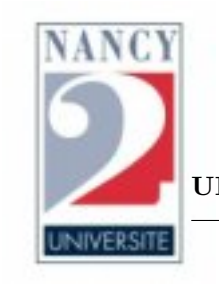
Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information

Sylvain Castagnos



UFR mathématiques et informatique **École doctorale IAEM Lorraine**

Département de formation doctorale en informatique

Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information

THÈSE

présentée et soutenue publiquement le 5 Novembre 2008

pour l'obtention du

Doctorat de l'université Nancy 2

(spécialité informatique)

par

Sylvain CASTAGNOS

Composition du jury

Rapporteurs :	Catherine BERRUT Alain MILLE
Examinateurs :	Makram BOUZID François CHARPILLET Kamel SMAILI Brigitte TROUSSE
Directrice :	Anne BOYER

Laboratoire Lorrain de Recherche en Informatique et Applications
Equipe MAIA/KIWI
Campus Scientifique, B.P.239 - 54506 Vandœuvre-lès-Nancy Cedex

Remerciements

Table des matières

Introduction

1 Constat de départ

Internet constitue un environnement évolutif, non structuré et quasi-infini proposant des documents hétérogènes notamment à travers le Web et les intranets documentaires. La quantité de données présentes sur ce réseau croît exponentiellement (20 téra-octets en 2000, 167 téra-octets en 2002 (Lyman et Varian, 2003), et au moins 480 téra-octets en 2005 rien que pour le Web de surface). A titre d'exemple, le nombre de pages référencées par Google passait de 1 à 8 trillions entre juin 2000 et août 2005. La recherche et l'accès à cette profusion de documents nécessitent d'assister l'utilisateur. Cependant, les outils actuels d'accès à l'information atteignent leur limite. En effet, la recherche par mots-clefs avec opérateurs logiques apparaît complexe pour une importante partie du grand public non formée à l'utilisation de ces outils informatiques. La plupart des internautes choisissent mal les mots-clefs et/ou ne savent pas les associer correctement au sein d'une même requête (également appelée "équation de recherche"). Le choix de l'équation de recherche est pourtant primordiale, car deux requêtes telles que "actualité New-York" et "information New-York" sont en apparence semblables – c'est-à-dire en lien avec le même thème – mais peuvent retourner des propositions très différentes. De fait, il devient difficile pour les usagers d'identifier les ressources les plus pertinentes (également appelées "items") dans un temps raisonnable, dans la mesure où le nombre de résultats retournés est colossal. Google propose par exemple plus de 5 milliards de liens pour la requête "news". Il reste encore 379 millions de sites à consulter pour les "news" relatives à New-York. D'un point de vue plus formel, chaque utilisateur a potentiellement accès à N ressources, N tendant vers l'infini. Non seulement il est impossible pour un individu de consulter toutes les ressources disponibles, mais en plus les recherches d'un individu sur un thème donné peuvent ne jamais terminer puisque de nouvelles ressources apparaissent sans cesse. Or, dans une telle situation, un utilisateur fait preuve d'une rationalité limitée à l'ensemble de choix que peut envisager l'entendement humain et il s'arrête au premier choix qu'il juge satisfaisant (Simon, 1982). C'est la raison pour laquelle la pertinence optimale des résultats – en terme de satisfaction de l'utilisateur – n'est plus garantie dans les systèmes de recherche et d'accès à l'information actuels.

Ce constat vis-à-vis du nombre de ressources et de l'insatisfaction engendrée dépasse le simple cadre de la recherche et de l'accès à des documents sur le Web. Il s'applique plus globalement aux *e-services*, incluant les portails intranet d'entreprise, les services commerciaux en ligne (sites de *e-commerce* proposant des produits dans un catalogue), les services mobiles, etc.[1] Le constat

[1] Charton (2003); Boyer et al. (2008) définissent un service comme un intermédiaire établissant un lien entre uti-

s'étend également au nombre de services accessibles et au nombre d'usagers pouvant interagir entre eux. Aussi, la communauté scientifique s'emploie à repenser les services existants sous la dénomination de Web 2.0 (White, 2006). Pour plus de facilité, le terme "recherche et accès à l'information" désignera l'ensemble de ces services dans la suite de ce manuscrit.

En résumé, chaque utilisateur est en permanence confronté à une surabondance d'items, d'interlocuteurs ou de services sur internet. Il devient alors nécessaire de l'assister et de faciliter sa recherche de ressources pertinentes.

2 Problématique scientifique

Parmi les solutions envisagées pour relever ces deux défis, on retrouve l'adaptation des interfaces pour faciliter l'exploration et la recherche au sein d'un service (Gajos et al., 2008), des systèmes à base de navigation sociale (Millen et al., 2006), des sites proposant de l'information au contenu personnalisé (Montaner et al., 2003), les approches fondées sur le raisonnement à partir de cas pour l'aide à la navigation (Trousse et al., 1999), des systèmes présentant les résultats de recherche par groupes d'items (Bekkerman et al., 2007), des outils statistiques aidant les utilisateurs à mieux définir leurs recherches par mots-clefs (Smyth et al., 2005), etc. Une autre approche consiste à fournir aux usagers des items intéressants individuellement tels que publiés. La finalité d'une telle approche est de proposer une information ciblée, déduite d'une suite d'interactions avec les principaux intéressés. Contrairement à la personnalisation de contenu (Kurki et al., 1999), cette solution ne nécessite pas d'adapter les items au lectorat potentiel. Il s'agit de proposer une information, telle qu'elle a été éditée, aux personnes potentiellement concernées (diffusion ciblée). De même, il faut être capable de proposer à un utilisateur donné un ensemble de ressources pertinentes (filtrage pertinent).

Que l'on souhaite faire de la diffusion ciblée ou du filtrage pertinent, il convient d' "apprendre l'utilisateur courant"[2], c'est-à-dire de modéliser ses préférences à partir des observations collectées dans le cadre d'un *e-service*. La connaissance de l'utilisateur permet au système de fournir des items susceptibles de l'intéresser, c'est-à-dire répondant au minimum à l'un de ces impératifs :

- Etre utile à l'utilisateur (Horvitz et Klein, 1993) ;
- Correspondre à ses préférences présentes, qu'elles soient connues ou inconnues du système. Cela inclut les items qu'il ne connaît pas encore mais pourrait apprécier ;
- Accroître sa satisfaction vis-à-vis du service. Il s'agit de maximiser la probabilité que l'utilisateur courant réemploie et/ou fasse de la publicité pour ce service au vue des interactions passées. Sur Internet, la qualité de l'expérience utilisateur conditionne entre autres le succès d'un service et améliore la fidélisation des usagers (Bénard, 2002). Le paradoxe est qu'il faut fidéliser l'utilisateur pour le connaître et le connaître pour le fidéliser ;
- Répondre à ses attentes à court, moyen ou long terme.

lisateurs et ressources. Le service peut soit préexister, soit être construit dynamiquement à partir des interactions avec l'utilisateur et les ressources.

[2]Dans la suite de ce manuscrit, nous appelerons "utilisateur courant" un individu donné à qui nous souhaitons fournir des ressources pertinentes, par opposition au terme "utilisateurs" désignant la population globale.

Ces critères peuvent être considérés d'un point de vue quantitatif : il est possible d'évaluer numériquement l'utilité, la satisfaction, etc. On peut alors introduire la notion de pertinence : un item suggéré est d'autant plus pertinent qu'il répond à ces impératifs.

En résumé, la modélisation des préférences de l'utilisateur courant permet de comprendre ses besoins et attentes afin de proposer à l'utilisateur des items adéquats. Cette modélisation nécessite de recueillir des données brutes relatives à l'utilisateur courant (observations), soit en lui demandant de se décrire, soit en analysant les traces laissées lors des interactions avec le service (actions des utilisateurs). Etant donné la contrainte selon laquelle aucune compétence particulière n'est requise pour utiliser un tel système, il s'agit de solliciter l'utilisateur le moins possible et de collecter des observations disponibles pour caractériser une information de haut niveau, à savoir la connaissance de l'utilisateur. L'étape suivante consiste à identifier les données nécessaires et utilisables à l'apprentissage en situation de l'utilisateur courant. S'inscrire dans une approche d'Intelligence Artificielle rend ensuite possible la prédiction, voire l'anticipation des besoins de l'usager.

Les observations relatives aux utilisateurs sont nombreuses et variées. Selon la forme et la quantité des données accessibles, il est possible d'inférer des modèles de préférences plus ou moins précis (c'est-à-dire de modéliser l'utilisateur plus ou moins précisément). La difficulté de l'exercice réside dans la capacité à attribuer à chaque observation un degré de pertinence et d'y accorder plus ou moins de poids lors de l'élaboration des modèles. Cette étape est généralement réalisée au cas par cas en fonction des données collectables, c'est-à-dire disponibles et légales, conduisant à la conception de systèmes *ad hoc* difficilement transposables d'un cadre applicatif à un autre.

La problématique de base consiste donc à trouver un moyen générique de transformer des observations en modèles de préférences individuels synthétiques rigoureux et à les exploiter efficacement pour prédire les préférences inconnues.

3 Thèse soutenue

Un premier objectif de cette thèse est de démontrer qu'il est possible de généraliser à différents contextes applicatifs et différentes architectures l'exploitation des données disponibles caractérisant chaque utilisateur, afin d'améliorer les interactions avec un système d'accès et de recherche à l'information.

Nous posons pour hypothèse que chaque utilisateur a des goûts stables, mais pouvant varier au cours du temps. Les observations résultant des interactions entre l'utilisateur et le système représentent en quelque sorte des indicateurs probabilistes de ces goûts. Par conséquent, nous avons choisi d'inscrire la collecte des données relatives aux préférences des utilisateurs dans le cadre d'une modélisation stochastique. Toutefois, des difficultés viennent se greffer à la problématique de base énoncée précédemment. En effet, il n'est pas toujours aisé de recueillir suffisamment rapidement des traces sur les utilisateurs, afin de proposer des items pertinents dès les premières utilisations. Après synthèse des observations, les modèles utilisateurs, également appelés profils lorsqu'ils se présentent sous forme numérique, comportent un grand nombre de données manquantes. Mathématiquement parlant, cela se caractérise par un fort éparpillement des données

le long des axes du repère dans l'espace de représentation utilisateurs/items. En effet, chaque préférence inconnue pour un item se traduit par l'absence de représentation de l'utilisateur suivant l'axe associé à cet item (projection suivant l'axe). Or, ces préférences inconnues portent sur des items différents selon les utilisateurs, dispersant de ce fait les utilisateurs dans l'espace.

L'approche envisagée consiste à exploiter collaborativement les données relatives à une population pour pallier le manque d'information inhérent à chaque utilisateur. Les techniques de filtrage collaboratif, introduites pour la première fois par Goldberg et al. (1992), s'inscrivent dans cette approche. Elles consistent à observer un utilisateur dans un contexte donné, à le modéliser sous une forme numérique, puis à l'associer à un groupe d'individus similaires (i.e. les voisins les plus proches dans l'espace de représentation) pour lui suggérer des items pertinents. En d'autres termes, elles permettent de bénéficier de l'expérience et des interactions au sein d'une population pour améliorer les services et prédire les futurs agissements d'un individu (Bonnefoy et al., 2007). La phase de prédiction repose sur la construction de ces groupes d'individus, également appelés "communautés d'intérêts virtuelles"[3]. Cela s'apparente à de l'apprentissage non supervisé dans la mesure où il est impossible de valider les modèles communautaires *a posteriori*. La grande évolutivité des données diminue en effet l'impact des retours d'expérience. Cependant, les techniques de filtrage collaboratif présentent l'avantage d'offrir un service de personnalisation automatique et transparent pour l'utilisateur dans le cadre d'une approche incrémentale et probabiliste. Cette technologie constitue – avec d'autres approches complémentaires ou similaires telles que le filtrage par contenu (Melville et al., 2002; Basilico et Hofmann, 2004), la reformulation de requêtes (Hoashi et al., 2001), le raisonnement par cas (Trousse et al., 1999; Cordier et al., 2007), le traitement automatique des langues (Zitouni et al., 2003), les modèles markoviens (Boyer et Brun, 2007) ou les systèmes à base de critiques (Chen et Pu, 2007) – le cœur des systèmes de recommandations implantés au sein de services intelligents tels que les intranets documentaires, les assistants intelligents, les services de vente en ligne, etc.

Le filtrage collaboratif présente des caractéristiques intéressantes mais souffre de fortes limitations qui rendent son utilisation difficile dans un contexte industriel d'envergure. Ainsi, dans les approches de filtrage collaboratif traditionnelles, les calculs sont centralisés et le nombre d'individus pris en compte dans la recherche des plus proches voisins ne peut excéder quelques milliers de candidats (Sarwar et al., 2001). La constante croissance du nombre d'individus et de ressources ne doit pourtant pas pénaliser le système en terme de temps de réponse. **Le deuxième objectif de cette thèse est d'étudier l'apport de la distribution des procédés de filtrage collaboratif. Nous verrons notamment comment cette nouvelle approche permet d'assurer le passage à l'échelle, et faire face aux problèmes sous-jacents explicités dans la section suivante.**

[3]Sur le même principe, certains algorithmes de filtrage collaboratif construisent des groupes d'items similaires (c'est-à-dire communément évalués de façon semblable par les utilisateurs), plutôt que des groupes d'individus.

4 Enjeux

4.1 Passage à l'échelle

Le passage à l'échelle est la première motivation de la distribution des procédés de filtrage collaboratif. Les systèmes de recommandations doivent bien souvent prendre en compte des centaines de milliers voire des millions d'utilisateurs et/ou de ressources. Or, les temps de calcul lors de la constitution des communautés d'intérêts virtuelles doivent rester très courts comparativement au temps de navigation, afin de fournir aux usagers des suggestions en temps réel. L'objectif est en effet d'anticiper les besoins des utilisateurs en proposant les items pertinents le plus rapidement possible en amont de leurs recherches. Le problème du passage à l'échelle concerne également le nombre d'items accessibles au sein du système. La quantité de ressources partagées est souvent conséquente et ne cesse de croître avec le temps, augmentant de ce fait le nombre de dimensions dans l'espace de représentation utilisateurs/ressources. Les algorithmes de *clustering* (Ungar et Foster, 1998) permettant de construire des communautés virtuelles et/ou d'établir une classification des items basée sur les préférences sont souvent coûteux en terme de temps de calcul. L'approche classique, au sein d'architectures centralisées, consiste à effectuer les calculs les plus longs *offline* et à exploiter les résultats de ces calculs *online* en fonction du contexte. Toutefois, cette façon de procéder réagit mal à l'insertion de nouveaux utilisateurs et de nouveaux items. Prendre en compte les ajouts d'utilisateurs ou d'items requiert de réinitialiser les calculs. Il serait en effet beaucoup trop coûteux en temps de calcul de construire les communautés incrémentalement (i.e. utilisateur par utilisateur) dans une architecture centralisée. Une alternative possible est la distribution du filtrage collaboratif, permettant de paralléliser les calculs et d'exploiter les capacités de plusieurs ordinateurs. Cela permet d'une part d'accroître le nombre d'utilisateurs et d'items qu'il est possible de prendre en compte, et assure d'autre part la réactivité du système au sein d'algorithmes *anytime* ou temps réel avec des performances accessibles. La distribution des calculs pose en revanche de nouveaux problèmes, notamment en matière de sécurité (cf. infra, section 4.6).

4.2 Gestion des données manquantes

Les algorithmes de filtrage collaboratif exploitent les comportements connus d'une population pour prédire les futurs agissements d'un individu à partir de son observation dans un contexte donné. Par conséquent, si les observations se révèlent insuffisantes, les modèles utilisateurs comporteront trop de données manquantes affectant par là même la qualité des prédictions. La méthode de collecte des données doit être aussi complète et diversifiée que possible en exploitant toutes les sources pertinentes d'informations relatives aux préférences des utilisateurs, afin de minimiser le nombre de données manquantes.

La rareté des données est particulièrement problématique à l'initialisation, lorsque les modèles utilisateurs ne comprennent pas ou peu de préférences connues. Il s'agit du problème du démarrage à froid. La précision des calculs au sein d'algorithmes de filtrage collaboratif nécessite en effet de disposer de suffisamment d'informations sur l'utilisateur courant. Si le profil de ce dernier est vide, le système est incapable de lui fournir des recommandations autres qu'aléatoires.

Si ce même profil n'est pas vide mais contient trop peu de préférences connues, il sera difficile d'identifier les voisins les plus proches et les recommandations seront peu satisfaisantes. Au sein de plate-formes à base de votes, les usagers ne prennent généralement pas le temps de fournir explicitement des notes en assez grand nombre sur les items pour obtenir de bonnes prédictions. Le nombre minimum de votes requis est estimé à 20 (Schickel et Faltings, 2006) pour sélectionner la population similaire adéquate et disposer d'un bon niveau de pertinence dans les prédictions.

De même, les algorithmes de filtrage collaboratif rencontrent des difficultés à cibler le lectorat intéressé par un nouvel item, tant que ce dernier n'a pas été évalué par un certain nombre d'utilisateurs. On parle alors de problème de "latence".

4.3 Respect de la vie privée

En analysant les actions et préférences des utilisateurs dans le but d'inférer des modèles et de proposer des items intéressants, les systèmes de recommandations intelligents présentent un caractère intrusif vis-à-vis de la vie privée des gens. De fait, les utilisateurs affichent souvent une certaine réticence à communiquer volontairement des informations personnelles à de tels systèmes sans connaître l'emploi qui en est fait. En revanche, ils ne sont pas toujours conscients du fait que les traces qu'ils laissent au sein de ces services de personnalisation peuvent être exploitées (à bon ou à mauvais escient). Le refus fréquent des utilisateurs de fournir explicitement des données personnelles a fait émerger une dérive insidieuse consistant à collecter les données à l'insu des utilisateurs notamment par le biais de *cookies* et de *spywares*, y compris dans des applications commerciales de renom. Le but affiché est souvent de garantir la qualité du service ou d'en tirer profit à des fins marketing. Ces pratiques ont eu pour effet de susciter la méfiance du public et de ralentir la démocratisation d'outils novateurs pouvant pourtant faciliter l'exploration du Web. A titre d'exemple, une étude menée auprès de 900 citoyens américains par ChoiceStream en 2005 (ChoiceStream, 2006; Jacquet et Drouot, 2007) montre que le frein principal aux techniques de personnalisation reste la peur de divulgation des données personnelles.

Le respect de la vie privée constitue donc un défi d'ordre éthique devant être pris en compte lors de la conception de services intelligents de personnalisation, afin de s'assurer la confiance des utilisateurs et atteindre une large audience. Par ailleurs, dans certains pays, la préservation de la vie privée est définie dans le cadre de la loi et/ou pris en charge par des organisations gouvernementales vigilantes au respect des droits des citoyens. Ainsi, les pays membres de l'Union Européenne doivent par exemple se conformer aux directives du Conseil Européen en date du 28 Janvier 1981. La France a quant à elle promulgué la loi n°78-17 du 6 Janvier 1978, baptisée "Informatique et Libertés". La Commission Nationale de l'Informatique et des Libertés (CNIL) a pour mission de s'assurer du respect des dispositions de la loi sur la territoire français.

De l'état de l'art, il ressort que plusieurs approches ont été envisagées pour rendre les algorithmes de filtrage aussi peu intrusifs que possible dans la vie privée des utilisateurs, ou au moins garantir leur anonymat. Les méthodes les plus populaires telles que cryptage des données, l'altération des profils et l'emploi d'agrégats publics seront explicitées dans le chapitre 1 (cf. infra, 1.4 Données utilisables et proposition d'un modèle de préférences, p.24).

4.4 Qualité des prédictions

Internet s'adresse à un public exigeant et la plupart du temps peu formé aux outils informatiques. Au sein de systèmes de recherche et d'accès à l'information, les utilisateurs attendent des items pertinents dans un délai très court. Par ailleurs, ils cessent rapidement de recourir à un tel système dès lors qu'il ne leur fournit pas entière satisfaction. Pour fidéliser les usagers, il faut donc s'assurer d'atteindre rapidement un bon niveau de qualité dans le choix des recommandations.

Dans le cas du filtrage collaboratif, le service de personnalisation passe par une modélisation statistique et/ou stochastique des utilisateurs et s'améliorent au cours du temps. Plus le système dispose d'informations relatives aux préférences des usagers, plus la qualité des prédictions s'améliore. Fournir des recommandations optimales selon les critères d'utilité ou de satisfaction suppose que les usagers manipulent le système pendant un délai suffisant et de façon régulière, afin de collecter suffisamment de données. La durée de la phase d'initialisation pendant laquelle les suggestions sont encore incertaines dépend donc fortement de l'implication de l'utilisateur courant. Il est possible d'accélérer l'initialisation en faisant remplir à l'utilisateur courant un questionnaire lors de son inscription sur le service par exemple (Goldberg et al., 2000).

Outre le temps nécessaire à la bonne modélisation des préférences de l'utilisateur, il faut que ce dernier accepte de communiquer au système des informations le concernant. Il faut donc trouver un compromis entre le respect de la vie privée et la qualité des prédictions. L'étude menée par ChoiceStream en 2005 (ChoiceStream, 2006) a révélé que 80% des personnes interrogées se déclarent en faveur de la personnalisation. En dépit de la crainte de diffusion des informations personnelles, il ressort que les personnes sondées sont prêtes à certains compromis en matière de tolérance à la surveillance (analyse des traces), en échange d'un effort moindre pour renseigner des formulaires.

Bien évidemment, la qualité des prédictions ne dépend pas uniquement de la quantité de données collectées lors de la phase de modélisation des préférences, mais requiert également l'emploi de méthodes de filtrage performantes. Différentes métriques ont été imaginées dans le but de mesurer la qualité prédictive des algorithmes de filtrage collaboratif (cf. infra, 2.2.2 Les critères d'évaluation en simulation, p.43). La qualité des prédictions constitue une exigence prioritaire transversale aux enjeux en matière de gestion des données manquantes et de respect de la vie privée.

4.5 Confiance

Promouvoir à travers un système de recommandations des items avec un haut degré de pertinence est une condition nécessaire mais non suffisante à la fidélisation des utilisateurs. Il faut également s'assurer que le système de recherche et d'accès à l'information leur inspire confiance. Cela se traduit de deux manières.

Premièrement, le processus de modélisation des préférences doit se faire en accord avec les souhaits des usagers. Nous avons vu précédemment que les utilisateurs sont souvent prêts à des compromis en matière de collecte des données personnelles. Toutefois, les internautes veulent garder le contrôle sur le paramétrage du module de personnalisation (Jacquet et Drouot, 2007).

Ainsi, ils souhaitent autant que faire se peut choisir les modes de collecte autorisés (implicite et/ou explicite), pouvoir accéder au contenu de leur(s) profil(s), sélectionner expressément les données partageables par opposition à celles devant rester confidentielles et définir le niveau de granularité souhaité pour les recommandations. L'approche visant à rapatrier tous les profils sur un serveur distant pour pouvoir traiter ces informations donne parfois aux utilisateurs l'impression de perdre le contrôle sur ces données. La distribution du filtrage collaboratif en terme de contenu permet de laisser à leurs propriétaires les items et les profils. Les usagers restent ainsi maîtres de leurs données et peuvent décider de l'utilisation qui peut en être faite.

Deuxièmement, le système de recommandations doit veiller à accroître la satisfaction des utilisateurs plutôt que d'émettre des suggestions en rapport avec une politique commerciale. Cette situation peut notamment être rencontrée dans le contexte des applications de *e-commerce* où l'hébergeur cherche à influencer les utilisateurs. A titre d'exemple, Amazon a déjà reconnu avoir générer de fausses recommandations afin de favoriser les nouveaux articles vestimentaires de leurs partenaires (Wingfield et Pereira, 2002).

4.6 Sécurité

La distribution des procédés de filtrage collaboratif suppose un plus grand nombre d'échanges de profils et de contenus à travers le réseau. Cela pose bien évidemment des problèmes de confidentialité des données lors de ces échanges. Ces questions peuvent être résolues en rajoutant des protocoles de cryptage au module de communication du système. Les données sont également plus vulnérables lorsqu'elles sont stockées sur des ordinateurs individuels que sur un serveur sécurisé (Central, 2008). Là encore, les données persistentes peuvent être conservées sous forme cryptée, procédé qui a déjà fait ses preuves pour la sauvegarde des mots de passe dans les navigateurs web par exemple.

Toutefois, outre la protection des données qui sont échangées ou stockées, de récentes recherches ont montré que les algorithmes traditionnels de filtrage collaboratif sont vulnérables à des attaques de pirates d'un nouveau genre. Ces attaques affectent la qualité des prédictions et des recommandations pour nombre d'utilisateurs diminuant ainsi leur confiance dans le système de recommandations. Il a été notamment prouvé que la présence d'un pourcent d'utilisateurs malveillants sur la totalité des utilisateurs suffit à perturber grandement un tel système (Burke et Mobasher, 2005).

Ainsi, un scénario typique d'attaque consiste pour un pirate à récupérer le profil d'un autre utilisateur, puis à l'altérer légèrement, à y ajouter des contenus qu'il souhaite introduire de force dans la liste des recommandations et à le renvoyer à cette même personne. Un algorithme classique de filtrage collaboratif calculera la distance entre les deux profils et les trouvera suffisamment proches pour suggérer à l'utilisateur les contenus suggérés par le pirate. Cette manipulation de tendances, sorte de "spam" des systèmes de recommandations, prend toute son ampleur dés lors que les pirates peuvent s'attaquer au système depuis plusieurs ordinateurs en utilisant des vers. Ce type de scénario peut grandement diminuer la qualité des recommandations et donc la satisfaction des utilisateurs. L'apprentissage supervisé est une des principales approches étudiées pour résoudre partiellement ce problème à l'heure actuelle (Burke et al., 2006).

4.7 Recommandations non sclérosées

Nous avons vu dans la section 4.2 que les données manquantes peuvent affecter la précision des calculs de manière conséquente. A l'inverse, disposer de trop d'informations relatives à l'utilisateur courant peut également se révéler problématique. Collecter des informations sur une trop grande période de temps peut conduire à disposer de préférences contradictoires au sein d'un même profil. En effet, l'hypothèse d'évolutivité des goûts n'est pas en contradiction avec la contrainte de stabilité imposée dans les procédés de filtrage collaboratif, tant que les comportements ne sont pas erratiques. De ce fait, on peut retrouver des votes contradictoires sur des items corrélés pour un même utilisateur lorsque ce dernier évolue fortement au cours du temps, puisque nous avons choisi d'opérer sa modélisation dans un contexte stochastique. Si son profil devient incohérent, l'utilisateur courant ne disposera plus de recommandations pertinentes et cessera rapidement de recourir au service de personnalisation. La validité des données étant tributaire du facteur temps, il faut dans un premier temps s'assurer que le profil ne prend en considération que les préférences actuelles de l'utilisateur. Pour ce faire, les solutions classiques de la littérature consistent à appliquer au modèle utilisateur un cœfficient d'évaporation qui croît avec le temps (Ding et al., 2006). Cela permet au système d'atténuer le poids des votes les plus anciens jusqu'à l'oubli de ceux n'ayant pas fait l'objet d'une actualisation de la part du principal intéressé.

Toutefois, même un profil reflétant les préférences actuelles d'un individu peut se révéler problématique. Ainsi, une trop bonne modélisation des préférences de l'utilisateur engendrera des suggestions sclérosées. Quand un système de recommandations connaît parfaitement l'utilisateur courant, il a souvent tendance à favoriser les items que cet usager a noté le plus favorablement au détriment de la nouveauté. Or, un des aspects intéressants du filtrage collaboratif est sa capacité à fournir aux utilisateurs des items susceptibles d'êtres intéressants, mais qu'ils n'auraient pas consultés spontanément. Certaines personnes préfèrent disposer de suggestions variées, plutôt que d'une liste d'items répondant à un besoin régulier ou qu'ils consultent fréquemment. Prenons un exemple simple pour illustrer ce problème. Imaginons un utilisateur qui souhaite faciliter ses recherches d'information à l'aide d'un service de personnalisation dans le cadre de son travail. Ce service lui fournira tous les items dont il aura besoin pour effecter ses tâches quotidiennes. Au demeurant cet usager peut également, souhaitant parfaire sa culture personnelle, consacrer une partie de son temps à la lecture d'items plus généralistes ou moins en lien avec ses besoins et centres d'intérêt. Le service doit alors se montrer flexible et adaptable aux souhaits des utilisateurs lors de la fourniture de suggestions.

4.8 Portabilité et mobilité

Un autre aspect à prendre en compte est la portabilité du système. Ce dernier doit se montrer rapide et efficace en toutes circonstances, quelles que soient les caractéristiques matérielles des postes clients. Les utilisateurs peuvent accéder à l'information depuis un ordinateur de bureau plus ou moins puissant, un ordinateur portable, un PDA ou même un téléphone. Les calculs doivent être suffisamment légers pour permettre au système de fournir des recommandations en temps réel sur chacun de ces supports.

La portabilité peut être reliée au besoin de mobilité. Ainsi, un utilisateur disposant du même système de recommandations sur son ordinateur et son PDA peut souhaiter synchroniser ses profils sur les deux appareils, afin d'accroître la précision des prédictions. Il existe plusieurs moyens d'atteindre cet objectif. La solution la plus répandue consiste à stocker les profils des utilisateurs sur un serveur (Cöster et Svensson, 2005). Le logiciel client cherche alors à y accéder lors des différents traitements nécessaires à l'élaboration d'une liste de suggestions. Ce procédé de stockage des profils est par exemple employé au sein du service Windows Live^{TM} (Thurrott, 2006).[4] Une autre approche repose sur la sauvegarde des profils localement sur les postes clients. L'application propose alors une procédure d'importation/exportation permettant de récupérer facilement les données. Cela présente l'avantage de rendre la propriété des profils standardisés aux utilisateurs, de sorte qu'ils puissent les réexploiter sur plusieurs supports et dans plusieurs applications. Le critère de mobilité est en lien avec les enjeux de sécurité et de confiance explicités précédemment.

4.9 Contexte

Nous définissons le contexte comme l'ensemble des circonstances d'utilisation et des attributs de l'usager pouvant influer sur son comportement[5]. Cette notion intègre les contraintes techniques, temporelles et humaines. La prise en compte de ce contexte d'utilisation peut avoir de l'importance lors de la phase de calcul. En effet, les usagers n'ont pas les mêmes attentes selon des critères tels que l'heure de la journée, leur localisation, leur humeur ou leurs caractéristiques personnelles (âge, sexe, profession, etc.). Ainsi, des travaux (Rasheed et al., 2005; Vildjiounaite et Kyllönen, 2006) ont illustré l'intérêt de prendre en compte le contexte temporel (matin, après-midi, soir) dans le processus de filtrage de l'information. La plupart des lecteurs ont par exemple besoin d'une information succincte le matin, afin d'être informés rapidement des événements majeurs. En revanche, ils disposent de plus de temps en fin de journée pour consulter des articles de fond.

Prenons à présent l'exemple d'un système de recommandations musicales, reposant sur le même principe que "Yahoo ! Music"[6]. L'utilisateur courant peut fournir un certain nombre d'informations sur les catégories musicales qu'il apprécie, ainsi que sur les morceaux et groupes qu'il préfère. Le système propose ensuite un service radiophonique personnalisé diffusant de la musique en rapport avec ses goûts. Toutefois, on peut considérer que cet auditeur n'aura pas les mêmes envies selon qu'il se trouve au travail ou à domicile, en vacances ou non, etc. De même, son humeur peut orienter ses choix. L'introduction de la notion de contexte prend alors tout son sens.

Enfin, il peut être intéressant de prendre en compte les compétences de l'utilisateur. Les intérêts de ce dernier peuvent par exemple s'orienter vers les langages de programmation orientés

[4] http ://www.microsoft.com/presspass/press/2005/nov05/11-01PreviewSoftwareBasedPR.mspx

[5] Boyer et al. (2008) fournissent une définition plus complète du contexte, comme étant l'ensemble des informations sur le dispositif d'accès (terminal, réseau, coût éventuel de connexion ou de transfert, qualité de la communication), la localisation géographique, le moment (date ou période de la journée ou de la semaine), la disponibilité (des ressources, du réseau, des serveurs), le coût (par exemple de l'accès à une ressource), les modalités possibles ou souhaitables en fonction de l'environnement, ...

[6] http ://new.music.yahoo.com/

objet. Au demeurant, le système ne devrait pas soumettre les mêmes recommandations selon qu'il s'agisse d'un étudiant qui découvre l'informatique ou au contraire d'un expert en Java.

La thèse soutenue dans ce manuscrit se décompose en trois points :
- **il est possible de modéliser les utilisateurs de façon générique à partir des données collectables ;**
- **il est possible de rendre ces modèles exploitables par un algorithme de filtrage collaboratif (i.e. sous une forme numérique) ;**
- **il est possible de rendre le filtrage collaboratif applicable à grande échelle. L'exploitation de ces modèles dans un cadre distribué permet en effet de faire face aux problèmes de passage à l'échelle tout en garantissant la gestion des données manquantes, le respect de la vie privée, la qualité des prédictions, la confiance dans le système, la sécurité des données, les recommandations non sclérosées, la portabilité du système, la mobilité des profils et la gestion du contexte.**

5 Contributions

Les contributions de cette thèse sont les suivantes :

- nous introduisons un formalisme unifiant les différents travaux menés jusqu'alors en matière de filtrage collaboratif (cf. chapitre 3) ;
- nous proposons un procédé de modélisation des préférences générique et respectueux de la vie privée des gens ;
- les algorithmes FRAC et FSB présentent la particularité d'être décentralisés et adaptés à une architecture client/serveur. Ces modèles s'attachent tout particulièrement à résoudre les problèmes de passage à l'échelle, de respect de la vie privée et de gestion des données manquantes ;
- l'algorithme AURA a été conçu pour distribuer les procédés de filtrage collaboratif en termes de calculs et de contenus au sein d'architectures pair-à-pair. Cette algorithme fournit des résultats en temps réel avec une bonne qualité de prédictions. Il prend en compte les problèmes de passage à l'échelle, de respect de la vie privée, de gestion des données manquantes, de portabilité, de confiance et évite les recommandations sclérosées en introduisant de la nouveauté. Il a fait l'objet d'une intégration dans notre plate-forme documentaire expérimentale SofoS conçue pour les besoins de cette thèse.

Les différents modèles proposés dans ce manuscrit ont fait l'objet d'intégrations dans des contextes industriels réels et présentant des contraintes différentes (SES ASTRA, Crédit Agricole S.A., Sailendra S.A.S., etc.). Cette mise en application a permis de démontrer la faisabilité d'un déploiement à grande échelle, de valider les modèles et d'appuyer les propos tenus dans cette thèse.

6 Organisation du manuscrit

Nous avons vu précédemment que la première étape lors de l'élaboration d'un service personnalisé consiste à "apprendre les utilisateurs". Nous commencerons donc par une discussion sur les différentes façons de concevoir des modèles de préférences exploitables dans un système à base de filtrage collaboratif (cf. chapitre 1). Nous présenterons en particulier une typologie des données collectables et la façon de les obtenir. Nous chercherons ensuite à identifier les données utilisables et nécessaires au bon fonctionnement d'un système de recommandations. Cette discussion nous conduira à proposer des notations et un méthode générique de modélisation des préférences respectueuse de la vie privée des utilisateurs.

La partie 2 réunira un ensemble de pré-requis relatifs au filtrage collaboratif et utiles à la bonne lecture de cette thèse. Nous introduirons un formalisme du filtrage collaboratif. Puis nous exposerons les différents critères d'évaluation et les corpus de tests utilisés pour valider les modèles ou pour comparer différentes approches.

Dans un troisième temps, nous présenterons un état de l'art détaillé des modèles à base de filtrage collaboratif. Nous montrerons comment les différentes problématiques scientifiques ont conduit les chercheurs à imaginer différentes familles d'algorithmes, depuis les premières méthodes purement statistiques jusqu'aux approches distribuées.

Les chapitres 4 et 5 présenteront nos modèles respectivement adaptés aux architectures client/serveur et aux grilles de calcul, et répondant au moins partiellement aux enjeux mentionnés plus haut. Des tests d'évaluation seront présentés, ainsi que l'intégration de ces modèles dans des applications industrielles réelles (diffusion satellitaire de sites Web, intranet bancaire).

Enfin, la dernière partie sera dédiée à la conclusion et aux perspectives de recherche.

Chapitre 1

Modélisation utilisateur

1.1 Objectif

L'objectif de nos recherches est d'améliorer les interactions entre le grand public et les services de recherche et d'accès à l'information. Cela peut passer par l'intégration d'un système de recommandations efficace au sein de l'architecture logicielle. Le rôle d'un système de recommandations est double :

- il s'agit d'un moteur de personnalisation proposant à chaque usager des ressources adaptées à ses attentes. Les techniques d'apprentissage automatique permettent de sélectionner l'information pour un utilisateur donné dans le but de lui procurer un gain de temps : l'information vient directement à lui. Cette "sélection" n'est pas nécessairement restrictive au niveau de l'accès aux documents. Elle peut consister à réorganiser l'ordre des propositions du système afin de mieux faire ressortir les ressources pertinentes parmi la masse de ressources. Les items sont donc triés selon un ordre de pertinence adapté à l'utilisateur. Ces mêmes techniques assurent également la présentation de chaque item auprès des personnes potentiellement intéressées ;
- un tel système comporte par ailleurs une fonction de prédiction. Il est en effet possible de suggérer à l'utilisateur courant des items pertinents, mais qu'il n'aurait pas consultés spontanément. Il s'agit là d'une façon d'anticiper ses besoins en prédisant l'intérêt qu'il aurait porté à tel ou tel document s'il l'avait déjà parcouru.

Dans les deux cas, nous parlerons de "filtrage" de l'information[7], le type de filtrage restant à définir. Afin de remplir son rôle, le système de recommandations doit apprendre à connaître l'utilisateur courant. La figure 1.1 illustre le fonctionnement d'un tel système, celui-ci s'apparentant à une plate-forme documentaire interactive, le principe s'appliquant également à la recherche et à l'accès aux ressources du Web.

Le procédé de filtrage se décompose comme suit :

[7]Dans un souci de simplification, on emploie souvent l'un ou l'autre des termes "service de personnalisation" et "système de recommandations" pour désigner les deux à la fois dans la mesure où le fonctionnement reste le même. Une définition rigoureuse ferait une distinction en terme de stratégies *Push-and-Pull*. Un service de personnalisation permet à l'utilisateur de "tirer" l'information pertinente à lui, alors qu'un système de recommandations le pousse à consommer de nouvelles ressources potentiellement intéressantes.

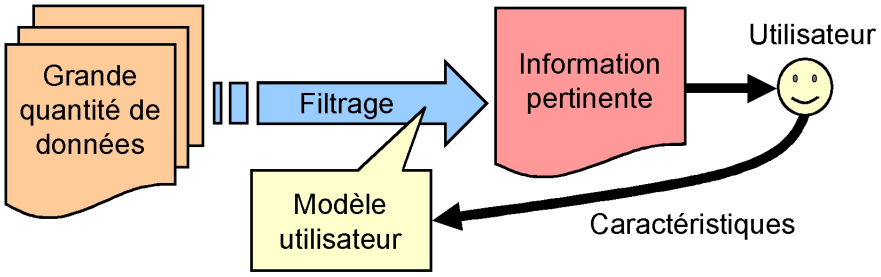


FIG. 1.1 – Filtrage personnalisé de l'information.

1. le système collecte un certain nombre de données relatives à l'utilisateur courant ;
2. ces données sont ensuite analysées et transformées sous une forme exploitable, appelée "modèle utilisateur". Dans le cas d'un système de recommandations à base de filtrage collaboratif, ce modèle se présente sous une forme numérique (on parle alors de "profil utilisateur") ;
3. le module de filtrage se charge ensuite d'identifier, à partir du modèle généré, les ressources pertinentes au sein du flux entrant d'informations (i.e. l'ensemble des items disponibles). Nous verrons par la suite que, dans le cas d'un algorithme de filtrage collaboratif, il est également possible de constituer des communautés d'intérêts virtuelles et donc de rapprocher les utilisateurs au même titre que les items.

Dans ce chapitre, nous introduirons une typologie des données qu'il est possible de collecter sur un utilisateur. Puis, nous discuterons des moyens d'obtenir celles-ci au sein du service de personnalisation. Par la suite, nous nous interrogerons sur la possibilité d'exploiter ces données. Leur utilisation est en effet soumise à certaines règles telles que le respect de la vie privée de l'usager. Nous identifierons alors les données que nous estimons utilisables et pertinentes au bon fonctionnement du système de recommandations dans le cas spécifique d'un algorithme de filtrage collaboratif.

1.2 Typologie des données collectables

Chaque utilisateur du service évolue dans un environnement dynamique dont il n'a qu'une connaissance partielle. Il peut partager des ressources avec les autres usagers ou accéder à de nouveaux documents via une plate-forme documentaire ou tout autre système de recherche et d'accès à l'information sur Internet. Les items disponibles sont hétérogènes (documents texte, vidéos, fichiers audio, sites web, blogs, etc.) et transitoires dans la mesure où leur durée de vie peut être limitée. Chaque item est considéré comme une unité élémentaire et insécable. Nous posons pour contrainte que le système ne requiert pas de connaissance *a priori* sur le contenu de ces items. Au demeurant, un identifiant décrivant de façon unique chaque item est nécessaire pour le localiser et le distinguer des autres ressources.

De ce fait, nous introduisons les notations suivantes :
– $U = \{u_1, u_2, \ldots, u_n\}$ est l'ensemble des n utilisateurs du système ;

- $u_a \in U$ désigne l'utilisateur courant et u_i un utilisateur quelconque ;
- $I = \{i_1, i_2, \ldots, i_m\}$ est l'ensemble des m items disponibles à un moment ou à un autre sur le système ;
- $I_t = \{i_a, i_b, \ldots, i_{m'} \mid m' \leq m\}$ est l'ensemble des m' items disponibles au sein du système à l'instant t. m' est variable dans la mesure où certains items peuvent être supprimés et de nouvelles ressources peuvent faire leur apparition. Ainsi, on a la relation d'ordre $I_t \subseteq I$;
- $< i_j, id_j >$ est le couple composé de l'item i_j et de son identifiant unique id_j ;
- $T_{i/id} = \{< i_j, id_j > | \ \forall i_j \in I\}$ est la table de correspondance item-identifiant. Celle-ci est généralement stockée sur un serveur.

Un système de recommandations peut à la fois être acteur et spectateur lors des interactions avec les utilisateurs. Il est acteur lorsqu'il a un comportement pro-actif (par le biais de sa fonction prédictive) ou lorsqu'il amorce une démarche d'acquisition de connaissances (en posant des questions à un utilisateur pour apprendre par exemple[8]). Le système est spectateur lorsqu'il se contente de collecter des données déjà disponibles à travers les interactions avec l'utilisateur : les traces récupérables au sein du service comprennent les usages, les goûts et le contexte. Nous appellerons "préférences" la synthèse de ces observations.

Les usages explicitent les relations entre les utilisateurs et les items, permettant entre autres de faire ressortir les habitudes de consultation. Ces usages peuvent être collectés pour chaque utilisateur grâce à une liste des actions effectuées et/ou à des annotations recueillies.

la liste d'actions effectuées. Elles illustrent les relations volontaires à durée limitée. Ces actions sont collectées dans des fichiers *access log* sur des serveurs. Un programme s'exécutant en arrière-plan appelé "démon http" se charge d'ajouter automatiquement une nouvelle entrée dans un ou plusieurs fichiers log à chaque fois qu'une requête atteint le serveur[9]. La syntaxe et les informations contenues dans ces fichiers log peuvent varier selon les systèmes. Toutefois, il y a un dénominateur commun à toutes ces variantes : on retrouve systématiquement les informations relatives à l'adresse IP du demandeur, l'url ou le chemin du fichier correspondant à la requête, ainsi que la date à laquelle cette demande a été formulée. La syntaxe la plus communément utilisée dans les fichiers log est le format CLF (W3C, 1995) que l'on retrouve sur des serveurs de type `Apache`. Chaque ligne du fichier log contient les champs suivants :

host user authuser [date :time] "request" status bytes

La table 1.1 fournit un exemple de fichier log contenu sur un serveur distant.

Dans la suite de ce chapitre, nous utiliserons la notation définie par Masseglia et al. (2004) pour les logs :

« Soit *Log* un ensemble d'entrées dans le fichier *access log*. Le fichier est composé de plusieurs lignes l_k. Une entrée g, $g \in Log$, est un tuple $g =< ip_{u_i}, \{(l_1^{u_i}.URL_{i_j}, l_1^{u_i}.time), \ldots, (l_q^{u_i}.URL_{i_j}, l_q^{u_i}.time)\} >$

[8] A titre de démonstrateur, les sites web comme *20q.net* et *akinator.com* se font fort de deviner les pensées des internautes en moins de 20 questions.

[9] Il est également possible d'implanter un démon http ou un plugin côté client dont le rôle sera de lister une action à chaque fois qu'un document sera affiché dans le navigateur.

192.168.0.5 - - [15/Feb/2008 :15 :50 :53 +0100] “GET http ://www.news.at/emedia/index.htm HTTP/1.1” - -
192.168.0.5 - - [15/Feb/2008 :15 :50 :53 +0100] “GET http ://www.news.at/nw1/ad_js.inc ?050127 HTTP/1.1” - -
192.168.0.5 - - [15/Feb/2008 :15 :51 :32 +0100] “GET http ://www.news.at/channels/540/main.shtml HTTP/1.1” - -
192.168.0.3 - - [15/Feb/2008 :15 :56 :20 +0100] “GET /loria/kiwi/castagnos/logs.txt HTTP/1.1” - -

TAB. 1.1 – Exemple de fichier log.

tel que pour $1 \leq k \leq q, l_k^{u_i}.URL_{i_j}$ représente l'objet demandé par l'utilisateur u_i à la date $l_k^{u_i}.time$, et pour tout $1 \leq t \leq k, l_k^{u_i}.time > l_t^{u_i}.time$. URL_{i_j} est une composante de l'item i_j. Si i_j est un site web, alors URL_{i_j} désignera une page de ce site. Si i_j est un document indivisible comme une vidéo par exemple, alors les notations URL_{i_j} et i_j seront équivalentes ».

les annotations. L'annotation est une interaction amorcée à l'initiative de l'utilisateur courant, lui permettant notamment d'exprimer explicitement son intérêt ou son désintérêt pour une ressource. Celles-ci peuvent prendre la forme d'un ensemble de notes ou de mots-clefs introduisant la notion d'*intention* de l'utilisateur.

Les notes (également appelées votes) sont des entiers naturels, relatifs ou des nombres réels fixés sur une échelle de valeurs arbitraire. Un vote élevé souligne la volonté de l'utilisateur de consulter l'item correspondant ou l'attrait qu'il a représenté pour lui si ce document est déjà connu. Un vote bas exprime au contraire le souhait de ne pas ou plus accéder à cette ressource. Lorsque l'utilisateur n'a pas fourni de note pour une ressource donnée, on lui attribue en général la valeur nulle. Nous appellerons $v_{i,j}$ le vote de l'utilisateur u_i sur la ressource i_j, borné par les valeurs v_{min} et v_{max}. Ce type de traces est souvent employé sur les sites de vente en ligne comme `Amazon` (cf. figure 1.2)[10].

Les mots-clefs sont des chaînes de caractères que les utilisateurs peuvent associer à des items. Ils peuvent ainsi fournir un retour d'expérience sur les points ayant attiré leur attention lors de la consultation des documents, sous une forme assez libre et propre à chaque individu. En d'autres termes, le choix des mots-clefs peut refléter l'intérêt suscité[11], la compréhension et l'effort de synthèse d'un utilisateur vis-à-vis d'une ressource. Cette approche permet de classer collaborativement les items selon une nomenclature plus fine que les catégories figées habituelles grâce à la participation de tous les lecteurs. Nous noterons $tags_{i,j} = \{tag_1, \ldots, tag_x\}$ le vecteur de mots définis par l'utilisateur u_i pour caractériser l'item i_j. Afin de faciliter les interactions entre les utilisateurs et le système, il est possible de proposer un lexique dans lequel choisir les mots-clefs. Cette approche est la clef de voûte des systèmes à base de navigation sociale (Millen et al., 2006). L'emploi de mots-clefs individualisés s'est démocratisé sur de nombreux sites web à caractère informatif, bien que l'exploitation automatique de ces *tags* fasse bien souvent encore défaut (cf. figure 1.3).

[10] http ://www.amazon.com/
[11] Rien que de mettre un mot-clef témoigne d'un intérêt de la part de l'utilisateur.

FIG. 1.2 – Exemple de site proposant aux clients d'attribuer des notes aux items.

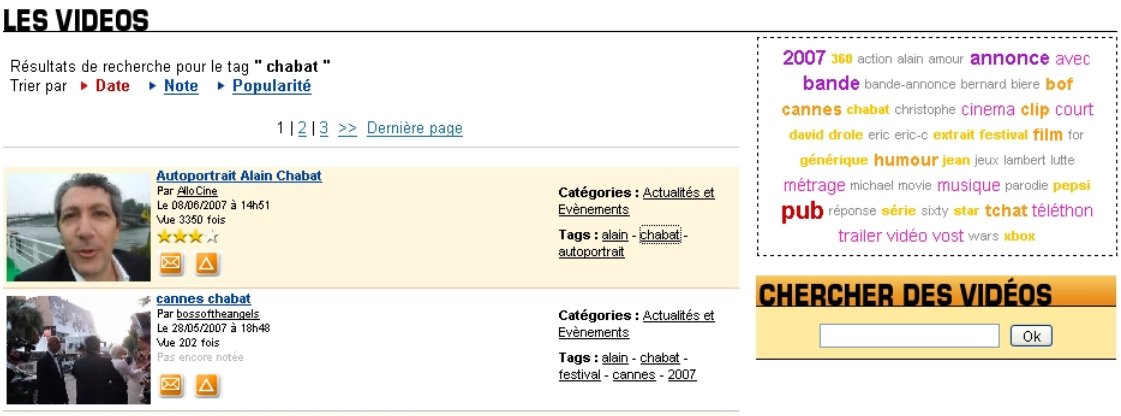


FIG. 1.3 – Exemple de site à base de mots-clefs (Allociné).

Les goûts expriment l'inclination des utilisateurs vis-à-vis de certains items ou catégories[12]. Ces informations peuvent être redondantes avec les usages, et en particulier avec les notes. Il s'agit, par exemple, de définir deux classes d'objets "Aime" et "Aime pas", notées respective-

[12]On appelle "Catégorie" un groupement d'items ayant au moins un attribut en commun. Dans le cas d'une base de films, il peut par exemple s'agir du genre cinématographique (films d'horreur, science-fiction, aventure), du réalisateur (films d'Alfred Hitchcock), etc.

ment C_{aime} et C_{aime_pas}. L'utilisateur a alors la possibilité de répartir l'ensemble des items ou des catégories dans une classe ou l'autre, au gré de ses envies. A titre d'exemple, l'utilisateur peut marquer les objets qu'il aime en cochant des cases via l'interface de certains sites comme `Yahoo ! Music` (cf. figure 1.4)[13]. Dans ce cas, les goûts de l'utilisateur ne peuvent être considérés comme des votes booléens, puisqu'une case non cochée revêt une double signification : soit l'utilisateur n'apprécie pas la ressource ou la catégorie non marquée, soit il n'a pas pris le temps d'exprimer une opinion. Par conséquent, l'ensemble des objets contenus dans C_{aime} et C_{aime_pas} ne recouvrent pas nécessairement la totalité des items et catégories proposés. Soit $IC = \{i_1, i_2, \ldots, i_m, c_1, \ldots, c_p\}$ l'ensemble des m items et des p catégories. On a : $C_{aime} \cup C_{aime\_pas} \subseteq IC$ et $C_{aime} \cap C_{aime\_pas} = \emptyset$.

FIG. 1.4 – Exemple de site proposant aux utilisateurs de sélectionner des catégories.

Le contexte permet de caractériser plus avant la situation et les moyens mis en œuvre lors des interactions entre un individu et son environnement. La proximité des évaluations des utilisateurs n'est en effet pas le seul critère applicable dans un système de filtrage (Nguyen et al., 2006). Lors de l'initialisation du modèle utilisateur, il est par exemple possible de collecter des informations descriptives (Lhoste, 2003) dont voici une liste non exhaustive : l'âge, le sexe, la situation géographique, la profession, le domaine d'expertise, la situation familiale, le tempérament. De même, le terme "contexte" peut également désigner des données décrivant l'environnement dans lequel est immergé l'utilisateur (au travail ou à domicile, seul ou en groupe, heure de la journée, données météorologiques, etc.) ou les caractéristiques du matériel dont il dispose (PC, PDA, téléphone portable, formats d'items acceptés, contraintes de taille, etc.).

[13] http ://new.music.yahoo.com/

L'ensemble de ces observations constituent des données brutes disponibles pouvant être exploitées afin de construire les modèles utilisateurs.

Les préférences hiérarchisent les goûts et les usages par ordre d'importance selon un critère tel que l'utilité. Horvitz et Klein (1993) ont introduit ce critère d'utilité comme suit :

« Soit $H_1,\ldots,H_n$ états exhaustifs et mutuellement exclusifs caractérisant le monde. L'action A_i consistant à lire une recommandation, prise dans le contexte d'un état du monde H_i, produit un résultat (A_i, H_j). La fonction $util(A_i, H_j)$ fait alors référence à l'utilité - c'est-à-dire la valeur ajoutée - pour l'utilisateur d'effectuer l'action (ou l'ensemble d'actions) A_i quand l'état H_j est vrai ».

Dans le cas présent, nous cherchons à modéliser l'utilisateur courant u_a. Les états du monde correspondent donc à des sous-ensembles de traces pouvant potentiellement être laissées par u_a et permettant de le caractériser (contexte, fréquence de consultation de tel ou tel document, etc.). Pour vulgariser, nous pourrions par exemple considérer que H_1 correspond à l'ensemble des traces pour un amateur de films de science-fiction, H_2 pour un lecteur assidu de littérature anglaise, etc. Bien sûr, dans la réalité, ces différents états ne sont pas aussi consistants et il n'est pas aisé de les étiqueter de la sorte. Dans ce contexte, la fonction d'utilité peut faire référence à une mesure *a priori* (on parle alors d'intérêt estimé) ou *a posteriori* (il s'agit dans ce cas de la satisfaction de u_a). La mesure de satisfaction vis-à-vis d'une ressource, bien que plus pertinente, n'est pas toujours accessible car elle nécessite un retour d'expérience de la part de u_a.

Les préférences intègrent à la fois les retours d'expérience et les votes estimés (Castagnos et Boyer, 2007c). Dans le cas d'une approche numérique telle que le filtrage collaboratif, les retours d'expérience peuvent se présenter sous la forme de votes réels fournis explicitement par l'utilisateur. Il est également possible de les inférer implicitement à partir des traces collectées. Cette étape requiert une fonction de transformation $f_{transf} : traces \rightarrow \mathbb{R}$. Ces votes, qu'ils soient fournis explicitement ou implicitement, constituent en fait une mesure d'utilité *a posteriori* puisqu'il s'agit de l'intérêt que porte l'utilisateur courant à ces ressources.

Les estimations constituent le cœur des systèmes de recommandations. Afin de présenter à l'utilisateur les ressources les plus pertinentes, le moteur doit être capable d'évaluer l'utilité *a priori* d'un item pour u_a, connaissant son passé.

Nous appellerons "profil utilisateur" l'ensemble des votes réels et estimés de l'utilisateur courant et il sera noté $\vec{u_a} = \{v_{a,j} \mid \forall i_j \in I\}$ tel que $v_{min} \leq v_{a,j} \leq v_{max}$.

1.3 Modes de collecte de données

La première étape, lorsque l'on cherche à modéliser les préférences de l'utilisateur courant, consiste à identifier les données que l'on souhaite collecter en complément des traces naturellement disponibles et une façon appropriée de le faire. Comme dans le domaine statistique, les enquêtes d'opinion semblent un moyen naturel et efficace de faire de l'élicitation de préférences (Chen et Pu, 2006; Castagnos et Boyer, 2007a). Cependant, la forme des questionnaires proposés peut grandement influencer les résultats. A titre d'exemple, les questions ouvertes

offrent une plus grande liberté aux personnes sondées mais engendrent des réponses complexes. A l'inverse, les questions fermées permettent de recueillir plus aisément les informations souhaitées mais constituent un biais en influençant les sujets interviewés. Même en trouvant un bon compromis, la modélisation complète des préférences de chaque utilisateur nécessiterait de poser des centaines de questions. Les usagers ne prennent généralement pas le temps de mener à bien un processus de modélisation aussi long. Par ailleurs, un questionnaire soumis à un instant t ne permet pas de tenir compte de l'évolution des préférences inhérente au contexte dans lequel l'utilisateur courant est immergé. Il faudrait donc soumettre périodiquement un questionnaire assez lourd. Toutefois, il peut être intéressant de proposer à chaque utilisateur une courte enquête lors de la phase d'initialisation de son profil. Cela permet de pallier partiellement le problème du démarrage à froid, en étant à même de proposer des items corrélés avec ce modèle minimaliste et en attendant de mieux définir l'utilisateur.

Une alternative aux enquêtes d'opinion consiste à assister l'utilisateur courant au fur et à mesure de sa recherche en lui posant des questions courtes et appropriées. Cette approche se révèle particulièrement pertinente dans le domaine du commerce en ligne (Chen et Pu, 2006). Certains produits vendus sur les sites marchands deviennent en effet complexes et comportent de nombreuses caractéristiques descriptives (appareils photographiques numériques, téléphones portables, consoles de jeux, etc.). Il devient alors difficile pour le consommateur non initié à ces technologies de faire son choix. Le risque financier associé à cette transaction est donc accru. C'est de ce constat qu'ont émergé les agents de recommandations basés sur les critiques (Chen et Pu, 2006) (cf. figure 1.5). Leur but est d'aider le client à trouver le produit idéal associé à ses besoins. Pour ce faire, ils mettent en place un mécanisme de retour d'expérience efficace. Chaque agent simule en quelque sorte un vendeur qui suggérerait des options en se référant aux préférences actuelles de l'utilisateur et en suscitant un retour d'expérience progressif sous la forme de critiques (par exemple, en souhaitant un produit moins cher ou avec une meilleure autonomie). Ces critiques permettent à l'agent d'affiner l'estimation des besoins spécifiques du client. Le système est obligé de procéder par étapes, dans la mesure où le consommateur est souvent incapable de spécifier dès le départ la liste complète de ses exigences (en particulier lorsqu'il n'est pas familiarisé avec ce type de produits). Cette tâche serait d'ailleurs trop fastidieuse. L'agent construit donc incrémentalement le modèle de préférences.

Il existe plusieurs familles de systèmes à base de critiques (Chen et Pu, 2007). Ainsi, certains disposent d'une interface dite "dynamique", présentant des critiques unitaires (relativement à une caractéristique du produit) ou composées. Les critiques composées constituent des règles d'association estimées statistiquement entre plusieurs critiques unitaires. Par exemple, le choix d'un appareil photographique moins cher signifie implicitement une résolution moindre. Dans ce contexte, l'utilisateur se contente de fournir des critiques unitaires. Le système lui propose alors un produit et lui permet de choisir parmi un ensemble de critiques composées dans l'hypothèse où le produit présenté ne lui convient pas. Les critiques portent donc à chaque pas de temps sur une seule recommandation. L'algorithme de recherche exploite des mesures de similarité et de compatibilité, ainsi que des critiques qualitatives ("moins cher", "fabricant différent", "résolution moins élevée", etc.).

Une autre famille de systèmes propose une interface disposant d'exemples de produits. Dans ce type de situation, l'utilisateur choisit à chaque pas de temps un produit parmi les exemples

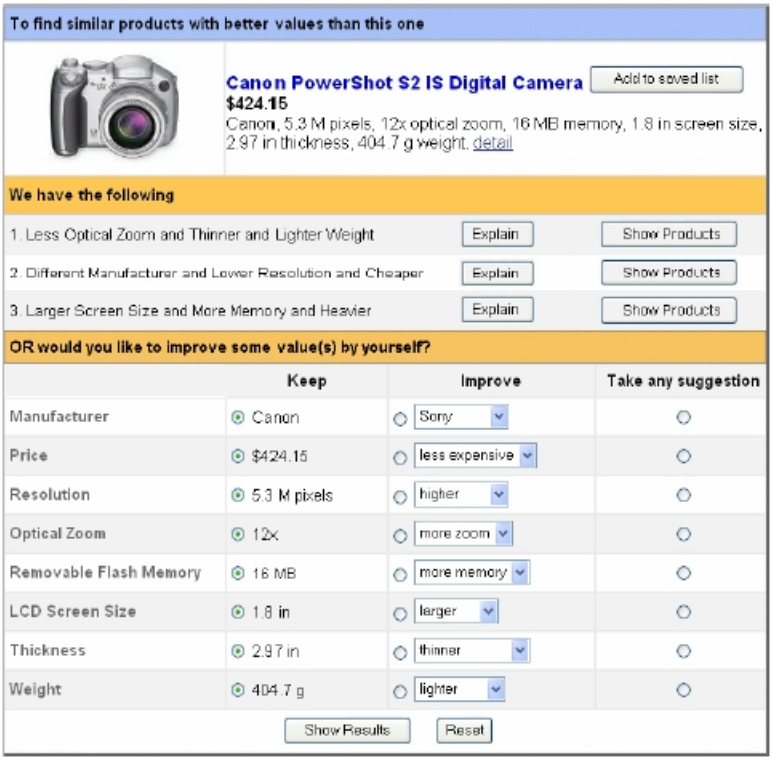


FIG. 1.5 – Exemple d'interface basée sur des critiques (Chen et Pu, 2006).

proposés. Il a ensuite la possibilité d'émettre et de combiner librement des critiques sur ce produit pour améliorer simultanément un ensemble de critères. Les critiques peuvent être qualitatives, basées sur la similarité ("semblable à ce produit") et/ou quantitatives ("moins cher de 100 euros"). L'algorithme de recherche procède alors par élimination.

On retrouve également des algorithmes hybrides combinant les approches dynamiques et à base d'exemples. Les systèmes reposant sur les critiques se révèlent performants lorsque l'utilisateur courant a une idée précise du type d'item recherché (ex : appareil photographique), à condition de disposer d'une description détaillée des produits et d'avoir un ensemble figé de critères de comparaison. Cela se rapproche du raisonnement par cas (Leake, 1996; Fauré et al., 2006b,a), fournissant des résultats très pertinents dans un domaine précis lorsque l'on dispose d'une base de connaissances[14]. Cette approche n'est donc pas nécessairement adaptée aux systèmes de recherche et d'accès à l'information grand public, offrant un cadre d'exploitation plus large pour lequel il n'est pas aisé de trouver des experts.

Une autre solution consiste à laisser les utilisateurs annoter explicitement et librement pour les items de leur choix. Chaque usager peut accéder à la liste des ressources qu'il a consultées, qu'il partage ou qui sont rendues accessibles par autrui. Il peut alors attribuer, pour chacun de ces items, des mots-clefs ou une note sur une échelle de valeurs arbitraire sans contrainte temporelle ou d'ordonnancement (cf. supra, section 1.2). Il peut également réévaluer ces items quand bon lui

[14]Le raisonnement par cas résout les problèmes en retrouvant des cas analogues dans sa base de connaissances et en les adaptant au cas considéré.

semble, de sorte que la modélisation des préférences tiendra compte de l'évolutivité des goûts. Ces annotations, quelles soient numériques ou symboliques, permettent entre autres à des algorithmes de filtrage d'organiser automatiquement les items et de faciliter les recherches. Cutrell et al. (2006) proposent par exemple un système personnel de recherche et d'accès à l'information dont la classification automatique des items repose sur les tags. Les notes permettent quant à elles de réordonner les items par ordre de préférence (ex : musique préférée sur `iTunes`[15], films favoris sur `allociné`[16], etc.). Mais l'utilité de ces annotations ne s'arrête pas à la simple fonction de classification ou d'ordonnancement. Elles constituent notamment les données entrantes dans les systèmes de recommandations à base de navigation sociale. Dans le cas de la navigation sociale, des algorithmes de filtrage se chargent d'analyser la fréquence des tags pour une ressource donnée, de hiérarchiser les mots-clefs sous la forme d'une ontologie et/ou de rechercher des similarités entre les tags pour réduire leur nombre (Millen et al., 2006). Le système présente ensuite à l'utilisateur un nuage de tags. Chaque mot-clef y est représenté avec une taille proportionnelle à sa popularité. Les tags populaires sont des pointeurs vers un ensemble d'items suggérés. Des sites et/ou logiciels dédiés à la conservation des favoris et reposant sur cette technologie ont ainsi vu le jour, tels que `Del.icio.us`[17] ou `Scuttle`[18].

Ces modes de collecte apparaissent également dans le filtrage collaboratif, exploitant pour sa part des données numériques. Le plus souvent, il s'agit d'inférer les suggestions à partir des votes réels fournis par les utilisateurs (cf. "annotations", p.16). Les communautés et les groupes d'items sont alors formés selon un seul critère, tel que la proximité des évaluations des utilisateurs (Perugini et al., 2003). Parfois, en combinant filtrage collaboratif et filtrage par contenu (Schein et al., 2002) ou simplement en étiquetant les items, les goûts exprimés relativement à une catégorie de ressources (cf. supra, les goûts, p.17) permettent de pallier partiellement le problème du démarrage à froid. En effet, lors de la phase d'initialisation, le profil de l'utilisateur courant ne comporte pas ou peu de votes. Nous avons vu en introduction que les données manquantes présentaient un impact fort sur la qualité des recommandations. Lorsque le système ne dispose d'aucun vote, il est contraint de proposer des ressources globalement populaires pour l'ensemble des utilisateurs ou des items sélectionnés au hasard. Intégrer les goûts (i.e. l'intérêt de l'utilisateur pour certaines catégories d'items) en tant que retour d'expérience permet d'affiner cette sélection aléatoire en soumettant des items appartenant à des thèmes considérés comme intéressants pour l'utilisateur courant. Cela suppose que ce dernier ait pu renseigner ces catégories lors de son inscription au service. A titre d'exemple, lors de l'initialisation du profil sur `Yahoo ! Music`, le système n'est pas capable de déterminer l'intérêt de l'utilisateur pour une chanson spécifique, mais il pourra se rapprocher de ses attentes en diffusant des morceaux de musique classés dans des thèmes qu'il apprécie.

Une approche plus originale consiste à tirer profit d'une variété de critères (également appelés "attributs") lors de la formation des communautés. Certains systèmes de filtrage hybride disposent entre autres de données contextuelles (cf. supra, le contexte, p.18), parmi lesquelles les données démographiques (Bouzeghoub et Kostadinov, 2005). Ainsi, Nguyen et al. (2006)

[15] http ://www.apple.com/itunes/
[16] http ://www.allocine.fr/
[17] http ://del.icio.us/
[18] http ://sourceforge.net/projects/semanticscuttle/

explorent les façons d'intégrer ces données, ainsi que les goûts, lors de la formalisation des espaces de communautés. Leur approche repose sur la théorie des ensembles d'approximation, permettant par là-même d'identifier les dépendances entre les attributs des données. L'introduction de ces dépendances compense l'absence de certaines valeurs d'attributs en instaurant des règles de décision. Il faut alors distinguer les attributs dits "conditions" (correspondant aux critères dont les valeurs sont disponibles) d'un attribut "décision" (c'est-à-dire la valeur manquante). Pour illustrer ce principe, considérons un système caractérisé par l'ensemble des attributs $A = \{Profession, Ville, Genre, Evaluation\}$. Un exemple de règle de décision serait : $(Profession = ATER, Ville = Nancy, Genre = Fiction) \rightarrow (Evaluation = Groupe5)$. Lorsque plusieurs valeurs sont manquantes ou douteuses, il est nécessaire d'élaborer une stratégie pour fixer l'ordre dans lequel compléter les attributs "décisions". Nguyen et al. (2006) proposent de les ordonner en fonction d'une mesure de qualité (consistance, consistance approximative, réductions approximatives). Cette approche se révèle intéressante dans les situations où l' utilisateur courant fournit des informations personnelles. Ce genre de circonstance n'est au demeurant pas systématique au sein des systèmes de recherche et d'accès à l'information. Rappelons qu'il est également techniquement possible de récupérer ces informations à partir des logs (cf. supra, listes d'actions effectuées, p.15) lors des interactions avec le système. En effet, certains scripts[19] déduisent de l'adresse IP un certain nombre d'informations relatives à l'utilisateur : heure locale, fuseau horaire, pays, ville, province, latitude et longitude, monnaie (euros, etc.), nationalité, fournisseur d'accès, nom de domaine. Ces scripts profitent également des échanges entre le client et le serveur pour récupérer les caractéristiques de la machine[20] : types de fichiers et d'encodages acceptés, autorisations relatives au Javascript et aux cookies, taille de la fenêtre, nombre de couleurs, langue du navigateur, nom et version du navigateur, numéro et nom de l'onglet (Firefox 2), nom de l'utilisateur (session windows), nom de l'ordinateur, système d'exploitation, type de processeur, taille de l'écran, etc. Toutefois, ce mode de récupération implicite des données peut dans certains cas entrer en conflit avec les contraintes légales en matière de respect de la vie privée, en particulier si l'utilisateur n'a pas donné expressément son accord.

Une approche plus respectueuse de la vie privée consisterait donc à se contenter d'une définition explicite de la part de l'utilisateur. Cependant, une collecte explicite des données peut s'avérer insuffisante, notamment pour des raisons d'ergonomie[21]. Le choix d'un mode de collecte explicite pose également problème en terme de validité des votes fournis. Des études psychologiques (Payne et al., 1993) ont montré que les gens bâtissent leurs préférences en découvrant les items disponibles. Cela signifie que les votes renseignés *a priori* ne sont pas nécessairement pertinents. Or, peu d'utilisateurs fournissent un retour d'expérience relatif à leurs consultations.

Un des défis à relever lors de la phase de modélisation consiste à trouver un bon compromis entre respect de la vie privée et qualité du modèle. En d'autres termes, il s'agit d'identifier les données utilisables légalement et nécessaires au bon fonctionnement d'un système de recommandations. La section suivante est dédiée à une discussion sur ce point : après un état de l'art

[19] Javascript, VBScript, ActiveX

[20] Ces informations peuvent être utilisées pour déterminer le type de ressources qu'il est possible de suggérer à l'utilisateur en fonction des capacités de sa machine (document vidéo, audio, textuel, etc.).

[21] Les choix d'interfaces et les politiques stratégiques et commerciales d'une entreprise influent grandement sur les informations que l'utilisateur peut fournir. Nous avons par exemple posé pour hypothèse qu'il n'était pas toujours possible de disposer des informations démographiques.

des principales approches existantes en matière de respect de la vie privée, nous proposons une méthode générique et non intrusive permettant d'apprendre à connaître l'utilisateur courant.

1.4 Données utilisables et proposition d'un modèle de préférences

Le respect de la vie privée constitue à la fois un problème éthique et une contrainte légale. Dans la mesure où les lois varient selon les pays, nous posons comme pré-requis qu'un système de recommandations doit s'avérer aussi strict et contraignant que la plus rigoureuse des lois[22]. Westin (1967) a fourni un cadre de travail en définissant la confidentialité comme la capacité à déterminer pour nous-mêmes quand, comment et dans quelle limite une information personnelle peut être communiquée à autrui. Cranor (2005) a complété cette définition en considérant les façons de collecter, stocker et utiliser les données. Afin de mesurer le degré de vigilance d'un système de recommandations sur ce point, elle introduit quatre axes de personnalisation :

- la méthode de collecte des données ;
- la durée de conservation des données ;
- l'implication de l'utilisateur dans le processus de personnalisation ;
- la confiance dans les suggestions.

Comme illustré précédemment, la méthode de collecte peut être explicite ou implicite. Dans le premier cas, la personnalisation s'appuie sur des informations fournies expressément par l'utilisateur, tels que des votes ou des données démographiques. Par exemple, un utilisateur peut attribuer une note à un échantillon d'items dans le but de recevoir des suggestions de nouvelles ressources potentiellement intéressantes. A l'inverse, une personnalisation basée sur une collecte implicite des données infère les préférences inconnues de l'utilisateur sur la base de ses usages (historique de navigation, historique d'achats, mots-clefs et équations de recherche saisis dans les moteurs, etc.).

La durée de conservation des données peut se mesurer à l'échelle d'une tâche à réaliser ou d'un profil à constituer. A l'échelle d'une tâche, le procédé de personnalisation ne détient les informations personnelles que pour une durée très limitée. Il peut par exemple s'agir de se restreindre à l'exploitation des actions de l'utilisateur lors de la session en cours. A l'échelle d'un profil, il s'agit au contraire de garder toutes les préférences, quelles que soient leur forme (implicite ou explicite, numérique ou symbolique), pour une durée indéterminée voire illimitée. Les *cookies* appartiennent à cette catégorie en reconnaissant automatiquement les visiteurs d'un site web, même lorsque leur dernière visite remonte à une période lointaine.

Le troisième axe de personnalisation distingue les procédés initiés par l'utilisateur de ceux initiés par le système. Dans la première catégorie, les utilisateurs ont tout loisir de définir leurs propres paramètres. A titre d'exemple, ils peuvent choisir le niveau de personnalisation exigé (proche de leurs besoins et intérêts, ou au contraire ouverts à la nouveauté), le nombre d'items à

[22] Les lois régissent à la fois les besoins organisationnels et techniques (en termes d'acquisition des données, d'objectif d'utilisation, de transfert et de traitement des données) pour les systèmes de recherche et d'accès à l'information stockant et/ou exploitant des données personnelles afin d'assurer leur protection (voir Wang et Kobsa (2006) pour une vue d'ensemble des lois en vigueur).

afficher, la partie des préférences déduites qu'il est possible de partager avec le reste du monde, etc. Dans le second cas, le système fournit systématiquement un même niveau de personnalisation à chacun, sans se préoccuper des particularités propres aux attentes individuelles.

Enfin, Cranor discerne la confiance dans les suggestions entre les deux grandes approches de filtrage, à savoir les systèmes basés sur les prédictions (ou filtrage collaboratif) et ceux basés sur le contenu. Une personnalisation reposant sur des prédictions confronte les préférences de l'utilisateur courant avec d'autres profils similaires afin de prédire ses futurs intérêts et fournir des recommandations basées sur les préférences d'autrui. Bien évidemment, qui dit prédiction, dit possibilité d'erreur (Zaslow, 2002). Toutefois, ce type de personnalisation fournit globalement de bons résultats, à condition qu'il y ait suffisamment de données à comparer. Cette approche permet également de suggérer des items potentiellement intéressants pour l'utilisateur, mais pas directement en lien avec ses consultations passées. A l'inverse, la personnalisation basée sur le contenu (ou "filtrage à base de contenu") favorise les similarités entre items, plutôt qu'entre utilisateurs. Ainsi, pour illustrer schématiquement, si un utilisateur achète un livre sur les réalisateurs de films, un tel système proposera d'autres livres sur le même thème. Cette approche est beaucoup plus respectueuse de la vie privée des gens, car elle ne nécessite pas de communiquer des informations personnelles aux autres usagers. Un algorithme basé sur le contenu commence par indexer les ressources. Dans le cas de ressources textuelles, une façon de le faire consiste à parcourir les documents, éliminer les mots-clefs non pertinents (les mots de liaisons, les articles définis et indéfinis, les pronoms, etc.) et compter le nombre d'occurrences des mots restants (Salton et McGill, 1983; Montaner et al., 2003). Ces informations peuvent par exemple être compilées sous forme de vecteurs de mots constituant les profils des documents. Le modèle de l'utilisateur synthétise alors l'ensemble des profils de documents consultés. Le procédé de filtrage peut ensuite comparer les nouveaux items au modèle de chaque utilisateur et de recommander les documents les plus proches. Cette famille d'algorithmes est pertinente dans les cas où le système dispose d'un catalogue de documents assez figé, ce qui constitue une hypothèse forte. Contrairement aux approches purement numériques, elle permet également de suggérer de nouvelles ressources, en dépit du fait qu'elles n'ont jamais été consultées. En revanche, elle ne fournit que des documents thématiquement proches de l'historique de consultation. Il n'est pas aisé de détecter des opportunités d'introduction de nouveautés parmi les documents déjà en partage (c'est-à-dire les items appréciés par une communauté mais encore ignorés d'un individu de cette communauté). Des approches hybrides exploitant le filtrage par contenu et des bouclages de pertinence (i.e. des retours d'expérience de la part de l'utilisateur) permettent de détecter des nouveautés potentiellement pertinentes (Kassab et al., 2005).

Le filtrage collaboratif et le filtrage basé sur le contenu sont les deux grandes approches existantes au sein des systèmes de recommandation adaptatifs. Elles présentent toutes deux des avantages et des inconvénients. Elles s'avèrent même bien souvent complémentaires, ce qui explique l'émergence de nombreux systèmes hybrides combinant ces deux familles. En général, les profils sont orientés contenu, et la comparaison entre ces profils donne lieu à la formation de communautés permettant le filtrage collaboratif (Burke, 2002). A l'ère de l'intelligence artificielle rhizomatique (Burke, 2006)[23], de nombreuses méthodes reposant sur les mêmes concepts ont vu

[23]La communauté scientifique s'accorde à penser qu'il y a eu à l'heure actuelle trois générations de systèmes d'Intelligence Artificielle. La première vague repose sur des modèles symboliques (règles, théorèmes, opérations

le jour et fournissent de bons résultats, qu'elles soient numériques ou symboliques. Dans la suite de ce manuscrit, nous nous focaliserons principalement sur les modèles numériques et le filtrage collaboratif, non pas parce qu'ils sont plus performants mais parce qu'ils constituent le cœur de nos travaux de recherche. Par conséquent, nous ne nous référerons qu'aux trois premiers axes de personnalisation de Cranor pour définir formellement une méthode générique de modélisation des préférences respectueuse de la vie privée des utilisateurs.

De nombreuses recherches ont été menées pour rendre les algorithmes de filtrage collaboratif aussi peu intrusif que possible. Dans la sous-section suivante, nous proposons une vue d'ensemble des principaux travaux en matière de confidentialité. Plusieurs approches sont considérées, les plus populaires étant le cryptage des données, les agrégats publics et l'altération des profils.

1.4.1 Filtrage Collaboratif et préservation de la vie privée

Canny (2002a,b) se concentre sur les façons d'assurer une protection efficace de la vie privée en calculant un agrégat public pour chaque communauté, sans divulguer de données individuelles. Il construit des modèles probabilistes de comportement utilisateur à partir des observations et extrapole les votes à partir de ces observations pour le filtrage collaboratif. Il a prouvé que le filtrage collaboratif peut être considéré comme un problème d'analyse factorielle linéaire, généralisant la décomposition en valeurs singulières et la régression linéaire. Il propose alors d'utiliser l'algorithme EM (*Expectation Maximization*) pour effectuer l'analyse factorielle, puisqu'il permet de traiter les données manquantes sans requérir de valeurs par défaut. Par ailleurs, l'algorithme EM a une définition récursive simple appropriée pour sa méthode de protection de la vie privée. Son approche repose sur du cryptage de données homomorphique. Cette propriété est utilisée dans de nombreux algorithmes de chiffrement, tel que le protocole RSA. Dans son modèle, la protection de la vie privée est garantie par un protocole pair-à-pair. Chaque utilisateur commence avec ses propres préférences et la connaissance des pairs de sa communauté. Les utilisateurs échangent divers messages cryptés, contenant leurs préférences, lors de l'exécution du protocole. Ce dernier consiste à multiplier plusieurs messages chiffrés afin d'obtenir le cryptogramme de leur somme, ce qui correspond au chiffrement des préférences de la communauté. Le procédé de décryptage repose sur un partage de clé. La clé requise pour décrypter l'ensemble n'est détenue par personne, mais partagée entre plusieurs utilisateurs. A la fin du processus, chaque utilisateur dispose d'une copie décryptée des préférences de la communauté. Les pairs, c'est-à-dire les différentes ordinateurs composant le réseau[24], peuvent alors calculer leurs propres recommandations personnelles sans avoir à révéler les données individuelles. Cependant, Canny pose pour hypothèse qu'il y a un nombre raisonnable de clients connectés simultanément, ce qui peut s'avérer contraignant dans la mesure où le système a besoin de disposer de suffisamment d'utilisateurs partageant des documents entre eux pour obtenir la clé de décryptage complète.

logiques). La deuxième vague est qualifiée d' "intelligence artificielle comportementale" (réseaux de neurones, algorithmes génétiques, méthodes statistiques, etc.). La troisième génération dite "rhizomatique" exploite les différentes parties de l'Intelligence Artificielle (représentations, techniques de raisonnement, types de connaissance) afin d'interagir sur le long terme et de multiples façons avec un environnement déstructuré. Le choix du nom provient du rhizome, une tige souterraine généralement horizontale de certaines plantes vivaces produisant des tiges aériennes et des racines adventives.

[24] chaque utilisateur est associé à un pair.

Dans (Polat et Du, 2004), les auteurs adoptent également le point de vue selon lequel la confidentialité des données est une question de sécurité. Ils définissent des protocoles de communication spécifiques pour aborder ce problème au sein d'algorithmes de filtrage collaboratif basés sur une décomposition en valeurs singulières. Leur but est de garantir à la fois la vie privée des utilisateurs et la qualité des prédictions. Au demeurant, ils considèrent qu'il s'agit là d'objectifs antithétiques et proposent une technique pour tenter d'atteindre un compromis entre les deux. Leur modèle diffère de celui de Canny (2002b), dans la mesure où ce dernier se positionnait dans un contexte pair-à-pair au sein duquel les différents pairs participaient activement au procédé de filtrage collaboratif. Dans (Polat et Du, 2004), les utilisateurs envoient leurs données à un serveur et ne prennent pas part à la phase de calcul. Le module de communication exploite des techniques de perturbation aléatoire pour préserver la vie privée. La randomisation perturbe les données de façon à ce que le serveur ne connaisse que la variation des données. Ainsi, le serveur ignore les votes réels de chaque utilisateur, mais collecte des tendances (u_a apprécie un peu plus une ressource qu'une autre, etc.). Les nombres aléatoires sont générés en utilisant une distribution uniforme ou gaussienne, et remplacent partiellement les votes réels. Cette approche nuit toutefois à la cohérence des données et donc à la qualité globale des prédictions.

Berkovsky et al. (2006) proposent un modèle pair-à-pair avec comme objectif affiché la confidentialité des données. L'algorithme consiste à scinder l'ensemble des utilisateurs en sous-populations constituées aléatoirement. Chaque sous-population est sous la responsabilité d'un super-pair. Ce sont ces super-pairs qui se chargent de collecter les profils des utilisateurs de la sous-population correspondante, qui calculent les profils moyens des sous-populations servant au processus de recommandations individuelles. Le système traite également le problème de confidentialité en modifiant aléatoirement le contenu des profils utilisateurs. Cet algorithme permet une distribution partielle des calculs et des profils, mais souffre d'une diminution importante de la qualité des recommandations par rapport aux algorithmes de filtrage collaboratif traditionnels, en raison de l'utilisation de profils moyens et de l'introduction de bruit dans les matrices de votes. Le fonctionnement de cet algorithme sera présenté en détail dans la section 3.6.1.

Miller et al. (2004) proposent une méthode en accord avec le premier et le quatrième axe de personnalisation de Cranor. Ils exploitent les votes explicites et proposent une adaptation de l'algorithme item-item (Sarwar et al., 2001) pour les architectures pair-à-pair. Les corrélations y sont donc calculées entre les items, plutôt qu'entre les utilisateurs. Cela s'apparente à des suggestions basées sur le contenu, puisque le système de recommandations proposent des items en rapport avec les ressources précédemment appréciées par l'utilisateur courant. Il ne se réfère donc pas à un unique voisin. Au demeurant, cette approche n'est viable que dans le cas où les items sont persistants. En effet, si certains items sont supprimés de la plate-forme, il n'est plus possible de suggérer les ressources en lien avec ces items. Par ailleurs, le système réagit mal à l'introduction de nouveaux items car cela lui impose de recalculer l'intégralité du modèle de correspondances entre ressources.

Une autre façon d'aborder le problème de la confidentialité est celle de Han et al. (2004). Ils proposent une méthode de protection uniquement basée sur la distribution des données au sein d'un algorithme de filtrage collaboratif distribué, appelé `PipeCF`. La gestion de la base de données utilisateurs et le calcul des prédictions sont tous deux répartis sur plusieurs ordinateurs. Cet algorithme a été implanté dans des réseaux pair-à-pair par le biais d'une table de hachage

distribuée.

1.4.2 Proposition d'un procédé générique de modélisation

Dans la section précédente, nous avons présenté des méthodes efficaces de préservation de la vie privée au sein d'algorithmes de filtrage collaboratif. Cependant, ces techniques ont été développées de façon *ad hoc*, ne s'adaptent pas à tout type de situation et peuvent souffrir de certaines limitations. Les approches basées sur l'altération des données (telle que l'offuscation aléatoire des profils) produisent des recommandations moins précises que les algorithmes de filtrage collaboratif traditionnels. Les techniques basées sur le cryptage des données supposent souvent de partager la clé de décryptage, ce qui nécessite un grand nombre de connexions simultanées. Enfin, les agrégats publics garantissent la vie privée des utilisateurs sans réduire la qualité des prédictions, à condition que ces agrégats illustrent les préférences de communautés d'intérêts virtuelles plutôt que des sous-populations constituées aléatoirement.

Dans cette section, nous nous proposons d'introduire un processus de modélisation des préférences générique et efficace présentant les mêmes avantages que les algorithmes précédemment mentionnés. Ce processus doit pouvoir s'adapter à diverses architectures (client/serveur ou pair-à-pair) et doit, de ce fait, être dissocié de l'algorithme de filtrage à proprement parlé. Par ailleurs, si l'on synthétise ce que nous avons vu précédemment, voilà la liste des contraintes à prendre en compte :

- Les critères de qualité des prédictions et de respect de la vie privée sont difficilement conciliables. Il s'agit de trouver un bon compromis entre ces deux objectifs ;
- Une collecte purement explicite des données est insuffisante pour deux raisons. Premièrement, les utilisateurs ont du mal à se décrire correctement, même par le biais d'interfaces simples et intuitives (par exemple, une personne abonnée à la presse déclarant aimer les articles de politique étrangère, mais qui se contente souvent de consulter les dernières sorties au cinéma). Deuxièmement, les usagers consacrent peu de temps à la constitution de leur profil. Ils se contentent bien souvent de fournir le minimum d'informations requises lors de l'inscription au service de personnalisation. Même un utilisateur assidu disposera d'un profil très incomplet car la collecte des préférences est extrêmement chronophage en raison du nombre de ressources disponibles ;
- Les préférences individuelles évoluant au cours du temps, nous avons choisi de nous positionner dans un contexte de modélisation stochastique. Certaines informations collectées peuvent s'avérer caduques après un délai plus ou moins long ;
- Les ressources au sein d'un système de recherche et d'accès à l'information sont nombreuses et volatiles. On suppose que l'ensemble des items n'est pas figé (certains documents sont supprimés, d'autres sont créés). Ce dernier point milite en faveur d'algorithmes de filtrage collaboratif basés sur la corrélation entre utilisateurs, plutôt qu'entre items, mais n'entre pas directement en ligne de compte dans le choix du mode de collecte des préférences.

Dans l'optique d'éviter un procédé de modélisation laborieux et explicite, un moyen de collecter automatiquement les préférences *a posteriori* consiste à analyser les usages. Cette analyse est transparente pour l'utilisateur et permet de déduire implicitement certaines préférences. A titre d'exemple, il est possible de collecter les actions de l'utilisateur courant au sein d'une plate-

forme documentaire et d'en extraire des informations sur la fréquence, la récence ou la durée de consultation pour chaque item. Ces informations peuvent ensuite être exploitées pour exprimer les préférences de l'utilisateur sous une forme numérique. En d'autres termes, il ne s'agit plus d'exploiter plusieurs critères lors de la formation des communautés (Nguyen et al., 2006), mais de synthétiser ces critères en un seul attribut exploitable par un algorithme de filtrage. Une modélisation purement implicite présente toutefois dans certains cas le risque de s'éloigner des véritables attentes de l'utilisateur courant. Par exemple, si ce dernier consulte le site du Louvre et reçoit ensuite un appel téléphonique pendant une heure tout en laissant son navigateur web ouvert, le système pensera que ce musée présente pour lui un fort attrait. Dans les faits, l'utilisateur va peut-être consulter le site cinq minutes et s'apercevoir que ce qu'il cherche se trouve plutôt dans le musée du quai Branly. Il est donc préférable de confronter régulièrement l'utilisateur aux données collectées implicitement à travers un retour d'expérience. Cela n'est malheureusement pas toujours possible.

Au final, il ressort qu'une bonne façon de construire les modèles de préférences individuels consiste à combiner les différentes méthodes de collecte de données possibles pour maximiser la pertinence des profils. La grande diversité des sources d'information implique de savoir traiter des données hétérogènes voire hétéroclites, et potentiellement contradictoires entre elles. En effet, certaines des données collectées peuvent être bruitées (en raison d'une interface homme/machine inappropriée ou des capacités du matériel utilisé par exemple) ou souffrir d'un niveau d'incertitude (entre autres choses, la durée de consultation d'un item ne garantit pas l'attention de l'utilisateur pendant ce laps de temps comme nous l'avons vu précédemment). De plus, la redondance des données caractérisant une même préférence peut parfois sur-expliquer les évidences. La problématique scientifique consiste donc à définir le degré de pertinence de chaque information et à mesurer la qualité des estimations notamment basées sur les usages afin de valider le modèle. Il s'agit également de synthétiser les données en réduisant le nombre de dimensions dans l'espace de représentation des utilisateurs.

En résumé, nous proposons de combiner les sous-ensembles disponibles et utilisables respectivement des données implicites et des données explicites en les synthétisant sous une forme numérique exploitable par un algorithme de filtrage collaboratif. Nous avons vu précédemment qu'il était possible d'analyser les fichiers log afin de récupérer des données utiles. Ces fichiers sont facilement et légalement récupérables. Dans la mesure où les usages collectés représentent des données très sensibles, il est préférable de conserver ces informations du côté client lorsque cela s'avère possible. Dans une architecture Client/Serveur traditionnelle, il est au demeurant plus fréquent de lister ces actions par ordre chronologique pour tous les utilisateurs confondus du côté serveur. En tout état de cause, ces traces doivent rester transitoires et être exploitées dans le respect de la législation en vigueur en matière de respect de la vie privée. Traditionnellement, l'analyse de ces traces repose sur des techniques de *Web Usage Mining*[25]. Ces dernières s'intéressent à la recherche de motifs ou de connaissances sur les comportements des utilisateurs

[25]Le *Web Mining* est l'application de techniques de fouille de données pour la recherche de motifs sur le Web. Le *Web Mining* comporte trois approches que sont le *Web Usage Mining*, le *Web Content Mining* et le *Web Structure Mining*. Le *Web Content Mining* (Liu, 2005) se charge de découvrir les informations utiles dans le contenu d'une page Web. Le *Web Structure Mining* (Bekkerman et al., 2007) exploite la théorie des graphes pour analyser la structure du Web (nœuds et connexions).

d'un site Web afin d'extraire des relations entre les données stockées (Masseglia et al., 2004). Une approche populaire consiste à extraire des motifs séquentiels (Agrawal et Srikant, 1995) permettant d'aboutir à des relations du type :

« *Sur ce site web marchand, 10% des clients ont visité dans l'ordre la page d'accueil, la page des DVD, la page des séries TV et enfin la page des nouvelles sorties*».

Pour obtenir ce résultat, la démarche se décompose en trois phases principales. Dans un premier temps, un pré-traitement est nécessaire pour éliminer les informations inutiles dans le fichier de données brutes. Par la suite, des algorithmes de *data mining* à base de classification et d'extraction de motifs sont utilisés pour identifier les séquences fréquentes. Enfin, les résultats sont exploités par un outil de requête et de visualisation.

Dans (Castagnos et Boyer, 2006a, 2008), nous proposons d'adapter des techniques de *Web Usage Mining* (WUM) afin de prendre en compte les critères implicites lors de la modélisation des préférences individuelles. Selon les méthodes de classification utilisées pour le WUM, on constate que la granularité des données traitées peut varier. Ainsi, dans (Mobasher et al., 2002), les auteurs raisonnent en terme de sessions. Celles-ci sont représentées par des vecteurs du type $< w(p_1, s), w(p_2, s), \ldots, w(p_n, s) >$ où $w(p_i, s)$ est le poids de la page p_i dans la session s. Le poids peut alors prendre une valeur binaire dépendant de la présence ou de l'absence de cette page dans la session, ou bien fonction du temps de chargement et d'affichage de la page par exemple. Dans (Fu et al., 2000), les pages sont organisées dans une hiérarchie basée sur la syntaxe des URL. Par exemple, la page **kiwi.loria.fr/publier/publicationsKiwi.php** est positionnée au niveau du nœud **kiwi.loria.fr** → **publier**. Une étape de généralisation permet ensuite de réduire les dimensions des données en remplaçant les pages par leur nœud parent. **kiwi.loria.fr/publier/publicationsKiwi.php** devient ainsi **kiwi.loria.fr/publier/**. Afin d'étendre ce principe de classification au-delà des seules pages Web, nous retiendrons de ces deux références bibliographiques (Mobasher et al., 2002; Fu et al., 2000) qu'il faut déterminer la granularité possible et/ou souhaitée pour chaque item, ainsi que le poids à accorder à chaque information. Notre approche s'articule en deux temps :

1. les données brutes, qu'elles soient présentes dans les logs ou fournies explicitement, permettent de construire des modèles utilisateurs sous la forme d'informations de plus haut niveau, également appelées « critères » (temps de consultation d'un item, fréquence, liste des favoris, etc.) ;
2. les modèles utilisateurs doivent ensuite être transposés sous une forme moins intrusive. Si l'on se réfère au premier axe de personnalisation définie par Cranor, cette collection de données implicite doit être transformée en votes estimés explicites$\in [v_{min}, v_{max}]$ constituant les profils utilisateurs. Cette opération revient à estimer les votes que les utilisateurs attribueraient aux items sur la base de plusieurs critères.

Nous proposons la fonction de transformation de l'étape 2, mesurant l'intérêt présumé de l'utilisateur courant u_a pour chaque item i_k, comme suit :

$$f_{transf}(u_a, i_k) = v_{min} + \frac{\sum_{c \in criteres}(p(c) * c(u_a, i_k))}{\sum_c p(c)} * \frac{(v_{max} - v_{min})}{c_{max}} \quad (1.1)$$

où les $c(u_a, i_k)$ correspondent aux valeurs normalisées attribuées à i_k suivant chacun des critères. c_{max} est la valeur maximale que peut prendre $c(u_a, i_k)$, tous critères confondus. Chaque

$p(c)$ représente le poids du critère associé, c'est-à-dire l'importance que revêt ce paramètre dans la formule. Ce poids peut s'avérer proportionnel à la confiance qu'on peut accorder à cette estimation. Par exemple, le temps de consultation est-il un indicateur précis et révélateur de l'intérêt porté par u_a sur i_k ? Le résultat de la fonction $f_{transf}(u_a, i_k)$ doit être arrondi à l'entier le plus proche et exprime le vote estimé pour l'item i_k.

La formule 1.1 est une généralisation des travaux menés par Chan (1999) pour les bases de données et les pages Web. Le choix des critères et leurs poids respectifs s'adaptent au contexte d'utilisation et aux informations qu'il est possible de récupérer. L'ensemble des critères applicables varie donc en fonction du type d'item consulté, de l'architecture du système et des paramètres locaux définis par l'utilisateur courant. Voici une liste non exhaustive des critères pouvant être pris en compte :

– **la récence de consultation d'un item** i_j ;

$$Recence(u_a, i_k) \;=\; \frac{\text{date(derniere visite) - date(debut log)}}{\text{date(present) - date(debut log)}} = \frac{l_k^{u_a}.time \;-\; l_1^{u_a}.time}{date(present) \;-\; l_1^{u_a}.time} \tag{1.2}$$

où $l_k^{u_a}$ correspond à la dernière ligne associée à l'item i_j dans les logs (i.e. contenant un objet URL_{i_j}). Selon la notation de (Masseglia et al., 2004), $l_1^{u_a}$ correspond à la première ligne du fichier log de l'utilisateur courant, c'est-à-dire la date de première exécution du module de profilage (début log).

– **la durée de consultation moyenne d'un item** i_j ;

$$Duree(u_a, i_k) \;=\; moyenne_{consultations}\left(\frac{\text{temps de lecture de l'item}}{\text{taille de l'item}}\right) \tag{1.3}$$

Il est nécessaire de tenir compte de la taille de l'item, sous peine de considérer qu'un gros ouvrage est systématiquement plus intéressant qu'un petit fascicule par exemple. Notons que l'on pourrait tenir compte du temps minimum ou du temps maximum de consultation, en lieu et place de la moyenne. Le temps de consultation d'un item est calculé à partir du fichier log en calculant l'écart temporel entre la première ligne l_k faisant référence à un objet URL_{i_j} (requête de la part de u_a) et la première ligne l_q de l'item suivant avec un objet $URL_{i_{j+1}}$ (à titre d'exemple, s'il s'agit d'un site Web, chaque accès à une page ou une image du site générera une ligne associée au même item. S'il s'agit d'un document vidéo, une seule ligne y fera référence). Dans le cas d'une navigation multi-onglets (consultations croisées), on pourra agréger plusieurs consultations d'une même ressource à condition que celles-ci ne soient pas trop espacées dans le temps (cf. consultations 1 et 3, figure 1.6).

FIG. 1.6 – Exemple de consultations dans un fichier log.

L'introduction d'une limite de temps t_{max} peut également être pertinente pour borner le temps de consultation. Ainsi, on minimisera l'erreur d'estimation dans le cas où l'utilisateur a passé un long moment à consulter un item sans être attentif (ex : navigateur ouvert sur un site Web pendant qu'il est au téléphone). La valeur t_{max} doit être paramétrée en fonction du délai maximum plausible de consultation du type de ressource considéré (la lecture d'un livre requiert plus de temps que de parcourir un article de presse).

– **la fréquence de consultation d'un item** ;

$$frequence(u_a, i_k) = \frac{1}{periode} = \frac{1}{e(u_a, i_k)} \tag{1.4}$$

avec $e(u_a, i_k)$ la moyenne des écarts entre chaque début de consultation de i_k par u_a.

– **l'appartenance de l'item considéré à la liste des favoris de** u_a. Ce critère peut par exemple tenir compte du fait que l'item a été ajouté aux marque-pages du navigateur (s'il s'agit d'un site Web). Il peut aussi intégrer le fait que l'utilisateur a voté explicitement et positivement pour cette ressource (vote élevé s'il s'agit d'un produit vendu en ligne, case cochée s'il s'agit d'un goût exprimé vis-à-vis de telle ou telle catégorie, etc.). Ce critère, bien que reposant sur des informations fournies explicitement, ne doit idéalement représenter qu'une partie des critères pris en compte dans la fonction de transformation. Nous avons effectivement précisé précédemment que l'utilisateur n'était pas toujours capable de décrire correctement ses préférences *a priori*. Au demeurant, les informations fournies explicitement constituent un signal d'intérêt fort et doivent de ce fait être pondérées par un poids p(c) plus élevé. Dans certains cas, on préférera remplacer purement et simplement le vote estimé par le vote réellement fourni par u_a ;
– **le pourcentage de liens visités**, s'il s'agit d'une page Web ou d'un site Web ;

On peut imaginer d'autres critères prenant en compte le fait que l'item a été imprimé, enregistré sur le disque dur, envoyé par messagerie électronique, etc.

Il n'est pas toujours possible d'exploiter les critères implicites sans identification de la part de l'utilisateur. En effet, la modélisation basée uniquement sur l'adresse IP au sein des fichiers log côté serveur se heurte à de nombreux problèmes (Masseglia et al., 2004) : la présence du cache sur la machine de l'utilisateur, les *proxies* (jouant le rôle de caches de niveau régional, entreprise, etc.), l'existence de moteurs de recherche extérieurs permettant à l'utilisateur d'accéder directement à la partie du site les concernant, etc. Une modélisation effectuée à partir de logs côté client ne rencontre pas ces problèmes-là.

L'estimation des votes nécessite de stocker la liste des actions (logs) et les critères. Dans le cas idéal, ces informations sont enregistrées côté client pour garantir leur confidentialité. Afin de se conformer au deuxième axe de personnalisation de Cranor, nous recommandons de supprimer périodiquement les actions et informations les plus anciennes dans la collection de données implicites. Le caractère non permanent de ces données présente également l'avantage de tenir compte de l'évolutivité des préférences des utilisateurs au cours du temps. Si l'on conserve trop longtemps les traces, on risque d'être confronté à une situation contradictoire du type « j'aime l'item i_k » et « je n'aime pas l'item i_k ». La durée de conservation idéale des données dépend des habitudes de consultation de chaque utilisateur (utilisation fréquente ou sporadique). Bien évidemment, pour éviter un oubli trop brutal des informations, il est possible de procéder par évaporation (en divisant chacun des critères par la récence de consultation de l'item, ou en divisant par la durée depuis l'actualisation du vote de ce même item).

Enfin, conformément au troisième axe de personnalisation de Cranor, nous préconisons de laisser les échanges éventuels d'informations relatives aux préférences individuelles à l'initiative de l'utilisateur. Dans le cas d'un système de recommandations à base de filtrage collaboratif, seuls les profils numériques des utilisateurs sont communiqués aux voisins avec l'accord explicite de leurs propriétaires. De plus, afin d'accroître la confiance des utilisateurs dans le système, il est important de les autoriser à visualiser à tout instant le contenu de leurs profils respectifs et de leur laisser éventuellement l'opportunité de les modifier par le biais d'un retour d'expérience. Ils peuvent ainsi influer sur le calcul du profil en modifiant ou en détruisant certaines informations. Selon le contexte applicatif, il est également possible d'introduire la notion de profil public, correspondant à la partie du profil personnel que l'utilisateur courant accepte de partager avec autrui. Cette solution constitue alors un bon compromis entre qualité des prédictions et respect de la vie privée lorsque le profil personnel est stocké localement sur la machine de l'utilisateur. En effet, la totalité des préférences est exploitée pour inférer des recommandations, mais seule une petite partie de ces préférences est partagée avec le reste du monde. Pour plus de sécurité, les profils stockés localement peuvent être cryptés.

Plutôt que de détériorer les profils échangés comme le suggèrent Berkovsky et al. (2006), ce qui a pour conséquence de réduire la précision du système, nous recommandons d'envoyer les profils anonymement. Cette approche présente l'avantage de rester conforme à la plus stricte des lois en matière de confidentialité des données sur le plan international. Cependant, cela nécessite d'associer un identifiant unique à chaque utilisateur et par voie de fait à chaque profil, afin d'éviter les duplications de données. Nous proposons ici une procédure garantissant l'anonymat des utilisateurs, en dépit du fait que chacun d'eux dispose d'un identifiant unique. Dans notre scénario, les utilisateurs doivent ouvrir une session à partir d'un nom et d'un mot de passe, afin d'utiliser le système de recommandations. De cette manière, plusieurs personnes peuvent se servir du même ordinateur (par exemple, les différents membres d'une même famille) sans interférer dans leurs profils respectifs. Ainsi, ils ne peuvent ni accéder aux autres profils cryptés stockés sur le disque dur, ni altérer les recommandations d'autrui de par leurs consultations sur l'ordinateur communément utilisé. En résumé, chaque utilisateur dispose sur un ordinateur donné de son propre profil et d'un identifiant unique. Ce dernier est généré à l'aide d'une fonction de hachage prenant en paramètre l'adresse IP et le nom de connexion.

On prend pour hypothèse qu'un même utilisateur exploitant les capacités du système depuis

plusieurs ordinateurs n'attend pas les mêmes suggestions selon le contexte d'utilisation. Ainsi, il s'attendra à avoir des recommandations différentes selon s'il est au travail ou à la maison. Il aura donc un profil professionnel sur l'ordinateur de sa société et un profil personnel chez lui. Cette hypothèse est compatible avec le fait de stocker localement les profils et de générer partiellement l'identifiant à partir de l'adresse IP. Au demeurant, notre procédé de modélisation s'adapte également au cas où l'on souhaite avoir un seul profil par utilisateur tout en gérant la contrainte de mobilité. Il s'agira alors de générer l'identifiant uniquement à partir du nom et de prévoir le stockage du profil crypté sur un serveur ou une procédure d'importation/exportation sur le poste client.

L'utilisation d'une fonction de hachage H pour générer un identifiant unique est appropriée, puisqu'elle présente les caractéristiques suivantes :

- elle est non réversible : sachant "y", il est difficile de trouver "x" tel que H(x)=y. Dans notre cas, "x" correspond aux informations confidentielles passées en paramètre (nom, adresse IP), et "y" à l'identifiant connu du reste du monde ;
- il n'y a pratiquement pas de collision : il est difficile de trouver un "x_1" et un "x_2" tels que $H(x_1) = H(x_2)$, garantissant par là même l'unicité de chaque identifiant ;
- connaissant "x" et la fonction H, il est facile de calculer H(x) et donc de générer les identifiants ;
- H(x) a une taille fixe, permettant de stocker facilement dans une liste les identifiants ayant tous le même format.

Le procédé de modélisation des préférences que nous proposons est donc respectueux de la vie privée des gens sans avoir à altérer les données, réduisant ainsi la qualité des prédictions. Par ailleurs, il se contente d'exploiter les données disponibles lors de l'élaboration du profil, déchargeant autant que faire se peut les utilisateurs de la tâche fastidieuse consistant à se décrire complètement et régulièrement. Nous verrons dans la suite de ce manuscrit qu'il est possible de déployer cette approche aussi bien dans des architectures client/serveur qu'au sein de grilles de calcul (architectures pair-à-pair). Le résultat de ce procédé de modélisation est un profil numérique exploitable par un algorithme de filtrage collaboratif, grâce à des échanges anonymes et standardisés. Le chapitre suivant vise à introduire un formalisme du filtrage collaboratif et présente les différentes méthodologies d'évaluation qu'il est possible de mettre en place.

Chapitre 2

Formalisme et évaluation du filtrage collaboratif

2.1 Formalisme du filtrage collaboratif

Boyer (2004) propose dans son programme de recherche d'aborder le filtrage collaboratif comme une technique d'apprentissage par renforcement collaboratif et décentralisé : le signal de renforcement provient des autres usagers et non pas de l'utilisateur courant, à la différence d'une approche classique. Toutefois, il est également possible de voir le filtrage collaboratif comme une méthode d'apprentissage automatique non supervisé consistant à rapprocher les bons interlocuteurs afin de les faire bénéficier d'une expérience commune accrue. Nous parlons alors de stratégies d'interactions sociales : les utilisateurs sont vus comme des agents humains autonomes et sociaux immergés dans un système multi-agents où les interactions sont transparentes pour les usagers.

Dans le cas d'un système de recherche et d'accès à l'information, il s'agit dans un premier temps d'identifier les utilisateurs ayant des intérêts en commun avec l'utilisateur courant. On peut ensuite exploiter leurs connaissances pour faire de la personnalisation à la volée, c'est-à-dire des recommandations individualisées en temps réel répondant au critère d'utilité défini précédemment.

Dans l'apprentissage non supervisé, il y a en entrée un ensembles de données collectées, appelé « échantillon d'apprentissage »(Russell et Norvig, 1995). Ensuite, le programme traite ces données comme des variables aléatoires et construit un modèle maximisant la vraisemblance des données. L'apprentissage non supervisé se distingue de l'apprentissage supervisé par le fait qu'il n'y a pas de valeurs spécifiques de sortie fournies *a priori* : les données ne sont pas étiquetées de sorte que l'on ne dispose pas d'exemples avec des valeurs de sortie correctes associées.

Dans notre contexte, les données utilisateurs sont éparses dans la mesure où elles sont potentiellement disséminées sur plusieurs machines (sur des serveurs ou sur plusieurs postes client pour un même utilisateur). $U_t = \{u_1, u_2, \ldots, u_{n'}\}$ est l'ensemble des n' utilisateurs du système à l'instant t. On suppose en effet que les utilisateurs peuvent s'inscrire ou se désinscrire de la plate-forme. "n" est donc variable et $U_t \subset U$. Les variables d'entrée correspondent aux votes

associés aux ressources. Les profils utilisateurs constituent donc une variable aléatoire composée de sous-variables. Ces sous-variables sont discrètes lorsqu'il s'agit de votes explicites (valeurs entières réparties sur une échelle de votes) et continues lorsqu'il s'agit de votes estimés à partir de la fonction de transformation f_{transf} (valeurs réelles). Nous devons donc projeter l'ensemble de ces variables d'entrée (votes connus) dans un espace continu. Notons $U_a = \{V_1, \ldots, V_m\}$ cette variable représentant les préférences de l'utilisateur courant. Nous considérons ici u_a comme un utilisateur lambda $\in U_t$ à qui l'on souhaite fournir des recommandations. V_i exprime la préférence pour chacun des m items disponibles dans l'ensemble I ou la préférence pour un groupement d'items. Une observation de U_a sera le profil $\vec{u_a}$, constitué d'un ensemble de votes spécifiques à l'utilisateur u_a. Notons que nous nous plaçons dans un contexte de modélisation stochastique partiellement observable : le système n'est pas capable de déterminer l'ensemble des préférences de u_a à un instant donné (c'est-à-dire son intérêt ou son désintérêt pour toutes les ressources dont il a déjà connaissance). Ses goûts peuvent évoluer au cours du temps. Aussi, nous noterons $\vec{u_{a,t}}$ son profil à l'instant t.

Les profils utilisateurs sont des vecteurs de positionnement des usagers dans l'espace de représentation utilisateurs/ressources $R_{u/r}$. Ce dernier est un espace à m dimensions où chaque axe correspond à un item de I. Les votes de u_a correspondent donc à ses coordonnées dans l'espace. Les recommandations faites à l'utilisateur courant ne doivent concerner que les items disponibles à l'instant t ($\in I_t$). Les prédictions associées sont alors les valeurs déterminées par le système pour les items inconnus de u_a (votes nuls dans le profil). Ces prédictions prennent des valeurs dans un intervalle borné de $\mathbb{R}$. Elles sont obtenues à partir d'une fonction de prédiction $f_{predict} : \mathbb{R}^m \times I_t \times R_{u/r} \rightarrow \mathbb{R}$. Elle prend en paramètre le profil de l'utilisateur courant, l'item pour lequel on souhaite prédire le vote et l'espace de représentation pour bénéficier de la connaissance des utilisateurs similaires. Pour des raisons de simplification, nous considérerons dans la suite de ce manuscrit $R_{u/r}$ comme implicitement inclus dans les paramètres et nous ne désignerons une prédiction que par l'utilisateur et l'item associés. L'effet de l'action de recommandation sur u_a est incertaine dans le sens où un item suggéré, bien que potentiellement intéressant, n'est pas forcément consulté par l'utilisateur courant.

L'apprentissage consiste à identifier les profils de $R_{u/r}$ qu'il est pertinent de prendre en compte pour effectuer une recommandation. Cela revient à construire implicitement ou explicitement des communautés d'intérêts virtuelles. Elles sont implicites lorsque le système se contente de décider au cas par cas s'il inclut ou non un profil dans le calcul de la prédiction. Elles sont explicites si le système bâtit un modèle qu'il exploite ensuite dans la phase de prédiction (approche probabiliste ou partitionnement des données). Nous avons donc une variable de sortie correspondant aux communautés $\{c_1, \ldots, c_j\}$ que l'on souhaite discerner dans $R_{u/r}$ tout en maximisant la vraisemblance des données (variables d'entrée). Nguyen et al. (2006) définissent une communauté comme un groupe d'utilisateurs qui prennent la même valeur pour un critère donné. Dans le cas du filtrage collaboratif, ce critère est la similarité existante entre deux profils. Il faut donc introduire une fonction de similarité $f_{simil} : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$.

Nous aurions un apprentissage supervisé si le système était capable de connaître *a priori* le nombre de communautés et s'il disposait d'exemples de membres appartenant à telle ou telle communauté. Cela n'est pas envisageable car il est très difficile de bénéficier d'un retour d'expérience explicite de la part des utilisateurs et qu'une validation des modèles *a posteriori*

n'est pas suffisante. En effet, les utilisateurs ont pu évoluer ou subir l'influence du système entre le moment où ils ont renseigné leurs préférences et le moment où ils commentent les prédictions du système. Toutefois, l'objectif premier d'un système de recommandations est de maximiser la satisfaction de chaque utilisateur. Une telle évaluation *a posteriori*, lorsqu'elle est possible, est donc souhaitable. Nous verrons dans la section suivante qu'il existe d'autres méthodes d'évaluation du modèle plus faciles à mettre en œuvre et basées, entre autres, sur des mesures d'erreur.

2.2 Méthodologies d'évaluation

Les algorithmes de filtrage collaboratif peuvent être évalués en terme de qualité des prédictions, de temps de calcul nécessaire, d'occupation mémoire, de complexité algorithmique, de trafic réseau généré par les échanges entre les différents agents et de taux de couverture. Ces évaluations peuvent se faire à partir de corpus de tests ou bien en situation réelle. Les corpus permettent de tester les algorithmes hors ligne en simulant des utilisations réelles et en jaugeant les résultats et le comportement du système. Les essais en situation réelle permettent de mesurer ponctuellement la satisfaction des utilisateurs vis-à-vis du service.

2.2.1 Corpus de tests utilisés

GroupLens

Afin d'évaluer la qualité des prédictions d'un algorithme de filtrage collaboratif, il est possible d'utiliser la base de données `MovieLens` fournie par l'équipe de recherche américaine GroupLens[26]. `MovieLens`[27] est un site Web de recommandation de films. Les utilisateurs ont la possibilité de partager leurs préférences en votant explicitement pour des items sur une échelle de valeurs entières ($\in \mathbb{N}$) comprise entre $v_{min} = 1$ et $v_{max} = 5$. Le service exploite ces votes pour générer des recommandations personnalisées. Le jeu de données extrait de cette plate-forme a été largement utilisé par la communauté scientifique pour évaluer et comparer les algorithmes de filtrage collaboratif. Il présente en effet l'avantage de reposer sur des votes réels et fournit de ce fait un bon support de validation. Il est composé d'un ensemble de 100.000 votes. Chaque utilisateur a voté pour au moins 20 items. La matrice de votes M comprend 943 utilisateurs (U) et 1682 items (I). Ainsi, il y a 93,7% de données manquantes[28]. La distribution des votes est présentée dans le tableau 2.1. Une grande proportion des votes sont des 3 et 4 (votes médians ou positifs). Il y a globalement peu de votes négatifs (1 ou 2).

Le jeu de données a été divisé à 5 reprises en un ensemble d'apprentissage (également appelé "base") et un ensemble test. L'idée est d'exploiter l'ensemble d'apprentissage pour générer des prédictions à partir d'un algorithme de filtrage collaboratif et de comparer les résultats avec les

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

[27] http ://www.movielens.org/

[28] Ce pourcentage peut paraître exorbitant et peu réaliste en situation réelle. Pourtant, il reflète bien le peu de données dont un système de recommandations dispose pour effectuer ses prédictions. Face à la profusion de ressources, les utilisateurs ne votent explicitement que pour une poignée d'items ce qui pénalise grandement la qualité des recommandations. En général, les usagers votent pour les premiers items de la liste I.

Echelle de votes Jeux de données	1	2	3	4	5
U.data	6,11%	11,37%	**27,14%**	**34,17%**	21,20%
U1.base	5,90%	11,47%	**27,45%**	**34,25%**	20,93%
U2.base	6,06%	11,48%	**27,26%**	**34,12%**	21,07%
U3.base	6,15%	11,43%	**26,95%**	**34,05%**	21,40%
U4.base	6,24%	11,21%	**26,97%**	**34,26%**	21,31%
U5.base	6,19%	11,24%	**27,08%**	**34,19%**	21,29%

TAB. 2.1 – Distribution des votes dans la base de données MovieLens.

valeurs réelles contenues dans l'ensemble test. L'opération est répétée 5 fois afin de renforcer la validité de l'évaluation d'un point de vue statistique. En effet, certains votes présentent une plus grande représentativité que d'autres. A titre d'exemple, un vote élevé de la part d'un utilisateur qui a tendance à mettre des notes basses constitue un indicateur fort de ses préférences. Un vote proche de sa moyenne revêt au contraire une moindre importance. Dans la mesure où les votes sont séparés aléatoirement dans les ensembles *base* et *test*, la qualité des prédictions peut varier en fonction de la proportion de votes représentatifs contenus dans l'ensemble d'apprentissage. Les ensembles *base* et *test* contiennent respectivement 80% et 20% des votes globaux.

U.data correspond au jeu de données complet. *U[1-5].base* sont les 5 ensembles d'apprentissage générés.

Par le biais des algorithmes de filtrage collaboratif, il est possible de mesurer à partir de ces votes le degré de similarité entre 2 utilisateurs ou entre 2 items. Il existe de nombreuses métriques de similarité (dont certaines seront détaillées dans le chapitre 3) comme le cœfficient de corrélation de Pearson. En utilisant ce cœfficient de corrélation pour mesurer les similarités existantes entre les ressources, nous obtenons la distribution du tableau 2.2 dans la matrice item-item S. Les valeurs sont comprises entre -1 et 1. Une valeur proche de 1 en valeur absolue met en lumière une forte similarité, tandis qu'une valeur proche de 0 souligne la différence d'usage entre 2 ressources. Le tableau 2.2 montre qu'une grande proportion des items sont peu corrélés entre eux. Une distribution semblable est obtenu lorsqu'on calcule la similarité entre les utilisateurs. Cela signifie que les utilisateurs et les ressources sont extrêmement disséminés dans l'espace de représentation $R_{u/r}$. Ce constat est en grande partie lié à la quantité de données manquantes dans la matrice des votes.

La proportion de données manquantes est encore plus flagrantes lorsqu'on représente la matrice item-item sous forme graphique. Dans la figure 2.1, chaque pixel de coordonnée (x,y) exprime la similarité entre les items x et y. L'image présente donc une taille de 1682 $\times$ 1682 pixels. Plus les ressources sont corrélées, et plus le pixel affiche une couleur claire. Inversement, une couleur sombre indique l'absence de corrélation.

Lorsqu'on supprime les ressources dont la similarité est nulle avec tous les autres items, on obtient une image de 1059 $\times$ 1059 pixels (cf. figure 2.2).

Le jeu de données MovieLens constitue un bon cadre d'évaluation en matière de qualité des

Intervalle de similarité	Nombre de valeurs	Intervalle de similarité	Nombre de valeurs
[-1,0 ;-0,8)	104.724	(0,8 ;1,0]	187.852
[-0,8 ;-0,6)	21.246	(0,6 ;0,8]	53.383
[-0,6 ;-0,4)	29.424	(0,4 ;0,6]	73.505
[-0,4 ;-0,2)	42.507	(0,2 ;0,4]	95.740
[-0,2 ;-0,0)	67.357	[0,0 ;0,2]	737.983
Nombre total de similarités dans la matrice S : 1.413.721			

TAB. 2.2 – Distribution des similarités dans la matrice item-item.

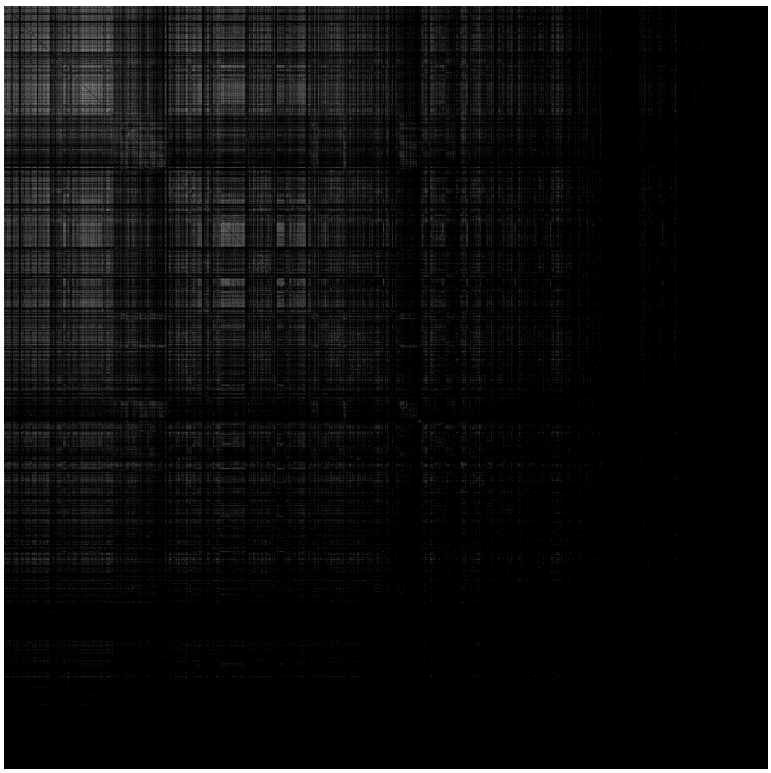


FIG. 2.1 – Similarité entre les items dans la base MovieLens.

prédictions. En effet, le fait qu'il s'agisse de votes réels portant sur des films assure la cohérence de la base. En revanche, elle comporte trop de données manquantes et trop peu d'utilisateurs et/ou de ressources pour tester le passage à l'échelle. Pour cette raison, nous avons choisi de mettre au point un outil de génération de corpus. D'autres corpus de plus grande taille que `MovieLens` ont depuis été mis à la disposition de la communauté scientifique, comme le corpus de blagues `Jester` (http ://eigentaste.berkeley.edu/) ou de films `Netflix` (http ://www.netflix.com/). L'intérêt de notre outil de génération par rapport à ces nouveaux corpus réside principalement dans les possibilités de paramétrage offertes.

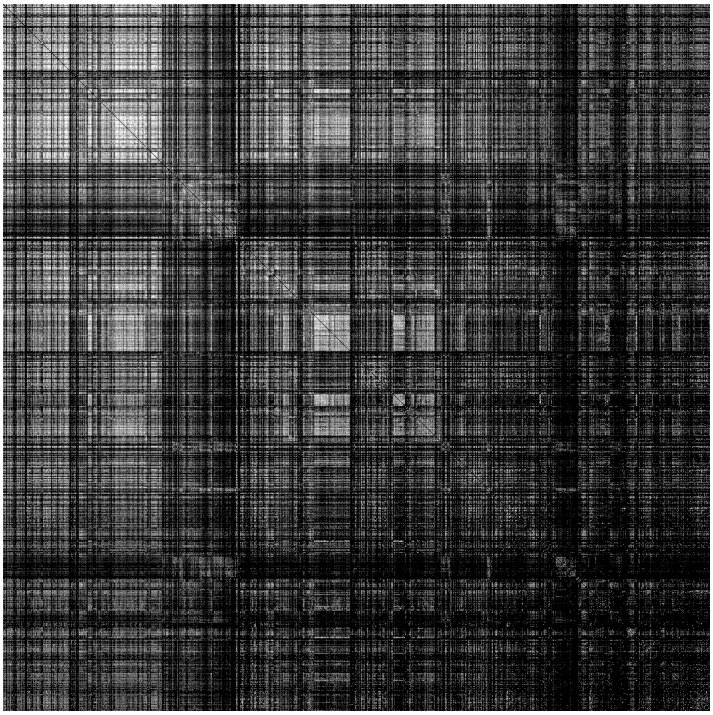


FIG. 2.2 – Similarités non nulles entre les items dans la base MovieLens.

Génération de corpus

Lors de la conception de cet outil de génération de corpus, nous nous sommes fixés pour objectif de simuler les votes d'une population de la façon la plus réaliste possible. Il s'est agi de développer un logiciel générant des bases de tests pseudo-aléatoirement en intégrant un grand nombre de paramètres, afin d'élargir le champ des possibles lors de l'évaluation des différents algorithmes (avec notamment une meilleure répartition des votes).

Notre outil permet ainsi de :

- générer des matrices de votes M correspondant à un instant donné (au format texte, XML ou dans une base de données MySQL) ;
- simuler l'évolution des votes au cours du temps (ensemble de matrices représentant chacune un pas de temps, ou bien séquences de votes nécessitant une plus faible occupation mémoire[29]). La prise en compte du facteur temporel permet de gérer l'évaporation des votes (oubli au-delà d'un certain délai) et la mise à jour des votes.

Dans les deux cas, il est possible de préciser le nombre d'utilisateurs, de ressources et l'échelle des votes (cf. figure 2.3).

Pour plus de réalisme, nous avons choisi de générer dans la mesure du possible[30] les votes suivant une distribution gaussienne pour chaque item (cf. figure 2.4). En effet, la loi normale est

[29] Il s'agit d'un mode de représentation plus optimal, étant donné que la matrice M est creuse. Cela permet de gérer un plus grand nombre d'utilisateurs et de ressources. Toutefois, ce mode de représentation peut pénaliser les algorithmes de filtrage collaboratif en terme de temps de calcul.

[30] C'est-à-dire si la méthode de génération et les paramètres ne sont pas contraires à cette directive.

FIG. 2.3 – Paramétrage de la taille de la matrice.

la loi qui se rencontre le plus fréquemment en statistique (Lethielleux, 2005) et c'est également celle que l'on retrouve dans le corpus de test `MovieLens`.

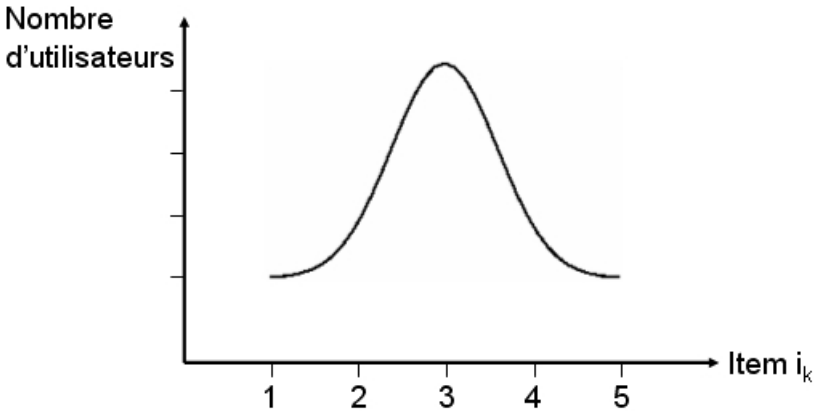


FIG. 2.4 – Distribution gaussienne des votes.

Par ailleurs, nous avons défini 5 types d'utilisateurs ayant des comportements différents :
- ceux qui mettent uniquement des notes basses ;
- ceux qui mettent uniquement des notes élevées ;
- ceux qui mettent des notes extrêmes (basses ou élevées) ;
- ceux qui mettent des votes moyens ;
- ceux qui votent uniformément sur l'ensemble des ressources (chaque note a autant de chances d'être votée que les autres).

Notre outil permet de définir la proportion d'utilisateurs dans chacune de ces catégories (cf. figure 2.5). Par défaut, la taille de ces 5 sous-ensembles est équivalente (20% pour chaque type d'utilisateurs).

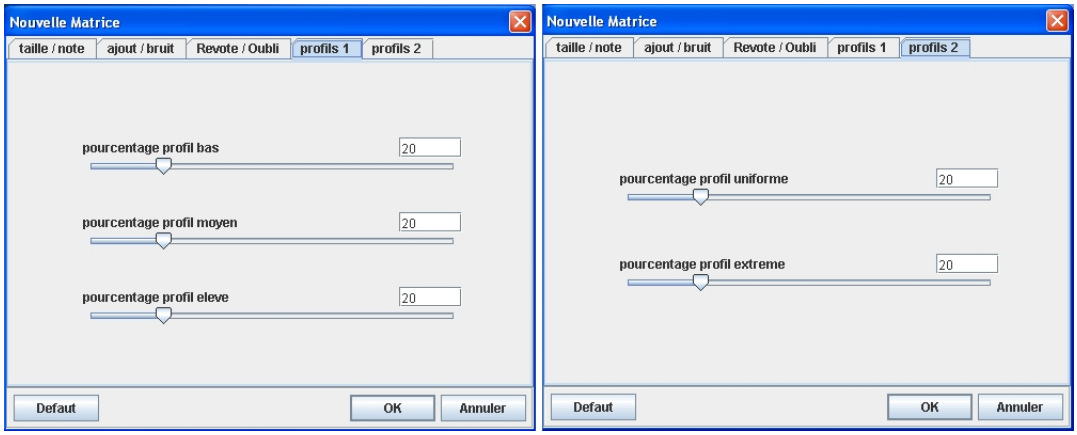


FIG. 2.5 – Répartition par types d'utilisateurs.

Il existe également une grande variété de paramètres (cf. figure 2.6), parmi lesquels :
- le nombre de pas de temps à simuler ;
- le pourcentage de vide dans la matrice ;
- le mode de représentation (ex : 0 en l'absence de vote, -1 en cas d'oubli, etc.) ;
- le pourcentage d'évaporation (nombres de votes oubliés), d'ajout de nouveaux votes et de mises à jour de votes existants à chaque étape ;
- le pourcentage de bruit (pour éviter une génération de corpus trop déterministe, nous avons en effet introduit un paramètre permettant à un utilisateur d'être plus ou moins proche du comportement que nous lui avons attribué).

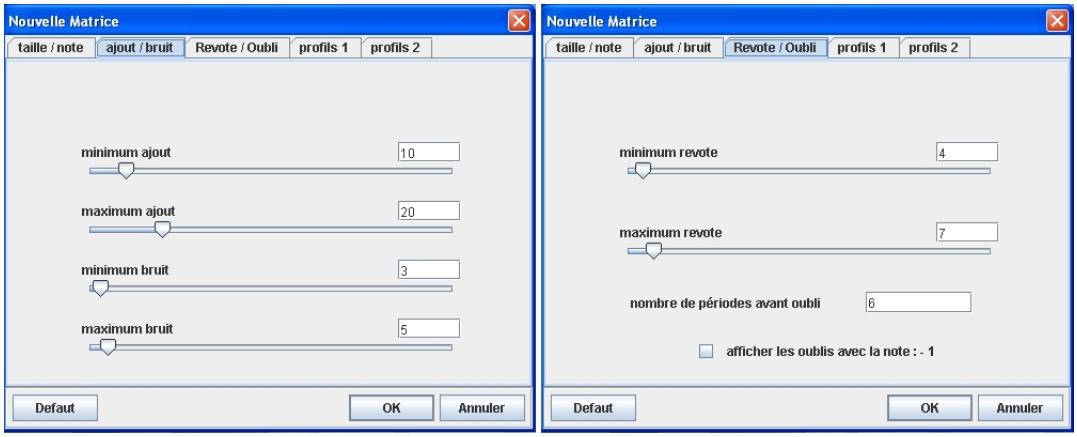


FIG. 2.6 – Paramétrages additionnels.

Enfin, nous avons imaginé plusieurs façons de générer les corpus en fonction des paramètres renseignés :

– une approche probabiliste, où chaque utilisateur a un certain pourcentage de chances de voter pour un item non noté et de revoter pour un déjà voté ;
– une méthode basée sur la taille totale de la matrice, où chaque utilisateur vote pour un nombre d'items proportionnel au nombre de ressources générales ;
– un remplissage en fonction de la taille restante, où chaque utilisateur vote à chaque pas de temps pour un nombre d'items proportionnel au nombre de ressources non encore votées par celui-ci ;
– une approche par pourcentage, où il est possible de définir la proportion de chaque note sur l'échelle de votes dans la matrice à générer.

Le logiciel a été conçu de façon modulaire, de sorte qu'il est possible de rajouter des paramètres et des méthodes de génération en fonction des besoins. Il est également possible d'apporter des modifications ponctuelles à la main dans les corpus via cet outil (ajout ou suppression d'utilisateurs, modifications de votes, blocage de certaines contraintes, etc.), comme en atteste la figure 2.7.

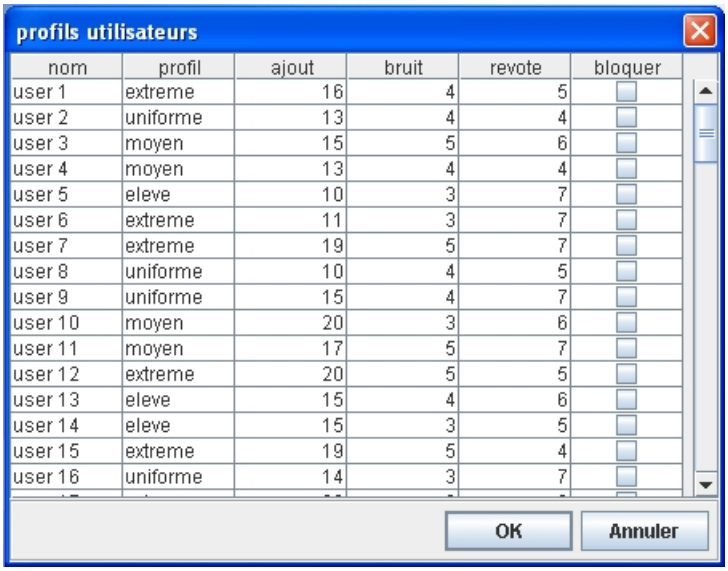


FIG. 2.7 – Paramétrage manuel.

Bien évidemment, la génération de corpus ne remplace pas les tests à partir de votes réels, mais permettent de simuler une application dans un contexte de grande envergure (grande quantité de données) ou d'observer le comportement d'un algorithme dans un cas spécifique.

2.2.2 Critères d'évaluation en simulation

Les corpus présentés dans la section précédente permettent d'effectuer un certain nombre de tests qualitatifs et quantitatifs sur les algorithmes de filtrage collaboratif.

Critères qualitatifs

Parmi les critères qualitatifs, on retrouve :
– la similarité moyenne (Miller et al., 2004) ;

Il existe plusieurs façons de calculer des prédictions à partir d'une même formule $f_{predict}$ dans un algorithme de filtrage collaboratif. Une première approche prend en compte en un seul calcul les préférences de tous les utilisateurs pondérées par la similarité avec l'utilisateur courant (Castagnos et al., 2008b,a). Lors de la phase d'évaluation (en terme de qualité des recommandations par exemple), une deuxième façon de procéder consiste à introduire progressivement et aléatoirement les profils des utilisateurs (Castagnos et Boyer, 2007a). Cela permet d'étudier le comportement du système lors de la montée en charge, depuis la phase d'initialisation jusqu'à un stade où il est pleinement opérationnel. Une dernière façon limite les calculs prédictifs aux préférences des plus proches voisins de l'utilisateur courant. Lors des tests, il est notamment possible de faire varier la taille de ce voisinage en fixant un seuil de similarité (tous les utilisateurs u_j tels que la similarité $s(u_a, u_j) > seuil$) comme dans (Castagnos et Boyer, 2007c) ou en définissant une taille (les K plus proches voisins[31]) comme dans (Miller et al., 2004). Lorsque l'on choisit cette dernière solution, on ne garantit pas la qualité de ce voisinage, et donc des prédictions. En effet, la valeur ajoutée des préférences de ces plus proches voisins est très dépendante de la topologie de l'espace de représentation $R_{u/r}$. Si les utilisateurs sont très éloignés les uns des autres, les prédictions seront de moins bonne qualité (Miller et al., 2004). Dans ce contexte, il est possible d'estimer la qualité du voisinage. La similarité moyenne est une mesure déterminant à quel point un groupe de voisins est proche du profil de l'utilisateur courant. Elle est calculé simplement en sommant les similarités entre u_a et ses voisins (lorsque les similarités peuvent prendre des valeurs négatives, il s'agira de sommer les valeurs absolues), puis en divisant par le nombre de voisins K :

$$Sim_{moyenne} = \frac{\sum_{j=1}^{K} |s(u_a, u_j)|}{K} \tag{2.1}$$

– la précision (Chee et al., 2001) ;

La précision correspond au pourcentage ou au nombre d'items suggérés et s'avérant véritablement pertinent pour l'utilisateur courant. Par exemple, si l'on considère une liste des meilleures N recommandations[32], la précision correspondra à la proportion d'items véritablement consultés et appréciés par l'utilisateur courant. La précision globale du système est calculée comme la moyenne des précisions individuelles. Lors de tests à partir de corpus, on considère qu'un item de l'ensemble *test* est apprécié

[31] On parle alors d'algorithme KNN, pour *K Nearest Neighbors*.
[32] "Meilleures" selon l'estimation du système. On parle de liste `top-N`.

dès lors que la note réelle est supérieure à un seuil (note supérieure ou égale à 4 sur une échelle de 1 à 5 par exemple).

– le rappel (Miller et al., 2004) ;

La mesure de rappel compte le nombre d'items dans la liste de recommandations `top-N` ayant déjà été votés par l'utilisateur courant (comme pour la précision, on peut aussi déterminer la proportion d'items relativement à N). La mesure de rappel globale du système correspond à la moyenne des mesures individuelles. Concrètement, lors de l'évaluation de l'algorithme, on énumère le nombre d'items dont le vote associé est non nul dans l'ensemble *test* et se retrouvant parmi les items suggérés. Cela suppose d'accepter les items dont les préférences sont d'ores et déjà connus (i.e. dont les votes sont contenus dans l'ensemble d'apprentissage) dans la liste `top-N`. Dans le cas contraire, la mesure de rappel serait de 100%. Intuitivement, cette mesure considère qu'il y a une forte probabilité que le système ait à suggérer un item dès lors que celui-ci a été vu par l'utilisateur courant.

Il existe une combinaison des mesures de précision et de rappel appelée `nombre F` (Salton et McGill, 1983; Chee et al., 2001). `F` est donnée comme étant égale à :

$$F = \frac{2 \times precision \times rappel}{(precision + rappel)} \tag{2.2}$$

Si la précision et le rappel sont exprimés en nombre d'items et que $F = N$, alors la qualité du système est considérée comme parfaite.

– la mesure d'erreur MAE (Resnick et al., 1994; Herlocker et al., 2004) ;

`MAE` est l'acronyme pour *Mean Absolute Error*. Comme son nom l'indique, il s'agit d'une métrique mesurant l'erreur entre les prédictions et les valeurs réels fournies par les utilisateurs. En d'autres termes, elle permet de déterminer la qualité de ces prédictions. Rappelons que les prédictions sont générées à partir d'un ensemble d'apprentissage. Dans le cas de la base de données `MovieLens`, on utilise les ensembles $U[1-5].base$ afin d'essayer de retrouver les votes contenus dans $U[1-5].test$. L'évaluation de la qualité des recommandations consiste alors à comparer les prédictions avec les votes contenus dans les fichiers $U[1-5].test$. Pour chaque paire prédiction-vote $< f_{predict}(u_j, i_k), v(u_j, i_k) >$, on calcule l'erreur absolue (i.e. la différence en valeur absolue $|f_{predict}(u_j, i_k) - v(u_j, i_k)|$). Afin d'obtenir la mesure de `MAE` d'un algorithme, on somme les erreurs absolues et on divise par le nombre N de paires prediction-vote prises en compte au numérateur. Au final, on a donc :

$$MAE = \frac{\sum_{j=1}^{n} \sum_{i_k \ t.q. \ v(u_j,i_k) \in M_{test}} |f_{predict}(u_j, i_k) - v(u_j, i_k)|}{N} \tag{2.3}$$

Avec : $u_j \in U$, $i_k \in I$, M_{test} la sous-matrice des votes correspondant au jeu de test (20% des votes) et N le nombre de paires $< f_{predict}(u_j, i_k), v(u_j, i_k) >$ telles que $f_{predict}(u_j, i_k) \neq 0$, $v(u_j, i_k) \neq 0$ et $v(u_j, i_k) \in M_{test}$.

Plus la valeur de `MAE` est basse, plus les prédictions sont précises et les recommandations de bonne qualité.

– la mesure d'erreur NMAE (Goldberg et al., 2000) ;

`NMAE` signifie *Normalized Mean Absolute Error*. Cette métrique se calcule de la façon suivante :

$$NMAE = \frac{MAE}{v_{max} - v_{min}} \tag{2.4}$$

Cette métrique permet de normaliser les MAE et comparer différents algorithmes dans le cas où les échelles de votes ne sont pas les mêmes (par exemple comparaison d'un algorithme avec des votes $\in [1;5]$ avec un autre algorithme dont les notes s'échelonnent sur l'intervalle $[1;7]$).

– la mesure d'erreur HMAE (Baltrunas et Ricci, 2007; Candillier et al., 2007).

Dans la littérature, la mesure `HMAE` (également appelée *High MAE*) est définie comme étant la `MAE` obtenue uniquement sur les votes élevés. On considérera par exemple que sur une échelle de votes de 1 à 5, les votes élevés sont les valeurs 4 et 5. Cette métrique évalue la qualité des prédictions véritablement importantes, c'est-à-dire celles qui figureront potentiellement dans le `top-N`. En effet, il est relativement peu important que la fonction $f_{predict}$ estime mal les items de votes faibles, puisque ceux-ci ne seront jamais suggérés par le système et s'avèrent inintéressants pour l'utilisateur courant. En revanche, l'erreur sur les votes élevés doit être la plus faible possible afin que le système soit capable de suggérer les bons items dans le bon ordre de préférence.

Nous avons introduit une définition un peu plus précise des `HMAE` dans (Castagnos et al., 2008b,a). Nous considérons deux métriques, respectivement appelées `HMAE1` (ou `TestHMAE`) et `HMAE2` (ou `RecHMAE`).

La mesure `HMAE1` porte sur les items dont les votes sont élevés dans l'ensemble *test* (votes réels fournis par les utilisateurs et que le système a cherché à réapprendre). Cette mesure se rapproche de la mesure de précision du paragraphe précédent, dans la mesure où elle porte sur les items appréciés par les utilisateurs. La mesure `HMAE1` est toutefois plus pertinente car elle permet d'estimer la qualité des prédictions sur tous les votes élevés et ne se restreint pas à une liste `top-N`. Comme nous l'avons vu dans le cas du corpus `MovieLens` (cf. supra, 2.2.1 GroupLens, p.37), il n'est pas rare que les utilisateurs fournissent des notes élevés sur les items. De ce fait, un grand nombre de ressources se retrouvent avec des votes équivalents. Lorsqu'on se restreint à une liste `top-N`, le nombre N est bien souvent très inférieur au nombre d'items ayant la note maximale v_{max}. Ce genre de situation impose de sélectionner les éléments de la liste `top-N` au hasard parmi les meilleurs items. Par ailleurs, la mesure `HMAE1` fournit une estimation de la qualité des prédictions en terme d'erreur,

et non pas en nombre d'items comme c'est le cas avec la précision, ce qui constitue une évaluation plus fine.

La mesure `HMAE2` est calculée pour les items dont les prédictions (valeurs obtenues à partir de l'ensemble d'apprentissage avec $f_{predict}$) sont élevées. Comme précédemment, une prédiction est considérée comme élevée lorsqu'elle est comprise entre 4 et 5 sur l'échelle de valeurs $[1; 5]$. Il s'agit d'une métrique plus fine que la mesure de rappel, puisqu'elle s'intéresse aux items supposés pertinents par le système et votés par les utilisateurs, mais exprimée en terme d'erreur plutôt qu'en nombre d'items sans se limiter à la liste `top-N`. Pour résumer, une valeur faible de `HMAE1` signifie que le système estime correctement l'intérêt que les utilisateurs avaient explicitement porté à leurs items favoris. En complément, une valeur faible de `HMAE2` indique que le système n'a pas sur-évalué des items dont l'intérêt est faible pour les utilisateurs.

Critères quantitatifs

Il est également possible d'évaluer quantitativement les algorithmes, par le biais :
– du taux de couverture (Miller et al., 2004) ;

La couverture mesure le pourcentage d'items pour lequel le système est capable de fournir des prédictions. En effet, il arrive que le système soit dans l'incapacité d'estimer un vote. Cette situation est par exemple rencontrée lorsqu'aucun utilisateur n'a fourni de vote pour un item donné dans l'ensemble d'apprentissage. Ce pourcentage peut porter sur l'ensemble des votes inconnus (valeurs nulles) dans l'ensemble d'apprentissage, ou bien se restreindre au pourcentage de votes qu'il est possible de retrouver parmi ceux contenus dans l'ensemble de test. Cette métrique est par exemple intéressante à mettre en place lors de l'évaluation d'algorithmes qui améliorent la qualité des prédictions en éliminant les données de l'ensemble d'apprentissage jugées non représentatives. Un système, même s'il fournit des prédictions avec une erreur nulle, sera peu utile s'il n'est capable d'estimer qu'un très faible pourcentage des votes manquants.

– de la complexité algorithmique en temps et en espace (Lavallée, 2008) ;

La théorie de la complexité algorithmique s'intéresse à l'estimation de l'efficacité des algorithmes. Elle permet de comparer la vitesse et les conditions d'exécution des algorithmes indépendamment des capacités de la machine et de la qualité du code produit par le compilateur. Pour cela, elle se fonde sur une estimation théorique des temps de calcul et des besoins en mémoire informatique. La théorie de la complexité établit des hiérarchies de difficultés entre les problèmes algorithmiques, dont les niveaux sont appelés des "classes de complexité". Elle utilise comme unité de comparaison des "opérations élémentaires". Parmi celles-ci, on retrouve par exemple

l'accès à une cellule mémoire, la comparaison de valeurs, les opérations arithmétiques ou les opérations sur des pointeurs. Le choix des opérations élémentaires peut varier selon la taille des données en entrée. Au demeurant, si les valeurs ont un codage de taille fixe (nombres de taille raisonnable), on considérera par exemple que les opérations arithmétiques telles que l'addition prennent un temps constant. Dans la plupart des cas, on peut supposer que les problèmes que nous considérons n'ont qu'une donnée. Cette donnée a une taille n qui est un nombre entier naturel. Il est alors possible d'analyser l'algorithme au sens du pire cas ou au sens de la moyenne. $t(n)$ est le temps d'exécution du pire cas et le maximum sur toutes les données de taille n. Ce temps $t(n)$ est habituellement compté comme le nombre d'étapes nécessaire à la résolution du problème. Par exemple, la comparaison d'un élément avec les n composants d'une liste prendra un temps $t(n) = n$. La comparaison de tous les éléments deux à deux d'une matrice de taille (n, n) prendra un temps $t(n) = n^2$. Dans la mesure où le plus mauvais peut en pratique n'apparaître que très rarement, il est également possible d'étudier l'espérance sur l'ensemble des temps d'exécution $tm(n)$. Toutefois, l'analyse mathématique de la complexité moyenne est souvent délicate car il est difficile d'évaluer la probabilité pour une entrée de se présenter. Par ailleurs, $s(n)$ représentera l'ordre de grandeur nécessaire à l'exécution du programme en espace (i.e. occupation mémoire). A titre d'exemple, pour calculer la taille mémoire d'une matrice en octets, il faut multiplier le nombre de cellules par la taille de la représentation interne du type d'élément contenu dans la matrice.

La notation de Landau permet de décrire le comportement asymptotique des fonctions. Par exemple, imaginons un algorithme dont le temps nécessaire afin de résoudre un problème de taille n est donné par : $t(n) = 4n^2 - 2n + 2$. En ignorant les constantes (trop dépendantes du matériel sur lequel le programme s'exécute) et les termes qui croissent le plus lentement, on peut dire que "$t(n)$ croit comme n^2". On écrit alors : $t(n) = O(n^2)$. Parmi les classes de complexité, on retrouve les fonctions suivantes par ordre de croissance de la plus lente à la plus rapide[33] : complexité constante $O(1)$ (indépendante de la taille de la donnée), complexité logarithmique $O(log(n))$, complexité linéaire $O(n)$, complexité quasi-linéaire $O(n.log(n))$, complexité quadratique $O(n^2)$, complexité cubique $O(n^3)$, complexité polynomiale $O(n^p)$, complexité quasi-polynomiale $O(n^{log(n)})$, complexité exponentielle $O(2^n)$ et complexité factorielle $O(n!)$.

– du temps de calcul réel (Castagnos et al., 2005a) ;

Nous avons vu que la complexité algorithmique permet d'évaluer le temps de calcul en nombre d'étapes, indépendamment de la machine utilisée. Toutefois, il s'agit d'un temps théorique. Il est également possible d'évaluer le temps de calcul réel d'un algorithme et de l'exprimer en unité de temps standard (secondes, minutes, heures, etc.).

[33] C'est-à-dire du temps d'exécution le plus rapide au plus lent. En effet, rappelons que l'on cherche à approximer le temps de calcul théorique $t(n)$. Si cette fonction croit rapidement alors le temps d'exécution sera conséquent, et inversement.

Cette évaluation temporelle est cette fois-ci dépendante des capacités matérielles et du langage de programmation utilisé. Tous les tests effectués dans cette thèse en terme de temps de calcul ont été réalisés sur un ordinateur portable `Dell Latitude D600` avec un processeur cadencé à 1,8 GHz et une mémoire vive de 1 Go[34]. Les algorithmes de filtrage collaboratif ont été codés en `Java`. Lors de l'exécution des algorithmes de filtrage, les capacités matérielles sont bien évidemment partagées avec les autres programmes fonctionnant sur l'ordinateur tels que le système d'exploitation. Il est toutefois possible d'allouer une taille mémoire fixe à la machine virtuelle `Java` en rajoutant des options dans la ligne de commande (options $-Xms$ et $-Xmx$ dans l'instruction visant à exécuter le programme).

– du trafic réseau (Castagnos et Boyer, 2007c).

Les échanges de messages à travers un réseau d'ordinateurs ont également un coût. Il est difficile d'intégrer au temps de calcul d'un algorithme le temps nécessaire à la transmission de toutes les informations à tous les interlocuteurs concernés, dans la mesure où cette durée additionnelle est très dépendante de la charge d'occupation du réseau (trafic plus ou moins dense), de la qualité de la ligne (plus ou moins de bruit et de paquets perdus), de la distance à parcourir (nombre de points intermédiaires : passerelles, proxys, ...), du protocole utilisé, etc. Au demeurant, il est possible d'estimer le trafic réseau généré en fonction de la taille des paquets, de la fréquence et de la stratégie de diffusion des messages.

2.2.3 Critères d'évaluation en situation réelle

Au-delà des estimations faites sur la qualité des recommandations d'un système à partir d'un corpus de test (*offline*), il importe lorsque cela s'avère possible de mesurer la satisfaction des utilisateurs par rapport aux ressources qui leur sont proposées. Nous proposons dans cette section une nouvelle méthode d'évaluation inspirée du domaine statistique permettant de valider les modèles en situation réelle (Castagnos et al., 2005c). Cette méthode suppose de disposer d'un ensemble suffisant de bêta-testeurs utilisant le système de façon régulière et prolongée dans le temps. Il s'agit d'une évaluation assez lourde à mettre en place. Elle ne peut donc servir qu'à estimer ponctuellement la qualité d'un service et non pas aider à améliorer en temps réel le processus d'apprentissage inhérent aux algorithmes de filtrage collaboratif, contrairement aux critères d'évaluation vus précédemment.

Afin d'illustrer cette méthodologie, nous allons nous positionner dans le contexte d'une plate-forme documentaire. Les items disponibles sur une telle plate-forme sont des documents textuels, audios ou vidéos que chaque utilisateur a la possibilité de consulter s'il le désire. Outre le gain

[34] Afin de valider le bon fonctionnement de nos algorithmes de filtrage collaboratif distribués, nous avons déployé nos applications sur plusieurs ordinateurs fixes et/ou portables, voire sur des serveurs. Toutefois, les temps de calcul ont systématiquement été déterminés sur un même ordinateur portable afin d'éprouver les algorithmes dans une situation contraignante où les capacités matérielles sont faibles.

de temps occasionné, l'intérêt d'un système de recommandations intégré à ce genre de plate-forme est de conseiller aux lecteurs des documents intéressants. La pertinence des suggestions du système est donc corrélée au fait que les usagers choisissent, à la simple lecture des noms/titres ou des résumés, d'accéder aux ressources préconisées par le module de filtrage collaboratif. Il faut également tenir compte de l'évolution du nombre de documents consultés au cours du temps. En effet, si la lecture des ressources prônées par le système ne satisfait pas l'utilisateur courant en dépit de résumés attrayants, sa confiance dans les prédictions diminuera et il consultera moins d'items.

La figure 2.8 illustre les modalités d'organisation d'une interface graphique adaptée dans le but d'évaluer la pertinence du module de filtrage. La liste à droite de l'écran énumère les items suggérés par le service de personnalisation. La colonne de gauche contient, quant à elle, les articles généraux, c'est-à-dire une sélection aléatoire des items non sélectionnés par le système de recommandations au vu du profil utilisateur. Il n'y a donc pas de recoupement entre les ressources présentées dans chacune des deux colonnes. Les lecteurs disposent du titre, de la catégorie et d'un court résumé de chaque article. Les utilisateurs ont la possibilité d'accéder aux documents en cliquant dans la colonne de gauche ou de droite, ou en sélectionnant une catégorie en haut de l'écran (limitant par là même la consultation aux items de la catégorie choisie). Les documents consultés apparaissent au centre de l'écran dans leur intégralité.

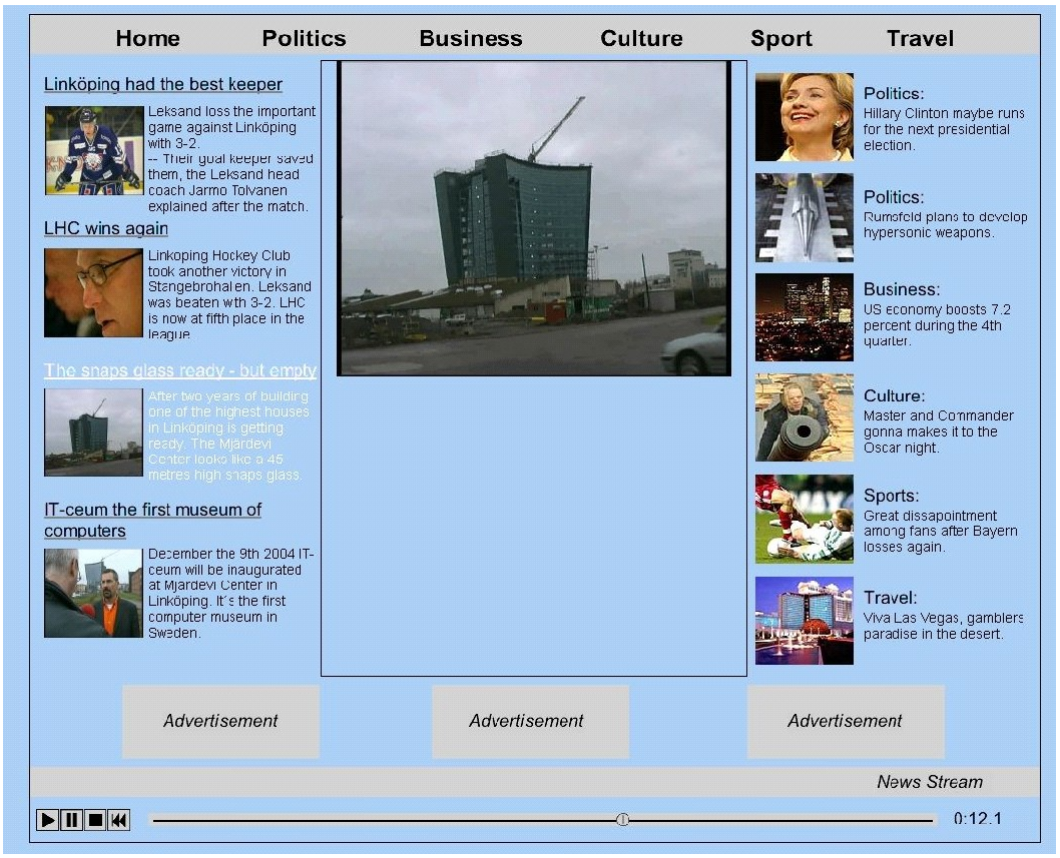


FIG. 2.8 – Interface graphique permettant l'évaluation en situation réelle du service.

Afin de mener à bien cette évaluation, un groupe de testeurs est requis. Plus ils sont nombreux et meilleure est l'analyse du système. Cet ensemble de personnes est divisé en trois sous-groupes de taille égale. Chacun de ces sous-groupes utilise régulièrement l'outil informatique sur une période de temps fixée et devant excéder plusieurs jours. En effet, un système de filtrage collaboratif évolue et apprend au cours du temps, en fonction des actions exercées par les utilisateurs. Il ne devient pertinent que lorsqu'il a recueilli suffisamment de traces. Le système n'est véritablement opérationnel qu'après un certain délai d'exploitation qui dépend du nombre d'utilisateurs et de ressources.

La distinction entre les trois sous-groupes de testeurs réside dans la manière de générer la liste d'articles à droite de l'écran (cf. figure 2.9). Cette subdivision est transparente pour les utilisateurs qui pensent tous disposer du même service de personnalisation.

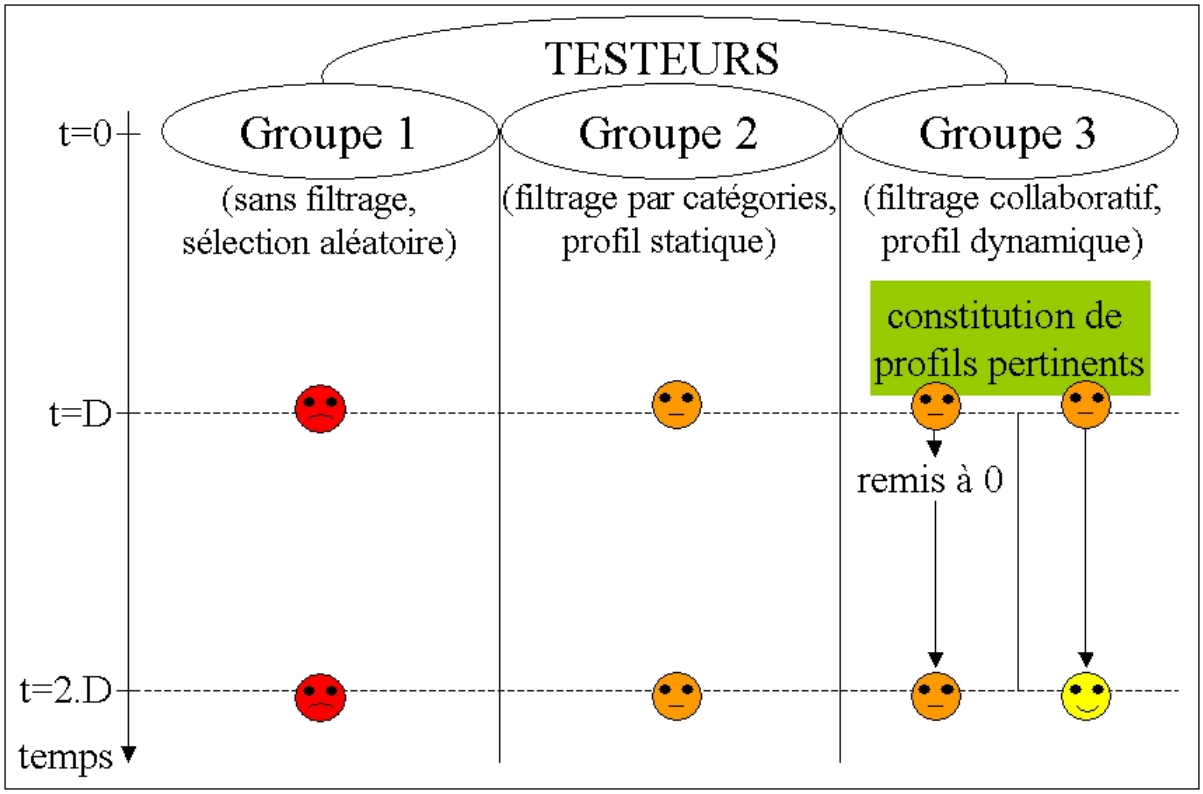


FIG. 2.9 – Mesure de satisfaction en fonction de l'évolution du nombre de liens visités dans la colonne de droite au cours du temps.

1. Pour le premier sous-groupe, qualifié de "témoin", les articles de la colonne de droite sont choisis au hasard parmi l'ensemble des documents contenus dans la base de données. Il n'y a donc pas de filtrage collaboratif ;
2. Pour le deuxième sous-groupe, le système dispose du profil statique de chaque lecteur. Ce profil contient la liste des catégories susceptibles de les intéresser et le système leur fournit les articles correspondants. Par exemple, un individu dont le centre d'intérêt principal est

le sport aura sa colonne de droite remplie d'articles ayant trait au sport. Ce procédé de filtrage repose sur le principe d'un moteur de recherche par catégories ;

3. Enfin, le dernier sous-groupe d'utilisateurs dispose d'une liste d'articles sélectionnés à partir de leurs préférences, grâce au module de filtrage collaboratif. Afin de mettre en évidence l'amélioration de ce procédé au cours du temps, il est également possible de subdiviser ce troisième groupe en deux. Au bout d'un certain délai D d'exploitation du système, la moitié des profils utilisateurs est remise à zéro, tandis que l'autre moitié est préservée. Puis les tests se poursuivent pendant une autre durée D. De cette façon, il est possible de comparer la satisfaction des utilisateurs ayant conservés leur profil avec les autres.

Pour mesurer l'efficacité des différents procédés de filtrage, il ne reste plus qu'à comparer le nombre de liens consultés dans la colonne de droite (relativement à la colonne de gauche) et son évolution au cours du temps et, ce, pour chacun des trois sous-ensembles de testeurs. Il est également possible de compléter l'enquête en soumettant un questionnaire à chaque utilisateur à l'issue des tests.

Cette méthodologie a été mise en place dans le cadre du projet IST ELIN[35] visant à réaliser un journal multimédia adapté à l'utilisateur (Casademont et al., 2005a,b). Les résultats ont montré un net accroissement du nombre de consultations dans la colonne de droite pour les utilisateurs disposant d'une personnalisation à base de filtrage collaboratif, dans une proportion plus importante que le groupe avec filtrage par catégories. Le groupe témoin avait, quant à lui, un nombre de consultations semblable entre la colonne de gauche et celle de droite. Les utilisateurs du groupe 1 ont été dans l'ensemble peu satisfaits par les recommandations du système, contrairement aux groupes 2 et 3. Par ailleurs, les utilisateurs du groupe 3 dont le profil a été réinitialisé à $t = D$ ont déploré une baisse de qualité dans le service et cela s'est traduit par un ré-équilibrage du nombre de consultations entre les deux colonnes.

Cette méthodologie d'évaluation peut donc s'avérer intéressante en complément des autres méthodes basées sur des corpus de test. Elle n'est toutefois pas aisée à mettre en place. En effet, dans un contexte industriel réel, il est rare de disposer d'un nombre de testeurs suffisants soit par manque de volontaires assidus (il ne s'agit pas de se connecter une fois sur le site, mais bien d'utiliser régulièrement le service) soit pour des raisons stratégiques liées à la politique de la société. Cette dernière peut ainsi privilégier une évaluation du système avant la phase d'exploitation, afin de préserver son image de marque. D'autres industriels comme `Google` ou `Amazon` valident leurs systèmes en situation réelle. `Google` a par exemple pour habitude de rediriger un petit pourcentage des utilisateurs du moteur de recherche du même nom vers des serveurs expérimentaux, tandis qu'`Amazon` évalue la qualité d'un nouveau service en fonction de l'évolution du chiffre d'affaires réalisé.

Ce chapitre a permis de formaliser le filtrage collaboratif et d'exposer les méthodes d'évaluation existantes. Le chapitre suivant présente un état de l'art chronologique des différents algorithmes existants.

[35] ELIN est l'acronyme de *Electronic NewsPaper Interactive*. Il s'agit d'un projet IST mené en collaboration avec des agences de presse espagnoles et suédoises.

Chapitre 3

Etat de l'art

3.1 Les origines du Filtrage Collaboratif

Goldberg et al. (1992) ont défini le filtrage collaboratif comme une technique employant les comportements connus d'une population pour prévoir les futurs agissements d'un individu à partir de l'observation de son attitude dans un contexte donné (critères explicites tels que les votes ou implicites comme, par exemple, le temps de consultation d'une page). Le principe général consiste, lorsqu'un nouvel individu se présente, à rechercher par comparaison les utilisateurs qui, par leurs préférences connues (votes réels ou estimés), semblent avoir des comportements similaires.

Les premières méthodes de filtrage collaboratif étaient purement statistiques : les votes des utilisateurs étaient contenus dans une base de données et permettaient d'identifier les documents les plus pertinents. Toutefois, le filtrage collaboratif doit aujourd'hui tenir compte de données incomplètes ou manquantes concernant les votes sur les items. C'est pourquoi les méthodes purement statistiques sont inadaptées. (Breese et al., 1998) ont identifié deux classes majeures d'algorithmes pour répondre à ce problème : les algorithmes basés sur la mémoire et ceux basés sur un modèle. Il existe également une classe d'algorithmes hybrides tirant partie de ces deux grandes familles. Cet état de l'art ne constitue pas une liste exhaustive des méthodes de filtrage collaboratif existantes, mais vise à présenter les grandes familles d'algorithmes avec leurs avantages et leurs inconvénients à travers des travaux pertinents et reconnus dans la communauté scientifique.

3.2 Algorithmes basés sur la mémoire

Les algorithmes basés sur la mémoire opèrent sur la totalité de la base de données des utilisateurs (Resnick et al., 1994). Ces systèmes nécessitent donc d'agréger l'ensemble des profils utilisateurs afin de connaître, à tout instant, l'ensemble des votes. Ces derniers sont collectés sous

forme d'une matrice où chaque ligne correspond à un utilisateur $u_j \in U_t$[36] (cf. infra, figure 3.1). Chaque colonne correspond à un item $i_k \in I$, et non pas simplement restreint à l'ensemble I_t des ressources disponibles à l'instant t. En effet, les usagers ont pu évaluer par le passé des items ne figurant plus dans I_t. Au final, la matrice représente les préférences connues de l'ensemble de la population à un instant t. Nous l'appellerons matrice des votes $M : U_t \times I \rightarrow \mathbb{R}$. Cette matrice contient des valeurs entières, puisqu'il s'agit de concaténer les votes explicites (valeurs entières sur une échelle de votes $[v_{min}; v_{max}] \subset \mathbb{N}$) et/ou les votes implicites obtenus à partir de la fonction de transformation f_{transf} (valeurs $\in \mathbb{R}$). Le vote d'un utilisateur u pour un item i sera noté v(u,i) ou $v_{u,i}$.

Au final, la matrice a donc la forme suivante : $M = \begin{bmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,m} \\ v_{2,1} & v_{2,2} & \dots & v_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n',1} & v_{n',2} & \dots & v_{n',m} \end{bmatrix}$

La prédiction pour l'utilisateur u_a sur la ressource i_k est donnée par la formule suivante :

$$f_{predict}(u_a, i_k) = \overline{v(u_a)} + \kappa \sum_{u \in U_t} f_{simil}(u_a, u) \left[v(u, i_k) - \overline{v(u)} \right] \tag{3.1}$$

Avec : $v(u, i_k)$ l'évaluation de l'item i_k par l'utilisateur u ;
$\overline{v(u)}$ la moyenne de l'ensemble des évaluations fournies par l'utilisateur u ;
$f_{simil}(u_a, u)$ le cœfficient de pondération liant l'utilisateur u_a et u ;
κ un cœfficient de normalisation ;
U_t l'ensemble des utilisateurs considérés.

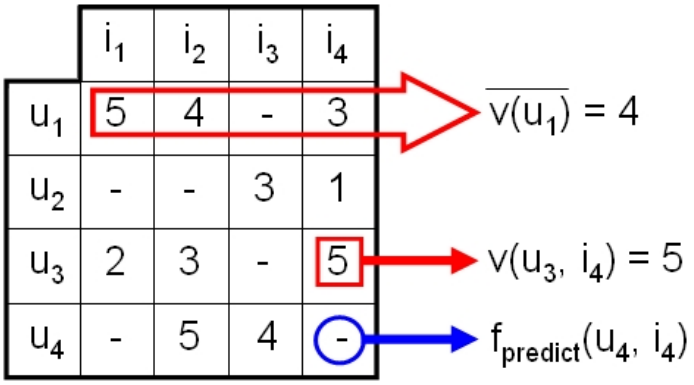


FIG. 3.1 – Matrice des votes des utilisateurs.

Dans la formule 3.1, le cœfficient κ permet d'harmoniser les votes, afin de minimiser l'influence des utilisateurs ayant tendance à noter de façon extrême (uniquement des notes très élevées ou très basses).

[36] $u_j \in U_t$ si l'on prend pour hypothèse qu'un utilisateur qui se désinscrit du service est supprimé de la base, afin de respecter les droits de l'usager.

Par ailleurs, le cœfficient de pondération $f_{simil}(u_i, u)$ (cf. supra, formule 3.1) représente la similarité existante entre l'utilisateur courant u_a et les autres utilisateurs u_j (comme précédemment introduit dans la section 2.1, Formalisme du filtrage collaboratif, p.35). Plus ils sont proches et plus le cœfficient est grand. Ainsi, le système tient d'autant plus compte des votes d'une personne que celui-ci est similaire à l'utilisateur courant. Le cœfficient de pondération peut prendre plusieurs formes, comme :

– le cœfficient de corrélation de `Pearson` (Resnick et al., 1994) ;

$$f_{simil}(u_a, u_j) = \frac{\sum_{i \in I_a \cap I_j} (v(u_a, i) - \overline{v(u_a)})(v(u_j, i) - \overline{v(u_j)})}{\sqrt{\sum_{i \in I_a \cap I_j} (v(u_a, i) - \overline{v(u_a)})^2 \sum_{i \in I_a \cap I_j} (v(u_j, i) - \overline{v(u_j)})^2}} \tag{3.2}$$

Les ensembles I_a et I_j correspondent aux items dont les préférences sont connues respectivement pour u_a et u_j. Le cœfficient de corrélation de Pearson évalue l'écart par rapport à la moyenne des votes de chaque utilisateur. Il est égal au rapport entre la covariance et le produit non nul des écarts types. On ne peut pas avoir une division par 0. Lorsque le produit des écarts types est nul, on considère que le cœfficient de corrélation l'est aussi. Le cœfficient de corrélation est compris entre -1 et 1. $f_{simil}(u_a, u_j)$ vaut 0 lorsque les deux utilisateurs sont décorrélés (on dit aussi que les variables sont linéairement indépendantes). $f_{simil}(u_a, u_j)$ vaut 1 lorsque les deux utilisateurs sont fortement corrélés et -1 lorsqu'ils sont fortement opposés. En d'autres termes, il est égal à 1 dans le cas où l'une des variables est fonction affine croissante de l'autre variable, à -1 dans le cas où la fonction affine est décroissante. Les valeurs intermédiaires renseignent sur le degré de dépendance linéaire entre les deux variables. Plus le cœfficient est proche des valeurs extrêmes -1 et 1, plus la corrélation entre les variables est forte. D'un point de vue géométrique, on peut considérer les paramètres comme des vecteurs entre l'origine du repère de l'espace $R_{u/r}$ et les positions des deux utilisateurs dans cet espace (respectivement $\overrightarrow{u_a}$ et $\overrightarrow{u_j}$). On appelle alors "vecteurs centrés" les vecteurs $\big(v(u_a, 1) - \overline{v(u_a)}, \ldots, v(u_a, m) - \overline{v(u_a)}\big)$ et $\big(v(u_j, 1) - \overline{v(u_j)}, \ldots, v(u_j, m) - \overline{v(u_j)}\big)$. Dans ce cas, le cœfficient de corrélation n'est autre que le cosinus entre ces deux vecteurs centrés. Si le cœfficient vaut 1, les deux vecteurs sont colinéaires. S'il vaut -1, les deux vecteurs sont colinéaires de sens opposé. S'il vaut 0, les deux vecteurs sont orthogonaux. Le cœfficient de corrélation n'est pas sensible aux unités de chacune des variables. En revanche, ce cœfficient de corrélation va être très sensible à la présence de valeurs aberrantes et/ou extrêmes dans l'ensemble de données (valeurs très éloignées de la majorité des autres, pouvant être considérées comme significatives).

– le cœfficient de corrélation forcé de `Pearson` (Shardanand et Maes, 1995) ;

$$f_{simil}(u_a, u_j) = \frac{\sum_{i \in I_a \cap I_j} (v(u_a, i) - v_x)(v(u_j, i) - v_x)}{\sqrt{\sum_{i \in I_a \cap I_j} (v(u_a, i) - v_x)^2 \sum_{i \in I_a \cap I_j} (v(u_j, i) - v_x)^2}} \tag{3.3}$$

Il existe plusieurs variantes du cœfficient de corrélation forcé de Pearson. Parmi ces variantes, on retrouve les cas où v_x vaut v_{max} ou $\frac{v_{min}+v_{max}}{2}$ ou n'importe quelle autre valeur fixe. A titre d'exemple, si l'on prend $v_x = 4$, alors on intègre le caractère positif ou négatif d'un vote. Un vote supérieur à 4 sera considéré comme positif puisque le terme $(v(u_a, i) - v_x)$ sera positif, et inversement si le vote est inférieur à 4. Le cœfficient de corrélation forcé de Pearson est préférable lorsque les utilisateurs ont tendance à toujours attribuer la même note pour chaque item (votes très proches voire égaux à leur moyenne de votes).

– les moindres carrés (Shardanand et Maes, 1995) ;

$$f_{simil}(u_a, u_j) = \frac{1}{\sum_{i \in I_a \cap I_j} (v(u_a, i) - v(u_j, i))^2} \tag{3.4}$$

Cette métrique calcule le degré de dissemblance entre deux profils utilisateurs. Bien évidemment, plus ce degré de dissemblance est élevé et plus le cœfficient de similarité est petit.

– le vecteur cosine (Breese et al., 1998) ;

$$f_{simil}(u_a, u_j) = \sum_{i \in I_a \cap I_j} \frac{v(u_a, i).v(u_j, i)}{\sqrt{\sum_{i' \in I_a} v(u_a, i')^2 \sum_{i' \in I_j} v(u_j, i')^2}} \tag{3.5}$$

Dans ce cas, les profils sont considérés comme des vecteurs dans l'espace de représentation utilisateurs/ressources $R_{u/r}$. La similarité cosine entre u_a et u_j est alors calculée comme étant le cosinus de l'angle entre $\overrightarrow{u_a}$ et $\overrightarrow{u_j}$ (cf. infra, equation 3.6). Cette valeur est donc comprise entre 0 et 1. Le cœfficient vaut 0 lorsque les variables sont indépendantes et 1 lorsque les profils sont similaires.

$$sim(i, j) = cos(\overrightarrow{u_a}, \overrightarrow{u_j}) = \frac{\overrightarrow{u_a} \cdot \overrightarrow{u_j}}{||\overrightarrow{u_a}|| \ \ ||\overrightarrow{u_j}||} \tag{3.6}$$

L'état de l'art (Shardanand et Maes, 1995; Breese et al., 1998) montre que la métrique de Pearson fournit de bons résultats en terme de qualité des prédictions. Toutefois, les prédictions risquent de perdre en pertinence (toutes métriques confondues) si la matrice M possède un faible nombre de ressources communément évaluées. Il existe différents moyens de pallier les données manquantes. Une solution consiste à allouer à chaque ressource une note par défaut. Toutefois, cette méthode générera du bruit dans le calcul. Une autre idée serait d'amplifier le cœfficient $f_{simil}(u_a, u_j)$ en pondérant l'évaluation par la fréquence utilisateur inverse : $\frac{1}{log\left(\frac{\text{nombre total d'utilisateurs}}{\text{nombre d'utilisateurs ayant évalué la ressource}}\right)}$. On valorise ainsi les documents pour lesquels un grand nombre de personnes ont voté.

Pour cette famille d'algorithmes, l'apprentissage des communautés se fait de façon implicite. Il s'agit de tenir d'autant plus compte de l'avis d'un individu qu'il est proche de l'utilisateur courant, ce qui revient à identifier les interlocuteurs pertinents (plus proches voisins) sans les regrouper explicitement en communautés.

Les algorithmes basés sur la mémoire présentent l'avantage d'être simples et d'évoluer dynamiquement en fonction des profils utilisateurs. Ils sont en effet très réactifs puisque la mise à jour d'un profil se répercute immédiatement dans le calcul de prédiction. Breese et al. (1998) s'accordent toutefois à trouver leur passage à l'échelle problématique : si ces méthodes fonctionnent bien sur des exemples de tailles réduites, il est difficile de passer à des situations caractérisées par un grand nombre de ressources ou d'utilisateurs. En effet, la complexité de ces algorithmes en temps et en mémoire est beaucoup trop importante pour les grosses bases de données.

3.3 Algorithmes basés sur un modèle

Les algorithmes basés sur un modèle constituent une alternative au problème de complexité combinatoire. Dans cette approche, le filtrage collaboratif peut être vu comme le calcul de la valeur attendue d'un vote, compte tenu des préférences de l'utilisateur courant. Ces algorithmes créent des modèles descriptifs corrélant les individus, les ressources et les votes associés. Lorsque la corrélation porte sur les individus, on parle de communautés d'intérêts virtuelles. Une autre approche, fonctionnant sur le même principe, vise à rapprocher les items en fonction des usages (ressources co-consultées et appréciées par plusieurs utilisateurs). On parle alors de groupes d'items (en anglais *clusters*). Les prédictions sont ensuite inférées depuis ces modèles. Ces derniers sont le plus souvent probabilistes ou basés sur du *clustering* (Lumineau, 2003).

3.3.1 Approche probabiliste

L'approche probabiliste (Breese et al., 1998) repose sur le même principe que la formule 3.1 vue précédemment (cf. supra, 3.2 Algorithmes basés sur la mémoire, p. 53). Etant donné que les algorithmes basés sur les modèles ne disposent pas forcément de la totalité des informations relatives aux utilisateurs, on raisonne désormais en terme d'évaluation espérée $E\big(v(u_a, i_k)\big)$:

$$f_{simil}(u_a, i_k) = E\left(v(u_a, i_k)\right) = \sum_{j=v_{min}}^{v_{max}} j.P\left(v(u_a, i_k) = j \mid eval(u_a, i), i \in I_a\right)$$

$$\text{Avec : } I_a = \{\text{ressources évaluées par } u_a\}. \quad (3.7)$$

Le terme P(...) dans la formule 3.7 représente la probabilité que l'utilisateur u_a donne la note j à la ressource i_k connaissant ses évaluations antérieures.

De plus, la variable v_{max} correspond à la valeur maximale d'une évaluation (les notes attribuées par les utilisateurs sont réparties sur une échelle allant de 0 à 5, de 0 à 7, ou de 0 à 10, par exemple).

Les modèles probabilistes peuvent, entre autres, revêtir la forme de modèles de classes ou de modèles graphiques (réseaux bayésiens, arbres de décision, etc.).

Classifieur naïf de Bayes

L'approche par classifieur nécessite d'introduire le concept de classe (Breese et al., 1998). L'idée de base s'appuie sur le fait qu'il existe des groupes ou types d'utilisateurs ayant un ensemble commun de préférences et de goûts. L'utilisation d'un classifieur naïf de Bayes repose sur l'hypothèse d'indépendance des préférences (exprimées sous la forme de votes sur divers items) pour une classe donnée. Le modèle reliant la probabilité jointe d'une classe et des votes à un ensemble malléable de distributions conditionnelles et limitées est la formulation standard "naïve" de Bayes :

$$P(C = c, v_{a,1}, ..., v_{a,m}) \; \alpha \; P(C = c) \prod_{i=1}^{m} P(v_{a,i} \mid C = c) \tag{3.8}$$

Le terme à gauche de la formule 3.8 correspond à la probabilité d'observer un individu dans une classe précise c avec cet ensemble de votes. Dans ce contexte, il est aisé de calculer les probabilités requises pour l'équation 3.7. Les paramètres du modèle, à savoir les probabilités $P(C = c)$ d'appartenance à une classe et les probabilités conditionnelles $P(v_{a,i} \mid C = c)$ des votes étant donnée la classe, sont estimées à partir d'un corpus d'apprentissage (votes de l'utilisateur).

Dans la mesure où on ne peut pas observer les variables de classes dans la base de données utilisateurs, il est nécessaire de recourir à des méthodes apprenant les paramètres pour des modèles avec des variables cachées. L'algorithme EM[37] permet par exemple d'apprendre les paramètres pour une structure de modèle avec un nombre fixé de classes. Le choix du nombre de classes peut être basé sur la mesure de vraisemblance des données (Cheeseman et Stutz, 1996; Chickering et Heckerman, 1997).

Miyahara et Pazzani (2000), quant à eux, proposent de travailler sur la classification des items pour l'utilisateur courant à partir des votes des autres utilisateurs. Ils définissent deux classes « Aime » et « AimePas ». Cela suppose de travailler sur une matrice de votes booléens, en effectuant la transformation de la figure 3.2. Lorsque l'on a $v(u_a, i_k) \geq v_{seuil}$, alors on considère que u_a aime i_k. u_a prend donc la valeur 1 pour la classe « Aime » et 0 pour la classe « AimePas ». A l'inverse, si $v(u_a, i_k) < v_{seuil}$, u_a prend donc la valeur 0 pour la classe « Aime » et 1 pour la classe « AimePas ». En l'absence d'information sur les préférences de u_a pour i_k, on met 0 dans les deux classes. Dans l'exemple de la figure 3.2, la valeur v_{seuil} est fixée à 4.

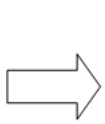
Pour déterminer la classe prédite à l'item i_k dans l'exemple de la figure 3.2 (pour les cas où les deux classes prennent une valeur nulle dans la matrice booléenne), il suffit de recourir à une adaptation de la formule 3.8 :

$$P(C = c \mid f_1, ..., f_n) \; \propto \; P(C = c) \prod_{i=1}^{n} P(f_i \mid C = c)$$

$$\text{Avec : } f_i = \begin{cases} u_p \text{ « AimePas » } i_k & \text{si } i = 2p, \; p \in \mathbb{N}, \\ U_{p+1} \text{ « Aime » } i_k & \text{si } i = 2p+1, \; p \in \mathbb{N}. \end{cases} \tag{3.9}$$

[37] EM pour *Expectation-Maximization*.

	i_1	i_2	i_3	i_4
u_1	5	4	-	3
u_2	-	-	3	1
u_3	2	3	-	5
u_4	-	5	4	-



	i_1	i_2	i_3	i_4
u_1 Aime	1	1	0	0
u_1 AimePas	0	0	0	1
u_2 Aime	0	0	0	0
u_2 AimePas	0	0	1	1
u_3 Aime	0	0	0	1
u_3 AimePas	1	1	0	0
u_4 Aime	0	1	1	0
u_4 AimePas	0	0	0	0

FIG. 3.2 – Transformation en matrice booléenne.

Comme dans le cas de la classification des utilisateurs (Breese et al., 1998), une phase d'apprentissage est requise pour utiliser la formule 3.9 : il s'agit de calculer les probabilités $P(C = c)$ et $P(f_i \mid C = c)$ par dénombrement. Il est alors possible de déterminer la classe la plus probable de i_k pour l'utilisateur courant.

Le classifieur naïf de Bayes présente l'avantage d'être simple à mettre en œuvre et peu coûteux. Au demeurant, l'hypothèse d'indépendance des votes s'avère peu réaliste. Breese et al. (1998) démontrent que les approches à base de réseaux bayésiens surpassent le classifieur de Bayes et les méthodes par similarité des vecteurs (vecteur cosine explicité plus haut) en terme de qualité des prédictions.

Arbres de décision et réseaux bayésiens

$B = (G, \theta)$ est un réseau bayésien si $G = (X, A)$ est un graphe acyclique dirigé (**DAG**) dont les sommets représentent un ensemble de variables aléatoires X $= \{X_1, ..., X_n\}$, et si $\theta_i = [P(X_i/X_{Pa(X_i)})]$ est la matrice des probabilités conditionnelles du nœud i connaissant l'état de ses parents $Pa(X_i)$ dans G (Huang et Darwiche, 1994). A correspond à l'ensemble des arêtes du graphe.

Un réseau bayésien B représente donc une distribution de probabilité sur X qui admet la loi jointe suivante :

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i/X_{Pa(X_i)}) \tag{3.10}$$

L'approche proposée par Breese et al. (1998) consiste à construire un réseau bayésien (cf. infra, figure 3.3) où chaque nœud correspond à une ressource et l'état du nœud à une valeur possible d'évaluation. La phase d'apprentissage consiste à rechercher les dépendances entre les

ressources (structure du réseau bayésien) et à construire les tables de probabilités conditionnelles. Dans le cas de la consultation d'items par exemple, les ressources consultées sont dépendantes entre elles si elles sont toutes consultées par un même groupe de personnes.

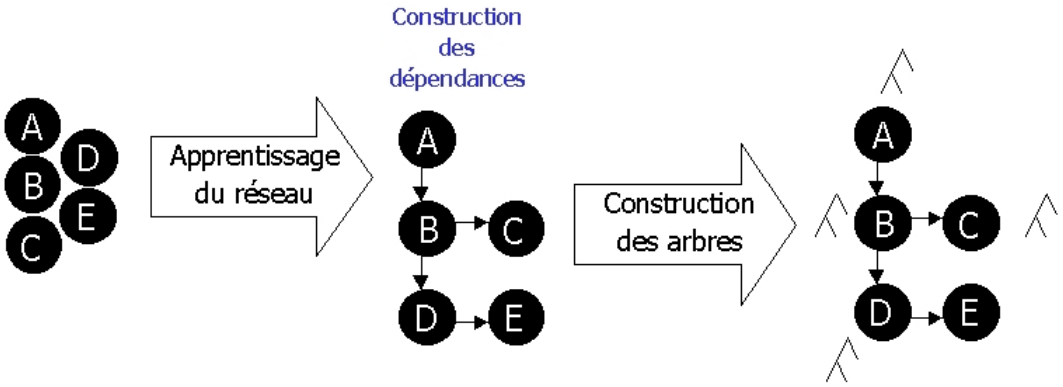


FIG. 3.3 – Construction du réseau bayésien (Breese et al., 1998; Lumineau, 2003).

En chaque nœud associé à une ressource i_k, il est ensuite possible d'injecter les évidences (i.e. les votes connus de l'utilisateur courant vis-à-vis des items associés aux nœuds parents). Cela permet de déduire, à partir de la table de probabilités conditionnelles du nœud considéré, la probabilité que u_a apprécie i_k en fonction d'un sous-ensemble pertinent de ses préférences connues.

Pour plus de lisibilité, il est également possible de représenter chaque table de probabilités conditionnelles sous la forme d'un arbre de décision (cf. infra, figure 3.4) en fonction des prédécesseurs qui sont les plus à même de prédire l'état du nœud. Un parcours de l'arbre de décision en fonction de l'appréciation qui a été faite des prédécesseurs (sur la figure 3.4, il s'agit des items `A` et `B`) permet de prédire l'intérêt porté par l'utilisateur à la ressource du nœud courant (item `D` sur la figure 3.4). Dans l'exemple de la figure 3.4, si l'utilisateur a aimé les ressources `A` et `B`, alors il y a une forte probabilité qu'il n'apprécie pas le document `D`.

D'après les expérimentations menées par (Breese et al., 1998), l'approche à base de réseaux bayésiens fournit des recommandations de très bonne qualité. Par ailleurs, les réseaux bayésiens sont appréciables pour leur capacité à mêler les probabilités issues d'un traitement statistique de retour d'expérience et les probabilités subjectives. Ils permettent enfin d'expliciter les relations existantes entre les ressources. Toutefois, le temps de calcul nécessaire lors de la construction du réseau bayésien explose et devient totalement inapproprié à un contexte industriel lorsqu'il y a un trop grand nombre de ressources (variables aléatoires). De plus, cette approche nécessite une phase d'apprentissage qui pénalisera l'utilisateur en ne lui fournissant pas des items pertinents dès les premières requêtes.

Une alternative aux approches probabilistes est le *clustering*.

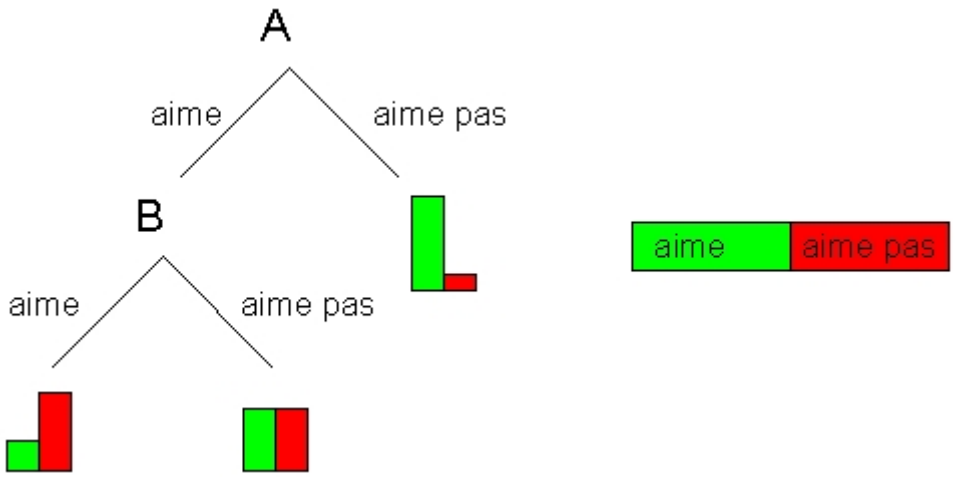


FIG. 3.4 – Arbre de décision (exemple au niveau du nœud D).

3.3.2 *Clustering*

Les méthodes dites de *clustering* permettent de limiter le nombre d'individus considérés dans le calcul de la prédiction. Ainsi, le temps de traitement sera plus court et les résultats seront potentiellement plus pertinents puisque les observations porteront sur un groupe (également appelé *cluster*) plus proche de l'utilisateur courant. Pour vulgariser, cela revient à demander leur avis à un groupe de personnes ayant les mêmes goûts que l'utilisateur, plutôt que de consulter l'ensemble de la population. Les groupes d'utilisateurs constitués correspondent aux communautés d'intérêts virtuelles dont nous avons parlé précédemment. Nous verrons dans cette section qu'il est également possible d'effectuer du *clustering* basé sur la corrélation entre items plutôt qu'entre utilisateurs (Sarwar et al., 2001).

K-Means

La méthode des plus proches voisins, également appelée `K-means` (Herlocker et al., 1999), consiste dans un premier temps à choisir aléatoirement k centres dans l'espace de représentation utilisateurs/ressources $R_{u/r}$. Par la suite, chaque utilisateur est positionné dans le *cluster* de centre le plus proche (figure 3.5). Rappelons que dans $R_{u/r}$, chaque point correspond à un utilisateur dont les coordonnées sont les votes contenus dans son profil. Une fois les groupes de personnes ainsi formés, on recalcule la position des centres pour chaque *cluster* et on réitère l'opération depuis le début jusqu'à obtenir un état stable (où les centres ne bougent plus après recalcul de leur position). Cet algorithme est en $o(knt)$ pour k *clusters*, n individus et t itérations.

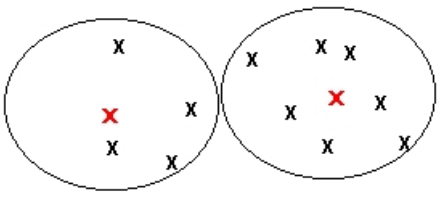


FIG. 3.5 – Espace de représentation utilisateurs/items.

Pour le calcul des prédictions, on procède comme pour les algorithmes basés sur la mémoire

avec une formule similaire à l'équation 3.1. Au demeurant, la somme ne porte plus sur l'ensemble des utilisateurs de U_t. On ne considère plus que les profils des utilisateurs appartenant au même *cluster* que u_a.

Il existe d'autres algorithmes très proches des `K-means`, tel que `Firefly` (Shardanand et Maes, 1995). Ce dernier consiste à sélectionner uniquement les profils dont la valeur de corrélation de `Pearson` par rapport à l'utilisateur courant est supérieure à un seuil paramétré.

La méthode des `K-means` est utilisée dans de très nombreux systèmes de recommandations. Elle présente toutefois de nombreux défauts. D'une part, il est très difficile de connaître le nombre k de centres appropriés. D'autre part, cet algorithme est coûteux en temps de calcul et présente des problèmes de convergence. En effet, les *clusters* générés sont très dépendants de la phase d'initialisation (choix initial des k centres) et il est rare de tomber sur un optimum global minimisant les distances intra-groupes et maximisant les distances inter-groupes. D'une façon générale, il est même fréquent que l'algorithme ne termine pas. Une façon d'optimiser le choix des k centres, et d'atténuer un peu ces inconvénients, consiste à choisir aléatoirement 1% des points et à appliquer les `K-means` sur ce sous-ensemble (Bradley et Fayyad, 1998). Si l'algorithme termine, alors on peut réappliquer les `K-means` sur l'ensemble des points en initialisant avec les k centres obtenus sur les 1% d'utilisateurs. Si l'algorithme ne termine pas sur le sous-ensemble, il est toujours possible de choisir un autre sous-ensemble jusqu'à ce que le calcul aboutisse. Les différents essais avec 1% des points est peu coûteux puisqu'il porte sur un ensemble très réduit d'utilisateurs. Cela améliore le pourcentage de réussite, mais ne garantit en rien la convergence de l'algorithme vers une solution, ni l'optimalité de cette solution. Par ailleurs, les résultats sont non reproductibles (si on lance l'algorithme deux fois de suite, on n'obtiendra pas les mêmes résultats).

Bien que les `K-means` soit la méthode de *clustering* la plus populaire, il existe des algorithmes concurrents comme le *clustering* répété et le `Gibbs Sampling` (Ungar et Foster, 1998), ou encore le *clustering* hiérarchisé (Chee et al., 2001). Seule la constitution des *clusters* varient dans ces différentes méthodes. La phase de prédiction reste la même.

Repeated Clustering

L'algorithme de *clustering* répété (Ungar et Foster, 1998) consiste à effectuer des raffinements successifs des groupes. Ainsi, un premier regroupement des individus par rapport aux ressources votées aboutit à la création de classes d'utilisateurs. De même, les classes de ressources correspondent aux items regroupés par rapport aux utilisateurs qui les ont consultés. Puis, le processus est réitéré sur les classes issues de la première clusterisation.

Cette méthode présente un risque de surgénéralisation. En effet, au fil des raffinements successifs, on risque de regrouper des utilisateurs aux profils différents et ne devant pas s'influencer.

Gibbs Sampling

`Gibbs Sampling` (Ungar et Foster, 1998) est une méthode d'estimation de paramètres dans un modèle statistique.

L'algorithme se déroule en deux phases appelées « étape d'attribution » et « étape d'estimation » :

1. Dans un premier temps, le système prend un utilisateur ou une ressource au hasard et lui attribue une classe proportionnellement à la probabilité suivante :

$$P_k \prod_l P_{kl}^{\sum_{j:C}^{j=l} Y_{ij}} \cdot \left(1 - P_{kl}\right)^{\sum_{j:C}^{j=l} (1-Y_{ij})} \tag{3.11}$$

Avec : P_k la probabilité qu'un utilisateur soit dans la classe C_k ;
P_l la probabilité qu'une ressource soit dans la classe C_l ;
P_{kl} la probabilité qu'une personne de la classe C_k soit liée à une ressource de la classe C_l ;

$$Y_{ij} = \begin{cases} 1 & \text{si l'utilisateur i aime la ressource j,} \\ 0 & \text{sinon.} \end{cases}$$

2. Durant la seconde phase, le système prend P_k et P_l selon une β-distribution paramétrée par le nombre d'utilisateurs et de ressources dans chaque classe. Ensuite, il prend P_{kl} selon une β-distribution paramétrée par le nombre d'utilisateurs qui aiment et n'aiment pas la ressource.

Une β-distribution est une distribution continue des probabilités (comme une courbe gaussienne par exemple), avec la fonction de densité définie sur l'intervalle $[0, 1]$:

$$f(x) \;=\; [constante] \,.\, x^{a-1}(1-x)^{b-1} \tag{3.12}$$

Où a et b sont des paramètres.
Quand la « constante » est incluse explicitement, la densité ressemble à ceci :

$$\begin{aligned} f(x) \;&=\; \frac{x^{a-1}(1-x)^{b-1}}{\int_0^1 u^{a-1}(1-u)^{b-1}du} \\ &=\; \frac{\Gamma(a\;+\;b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} \;=\; \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} \end{aligned} \tag{3.13}$$

Où Γ et B sont respectivement la Γ-fonction et la β-fonction.
La Γ-fonction permet d'étendre le concept de « factoriel » aux nombres complexes :

$$\Gamma(z) \;=\; \int_0^\infty t^{z-1}e^{-t}dt, \text{ si la partie réelle du nombre complexe z est positive} \tag{3.14}$$

La β-fonction, également appelée « Intégrale d'`Euler` », est définie par :

$$B(x,y) \;=\; \int_0^1 t^{x-1}(1-t)^{y-1}dt \tag{3.15}$$

Cet algorithme a une convergence assurée vers la bonne distribution mais est extrêmement coûteux en temps de calcul.

RecTree

Le *clustering* hiérarchisé, également appelé `RecTree` (Chee et al., 2001), cherche à fractionner l'ensemble des utilisateurs en cliques. Celles-ci sont hiérarchisées sous forme d'un arbre comme l'illustre la figure 3.6.

La matrice des votes 3.1 correspondant à l'exemple proposé est représentée dans la figure 3.6.

TAB. 3.1 – Exemple de matrice des votes.

	i_1	i_2	i_3	i_4	i_5
u_1			7	6	
u_2			5	6	7
u_3			6	6	7
u_4	7	5			7
u_5	7	6			7

On commence par associer à la racine la répartition des votes de toute la population U_t. Ainsi, dans l'exemple de la figure 3.6, 40% des utilisateurs ont attribué la note 7 à l'item i_1 et 60% ne l'ont pas consulté, etc. Pour former l'arbre, on cherche ensuite à maximiser les similarités entre les membres d'une même clique et à minimiser celles entre les membres de deux cliques différentes. Ainsi, plus on descend dans l'arbre et plus les *clusters* sont spécifiques à un certain groupe d'utilisateurs similaires. Par conséquent, plus on parcourt l'arbre en profondeur, plus les individus partagent le même avis concernant l'attribution d'une certaine note à un article donné. La construction de l'arbre s'opère en $o(q.n.log_2 n)$, où n est le nombre d'utilisateurs et q une constante dépendante du nombre d'itérations maximum autorisé lors du fractionnement de la population en deux sous-ensembles.

La phase d'identification de l'utilisateur courant à l'une des cliques se fait par la suite en $o(b)$, où b correspond au nombre de personnes dans chaque clique. En effet, l'arbre est construit de telle sorte que les cliques contiennent à peu près le même nombre d'individus pour une profondeur donnée.

Sur la figure 3.6, les chiffres en rouge à droite de chaque clique correspondent à la liste des utilisateurs intégrant ce groupe. De plus, les probabilités d'atteindre chaque nœud de l'arbre, étant donné le nœud parent, sont affichées au-dessus de chaque clique. La probabilité $P(root)$ de la racine vaut évidemment 1.

Cette algorithme fournit des prédictions de qualité moyenne. Par ailleurs, il présente les mêmes problèmes de convergence que l'algorithme `K-means`. Il s'avère toutefois intéressant car il permet de subdiviser la tâche visant à constituer les communautés d'intérêts virtuelles. Il est potentiellement moins coûteux en temps de calcul que les `K-means` et il a une moindre complexité combinatoire.

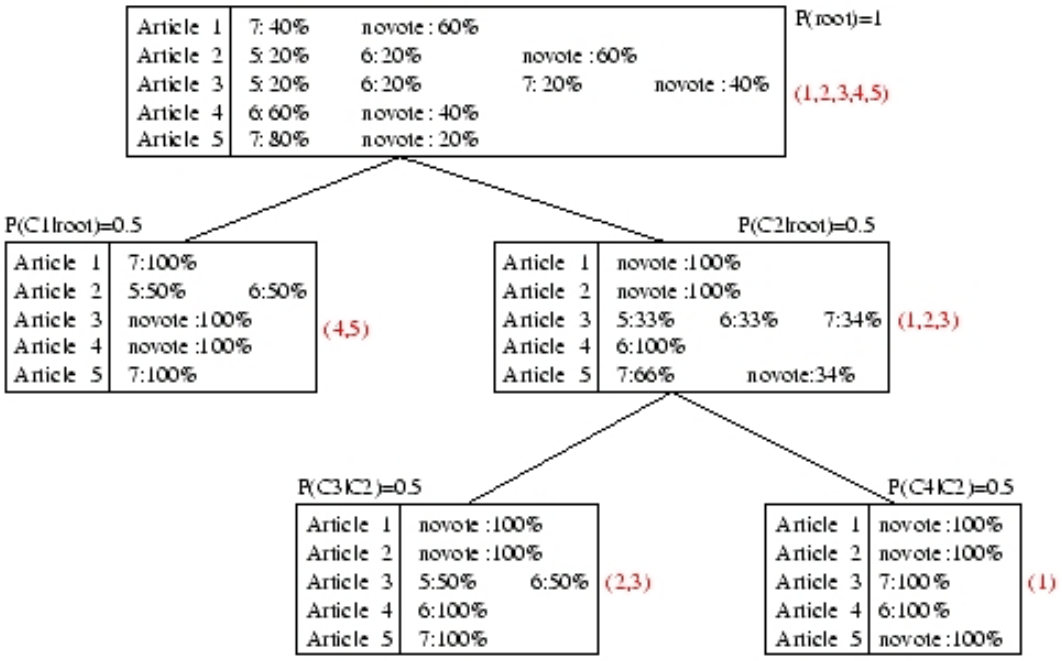


FIG. 3.6 – Hiérarchie de cliques.

3.3.3 Item-Item

Cette approche basée sur un modèle part du principe que la consultation d'un item engendre généralement chez le lecteur un intérêt pour une autre ressource. L'algorithme item-item (Sarwar et al., 2001) consiste à construire un modèle identifiant les relations existantes entre les ressources, à la différence des précédents algorithmes de *clustering* dont le rôle était de rapprocher les utilisateurs entre eux. Nous pouvons donc introduire une classification des algorithmes de filtrage collaboratif transversale à celle distinguant les algorithmes basés sur la mémoire et ceux basés sur un modèle : il y a les approches basées sur les utilisateurs et celles basées sur les items.

Une façon simple d'élaborer ce modèle consiste par exemple à compter le nombre de fois qu'une paire d'items est parcourue par un même utilisateur. Transposons cette technique dans le cadre d'un système de recommandations de films à des fins d'illustration. D'après le tableau 3.2), un grand nombre de cinéphiles ayant vu le film "Pulp fiction" se sont également déplacés pour le film "Memento". Ces deux œuvres cinématographiques semblent donc corrélées et il peut s'avérer pertinent de recommander "Memento" à une personne ayant visionné "Pulp fiction". Notons que ce type de modèle fait également ressortir des évidences : une grande proportion des spectateurs de "Harry Potter 1" se retrouvera également dans les salles obscures pour les autres opus de la saga. Certains items sont corrélés de fait, sans qu'il soit nécessaire d'utiliser des méthodes statistiques. Au demeurant, ces associations peuvent ressortir par le biais de ces modèles.

Une fois le modèle constitué (ici représenté par la matrice de corrélation du tableau 3.2), il est possible de recommander à l'utilisateur courant des items corrélés à ceux qu'il a consultés. Dans

TAB. 3.2 – Exemple de modèle item-item comptant les films co-visionnés.

	Pulp fiction	Memento	Harry Potter 1	Harry Potter 2	Harry Potter 3
Pulp fiction	-	**40**	7	6	9
Memento	**40**	-	5	6	7
Harry Potter 1	7	5	-	**30**	**29**
Harry Potter 2	6	6	**30**	-	**29**
Harry Potter 3	9	7	**29**	**29**	-

l'exemple précédent, on sélectionne les lignes de la matrice correspondant aux items consultés par l'utilisateur. Puis, en additionnant colonne par colonne les valeurs contenues dans ces lignes, on obtient un vecteur de corrélation entre les consultations passées de l'utilisateur et chacun des items disponible. Il suffit alors de trier ce vecteur pour obtenir la liste des N recommandations les plus pertinentes pour l'utilisateur courant, c'est-à-dire des N items associés aux valeurs les plus fortes (top-N).

Cette façon simple de procéder présente deux inconvénients :

1. les recommandations faites à l'utilisateur correspondent aux items les plus populaires, et pas forcément aux plus pertinents ;
2. il n'est pas possible d'estimer l'intérêt plus ou moins fort que peut représenter une recommandation par rapport à une autre (pas de formule de prédiction).

Il est donc possible de raffiner le modèle pour améliorer la qualité des prédictions. Dans cette optique, Sarwar et al. (2001) proposent de construire le modèle de corrélation non plus en fonction du décompte des co-consultations d'items, mais sur la base de votes explicitement fournis par les utilisateurs sur les items. Il s'agit alors de transformer la matrice de votes utilisateurs-items M en une matrice de similarité entre les items S (figure 3.7).

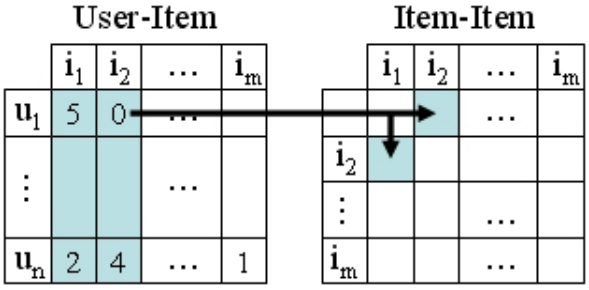


FIG. 3.7 – Transformation d'une matrice utilisateurs-items en une matrice items-items.

Appelons f_{simil_item} l'équivalent de la fonction de similarité f_{simil} vue précédemment, mais

adaptée au calcul entre items. On a $f_{simil\_item} : \mathbb{R}^{n'} \times \mathbb{R}^{n'} \rightarrow \mathbb{R}$ lorsque $card(U_t) = n'$. La matrice item-item S est symétrique (cf. formule 3.16) puisque $f_{simil\_item}(i_j, i_k) = f_{simil\_item}(i_k, i_j) = s_{i_j,i_k}, \forall i_j, i_k \in I$.

$$S = \begin{bmatrix} - & s_{1,2} & \dots & s_{1,m} \\ s_{1,2} & - & \dots & s_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1,m} & s_{2,m} & \dots & - \end{bmatrix} \tag{3.16}$$

La métrique préconisée[38] pour déterminer le score de similarité entre deux items est le vecteur cosine ajusté :

$$f_{simil\_item}(i_j, i_k) = \frac{\sum_{u \in U_t}(v(u, i_j) - \overline{v(u)})(v(u, i_k) - \overline{v(u)})}{\sqrt{\sum_{u \in U_t}(v(u, i_j) - \overline{v(u)})^2 \sum_{u \in U_t}(v(u, i_k) - \overline{v(u)})^2}} \tag{3.17}$$

Le vecteur cosine ajusté n'est autre qu'une adaptation du cœfficient de Pearson (cf. equation 3.2, p.55) au calcul de corrélation entre items.

Dans l'équation 3.17, i_j et i_k correspondent aux deux items pour lesquels on souhaite calculer la similarité.

L'intérêt estimé de l'utilisateur courant u_a pour une ressource i_k est calculé à l'aide de la formule de prédiction suivante :

$$f_{predict}(u_a, i_k) = \frac{\sum_{i \in I_a} |f_{simil\_item}(i_k, i)| \times v(u_a, i)}{\sum_{i \in I_a} |f_{simil\_item}(i_k, i)|} \tag{3.18}$$

I_a correspond à l'ensemble des items votés par u_a (c'est-à-dire les votes contenus dans le profil de u_a). La formule 3.18 exploite donc les votes connus de l'utilisateur courant pour prédire les votes encore inconnus. L'algorithme tient d'autant plus compte d'un vote $v(u_a, i)$ que l'item i associé à cette note est similaire à i_k.

Sarwar et al. (2001) avancent l'hypothèse qu'il est plus facile de rapprocher les items que les utilisateurs car il y a potentiellement moins d'items que d'utilisateurs et parce que le catalogue de ressources I_t est souvent plus stable. Une approche basée sur les items s'avère effectivement pertinente lorsque le nombre d'utilisateurs n est très supérieur au nombre de ressources m : $n \gg m$. Par ailleurs, l'algorithme item-item produit des prédictions de très bonne qualité. Toutefois, les raisons avancées par Sarwar et al. (2001) pour priviligier les algorithmes basés sur les items sont discutables. En effet, selon le contexte applicatif, il n'est pas rare que le nombre d'items dépasse le nombre d'utilisateurs. Ainsi, un intranet documentaire au sein d'une entreprise aura un ensemble limité d'utilisateurs (les employés de la société) mais un nombre potentiellement très grand de ressources accessibles. De même, la stabilité de l'ensemble des items I_t peut être remis en cause en dehors du contexte du commerce électronique. Si les produits vendus par un site marchand sont souvent les mêmes, les items accessibles sur une plate-forme documentaire (et plus globalement au sein d'un service de recherche et d'accès à l'information)

[38] On peut également utiliser d'autres métriques comme le vecteur cosine par exemple.

peuvent être extrêmement volatiles. Nous supposons donc que le choix d'une méthode basée sur les items ou sur les utilisateurs dépend du contexte applicatif (Castagnos et Boyer, 2006a, 2007c).

3.4 Algorithmes hybrides

Pennock et al. (2000a) s'accordent à penser que les algorithmes basés sur un modèle ont une moindre complexité combinatoire que ceux basés sur la mémoire. Toutefois, ils présentent globalement une évolutivité trop lente : la mise à jour du modèle est en effet coûteuse en temps de calcul et ne se fait pas en temps réel. Des algorithmes hybrides ont alors vu le jour, cherchant à combiner les avantages des approches basées sur la mémoire (réactives avec des prédictions de bonne qualité) et basées sur un modèle (moindre complexité).

3.4.1 Horting

Le filtrage collaboratif par `Horting` (Aggarwal et al., 1999) est une approche basée sur un graphe de relations de similarité (représentées par les arcs) entre les utilisateurs (correspondant aux nœuds du graphe). La notion d'influence (arcs) se décline sous forme de deux contraintes :

- la contrainte de `Horting` impose de ne considérer que les utilisateurs ayant un grand nombre d'évaluations communes ;
- la contrainte de prédictabilité ajoute à la notion de « `Horting` » une information sur le degré de ressemblance entre deux utilisateurs en se basant sur la distance de `Manhattan` (formule 3.19).

$$f_{simil}(u_a, u_j) \;=\; D_T(u_a, u_j) \;=\; \frac{\sum_{i \in I_a \cap I_j} \Big| v(u_a, i) - T\big(v(i_j, i)\big) \Big|}{card(I_a \cap I_j)} \tag{3.19}$$

Avec : $D_T(u_a, u_j)$ la distance de `Manhattan` entre les utilisateurs u_a et u_j ;
$v(u_j, i)$ l'évaluation de la ressource i par l'utilisateur u_j ;
T une transformation linéaire qui, à une évaluation définie dans l'ensemble $v_{min}, .., v_{max}$, associe une évaluation atténuée (pour éviter les notations excessives, c'est-à-dire les notes trop élevées ou trop basses) ;
I_j l'ensemble des ressources évaluées par l'utilisateur u_j.

Le parcours du graphe permet de filtrer les utilisateurs proches et ceux ayant une expérience importante. La notion de prédictabilité est plus contraignante que le concept de proximité car le système a besoin d'un échantillon suffisamment important de ressources communément évaluées.

3.4.2 Eigentaste

L'approche basée sur la clusterisation et la réduction de dimension, plus communément appelée système « `Eigentaste` » (Goldberg et al., 2000), se décompose en trois temps :

1. Chaque utilisateur commence par évaluer un ensemble de ressources « jauge » (il s'agit de documents décrits par des métadonnées pour connaître rapidement leur contenu) ;
2. la deuxième phase débute par une Analyse en Composantes Principales (**ACP**) afin d'avoir une réduction optimisée de l'espace de représentation utilisateurs/ressources. Puis, ce même espace réduit est partitionné selon le principe de clusterisation rectangulaire récursif (figure 3.8). Pour chacun des groupes ainsi créés, le système calcule la moyenne des évaluations autres que celles de l'ensemble jauge. Par le suite, il initialise une table de recommandations avec la liste des moyennes calculées, triées par ordre croissant ;
3. Enfin, la dernière étape consiste à rechercher en ligne le *cluster* le plus approprié (algorithme constant en temps de calcul = $o(1)$), à récupérer la prédiction dans la table des recommandations (afin que le système fournisse l'ensemble des ressources associées au *cluster*) et avoir un retour sur la satisfaction de l'utilisateur (*feedback*).

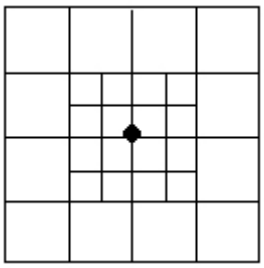


FIG. 3.8 – Clusterisation rectangulaire récursive.

Cette méthode présente deux inconvénients majeurs. Tout d'abord, la première phase dite de *profiling* est assez contraignante pour l'utilisateur qui doit évaluer un ensemble jauge. Par ailleurs, la deuxième phase s'effectue *offline* (afin de désynchroniser la partie des calculs coûteuse en temps) et peut donc engendrer des problèmes de rafraîchissement des informations.

3.4.3 Diagnostic de personnalité

Pennock et al. (2000b) ont introduit une méthode de modélisation de la personnalité permettant de considérer les évaluations des utilisateurs avec un bruit Gaussien. Ainsi, il est possible de prendre en compte des paramètres extérieurs tels que l'humeur de l'utilisateur, par exemple. Les évaluations sont considérées comme des symptômes et le type de personnalité comme une maladie. De cette manière, déterminer cette personnalité revient à identifier la cause la plus probable des évaluations.

Cet algorithme, bien que basé sur un modèle probabiliste, se gère comme une approche basée sur la mémoire car toutes les informations sont nécessaires pour les calculs de probabilité. Le principal bénéfice de cette approche réside dans le fait que les prédictions du modèle sont exprimées explicitement ce qui rend possible leur modification et leur validation.

3.5 Synthèse

Les algorithmes basés sur la mémoire maintiennent une base de données de tous les votes de tous les utilisateurs et chaque prédiction entraîne un calcul sur l'ensemble de cette source de données. Ces techniques présentent donc l'avantage d'être très réactives, en intégrant immédiatement au système les modifications des profils utilisateurs. Au demeurant, si ces méthodes fonctionnent bien sur des exemples de tailles réduites, il est difficile de passer à des situations proposant un grand nombre d'items ou d'utilisateurs. En effet, la complexité des algorithmes est beaucoup trop importante pour les grosses bases de données. L'alternative proposée, face au problème de complexité combinatoire, consiste à créer et exploiter des modèles capables d'effectuer les calculs de prédictions.

Pour ce faire, des algorithmes compilent tout d'abord les préférences des utilisateurs dans un modèle descriptif corrélant les individus, les ressources et les votes associés. Dans cette approche, les prédictions sont inférées depuis un modèle sous-jacent, calculé au préalable, des préférences de l'utilisateur. Ce modèle peut, par exemple, avoir la forme de réseaux de dépendances ou de réseaux bayésiens. Le grand avantage de ces derniers est de permettre la modélisation des relations non déterministes. Les réseaux bayésiens peuvent mêler les probabilités issues d'un traitement statistique de retour d'expérience et les probabilités subjectives.

Dans le cas des algorithmes basés sur un modèle, le filtrage collaboratif peut être vu comme le calcul de la valeur attendue d'un vote, étant donné ce que nous savons de l'utilisateur. Le modèle, lui-même, offre une valeur ajoutée au-delà de la seule fonction de prédiction. En effet, il peut mettre en lumière certaines corrélations dans les données, proposant ainsi un raisonnement intuitif pour les recommandations ou rendant simplement les hypothèses plus explicites. La méthode présentée dans la section 3.3.1 illustre, par exemple, la possibilité pour le réseau d'encoder les dépendances entre les ressources. On pourrait également recourir à la classification bayésienne pour regrouper les utilisateurs similaires en classes. Les méthodes à base de réseaux bayésiens fournissent de bonnes recommandations en terme de qualité, au même titre que les algorithmes basés sur la mémoire. Toutefois, la construction de la structure de l'arbre est très coûteuse : le temps nécessaire croit exponentiellement avec le nombre de ressources à considérer (i.e. le nombre de nœuds dans l'arbre).

Une autre façon d'aborder le problème du filtrage collaboratif consiste à classifier les utilisateurs et les documents en groupes. Pour chaque catégorie d'utilisateurs, il s'agit d'estimer la probabilité qu'une ressource soit choisie. Ces approches souffrent bien souvent de problèmes de convergence liés à l'initialisation des *clusters* et fournissent dans certains cas des recommandations de mauvaise qualité.

Les algorithmes basés sur un modèle minimisent le problème de la complexité combinatoire. Cependant, ces méthodes ne sont pas assez dynamiques et elles réagissent mal à l'insertion de nouveaux contenus dans la base de données.

A la lecture de cet état de l'art, la problématique principale du filtrage collaboratif reste le passage à l'échelle des systèmes. Une solution consiste à scinder de manière explicite les calculs dits « *on-line* » de ceux dits « *off-line* ». Ceci a abouti au développement d'algorithmes

hybrides cherchant à réduire le nombre de calculs « *on-line* » et à augmenter le nombre de calculs « *off-line* ». Cette scission a constitué les prémisses du filtrage collaboratif distribué au sein d'architectures client/serveur. La partie « *off-line* » est gérée sur le serveur, tandis que les utilisateurs obtiennent une réponse en temps réel côté client grâce à la partie « *on-line* » des calculs. L'étape suivante vers la distribution du filtrage collaboratif, et donc vers des systèmes à large échelle, est la répartition complète des calculs et des contenus sur un réseau d'ordinateurs. La section suivante vise à présenter des travaux menés dans le domaine du filtrage collaboratif dans un cadre totalement distribué sur des grilles de calcul (architectures pair-à-pair).

3.6 Les modèles distribués

3.6.1 Système avec topologie de voisinage hiérarchique

Berkovsky et al. (2006) proposent un algorithme de filtrage collaboratif distribué adapté aux architectures pair-à-pair et reposant sur une topologie de voisinage hiérarchique (cf. figure 3.9).

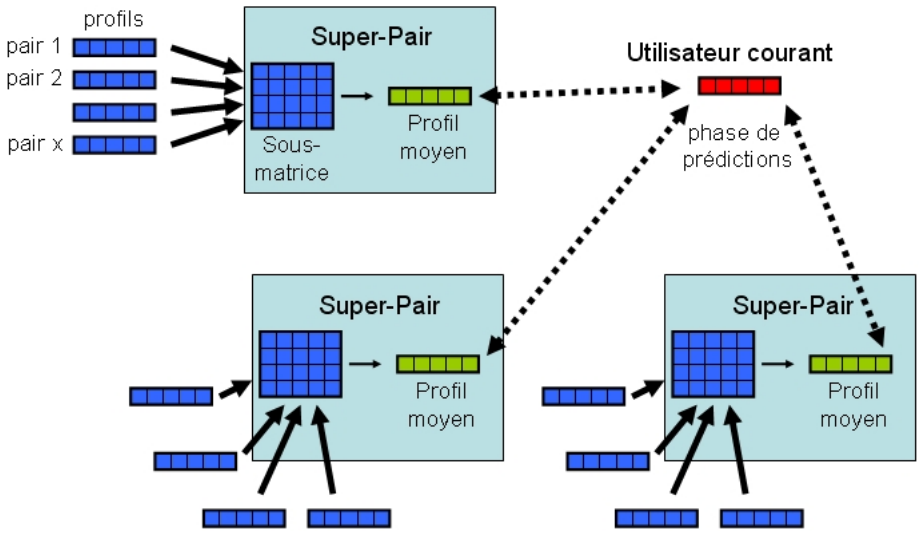


FIG. 3.9 – Système avec topologie de voisinage hiérarchique.

Leur modèle repose sur une méthode d'agrégation des profils améliorant la confidentialité au sein de systèmes de recommandations pair-à-pair. Le système commence par élire des super-pairs dont le rôle est de calculer un profil moyen pour une sous-population. Dans ce modèle, seuls les super-pairs ont accès aux profils des autres pairs. On peut par exemple considérer que le fournisseur du service alloue des serveurs pour jouer le rôle de ces super-pairs. Lors de la phase de prédiction, le système procède comme pour les algorithmes basés sur la mémoire (cf. supra, 3.2 Algorithmes basés sur la mémoire, p.53). Toutefois, il n'utilise que les profils moyens constitués, plutôt que des matrices regroupant l'ensemble des votes individuels. Ainsi, le procédé de filtrage collaboratif ne repose plus sur les préférences des plus proches voisins, mais se positionne sur une échelle macroscopique en exploitant directement les préférences moyennes de sous-populations. Chaque profil moyen simule en quelque sorte un super utilisateur qui aurait

les préférences de tous les membres de cette sous-population, garantissant par là même la confidentialité et améliorant les temps de calcul. Ce procédé a toutefois un impact sur la qualité des prédictions : calculer des profils moyens revient à perdre de l'information. La diminution de la qualité est d'autant plus conséquente que les sous-populations sont construites aléatoirement et ne constituent en rien des communautés d'intérêts virtuelles. Calculer la moyenne des votes d'une sous-population présente donc le risque de produire un super utilisateur qui aurait un avis mitigé sur toutes les ressources.

Berkovsky et al. (2006) traite également le problème de confidentialité des données en instaurant des politiques de modification du contenu des profils utilisateurs. Ils distinguent trois politiques :

- l'offuscation aléatoire uniforme permet au système de substituer des votes réels par des valeurs aléatoires choisies uniformément dans un intervalle de votes possibles ;
- l'offuscation aléatoire gaussienne remplace des votes réels par des valeurs aléatoires suivant une loi normale. Comme dans le domaine statistique, il s'agit de prêter attention à la moyenne et à la déviation standard des votes ;
- l'offuscation par défaut utilise une valeur constante "x" pour remplacer certaines données réelles.

Les politiques d'offuscation viennent renforcer la perte de qualité occasionné par les profils moyens en rajoutant du bruit dans les matrices de votes.

3.6.2 PocketLens

L'algorithme `PocketLens` (Miller et al., 2004) est une version distribuée de l'algorithme `Item-Item` (Sarwar et al., 2001) présenté dans la section 3.3.3. La distribution s'opère au sein d'un réseau pair-à-pair où chaque pair représente un utilisateur du système. Le pair de l'utilisateur courant se charge de communiquer avec les autres pairs pour récupérer les profils et construire le modèle. La clef de cet algorithme réside dans la construction incrémentale de la matrice de similarité. Afin de compléter cette dernière, `PocketLens` utilise le vecteur cosine pour déterminer la similarité entre deux items. Rappelons que dans un algorithme centralisé, le vecteur cosine entre deux items i_j et i_k est donné par la formule :

$$f_{simil\_item}(i_j, i_k) = \sum_{u \in U_t} \frac{v(u, i_j).v(u, i_k)}{\sqrt{\sum_{u' \in U_j} v(u', i_j)^2 \sum_{u' \in U_k} v(u', i_k)^2}} = \frac{\vec{i_j}.\vec{i_k}}{||\vec{i_j}||.||\vec{i_k}||} \tag{3.20}$$

Dans cette formule, U_j et U_k contiennent l'ensemble des utilisateurs ayant votés respectivement pour i_j et i_k. L'algorithme `PocketLens` a été conçu pour calculer incrémentalement ce score de similarité. Le système reçoit les profils des utilisateurs les uns après les autres et le score f_{simil_item} entre deux items n'est pas déterminé directement à la réception d'un profil. Il ne sera pas calculé avant le processus de recommandation. Concrètement, la matrice Item-Item contient divers attributs qui doivent être mis à jour et serviront donc à calculer la similarité plus tard (cf. figure 3.10). Chaque case (i_j, i_k) de la matrice contient les attributs suivants :

- la norme euclidienne partielle de $\vec{i_j}$, appelée `PtLenIj` ;
- la norme euclidienne partielle de $\vec{i_k}$, appelée `PtLenIk` ;

– le produit scalaire partiel `PartialDot` des vecteurs $\vec{i_j}$ et $\vec{i_k}$;
– le nombre de co-occurrence des deux items `Coocur`.

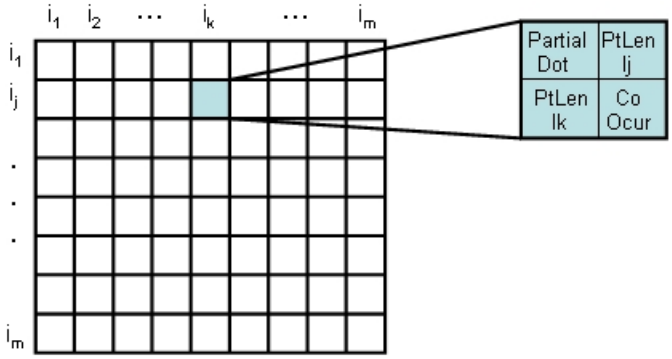


FIG. 3.10 – Calcul de la matrice Item-Item dans PocketLens.

Considérons que l'utilisateur courant reçoit le profil d'un autre pair $\vec{u_i}$. La matrice doit alors être mise à jour sur chaque ligne où se trouve un item commun à $\vec{u_a}$ et $\vec{u_i}$. Soit u la note de l'utilisateur pour l'item Oi et w la note fournie par le pair pour l'item Nj, la mise à jour des quatre attributs se fait selon les formules suivantes :

$$PartialDot(i_j, i_k) = PartialDot(i_j, i_k) + v(u_i, i_j).v(u_i, i_k) \tag{3.21}$$
$$PtLenIj(i_j, i_k) = PtLenIj(i_j, i_k) + v(u_i, i_j).v(u_i, i_j) \tag{3.22}$$
$$PtLenIk(i_j, i_k) = PtLenIk(i_j, i_k) + v(u_i, i_k).v(u_i, i_k) \tag{3.23}$$
$$Coocur(i_j, i_k) = Coocur(i_j, i_k) + 1 \tag{3.24}$$
$$\tag{3.25}$$

Le cœfficient de similarité est ensuite calculé par :

$$f_{simil\_item}(i_j, i_k) = k \times \frac{PartialDot(i_j, i_k)}{\sqrt{PtLenIj(i_j, i_k)}.\sqrt{PtLenIk(i_j, i_k)}} \tag{3.26}$$

Le cœfficient k sert à minimiser l'impact des items fortement similaires mais dont le nombre de notation est très faible au sein de la matrice. Il se calcule comme suit :

$$k = \begin{cases} 1 & \text{si } Coocur(i_j, i_k) >= 50, \\ \frac{Coocur(i_j, i_k)}{50} & \text{sinon} \end{cases} \tag{3.27}$$

Une fois le modèle construit et à jour, l'algorithme `PocketLens` est en mesure d'effectuer des recommandations sur le même principe que l'algorithme `Item-Item` (Sarwar et al., 2001).

Dans (Miller et al., 2004), les auteurs ont imaginé 5 topologies permettant une découverte plus ou moins rapide des autres usagers à travers le réseau tout en tentant de préserver leur vie privée :

– **une architecture avec serveur central** ;

Dans cette architecture, chaque utilisateur a un identifiant unique et persistant qui est utilisé pour associer ses notes aux items contenus dans la base de données du serveur central. Cette architecture est assez proche de celle du projet `SETI@home`[39] où les données sont stockées sur un serveur central et les calculs sont réalisés chez les pairs. Dans le cas présent, les votes sur les items sont stockés sur le serveur et chaque pair construit son propre modèle et calcule ses propres recommandations (cf. figure 3.11). Cette architecture ne permet donc pas une distribution complète du filtrage collaboratif. La préservation des données repose sur le fait que les votes sont collectés sur un serveur sécurisé.

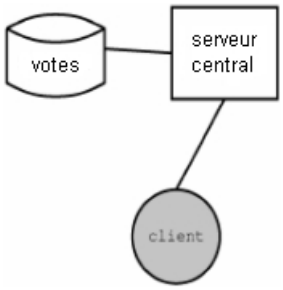


FIG. 3.11 – Architecture avec serveur central.

– **une architecture avec découverte aléatoire** ;

Cette architecture (cf. figure 3.12) permet à chaque utilisateur de rester anonyme. C'est une architecture similaire à celle utilisée par `Gnutella`[40]. Quand le client se connecte sur le réseau, il envoie un message de découverte. Lorsqu'un pair reçoit un de ces messages, il renvoie un message au client contenant la liste des pairs dont il a connaissance. Ainsi petit à petit le client trouve tous les autres pairs qu'il y a sur le réseau. Le client continue à envoyer des messages de découverte toutes les X secondes ou minutes, afin de découvrir les nouveaux pairs qui se sont connectés entre temps. Cette architecture ne nécessite pas de calculs supplémentaires au niveau des pairs mais génère potentiellement beaucoup de messages.

– **une architecture transitive transverse** ;

Cette architecture (cf.figure 3.13) se base sur celle avec découverte aléatoire mais permet en plus de connaître le voisin le plus similaire à chaque fois qu'un nouvel utilisateur est découvert. Afin de permettre de se souvenir du meilleur voisin, il est nécessaire de réintroduire un identifiant unique pour chaque utilisateur (identifiant qui n'était pas nécessaire pour l'architecture en découverte aléatoire). Cette identifiant peut revêtir n'importe quelle forme, du moment qu'il est unique et est

[39] http ://setiathome.berkeley.edu/
[40] http ://www.gnu.org/philosophy/gnutella.fr.html

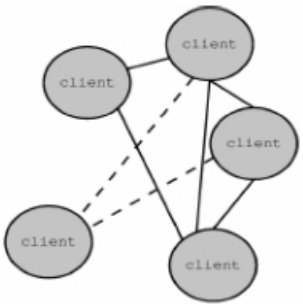


FIG. 3.12 – Architecture avec découverte aléatoire.

toujours le même pour un utilisateur donné. Avec cette architecture, l'algorithme de filtrage peut construire plus facilement la liste des voisins, en permettant aux pairs de transmettre leurs listes de voisins et ainsi construire le "voisinage" en utilisant une forme de transitivité. Cette approche se base sur le fait que les voisins du meilleur voisin d'un client, sont sûrement des bons voisins pour le client. Cette architecture construit une liste de voisins des voisins au fur et à mesure que le client parcourt ses voisins potentiels. Ainsi, si jamais le meilleur voisin n'est pas connecté au moment de la construction du modèle, la liste permet d'avoir des voisins de qualité à utiliser en remplacement. Cette architecture génère un peu moins de messages que la découverte aléatoire, mais nécessite des calculs supplémentaires afin de répartir au mieux les pairs sur le réseau.

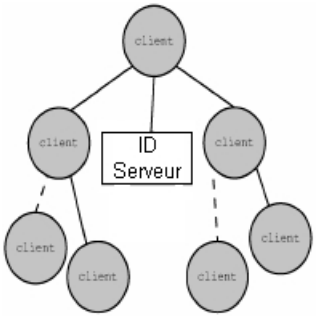


FIG. 3.13 – Architecture transitive transverse.

– **une architecture en anneau** ;

Cette architecture est basée sur l'algorithme de routage pour les systèmes de partage de fichier pair-à-pair tels que `Chord` (Stoica et al., 2001), `CAN` (Ratnasamy et al., 2001) ou encore `Pastry` (Rowstron et Druschel, 2001). Cet algorithme permet au réseau de fournir des garanties au niveau de la bande passante, du temps de recherche et de la disponibilité des données en positionnant les pairs sur un anneau virtuel (cf. figure 3.14). La solution imaginée par Miller et al. (2004) à partir de cette topologie s'avère au demeurant plus lourde que les précédentes car elle augmente la complexité

de l'algorithme pour rechercher le profil d'un pair spécifique tout en garantissant l'anonymat et donc pour construire le modèle. Ce processus est notamment basé sur des clefs de cryptage connues uniquement par les pairs successeurs dans l'anneau. Ce modèle souffre d'une contrainte forte : il est nécessaire qu'il y ait en permanence suffisamment de pairs connectés pour assurer la continuité du modèle. Par ailleurs, cette architecture est vulnérable aux attaques d'utilisateurs malveillants s'insérant dans l'anneau.

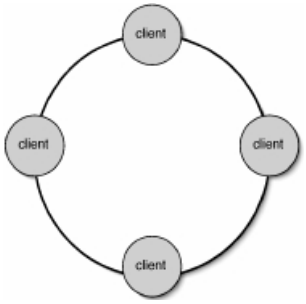


FIG. 3.14 – Architecture en anneau.

– **une architecture dite "tableau noir"** ;

Les algorithmes précédents se basaient sur l'anonymat et l'utilisation d'identifiant pour protéger la vie privée des utilisateurs. Avec cette architecture, les utilisateurs cryptent leurs profils avant de les envoyer aux autres utilisateurs. Les autres utilisateurs ont une fonction qui permet d'ajouter les profils dans le modèle sur le tableau noir sans devoir les décrypter. Enfin les autres utilisateurs peuvent décrypter le modèle complet. Ce modèle repose sur les travaux de Canny (2002a). Contrairement à l'algorithme de décomposition en valeurs singulières de ce dernier, `PocketLens` est capable avec cette architecture de rajouter de nouveaux votes incrémentalement sans réinitialiser le modèle. Au demeurant, ce modèle de prévoit pas de modification des votes existants. Il nécessite également de disposer de suffisamment de pairs simultanément pour que le modèle puisse être décrypté.

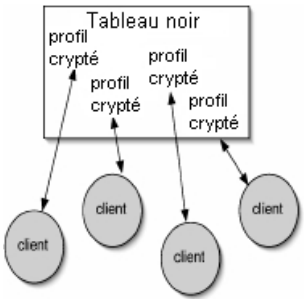


FIG. 3.15 – Architecture avec tableau noir.

Les architectures transitive transverse ou avec découverte aléatoire offrent un bon compromis entre simplicité et préservation des données personnelles. Elles sont moins vulnérables aux attaques d'utilisateurs malveillants et n'imposent aucune contrainte sur le nombre d'utilisateurs connectés simultanément. En outre, elles permettent une distribution complète des calculs et des profils, contrairement à l'architecture avec serveur central. Le caractère distribué de ces architectures permet à l'algorithme `PocketLens` de fournir des recommandations dans un délai très court, en contrepartie d'un nombre de messages accru au sein du réseau. Cela met en exergue l'intérêt de la distribution du filtrage collaboratif. `PocketLens` fournit également une excellente qualité de recommandations, puisqu'à l'inverse de l'algorithme de Berkovsky et al. (2006), il ne nécessite pas d'altérer les profils. Au demeurant, cet algorithme ne permet pas de gérer dynamiquement les mises à jour du modèle en cas de modifications des votes existants. Par ailleurs, comme toutes les approches basées sur les items, elles s'avèrent intéressantes dès lors que l'ensemble des items disponibles reste stable, ce qui n'est pas toujours le cas dans les systèmes de recherche et d'accès à l'information.

Chapitre 4

Filtrage collaboratif Client/Serveur

4.1 Architectures Client/Serveur

L'architecture Client/Serveur est la plus couramment utilisée sur Internet. C'est sur elle que repose le Web. L'étude de la distribution des procédés de filtrage collaboratif nous a donc naturellement conduit dans un premier temps à imaginer un moyen de répartir les calculs entre les clients et le serveur au sein d'une telle architecture. L'objectif premier de cette décentralisation des calculs est de s'attaquer à l'épineux problème du passage à l'échelle. Le nombre d'utilisateurs d'un service de recherche et d'accès à l'information se chiffre potentiellement en centaines de milliers voire millions. Le nombre de ressources peut varier de quelques centaines dans le cas de petits sites de e-commerce ou de communications satellitaires limitées en bande passante à plusieurs milliers – voire dizaines de milliers – dans le cas de la recherche documentaire.

L'introduction d'un système de recommandations dans un service de recherche et d'accès à l'information permet un gain de temps et d'énergie pour les utilisateurs. Ces derniers sont en effet assistés par le système et se voient suggérer une liste d'items adaptée à leurs attentes. L'ordre dans lequel les items sont exposés dans un système de recommandations présente une plus grande pertinence qu'un service sans personnalisation, occasionnant ce gain de temps lors des recherches des utilisateurs. Le gain d'énergie, lui, est lié au fait qu'il est possible de recommander des items intéressants aux utilisateurs sans qu'ils aient besoin d'effectuer une recherche. L'information vient directement à eux. Dans le cas d'un système où chaque utilisateur peut rechercher une information et où chaque information peut être soumise à un utilisateur, on parle de stratégie *Push and Pull*.

Les services de personnalisation ont donc un fort intérêt stratégique, commercial et/ou ergonomique selon le contexte d'utilisation. Toutefois, ils s'adressent à un public exigeant souhaitant obtenir des recommandations pertinentes sans attendre. Il s'agit donc de fournir des réponses en temps réel et d'atteindre rapidement (puis de conserver) un bon niveau de qualité des prédictions. En outre, un si grand nombre d'utilisateurs et de ressources implique nécessairement un grand nombre de données manquantes lors de la modélisation des préférences individuelles. Ce constat est d'autant plus vrai que les utilisateurs sont peu disposés à fournir de gros efforts lors de l'étape de modélisation. Par ailleurs, cette modélisation des préférences doit se faire dans le

respect de la vie privée des utilisateurs du service.

4.2 FRAC+

C'est dans ce triple objectif de passage à l'échelle, de gestion des données manquantes et de respect de la vie privée au sein d'une architecture client/serveur que nous avons conçu le modèle `FRAC+` (Castagnos et Boyer, 2006b) (cf. figure 4.4). Celui-ci prévoit, dans un premier temps, la modélisation des préférences de l'utilisateur courant côté client suivant le procédé générique présenté dans la section 1.4.2. Cette modélisation repose sur une analyse des usages. Les profils issues de cette première phase sont collectés sur le serveur et exploités par notre algorithme de *clustering* `FRAC`. Les communautés d'intérêts virtuelles obtenues permettent de suggérer des items à chaque utilisateur. L'identification de l'utilisateur courant à une communauté se fait côté client.

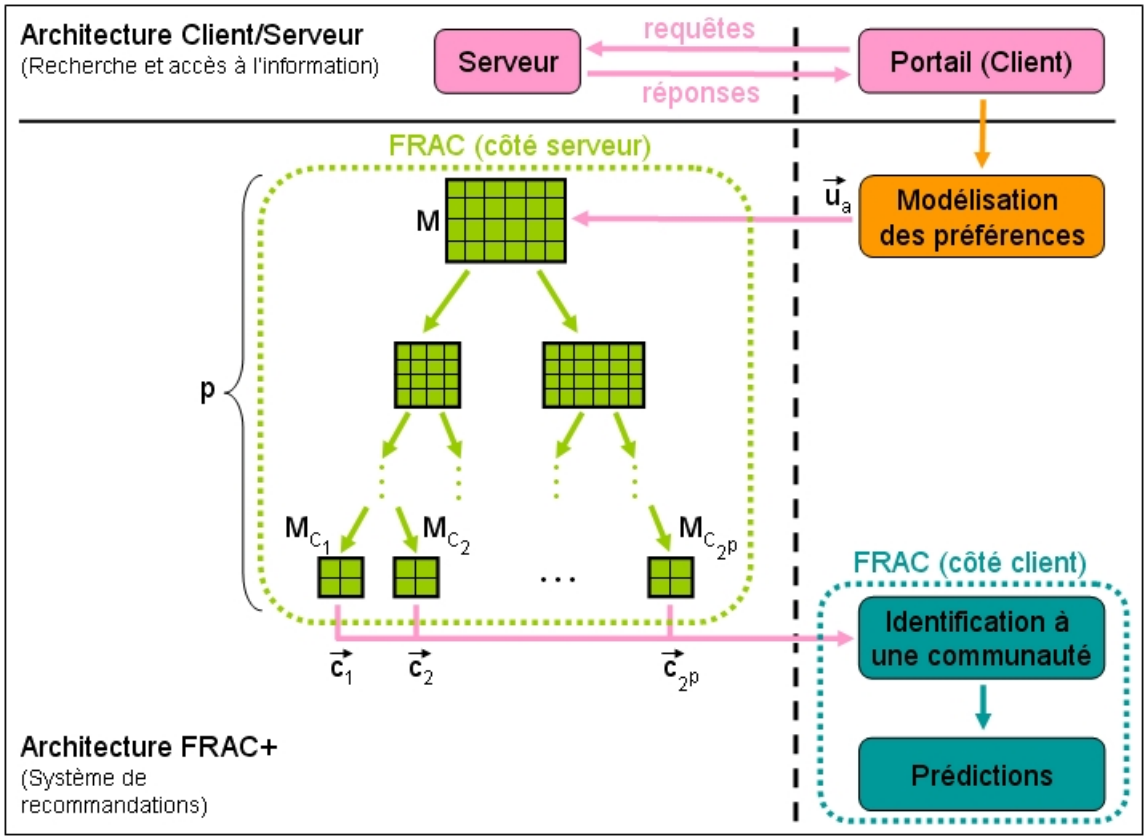


FIG. 4.1 – Modèle FRAC+.

Ainsi, le procédé de filtrage collaboratif est divisé en deux parties :
- la détermination des communautés est réalisé hors ligne côté serveur ;
- l'affectation d'un utilisateur à une communauté dans le but de bénéficier d'une expérience accrue et de lui faire des recommandations se fait en temps réel côté client ;

4.2.1 Apprentissage des communautés

Les profils utilisateurs collectés sur le serveur servent à construire les communautés d'intérêts virtuelles. Nous utilisons pour cela un algorithme de *clustering* inspiré de l'algorithme `RecTree` (Chee et al., 2001) (cf. supra, section 3.3.2, p.64) que nous appelons `FRAC`. Ce nom est l'acronyme de *Filtering with Recursive Algorithm of Clusterization.*

Les profils des utilisateurs sont agrégés sur le serveur sous la forme d'une matrice de votes `M`. De cette manière, le serveur ne dispose d'aucune information relative à la population, hormis les votes envoyés anonymement. Le critère de confidentialité est donc bien respecté.

L'algorithme `FRAC` côté serveur a pour but de réduire la quantité des données à traiter. Les calculs sont effectués hors ligne et permettent de fractionner l'ensemble des utilisateurs en sous-groupes. De cette façon, il n'est plus nécessaire de considérer l'intégralité de la matrice des votes, mais uniquement les votes des individus appartenant au groupe de l'utilisateur courant. Cela réduit le nombre d'individus considérés, mais également le nombre de ressources : il est inutile de conserver les ressources qui n'ont été consultées par aucun des membres du groupe.

Le tableau 4.1 propose une matrice de votes caractérisés par des entiers naturels allant de 1 à 7, pour reprendre l'exemple précédemment mentionné.

Tab. 4.1 – Exemple de matrice des votes.

	i_1	i_2	i_3	i_4	i_5
u_1		6	7	6	
u_2			4	7	7
u_3		7	6	5	
u_4	6	6			7

La figure 4.2 illustre les modalités d'organisation des groupes dans l'exemple ci-dessus. Les utilisateurs sont séparés en *clusters* selon la similarité déterminée entre chacun d'eux. De par la généricité de notre algorithme, il nous était possible d'intégrer n'importe quelle métrique de similarité f_{simil} présentée dans la section 3.2. Notre choix s'est porté sur le cœfficient de corrélation de Pearson (Resnick et al., 1994) (cf. supra, équation 3.3, p. 55), puisque cette métrique a été montrée comme étant une des plus performantes (Shardanand et Maes, 1995).

La première étape consiste à associer à la racine de l'arbre la matrice globale des votes des utilisateurs par rapport aux ressources. Par la suite, l'ensemble des utilisateurs sont répartis en deux sous-groupes à l'aide de la méthode des plus proches voisins, également appelée `K-means` Herlocker et al. (1999). Cette dernière consiste, dans un premier temps, à choisir aléatoirement k centres dans l'espace de représentation utilisateurs/ressources $R_{u/r}$ (cf. supra, section 3.3.2, p.61). Dans le cas présent, le nombre k vaut 2, puisqu'il faut subdiviser la population en deux sous-ensembles.

Comme précédemment mentionné, l'algorithme K-means classique présente des problèmes de convergence, car il est très dépendant de la phase d'initialisation consistant à choisir les centres initiaux. Nous proposons une méthode d'initialisation garantissant la convergence de

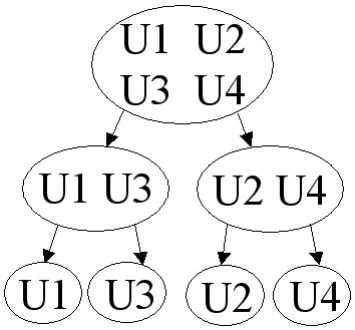


FIG. 4.2 – Hiérarchisation des utilisateurs.

l'algorithme quand K vaut 2 (Castagnos et Boyer, 2006a). Il s'agit dans un premier temps de prendre le milieu de l'espace de représentation utilisateurs/ressources (cf. figure 4.3). Ce milieu correspond à un utilisateur virtuel m qui aurait voté pour toutes les ressources avec la valeur médiane sur l'intervalle de votes (par exemple, valeur 4 sur une échelle de 1 à 7). Nous calculons ensuite la similarité entre tous les utilisateurs et ce milieu m. Nous choisissons le point de l'espace a correspondant à l'utilisateur le plus éloigné de m (similarité la plus faible). Puis nous déterminons le point b s'avérant le plus éloigné du point a. Ces points a et b constituent les centres initiaux pour les deux *clusters* dans l'algorithme des `2-means`.

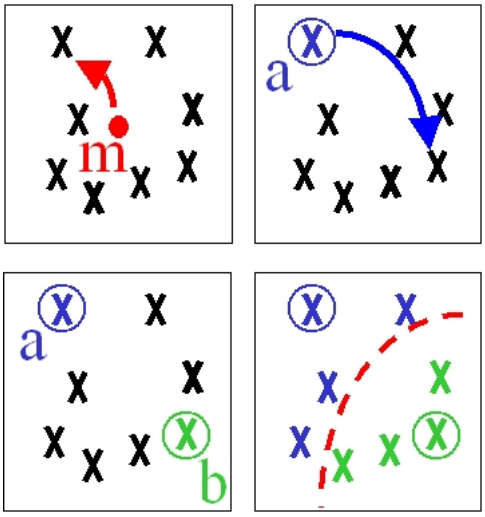


FIG. 4.3 – Initialisation de l'algorithme 2-means.

L'algorithme se décompose ensuite en deux phases :

1. Chaque utilisateur est ensuite positionné dans le *cluster* de centre le plus proche. Les utilisateurs indépendants (similarité nulle avec chacun des centres) et équidistants (même similarité avec les deux centres) sont temporairement écartés du calcul (au cours d'une seule et même itération des 2-means) ;
2. Puis nous recalculons les centres des deux *clusters* (cf. équation 4.1). Ceux-ci représentent

deux utilisateurs virtuels dont le profil contient la moyenne des votes des utilisateurs du même *cluster* pondérée par le degré de similarité. Ce sont donc les barycentres des deux groupes.

Ces deux phases sont répétées jusqu'à obtenir un état stable, c'est-à-dire jusqu'à ce que les centres des *clusters* ne bougent plus après recalcul de leurs positions respectives lors de la phase 2. Une fois l'état stable atteint[41], nous faisons en sorte que les deux groupes obtenus aient des tailles équivalentes (i.e. un même nombre d'utilisateurs à un utilisateur près, dans le cas où $card(U) = n$ est impair). Pour ce faire, nous ajoutons un à un les utilisateurs indépendants et les utilisateurs équidistants dans le plus petit des deux groupes. Si les deux groupes ainsi obtenus sont de tailles différentes, nous prenons les utilisateurs les plus éloignés du centre du plus gros *cluster* pour les mettre dans le plus petit *cluster* jusqu'à équilibre.

$$v(ct_{t+1}, i_k) = \alpha . \sum_{Y_{ct_t,u}} (v(u, i_k).|f_{simil}(ct_t, u)|) \tag{4.1}$$

Avec u un utilisateur lambda appartenant temporairement à ce *cluster* ;
$v(ct_{t+1}, i_k)$ la coordonnée du nouveau centre du *cluster* ct_{t+1} pour l'item i_k ;
$v(u, i_k)$ le vote de l'utilisateur u pour l'item i_k ;
$f_{simil}(ct_t, u)$ la similarité entre le centre précédent ct_t et u ;
$Y_{ct_t,u} = \{u | f_{simil}(ct_t, u) \neq 0\}$;
$\alpha = \frac{1}{\sum_{Y_{ct_t,u}} |f_{simil}(ct_t,u)|}$.

Une fois cette première subdivision effectuée, l'opération est renouvelée sur chacun des deux sous-groupes obtenus jusqu'à atteindre la profondeur de l'arbre p souhaitée. Plus on descend dans la structure et plus les *clusters* sont spécifiques à un certain groupe d'utilisateurs similaires. Par conséquent, plus on parcourt l'arbre en profondeur, plus les individus partagent le même avis concernant l'attribution d'une certaine note à un item donné. Nous pouvons fixer la profondeur de l'arbre soit en fonction du nombre de communautés souhaitées, soit en fonction du nombre d'utilisateurs maximum dans chaque communauté.

Au final, les feuilles de l'arbre représentent les 2^p communautés d'intérêts virtuelles constituées $\{c_1, \ldots, c_{2^p}\}$. Chacune communauté c_i est associée à une sous-matrice M_{c_i} restreinte aux utilisateurs et aux ressources de la communauté. Nous utilisons cette matrice M_{c_i} pour calculer le barycentre de c_i. Il s'agit du profil type de c_i, sorte de super utilisateur représentant les préférences de tous les utilisateurs de la communauté.

Les 2^p profils types $\{\vec{c_1}, \ldots, \vec{c_{2^p}}\}$ sont ensuite envoyés périodiquement côté client (à chaque fois que l'algorithme `FRAC` se termine).

4.2.2 Phase de prédictions

A chaque réception de nouveaux profils types sur le poste client, le système cherche à identifier l'utilisateur courant à l'une des communautés déterminées précédemment. Il faut pour cela calculer la similarité entre le profil de l'utilisateur courant $\vec{u_a}$ et chacun des profils types. Cette

[41] Cette phase d'équilibrage est réalisée après stabilisation pour garantir la convergence.

opération est rendue possible dans la mesure où le profil $\vec{u_a}$ est stocké sur le poste client et est, de ce fait, accessible à tout instant. Au final, l'utilisateur courant appartient à la communauté c_a dont le profil type $\vec{c_a}$ est le plus similaire à ses propres votes.

La phase de prédictions consiste alors à exploiter les votes contenus dans le profil type $\vec{c_a}$. Nous appliquons la formule suivante pour tous les items i_j dont le vote est inexistant dans $\vec{u_a}$:

$$f_{predict}(u_a, i_j) = max(v_{min}, min(vc_a, i_j + (\overline{u_a} - \overline{c_a}), v_{max}) \tag{4.2}$$

Avec $\overline{u_a}$ la moyenne des votes de l'utilisateur courant ;
$\overline{c_a}$ la moyenne des votes contenus dans le profil type $\vec{c_a}$;
vc_a, i_j le vote contenu dans $\vec{c_a}$ et associé à i_j ;
$min(a, b)$ et $max(a, b)$ les fonctions retournant respectivement le minimum et le maximum entre chacun des deux arguments.

Il serait également possible d'exploiter l'ensemble des profils types dans le calcul des prédictions. Plutôt que d'associer l'utilisateur courant à une communauté, nous pourrions déterminer dans un premier temps la similarité entre son profil et chacun de ses profils types. Puis, nous estimerions par exemple un vote inconnu de $\vec{u_a}$ comme la moyenne des votes des profils types sur l'item considéré pondérée par la similarité précédemment calculée :

$$f_{predict}(u_a, i_j) = max(v_{min}, min(\frac{\sum_k vc_k, i_j \ \times \ f_{simil}(c_k, u_a)}{\sum_k |f_{simil}(c_k, u_a)|}, v_{max}) \tag{4.3}$$

Dans le reste de ce chapitre, nous utiliserons la première formule de prédictions, afin de considérer que l'utilisateur courant n'appartient qu'à une seule communauté. La deuxième formule est, quant à elle, assez proche des formules utilisées dans les algorithmes `Item-Item` (Sarwar et al., 2001) (cf. supra, 3.3.3 Item-Item, p.65) et Berkovsky et al. (2006) (cf. supra, 3.6.1 Système avec topologie de voisinage hiérarchique, p.71).

4.3 Résultats

4.3.1 Temps de calcul

Il s'est ensuite agi de vérifier que les temps de calcul de notre algorithme étaient compatibles avec un grand nombre d'utilisateurs et de ressources pour s'assurer du passage à l'échelle. Pour cela, nous avons étudié la complexité algorithmique de `FRAC`, ainsi que les temps de calcul réels.

La complexité algorithmique de la partie *clustering* sur le serveur est quasi-linéaire.

Preuve :
Le choix des centres initiaux nécessite $2n$ opérations, avec n utilisateurs, puisqu'il s'agit de calculer la similarité entre les n profils et le point m, puis la similarité entre le point a et les n profils. Par la suite, il s'agit d'appliquer récursivement l'algorithme des `2-means`. La complexité de l'algorithme `K-means` est en $0(k.t.n)$ avec t le nombre d'itérations nécessaire (ou fixé dans le cas où nous choisissons de borner cette valeur). Dans notre cas, la complexité est donc en $0(2.t.n)$. Le choix de nos centres initiaux fait que le nombre d'itérations nécessaire est très restreint. En effet, dès la deuxième itération, les centres sont déjà très proches des positions à atteindre pour avoir un état stable. Cela est dû au fait que les centres initiaux sont nécessairement dans des groupes différents et sont les plus éloignés, et donc que la plupart des utilisateurs sont répartis dans les bons groupes à l'exception de quelques utilisateurs proches de la médiatrice du segment $[a - b]$. Nous pouvons donc considérer que le terme $(2.t)$ est fixe et que la complexité est linéaire lorsqu'il s'agit de répartir n utilisateurs en 2 groupes. De même, pour une profondeur de l'arbre donné, répartir t sous-ensembles d'utilisateurs en (2.t) sous-populations est également en $0(n)$. En effet, dans notre cas, la complexité $O(n_1 + n_2 + \ldots + n_x) = O(n)$ puisque les sous-populations sont disjointes et que l'union de leurs membres forme la population globale du système. Notre algorithme construit par ailleurs $(n/b) = 2^p$ communautés, avec b le nombre d'utilisateurs dans chaque communauté. La profondeur de l'arbre p est donc égale à $log_2(n/b)$. Par conséquent, la complexité globale pour la construction de l'arbre est en $O(n.log_2(n/b))$.

La complexité algorithmique de la partie client est, pour sa part, constante en $O(2^p)$, puisqu'il s'agit de comparer le profil de l'utilisateur courant avec les 2^p profils types. Cela permet d'avoir des temps de réponse instantanés côté client.

Nous avons également mesuré les temps de calcul réels de `FRAC` côté serveur et les avons comparé aux temps de calcul de `CorrCF` et d'`Item-Item` sur des populations simulées. Nous avons utilisé notre outil de génération de corpus (cf. supra, p.40) de sorte que les utilisateurs soient tous assez proches du milieu de l'espace de représentation et avec seulement 1% de votes manquants. Cette situation est, de ce fait, proche du pire cas que peut rencontrer notre algorithme. Les résultats sont présentés dans le tableau 4.2.

Items	100			150			1000		
Utilisateurs	FRAC	CorrCF	Item-Item	FRAC	CorrCF	Item-Item	FRAC	CorrCF	Item-Item
200	0”70	2”60	2”14	1”35	3”17	2”71	4”78	11”09	52”74
400	1”84	6”09	3”87	2”09	7”62	5”29	8”58	32”24	1’22”
600	3”60	11”78	5”59	4”10	15”21	7”34	15”43	1’04”	2’05”
800	7”03	19”98	7”23	7”34	25”67	10”53	30”17	1’52”	2’33”
1.000	8”30	30”22	8”56	9”45	40”68	12”84	39”71	3’06”	3’25”
1.400	11”21	1’00”	11”50	12”81	1’17”	18”10	49”47	6’04”	4’29”
10.000	**6’50”**	**7 :30’**	**1’22”**	9’12”	-	2’05”	**14’22”**	-	**49’28”**

Tab. 4.2 – Temps de calcul des différents algorithmes de filtrage collaboratif.

Parmi les résultats significatifs, notons que `FRAC` est capable de constituer les communautés, dont le nombre a été arbitrairement fixée à 10, et les profils types en moins de 7 minutes pour 10.000 utilisateurs et 100 ressources. En pareille situation, l'algorithme `CorrCF` a nécessité 7 heures et demi de calcul. L'algorithme `Item-Item` a, quant à lui, obtenu des résultats en 1 minutes et 22 secondes.

Pour 10.000 utilisateurs et 1000 items, la tendance s'inverse : `FRAC` réalise ses calculs en 14 minutes et 22 secondes, contre presque 50 minutes pour `Item-Item`. Pour 120.000 utilisateurs et 150 items, l'algorithme `FRAC` requiert près de 11 heures de calcul.

Les temps de calcul réels de `RecTree` n'ont pu être déterminés, car cet algorithme ne convergeait quasiment jamais.

4.3.2 Qualité des prédictions

La première étape dans la validation de notre modèle consiste à vérifier qu'il génère des recommandations de bonne qualité. Pour ce faire, nous avons utilisé le corpus `MovieLens` de 100.000 votes présenté dans le chapitre 2 (cf. supra, 2.2.1 GroupLens, p.37). Nous avons calculé la mesure d'erreur `MAE` obtenue en moyenne sur les 5 jeux de tests par notre algorithme, ainsi que par l'algorithme `Item-Item` (Sarwar et al., 2001) (cf. supra, p.65), l'algorithme basé sur la mémoire `CorrCF` (Resnick et al., 1994) (cf. supra, p.53) et l'algorithme `RecTree` (Chee et al., 2001) (cf. supra, p.64).

Les résultats sont affichés dans la figure 4.4 et montrent que notre algorithme sont comparables à l'algorithme `CorrCF`, ce qui témoigne d'une bonne qualité de prédictions.

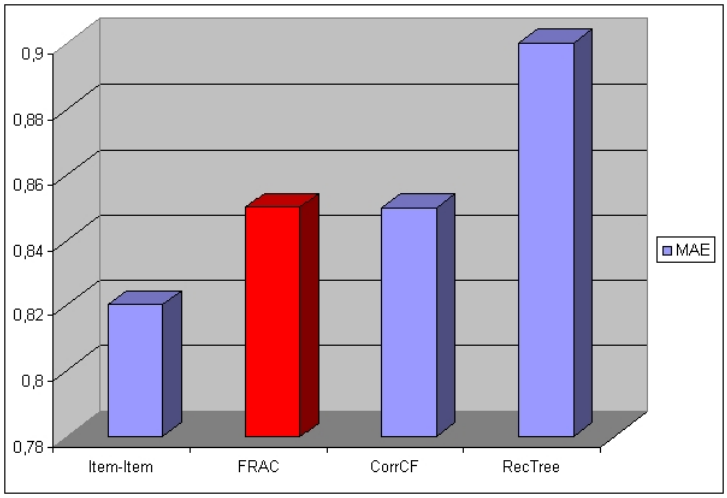


FIG. 4.4 – Erreur moyenne pour les différents algorithmes.

4.4 Discussion sur FRAC

Le modèle `FRAC+` a été imaginé dans le but de répartir les calculs entre le serveur et les postes clients. Cette décentralisation du filtrage collaboratif présente deux avantages majeurs :

- elle permet aux utilisateurs ne souhaitant pas partager leurs préférences de bénéficier du service de personnalisation, puisque son profil est tout de même construit localement et qu'il reçoit les profils types des communautés sur son poste ;
- elle permet d'autre part de réduire les temps de calcul en dissociant la phase de constitution des communautés d'intérêts virtuelles de la phase de prédictions. Le système est capable de fournir des réponses en temps réel aux utilisateurs, car la phase de calcul des prédictions côté client est extrêmement rapide. Les temps de calcul côté serveur sont, quant à eux, allégés bien qu'encore trop lourds pour être effectués en temps réel. Il est donc nécessaire de réaliser ces calculs hors ligne et de relancer périodiquement l'algorithme pour actualiser les communautés.

La décentralisation des calculs n'est pas le seul facteur permettant de réduire les temps de calculs, côté serveur. Notre algorithme de *clustering* `FRAC` utilise récursivement l'algorithme des plus proches voisins avec une méthode d'initialisation réduisant considérant le temps de calcul et garantissant la convergence. Par convergence, nous entendons ici que l'algorithme terminera à coup sûr tout en fournissant un optimum local (cf. infra, preuve de convergence de l'algorithme `FRAC`), contrairement à l'algorithme `RecTree`.

Preuve de convergence de l'algorithme `FRAC` :

- Quelles que soient les positions des utilisateurs dans l'espace de représentation $R_{u/r}$, il y aura toujours un point m correspondant au milieu.
- De même, à condition d'avoir au moins 2 utilisateurs dans l'espace, il y aura toujours un point a et un point b, tels que a est le moins similaire à m et b le moins similaire à a. Ces deux points sont donc les plus espacés dans l'espace de représentation, suivant cette métrique de similarité et seront donc nécessairement dans 2 groupes différents. Avec cette méthode d'initialisation, les points a et b obtenus seront toujours les mêmes si nous exécutons plusieurs fois l'application. Il peut éventuellement y avoir plusieurs candidats (similarités égales) pour ces centres initiaux. La reproductibilité des résultats passe alors par une méthode de tri : les utilisateurs pouvant être sélectionnés comme point a sont par exemple triés par identifiant croissant et nous prenons systématiquement le premier de la liste.

Preuve de convergence de l'algorithme `FRAC` :

- Une fois les points a et b choisis, l'algorithme est chargé de répartir les utilisateurs dans le groupe de centre provisoire le plus proche. Les points problématiques pouvant nuire à la convergence du système sont alors les utilisateurs dont les similarités avec les deux centres sont nulles ou égales (utilisateurs indépendants ou équidistants). Ces derniers sont donc écartés du calcul lors de la vérification de la stabilité de l'état. Ils sont ensuite réinjectés de façon à ce que les deux groupes aient des tailles équivalentes.
- Le fait de positionner un utilisateur dans une autre communauté que celle dont il est le plus proche lors de la phase de *clustering* afin d'avoir des groupes de taille égale n'est pas pénalisant, puisque cet utilisateur bénéficiera tout de même des recommandations du groupe le plus proche lors de la phase de prédictions. En revanche, ce mode de répartition permettra d'introduire un peu de nouveauté dans les groupes en ajoutant en petite proportion les items consultés par certains des utilisateurs les moins éloignés des membres effectifs des communautés.

L'algorithme `FRAC` converge donc systématiquement et présente l'avantage de proposer des résultats reproductibles. En effet, le choix des centres initiaux étant déterministe, nous sommes garantis de créer les mêmes communautés si nous exécutons plusieurs fois le programme avec le même corpus. La reproductibilité des résultats était un point qui faisait défaut aux algorithmes `RecTree` et `K-Means`, et globalement à la plupart des algorithmes de *clustering* basés sur une initialisation aléatoire. Toutefois, à l'instar de l'algorithme des `K-Means` (Wu et al., 2007), la solution obtenue par `FRAC` n'est pas nécessairement un optimum global, en particulier dans les cas où les communautés des utilisateurs ne sont pas clairement séparées dans l'espace de représentation, ou lorsque celles-ci s'entremêlent (cf.figure 4.5). Ce nonobstant, nos expérimentations montrent que notre algorithme maintient une bonne qualité de recommandations, équivalente à celle obtenue par l'algorithme `CorrCF`. La qualité des prédictions est un peu moins bonne que l'algorithme `Item-Item`. Cependant, il faut préciser que le corpus `MovieLens` dispose de plus d'items que d'utilisateurs et favorise donc les algorithmes basés sur les items. La différence de `MAE` est également due au fait que `FRAC` utilise des profils types, plutôt des sous-matrices pour générer ses prédictions. Ce choix pénalise la qualité des recommandations, mais est justifié par le fait qu'il serait trop lourd d'envoyer des matrices côté client et que cela serait moins respectueux de la vie privée des utilisateurs. En effet, ces derniers pourraient alors consulter les sous-matrices, même si les profils sont transmis sous le couvert de l'anonymat.

Les temps de calcul côté serveur montrent que notre algorithme est adapté à des corpus de grande taille. Il permet de s'atteler au problème du passage à l'échelle en gérant plusieurs dizaines voire centaines de milliers d'utilisateurs dans des temps raisonnables. Nos expérimentations ont mis en exergue le fait que `FRAC` termine ses calculs beaucoup plus rapidement que `CorrCF`. L'algorithme `Item-Item` reste plus rapide que `FRAC` pour 10.000 utilisateurs et 100 ressources. Nous constatons toutefois une inversion de tendance, dès lors que le nombre d'items augmente. Ainsi, pour 10.000 utilisateurs et 1.000 ressources, notre algorithme est déjà 3,5 fois plus rapide. Cet écart aurait pu être encore plus important si nous avions exploité le caractère parallélisable de nos calculs. En effet, une fois la première subdivision en deux sous-populations effectuée, il est aisé de répartir les calculs sur deux serveurs lors des appels récursifs suivants. Puis, plus nous

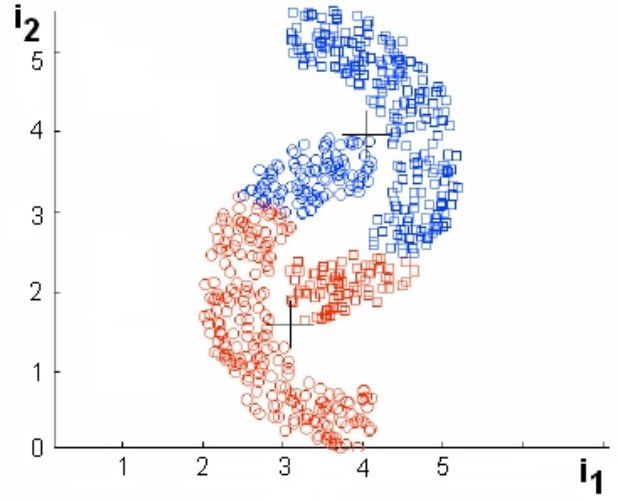


FIG. 4.5 – Exemple de répartition des utilisateurs difficile à identifier avec un algorithme de clustering.

descendons dans l'arbre de la figure 4.2, plus nous pouvons utiliser de serveurs de calcul. Au final, nous pouvons utiliser jusqu'à $\frac{\text{nombre de communautés}}{2} = 2^{p-1}$ serveurs. En outre, notre algorithme a été conçu pour sauvegarder régulièrement dans un fichier externe l'état d'avancement de ses calculs. Ainsi, il stocke à tout instant et pour chaque sous-population, la liste des identifiants des utilisateurs, la position des centres temporaires et le fait qu'il ait atteint un état stable ou non. Cela permet à l'algorithme de reprendre là où il en était en cas d'interruption volontaire ou involontaire.

Notons enfin que notre modèle repose sur la méthode générique de modélisation des préférences présentée dans le chapitre 1. Ainsi, l'emploi de critères implicites et explicites nous permet de réduire partiellement le nombre de données manquantes dans les profils, tout en veillant rigoureusement à respecter la vie privée des usagers.

4.5 Optimisation de la qualité des recommandations

4.5.1 Le modèle FSB

Principe général

Comme précisé dans la section précédente, l'emploi de profils types plutôt que de matrices de votes lors du calcul des prédictions réduit la qualité des recommandations. Au demeurant, il serait trop lourd d'envoyer systématiquement les matrices des communautés M_{c_i} côté client en raison de leur taille. Par ailleurs, cela présenterait le risque d'altérer la contrainte de respect de la vie privée, dans la mesure où tout un chacun pourrait consulter les votes d'autrui. Ce constat est atténué par le fait que chaque profil est anonyme. Toutefois, un utilisateur mal intentionné pourrait chercher à récupérer les identifiants des autres utilisateurs se connectant sur le même

ordinateur que lui et n'auraient alors plus qu'à parcourir la matrice à la recherche des lignes associées à ces identifiants.

Nous avons donc cherché un meilleur moyen de représenter les préférences des membres d'une communauté. Il s'agissait de trouver une forme plus compacte qu'une matrice permettant des envois à travers le réseau, respectueuse de la vie privée des usagers et préservant la richesse des informations contenues dans M_{c_i}. Notre modèle `FSB`(Castagnos et al., 2005b) répond à ces impératifs. Il permet de représenter les préférences d'une communauté sous forme d'un réseau bayésien, comme illustré sur la figure 4.6. Comme dans (Breese et al., 1998) (cf. supra, 3.3.1 Arbres de décision et réseaux bayésiens, p.59), les nœuds du réseau bayésien représentent les ressources et les arcs correspondent aux dépendances identifiées entre ces ressources. Le réseau bayésien de la communauté de u_a est ensuite utilisé côté client pour générer les recommandations, en lieu et place du profil type.

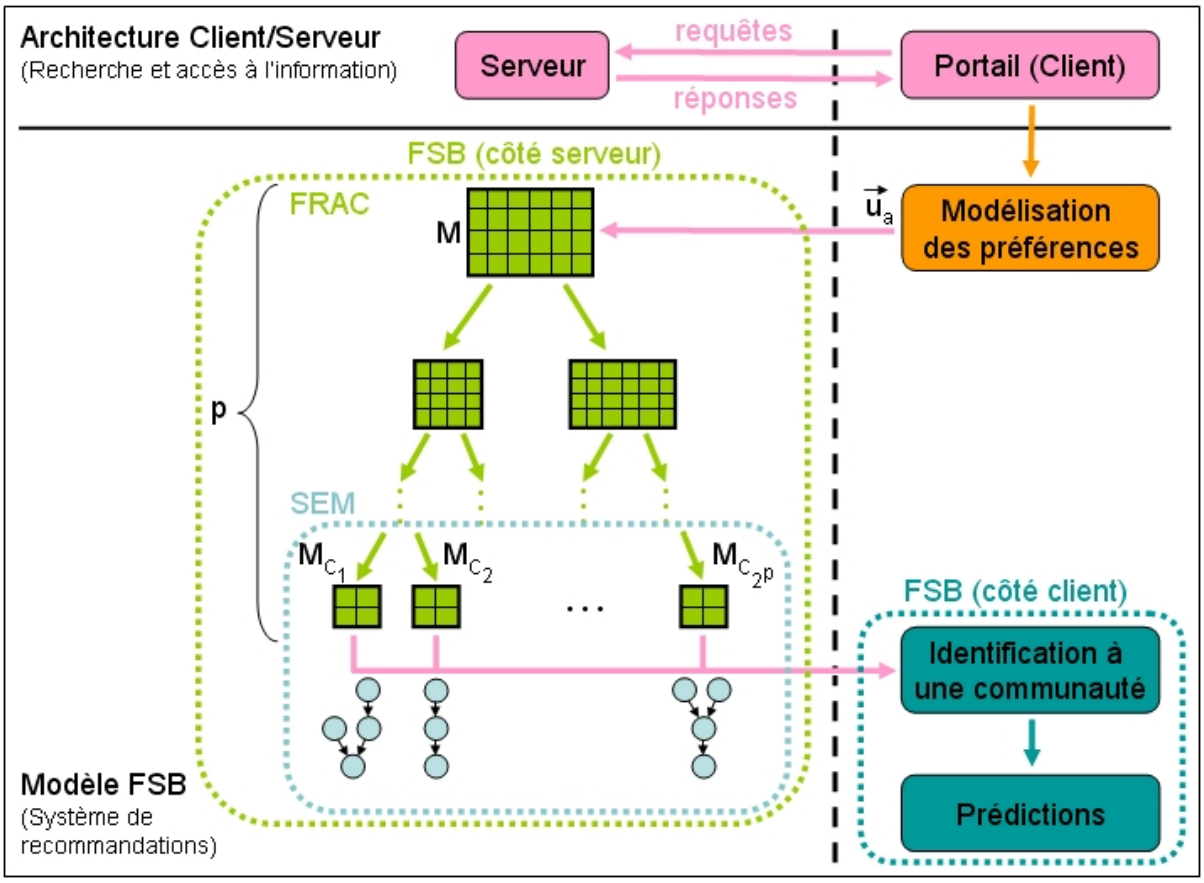


Fig. 4.6 – Modèle FSB.

`FSB` est l'acronyme de `FRAC`, `SEM`, réseaux **B**ayésiens. En effet, cet algorithme consiste dans un premier temps à constituer les communautés à l'aide de l'algorithme `FRAC`. Puis, l'algorithme d'apprentissage des structures des réseaux bayésiens `SEM` (Friedman, 1998) se charge de générer un réseau bayésien pour chaque communauté. Enfin, ces réseaux bayésiens sont envoyés et exploités côté client.

La section suivante a pour objectif de justifier notre choix concernant l'algorithme d'apprentissage des réseaux bayésiens. Nous présenterons ensuite un exemple de fonctionnement et discuterons des avantages et inconvénients du modèle `FSB`.

Choix de l'algorithme d'apprentissage des réseaux bayésiens

Nous avons vu précédemment (cf. supra, Arbres de décision et réseaux bayésiens, p.59) qu'un réseau bayésien est constitué à la fois d'un graphe (aspect qualitatif) et d'un ensemble de probabilités conditionnelles (aspect quantitatif). L'apprentissage d'un réseau bayésien se décompose donc en deux sous-problèmes (Naïm et al., 2004) :

- l'apprentissage des paramètres, étape au cours de laquelle on suppose que la structure du réseau a été fixée et où il faut estimer les probabilités conditionnelles de chaque nœud du réseau ;
- l'apprentissage de la structure, où le but est de trouver le meilleur graphe représentant la tâche à résoudre.

Nous nous situons dans le contexte d'un apprentissage à partir de données incomplètes, à savoir les votes contenus dans les profils utilisateurs.

Apprentissage des paramètres. Lors de la phase d'apprentissage des paramètres, nous cherchons à estimer les distributions de probabilités à partir des données disponibles. Dans le cas où toutes les variables sont observées[42], cette tâche peut par exemple être réalisée par apprentissage statistique.

L'apprentissage statistique consiste à estimer la probabilité d'un événement par la fréquence d'apparition de l'événement dans la base de données. Cette approche, appelée maximum de vraisemblance (MV), nous donne alors :

$$\hat{p}(X_i = x_k \mid pa(X_i) = x_j) \;=\; \hat{\theta}^{MV}_{i,j,k} \;=\; \frac{N_{i,j,k}}{\sum_k N_{i,j,k}} \tag{4.4}$$

où $N_{i,j,k}$ est le nombre d'événements dans la base de données pour lesquels la variable X_i est dans l'état x_k et ses parents dans la configuration x_j. De même, $\theta_{i,j,k}$ désigne la probabilité pour que X_i soit dans l'état x_k, conditionnellement au fait que l'ensemble de ses parents soit dans l'état x_j. L'accent circonflexe souligne le fait que nous cherchons à estimer cette probabilité.

Toutefois, cette estimation n'est valable que si les variables sont entièrement observées. Dans les applications pratiques, les bases de données sont très souvent incomplètes. Certaines variables ne sont observées que partiellement ou même jamais. L'approche la plus couramment utilisée dans ce cas là est l'algorithme *Expectation-Maximization* (`EM`) présenté dans (Dempster et al., 1977). Il s'agit d'un algorithme procédant par itérations successives. Dans ce contexte, nous noterons $X_v = \{X_v^{(l)}\}_{l=1...N}$ l'ensemble des N données observées et $\theta^{(t)} = \{\theta^{(t)}_{i,j,k}\}$ les paramètres

[42] Dans notre cas, cela signifierait de disposer d'une matrice de votes `M` entièrement pleine. Autrement dit, il faudrait connaître l'intégralité des votes $v(u_j, i_k), \forall u_j \in U et i_k \in I$.

du réseau bayésien à l'itération t. L'algorithme `EM` s'applique à la recherche des paramètres en répétant jusqu'à convergence les deux étapes *Espérance* et *Maximisation* présentées ci-dessous :

- Espérance : estimation des $N_{i,j,k}$ manquants en calculant leur moyenne conditionnellement aux données et aux paramètres courants du réseau (cf. équation 4.5). Cette étape revient à réaliser une série d'inférences en utilisant les paramètres courant du réseau, puis à remplacer les valeurs manquantes par les probabilités obtenues par inférence ;

$$N^*_{i,j,k} \;=\; E[N_{i,j,k}] \;=\; \sum_{l=1}^{N} p(X_i = x_k \mid pa(X_i) = x_j, X_v^{(l)}, \theta^{(t)}) \tag{4.5}$$

- Maximisation : en remplaçant les $N_{i,j,k}$ manquants par leur valeur moyenne calculée précédemment, il devient possible de calculer de nouveaux paramètres $\theta^{(t+1)}$ par maximum de vraisemblance (cf. équation 4.6).

$$\theta^{(t+1)}_{i,j,k} \;=\; \frac{N^*_{i,j,k}}{\sum_k N^*_{i,j,k}} \tag{4.6}$$

Au final, le pseudo-code correspondant à l'algorithme `EM` est le suivant :

Algorithme EM

Initialiser $\theta^{(0)}$

$t \leftarrow 0$

Répéter

$t \leftarrow t+1$

$N^*_{i,j,k} \leftarrow \sum_{l=1}^{N} p(X_i = x_k \mid pa(X_i) = x_j, X_v^{(l)}, \theta^{(t-1)})$

$\theta^{(t+1)}_{i,j,k} \leftarrow \frac{N^*_{i,j,k}}{\sum_k N^*_{i,j,k}}$

Tant que $|\theta^{(t)} - \theta^{(t-1)}|$ (critère d'arrêt)

TAB. 4.3 – Algorithme EM.

Apprentissage de la structure. François et Leray (2003) ont réalisé une étude comparative de différents algorithmes d'apprentissage de la structure de réseaux bayésiens. Lors de leurs expérimentations, ils ont utilisé les réseaux bayésiens `ASIA` (Lauritzen et Speigelhalter, 1988) et `INSURANCE`[43] dont la structure était déjà connue (cf. figure 4.7). A partir de ces réseaux bayésiens, ils ont généré plusieurs bases de données de différentes tailles pour tester l'influence du nombre d'exemples sur les différents algorithmes d'apprentissage. Ils ont ensuite cherché à retrouver les structures des réseaux à partir des bases ainsi générées. Le critère choisi pour comparer les structures apprises et les structures réelles était la distance d'édition, c'est-à-dire le nombre d'opérations nécessaires pour transformer le graphe obtenu en celui d'origine (l'ajout, le renversement ou le retrait d'un arc augmente de 1 la distance d'édition). Ils ont également supprimé aléatoirement 20% des valeurs des bases de données pour tester la gestion des données manquantes.

Nous nous sommes appuyés sur cette étude pour choisir une solution adaptée à notre situation parmi les algorithmes suivants :

[43] http ://www.cs.huji.ac.il/labs/compbio/Repository/networks.html

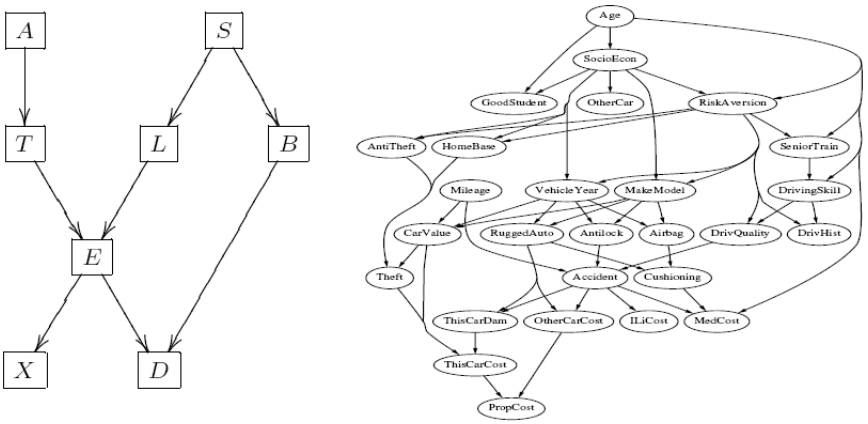


FIG. 4.7 – Structures originales des réseaux bayésiens `ASIA` (à gauche) et `INSURANCE` (à droite).

– Algorithme `PC` (Spirtes et al., 2000) ;

  Spirtes et al. (2000) ont mis au point cet algorithme en 1993 (cf. tableau 4.4). Un graphe complètement connecté sert de point de départ. Puis un test statistique est utilisé pour évaluer s'il y a indépendance conditionnelle entre chaque couple de variables. Si c'est le cas, l'arc correspondant est retiré. Cette technique permet d'obtenir de bons résultats et parvient même à trouver les dépendances faibles si la base de données est suffisamment grande. Toutefois, elle requiert un grand nombre de tests d'indépendance conditionnelle.

– Algorithme `MWST` (Chow et Liu, 1968) ;

  L'algorithme *Minimum Weight Spanning Tree* (`MWST`) est, comme son nom l'indique, dérivé de la recherche de l'arbre de recouvrement de poids minimal (cf. tableau 4.5). Cette méthode s'applique à la recherche de structure d'un réseau bayésien en fixant un poids W à chaque arête potentielle A-B de l'arbre. Ce poids peut par exemple correspondre à la variation du score local lorsqu'on choisit B comme parent de A : $W(X_A, X_B) = score(X_A, X_B) - score(X_A, \emptyset)$, où $score(X_A, X_B)$ est le score local en X_A en supposant que X_B est son parent et $score(X_A, \emptyset)$ le score en supposant que X_A n'a pas de parent. Une fois la matrice de poids définie, il suffit d'utiliser un des algorithmes standards de résolution du problème de l'arbre de poids minimum. On obtient un arbre devant être dirigé : il faut donc choisir une racine et faire un parcours en profondeur. Cet algorithme est peu sensible à la variation de la taille de la base de données. Par ailleurs, il permet d'obtenir des résultats très proches des structures initiales en dépit du fait qu'il parcourt un espace plus pauvre (les arbres n'ont ni V-structures $X_A \rightarrow X_C \leftarrow X_B$, ni cycles).

– Algorithme `K2` (Cooper et Hersovits, 1992) ;

Algorithme PC

Construction d'un graphe non orienté

Soit G le graphe reliant complètement tous les nœuds X

$i \leftarrow 0$

Répéter

Recherche des indépendances conditionnelles d'ordre i

$\forall \{X_A, X_B\} \in X^2$ tels que $X_A - X_B$ et $Card(Adj(G, X_A, X_B)) \geq i$

$\forall S \subset Adj(G, X_A, X_B)$ tel que $Card(S) = i$

si $X_A \perp X_B \mid S$ alors

suppression de l'arête $X_A - X_B$ dans G

$SepSet(X_A, X_B) \leftarrow SepSet(X_A, X_B) \cup S$

$SepSet(X_B, X_A) \leftarrow SepSet(X_B, X_A) \cup S$

$i \leftarrow i + 1$

Jusqu'à $Card(Adj(G, X_A, X_B)) < i, \forall \{X_A, X_B\} \in X^2$

Recherche des V-structures

$\forall \{X_A, X_B, X_C\} \in X^3$ tels que $\neg\overline{X_A X_B}$ et $X_A - X_B - X_C$,

si $X_C \notin SepSet(X_A, X_B)$ alors on crée une V-structure :

$X_A \rightarrow X_C \rightarrow X_B$

Ajout récursif de $\rightarrow$

Répéter

$\forall \{X_A, X_B\} \in X^2$,

si $X_A - -X_B$ et $X_A \rightsquigarrow X_B$, alors ajout d'une flèche à X_B :

$X_A \rightarrow X_B$

si $\neg\overline{X_A X_B}, \forall X_C$ tel que $X_A \rightarrow X_C$ et $X_C - X_B$ alors $X_C \rightarrow X_B$

Tant qu'il est possible d'orienter des arêtes

Notations :

X	ensemble de tous les nœuds
$Adj(G, X_A)$	ensemble des nœuds adjacents à X_A dans G
$Adj(G, X_A, X_B)$	$Adj(G, X_A) \setminus \{X_B\}$
$X_A - X_B$	il existe une arête entre X_A et X_B
$X_A \rightarrow X_B$	il existe un arc de X_A vers X_B
$\overline{X_A X_B}$	X_A et X_B sont adjacents $X_A - X_B$, $X_A \rightarrow X_B$ ou $X_B \rightarrow X_A$
$X_A \rightsquigarrow X_B$	il existe un chemin dirigé reliant X_A et X_B
$X_A \perp X_B$	indépendance d'ordre 0
$X_A \perp X_B \mid X_C$	indépendance d'ordre 1

TAB. 4.4 – Algorithme PC.

Le principe de l'algorithme `K2` consiste à maximiser la probabilité de la structure connaissant les données $P(G|D\ ) = \frac{P(G,D)}{P(D)}$. Or, nous savons que $P(G, D) \propto P(G)P(D|G)$ avec P(G) la probabilité *a priori* affectée à la structure G. Supposons que le graphe est composé de m nœuds. Si X_i est un nœud, alors r_i est le nombre de ses états possibles et q_j est le nombre d'états possibles de ses parents. $N_{i,j}$ est le

Algorithme MWST

Construction de l'arbre optimal

$\forall X_i, T(X_i) = \{X_i\}$

$B^o \leftarrow \emptyset$

$\forall (X_i, X_j) \in A$

si $T(X_i) \neq T(X_j)$ alors

$B^o \leftarrow B^o \cup (X_i, X_j)$

$T' \leftarrow T(X_i) \cup T(X_j)$

$T(X_i) \leftarrow T'$

$T(X_j) \leftarrow T'$

Orientation des arêtes

$B \leftarrow \emptyset$

$\{pa(X_i)\} \leftarrow ParcoursProfondeur(B^o, X_r)$

$\forall X_i,$

si $pa(X_i) \neq \emptyset$ alors ajout de $pa(X_i) \rightarrow X_i$ dans B

Notations :

A	liste des arêtes (X_i, X_j) dans l'ordre décroissant des W
$T(X_i)$	arbre passant par le nœud X_i
X_r	racine choisie pour orienter l'arbre
$pa(X_i)$	parent du nœud X_i
B^o	arbre optimal non orienté
B	structure finale obtenue par l'algorithme

TAB. 4.5 – Algorithme MWST.

nombre d'exemples dans la base D, tels que les parents du nœud X_i sont dans l'état x_j, indépendamment de l'état de X_i. On a alors :

$$P(D|G) = \prod_{i=1}^{m} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}! \tag{4.7}$$

En supposant un *a priori* uniforme sur les structures, la qualité d'un jeu de parents pour un nœud fixé pourra donc être mesurée par le score local de l'équation 4.8.

$$s(X_i, pa(X_i)) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}! \tag{4.8}$$

Ensuite, en imposant un ordre sur les nœuds de manière à ce qu'un nœud ne puisse être parent d'un autre que s'il précède celui-ci dans cet ordre (si X_i est avant X_j alors il ne pourra y avoir d'arc de X_j vers X_i), il est alors possible de réduire l'espace de recherche. L'algorithme `K2` teste l'ajout de parents en respectant cet ordre. Le premier nœud ne peut pas posséder de parents. Pour les nœuds suivants, il faut choisir l'ensemble de parents augmentant le plus le score bayésien parmi les parents possibles. Le pseudo-code de cet algorithme est présenté dans le tableau 4.6.

Algorithme K2

Pour i = 1 à m
- $pa(X_i) \leftarrow \emptyset$
- $s_{old} \leftarrow s(X_i, pa(X_i))$
- $OK \leftarrow vrai$
- Répéter
  - Chercher $X_j \in Pred(X_i) \setminus pa(X_i)$ qui maximise $s(X_i, pa(X_i) \cup \{X_i\})$
  - $s_{new} \leftarrow s(X_i, pa(X_i) \cup \{X_i\})$
  - si $s_{new} > s_{old}$ alors
    - $s_{old} \leftarrow s_{new}$
    - $pa(X_i) \leftarrow pa(X_i) \cup \{X_i\}$
  - sinon $OK \leftarrow faux$
- Tant que OK et $|pa(X_i)| < bs$

Notations :

$Pred()$	relation d'ordre sur les nœuds X_i
bs	borne supérieure du nombre de parents possibles pour un nœud
$s(X_i, pa(X_i))$	score local défini dans l'équation 4.8
$pa(X_i)$	ensemble des parents du nœud X_i

TAB. 4.6 – Algorithme K2.

Cet algorithme est très rapide, mais est trop sensible à l'initialisation. `K2` dépend fortement de l'ordre d'énumération passé en paramètre.

– Algorithme `GS` (Chickering et al., 1995) ;

L'algorithme de recherche gloutonne, ou `GS` comme *Greedy Search*, prend un graphe initial et définit un voisinage. Puis, il associe un score à chaque graphe du voisinage. Le meilleur d'entre eux est alors choisi comme point de départ de l'itération suivante. Puisque cet algorithme va calculer le score de graphes voisins, il faut utiliser un score décomposable[44] pour n'avoir à calculer que la variation de scores locaux (entraînée par la suppression ou l'ajout d'un arc) et non les deux scores globaux de ces deux graphes.

Dans ce cas, le voisinage d'un `DAG` est défini par l'ensemble de tous les graphes obtenus soit en ajoutant, soit en supprimant, soit en retournant un arc du graphe d'origine. L'algorithme s'arrête lorsque le graphe obtenu réalise un maximum local de la fonction de score (cf. tableau 4.7). Cette technique est robuste face à la variation

[44]Un score S est dit décomposable s'il peut être écrit comme une somme (ou un produit) de mesures qui sont fonction seulement du nœud et de ses parents. En clair, si m est le nombre de nœuds du graphe, le score doit avoir une des formes suivantes :

$$S(B) = \sum_{i=1}^{m} s(X_i, pa(X_i)) \text{ ou } S(B) = \prod_{i=1}^{m} s(X_i, pa(X_i)) \tag{4.9}$$

de la taille de la base de données. Toutefois, l'algorithme de recherche gloutonne est connu pour converger vers un optimum qui est souvent local et de mauvaise qualité.

Algorithme GS

Initialisation du graphe G
(Graphe vide, aléatoire ou arbre obtenu par MWST)
$Continuer \leftarrow vrai$
$score_{max} \leftarrow score(G)$
Répéter
Génération de V_G, voisinage de G, à l'aide d'opérateurs :
- Ajout d'arc, suppression d'arc, inversion d'arc
(les graphes ainsi obtenus doivent être acycliques)
Calcul du score pour chaque graphe de V_G
$G_{new} \leftarrow argmax_{G' \in V_G}(score(G'))$
si $score(G_{new}) \geq score_{max}$ alors
$score_{max} \leftarrow score(G_{new})$
$G \leftarrow G_{new}$
sinon
$Continuer \leftarrow faux$
Tant que $Continuer$

Notations :

$score()$	fonction de score sur les structures possibles
V_G	ensemble des DAG voisins du DAG G courant
G	structure finale obtenue par l'algorithme

TAB. 4.7 – Algorithme de recherche gloutonne (GS).

– Algorithme `SEM` (Friedman, 1998).

L'algorithme `Structural-EM` permet de traiter des bases d'exemples incomplètes. Il s'agit d'une méthode itérative dont la convergence a été prouvée par Friedman (1998). On part d'une structure initiale (pouvant être vide) pour estimer la distribution de probabilité des variables cachées ou manquantes grâce à l'algorithme `EM` classique (cf. supra, 4.5.1 Apprentissage des paramètres, p.92). L'étape suivante consiste à calculer l'espérance d'un score par rapport à ces variables cachées pour tous les réseaux bayésiens du voisinage (par ajout, suppression ou inversion d'un arc) afin de choisir la prochaine structure. Puis, on réitère l'opération. En résumé, `SEM` se décompose en deux phases répétées jusqu'à convergence de l'algorithme (cf.figure 4.8) :
– la phase `structurelle` consistant à sélectionner à chaque étape la meilleure structure $G^(b)$ maximisant le score utilisé dans l'ensemble $V_{G^(b)}$ des voisins de $G^(b)$. Le score d'une structure se calcule grâce au *Bayesian Information Criterion* (BIC Score) introduit dans (Schwarz, 1978; Friedman, 1997) :

Soit $G = (X, A)$ le graphe acyclique dirigé et $L(D|\theta)$ le critère de vraisemblance du réseau bayésien défini par l'algorithme EM. On a :

$$L(D|\theta) = \prod_{i=1}^{m} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{i,j,k}^{N_{i,j,k}} \tag{4.10}$$

Le score BIC s'écrit alors :

$$Score_{BIC} = -2 * log(L(D|\theta)) + log(|X|) \times \sum_{X_i \in X} 2^{nbparents(X_i)}$$

$$\text{Avec : } log(L(D|\theta)) = \sum_{i=1}^{m} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{i,j,k} \times log(\theta_{i,j,k}) \tag{4.11}$$

– la phase `EM` d'optimisation dans l'espace des paramètres $\theta^{(b,t)}$ à partir d'une structure $G^{(b)}$ fixée, où il faut trouver de manière itérative des paramètres $\theta^{b,t+i)}$ de plus en plus vraisemblables.

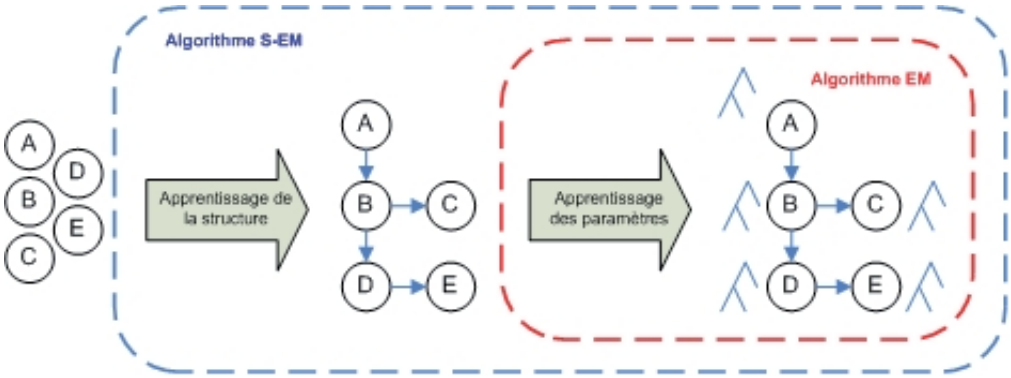


FIG. 4.8 – Construction du réseau bayésien avec SEM.

Dans le tableau 4.8, le système effectue à chaque itération t_{max} pas de l'algorithme `EM` puis un unique pas de recherche gloutonne. L'algorithme `SEM` permet d'obtenir des résultats très proches des véritables structures des réseaux bayésiens.

Au final, les algorithmes `MWST` et `SEM` sont les plus appropriés dans notre cas, puisqu'ils permettent d'obtenir de très bons résultats quelle que soit la taille de la base d'apprentissage. L'algorithme `SEM` présente l'avantage de gérer les données manquantes, ce qui est très intéressant dans notre contexte. Par conséquent, il s'agit de la méthode que nous avons choisi d'implanter.

Exemple de fonctionnement

Dans notre modèle, la mise en place de réseaux bayésiens intervient au niveau des communautés $\{c_1, \ldots, c_{2^p}\}$ déterminées par l'algorithme `FRAC`. Pour chaque communauté c_i, nous cherchons à déterminer le réseau bayésien qui explique le mieux les relations de dépendance entre les ressources votées par au moins un utilisateur de c_i. Les observations correspondent aux votes de la sous-matrice M_{c_i}. Afin de réduire la complexité de l'algorithme d'apprentissage `SEM`,

Algorithme SEM

Initialiser $\theta^{(0,0)}, G^0$

$b \leftarrow 0$

Répéter (itérations sur les structures)

$t \leftarrow 0$

Répéter (itérations sur les paramètres, avec la structure $G^{(b)}$)

$N^*_{i,j,k} \leftarrow \sum_{l=1}^{N} p(X_i = x_k \mid pa(X_i) = x_j, X_v^{(l)}, \theta^{(b,t)}, G^{(b)})$

$\theta^{(b,t+1)}_{i,j,k} \leftarrow \frac{N^*_{i,j,k}}{\sum_k N^*_{i,j,k}}$

$t \leftarrow t + 1$

Tant que $|\theta^{(b,t)} - \theta^{(b,t-1)}| \geq \epsilon$ ou $t < t_{max}$

Recherche d'une meilleure structure $G^{(b+1)}$

Génération de $V_{G^{(b)}}$, à l'aide d'opérateurs comme ajout d'arc, suppression d'arc, inversion d'arc (les graphes ainsi obtenus doivent être acycliques)

Calcul de $score(G')$ pour chaque graphe de $V_{G^{(b)}}$

$G_{new} \leftarrow argmax_{G' \in V_{G^{(b)}}}(score(G'))$

$b \leftarrow b + 1$

Tant que $score(G^{(b)}) \geq score(G^{(b-1)})$

$G^o \leftarrow G^{(b-1)}$

TAB. 4.8 – Algorithme SEM.

nous réduisons le nombre de valeurs possibles pour chaque variable en transformant la matrice M_{c_i} en une matrice booléenne MB_{c_i} suivant le principe énoncé dans la section 3.3.1 (cf supra, p.58). Nous utilisons les classes "Aime" et "Aime pas". Nous considérons qu'un utilisateur u_j aime une ressource i_k dès que lors que $v(u_j, i_k)$ est supérieur ou égal à un certain seuil. Dans le cas du corpus `MovieLens` où l'échelle de votes est comprise entre 1 et 5, ce seuil est fixé à 4. L'algorithme `SEM` se charge ensuite d'apprendre le réseau bayésien associé à c_i à partir des données contenues dans MB_{c_i}. Une fois les 2^p réseaux bayésiens appris, les structures et les paramètres sont envoyés côté client, en complément des profils types des communautés. Un calcul de distance permet d'associer l'utilisateur courant à une communauté. Par la suite, plutôt que d'utiliser la formule de prédiction de l'algorithme `FRAC`, on injecte les évidences (votes booléens connus de u_a) dans les tables de probabilités conditionnelles du réseau bayésien associé à la communauté de u_a. Cela nous permet de déterminer la probabilité que u_a aime telle ou telle ressource.

Afin d'illustrer le déroulement de la phase d'apprentissage des réseaux bayésiens, prenons un exemple simple où les données fournies M_{c_i} pour la communauté c_i sont les suivantes (les tirets représentent les données manquantes) :

La transformation en matrice booléenne MB_{c_i} donne :

Nous initialisons la structure du graphe $G^{(0)}$ en reliant les variables I_2 et I_4, correspondant aux items i_2 et i_4 (cf. figure 4.9).

Dans notre cas, chaque variable ne peut prendre que 2 valeurs possibles. Les paramètres à

TAB. 4.9 – Sous-matrice M_{c_i} de la communauté c_i.

	i_2	i_4
u_1	2	1
u_2	3	-
u_3	-	5

TAB. 4.10 – Sous-matrice booléenne MB_{c_i} de la communauté c_i.

	i_2	i_4
u_1 Aime	0	0
u_1 Aime pas	1	1
u_2 Aime	0	-
u_2 Aime pas	1	-
u_3 Aime	-	1
u_3 Aime pas	-	0

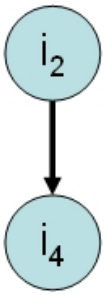


FIG. 4.9 – Initialisation de la structure du graphe ($G^{(0)}$).

estimer sont donc les suivants :

$$p(I_4 = 1 \mid I_2 = 0) = \theta_{(I_4=1 \mid I_2=0)} \quad \textbf{et} \quad p(I_4 = 0 \mid I_2 = 0) = 1 - \theta_{(I_4=1 \mid I_2=0)} \quad (4.12)$$

$$p(I_4 = 1 \mid I_2 = 1) = \theta_{(I_4=1 \mid I_2=1)} \quad \textbf{et} \quad p(I_4 = 0 \mid I_2 = 1) = 1 - \theta_{(I_4=1 \mid I_2=1)} \quad (4.13)$$

Concentrons-nous sur l'estimation des paramètres $\theta_{(I_4=1 \mid I_2=0)}$ et $\theta_{(I_4=1 \mid I_2=1)}$ avec l'algorithme EM.

Initialisation. Les valeurs initiales des paramètres sont :
$\theta^{(0)}_{(I_4=1 \mid I_2=0)} = 0,5$ et $\theta^{(0)}_{(I_4=1 \mid I_2=1)} = 0,5$.

Première itération. Le calcul de l'étape d'Espérance est résumée dans le tableau 4.11 (les valeurs sont obtenues par calcul des probabilités selon le modèle $\theta^{(0)}$).

TAB. 4.11 – Etape d'Espérance de l'algorithme EM (itération 1).

			$p(I_4 = 1 \mid I_2 = 0)$		$p(I_4 = 1 \mid I_2 = 1)$	
	i_2	i_4	$I_4 = 0$	$I_4 = 1$	$I_4 = 0$	$I_4 = 1$
u_1 Aime	0	0	1	0	0	0
u_1 Aime pas	1	1	0	0	0	1
u_2 Aime	0	-	0,5	0,5	0	0
u_2 Aime pas	1	-	0	0	0,5	0,5
u_3 Aime	-	1	0	1	0	1
u_3 Aime pas	-	0	1	0	1	0
N^*			2,5	1,5	1,5	2,5

L'étape de Maximisation nous donne :
$\theta^{(1)}_{(I_4=1 \mid I_2=0)} = \frac{1,5}{(2,5+1,5)} = 0,375$
et $\theta^{(1)}_{(I_4=1 \mid I_2=1)} = \frac{2,5}{(1,5+2,5)} = 0,625.$

Deuxième itération. Etape d'Espérance : L'étape de Maximisation nous donne :

TAB. 4.12 – Etape d'Espérance de l'algorithme EM (itération 2).

			$p(I_4 = 1 \mid I_2 = 0)$		$p(I_4 = 1 \mid I_2 = 1)$	
	i_2	i_4	$I_4 = 0$	$I_4 = 1$	$I_4 = 0$	$I_4 = 1$
u_1 Aime	0	0	1	0	0	0
u_1 Aime pas	1	1	0	0	0	1
u_2 Aime	0	-	0,625	0,375	0	0
u_2 Aime pas	1	-	0	0	0,375	0,625
u_3 Aime	-	1	0	1	0	1
u_3 Aime pas	-	0	1	0	1	0
N^*			2,625	1,375	1,375	2,625

$\theta^{(2)}_{(I_4=1 \mid I_2=0)} = \frac{1,375}{(2,625+1,375)} = 0,344$
et $\theta^{(2)}_{(I_4=1 \mid I_2=1)} = \frac{2,625}{(1,375+2,625)} = 0,656.$

Convergence Après quelques itérations de l'algorithme `EM`, les valeurs des paramètres convergent vers :
$\theta^{(t)}_{(I_4=1 \mid I_2=0)} = \frac{1,375}{(2,625+1,375)} = 1/3$
et $\theta^{(t)}_{(I_4=1 \mid I_2=1)} = \frac{2,625}{(1,375+2,625)} = 2/3.$

Evaluation de $G^{(0)}$. L'évaluation de la structure du graphe $G^{(0)}$ donne :

$$\begin{aligned} Score_{BIC} &= -2 * log(\sum_{j=0}^{1}\sum_{k=0}^{1} N_{i,j,k} \times log(\theta_{i,j,k})) + log(2) \times (2^1 + 2^0) \\ &= -2 * log(4/3) + 3 \times log(2) \\ &= 0,654 \end{aligned}$$

Génération du voisinage. Nous générons ensuite toutes les structures voisines possibles de $G^{(0)}$ en ajoutant un arc, supprimant un arc et inversant un arc. Cela nous donne les structures de la figure 4.10.

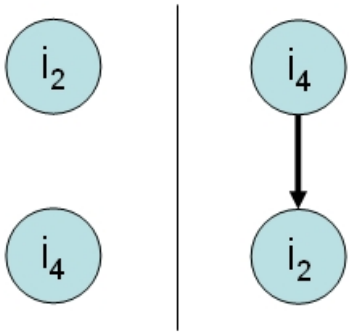


FIG. 4.10 – Voisinage $V_{G^{(0)}}$ de $G^{(0)}$.

Nous réitérons ensuite l'algorithme `EM` et évaluons le score `BIC` pour chacune de ces structures de $V_{G^{(0)}}$. S'il n'existe pas de graphe de $V_{G^{(0)}}$ avec un meilleur score que $G^{(0)}$ (état stable), alors l'algorithme se termine et la réseau bayésien appris est $B = (G^{(0)}, \theta^{(0,t)})$. Dans le cas contraire, le meilleur des graphes de $V_{G^{(0)}}$ devient $G^{(1)}$ et on réitère l'algorithme `SEM` jusqu'à obtenir un état stable.

Exploitation côté client. Ce réseau bayésien sera envoyé côté client, au même titre que les réseaux bayésiens des autres communautés. Supposons alors que u_a appartienne à la communauté pour laquelle nous avons déroulé l'algorithme `SEM` (le calcul de distance montre que le profil type de cette communauté est le plus similaire avec le profil de u_a) et que $G^{(0)}$ soit la structure obtenue avec `SEM`. Afin de déterminer l'intérêt de u_a pour i_4, on intègre les préférences connues de u_a pour les nœuds parents de i_4. Dans le cas présent, il s'agit de i_2. Si u_a aime i_2, alors il a une plus forte probabilité d'aimer i_4 ($p = 2/3$). Au final, nous pouvons classer la liste des recommandations par ordre décroissant des probabilités des items susceptibles d'être aimés par u_a.

Discussion sur FSB

L'objectif de cet algorithme est d'améliorer la qualité des recommandations de l'algorithme `FRAC` sans trop alourdir le trafic réseau, ni altérer la vie privée des utilisateurs. En effet, l'algo-

rithme `FRAC` perd un peu en qualité par rapport à un algorithme de clustering centralisé dans la mesure où il exploite le profil type de la communauté dans la formule de prédiction, en lieu et place des profils individuels des membres de cette communauté pondérés par leur distance avec u_a[45]. Ce choix se justifie de deux façons. D'une part, le calcul des prédictions se fait côté client et il sera coûteux en terme de trafic d'envoyer à chaque utilisateur l'intégralité des sous-matrices. D'autre part, si les utilisateurs recevaient ces sous-matrices sur leur machine, ils disposeraient alors des votes d'autrui. Cela serait contraire à la contrainte de respect de la vie privée que nous nous sommes fixés. La transformation de ces matrices sous la forme de réseaux bayésiens permet, quant à elle, de synthétiser efficacement l'intégralité de l'information contenue dans les M_{c_i} sans enfreindre la règle de confidentialité[46]. La taille d'un réseau bayésien reste raisonnable et l'état de l'art (Breese et al., 1998) prouve que cette approche garantit la bonne qualité des recommandations.

Au demeurant, la détermination de la structure d'un réseau bayésien à partir des données est un problème NP-difficile. Cette recherche n'est pas aisée car l'espace à parcourir est de taille super-exponentielle en fonction du nombre de variables. Ainsi, le nombre de structures différentes $r(m)$ pour un réseau bayésien possédant m nœuds vaut :

$$r(m) \;=\; \begin{cases} 1 & \text{si m = 0 ou 1} \\ \sum_{i=1}^{m} (-1)^{i+1} \binom{m}{i} 2^{i(m-1)} r(m-i) \;=\; m^{2^{O(m)}} & \text{si m > 1} \end{cases} \tag{4.14}$$

On a donc r(1)=1, r(2)=3, r(3)=25, r(5)=29281, r(10)=4.2x10^{18}, etc. Ceci explique que les méthodes d'apprentissage des réseaux bayésiens soient particulièrement coûteuses en temps. Cette solution n'est plus viable en terme de temps de calcul dès lors que le nombre de variables à prendre en compte, i.e. d'items votés dans la sous-matrice d'une communauté, dépasse 7 ou 8. On peut imaginer restreindre la liste des items à intégrer dans le réseau bayésien aux 8 plus populaires de la communauté, mais cela limiterait grandement l'intérêt de ce modèle. Pour cette raison, en dépit de l'intérêt que pouvait susciter au premier abord l'emploi de réseaux bayésiens, nous avons jugé que le modèle `FSB` n'était pas adéquat pour une utilisation en situation réelle. Nous avons alors cherché une alternative aux réseaux bayésiens et avons imaginé l'algorithme `RIBA`.

4.5.2 RIBA

Principe général

Pour concevoir ce modèle, nous sommes partis de deux constats :

– dans les réseaux bayésiens du modèle `FSB`, les arcs reliant deux items $i_j \rightarrow i_k$ présentent une faible valeur ajoutée par rapport au calcul de similarité $f_{simil\_item}(i_j, i_k)$ déterminant le degré de dépendance entre les deux variables. En revanche, certaines relations telles que les V-structures $i_j \rightarrow i_l \leftarrow i_k$ ne sont pas identifiables à partir d'une simple matrice `S` de similarité item-item et pourraient affiner les calculs de prédictions si elles étaient correctement prises en compte dans notre modèle ;

[45] Calculer la moyenne des profils des utilisateurs constituent en effet une perte d'information.

[46] Les tables de probabilités conditionnelles ne contiennent aucune information personnelle.

– un calcul de similarité s'effectue toujours entre deux variables. Or, il serait intéressant dans certains cas de mesurer les dépendances entre trois variables ou plus sur le même principe.

Le modèle `RIBA` est né de l'idée que les similarités entre deux items ne suffisent pas toujours à expliquer l'intérêt d'un utilisateur pour un item, mais que l'étude des dépendances au sein de triplets peut améliorer les recommandations du système (Castagnos et al., 2008b,a). Au moment de la rédaction de cette thèse, ce modèle n'a pas encore fait l'objet d'une intégration dans l'architecture `FRAC+` mais vise à valider cette hypothèse. Afin d'illustrer nos propos, nous pouvons considérer trois items i_j ="Cendrillon", i_k="Scary Movie" et i_l ="Shrek". En caricaturant, nous pouvons imaginer qu'un utilisateur du système adepte de contes de fées appréciera i_j, mais n'aimera pas i_l qui est une parodie de contes de fées. De même, un spectateur féru de parodies et de films d'horreur aimera i_k, mais votera négativement pour i_l qu'il jugera trop enfantin puisqu'il s'agit d'un film d'animation. Néanmoins, un utilisateur qui affectionnerait à la fois i_j et i_k, et de ce fait les contes de fées et les parodies, aurait de plus grandes chances de passer un bon moment devant la parodie de contes de fées i_l. En d'autres termes, nous cherchons à identifier les relations non triviales où i_j et i_k n'expliquent pas seuls l'attrait pour i_l, mais où l'intérêt conjugué pour i_j et i_k vient renforcer la probabilité d'aimer i_l.

Nous proposons alors de générer automatiquement des règles d'association portant sur des triplets d'items dans le but d'affiner certaines estimations lors de la phase de prédictions. Dans notre cas, une règle d'association s'exprime sous la forme de deux prémisses et d'une conclusion. La conclusion porte sur l'item pour lequel il faut affiner la prédiction. Nos règles d'association sont donc analogues aux V-structures des réseaux bayésiens. Bien évidemment, les règles d'associations peuvent s'intéresser aussi bien à l'intérêt des utilisateurs pour les prémisses et la conclusion, qu'à leur désintérêt. Notons $\mathring{I}_j$ le fait d'aimer i_j et $\overline{I_j}$ une aversion pour i_j. Nous considérons qu'un utilisateur u_i aime une ressource i_j dès lors que le vote $v(u_i, i_j)$ est supérieur ou égal à un seuil v_{aime}. Sur une échelle de votes de 1 à 5, ce seuil v_{aime} est fixé à 4. De même, nous estimons que u_i n'aime pas i_j lorsque $v(u_i, i_j) \leq v_{aime\_pas}$. v_{aime_pas} vaut 2 dans le cas de votes compris entre 1 et 5. Il existe alors potentiellement 8 règles d'association portant sur le triplet $< i_j, i_k, i_l >$ avec des prémisses portant sur $<i_j,i_k>$ et une conclusion concernant i_l :

$$< \mathring{I}_j, \mathring{I}_k > \Rightarrow \mathring{I}_l \tag{4.15}$$
$$< \mathring{I}_j, \overline{I_k} > \Rightarrow \mathring{I}_l \tag{4.16}$$
$$< \overline{I_j}, \mathring{I}_k > \Rightarrow \mathring{I}_l \tag{4.17}$$
$$< \overline{I_j}, \overline{I_k} > \Rightarrow \mathring{I}_l \tag{4.18}$$
$$< \mathring{I}_j, \mathring{I}_k > \Rightarrow \overline{I_l} \tag{4.19}$$
$$< \mathring{I}_j, \overline{I_k} > \Rightarrow \overline{I_l} \tag{4.20}$$
$$< \overline{I_j}, \mathring{I}_k > \Rightarrow \overline{I_l} \tag{4.21}$$
$$< \overline{I_j}, \overline{I_k} > \Rightarrow \overline{I_l} \tag{4.22}$$

Afin de générer automatiquement ces règles, nous avons considéré uniquement les triplets d'items pour lesquels les similarités deux à deux sont moyennes :

$$\begin{aligned} 0 < t_{min} \leq |f_{simil\_item}(i_j, i_l)| \leq t_{max} < 1 \\ 0 < t_{min} \leq |f_{simil\_item}(i_k, i_l)| \leq t_{max} < 1 \\ 0 < t_{min} \leq |f_{simil\_item}(i_j, i_k)| \leq t_{max} < 1 \end{aligned} \tag{4.23}$$

t_{min} et t_{max} se réfèrent ici aux seuils minimum et maximum acceptés[47] afin de juger un triplet $< i_j, i_k, i_l >$ comme candidat potentiel à l'élaboration d'une règle. Cette contrainte nous permet de restreindre le nombre de triplets à examiner lors de la phase de génération des règles et donc de réduire les temps de calcul. Ce choix est justifié par le fait qu'une forte similarité entre deux items témoigne d'un fort degré de dépendance. Ainsi, une forte similarité entre i_j et i_l signifierait qu'une règle du type $< \mathring{I}_j, \mathring{I}_k > \Rightarrow \mathring{I}_l$ peut se résumer à $\mathring{I}_j \Rightarrow \mathring{I}_l$. A l'inverse, des similarités faibles $|f_{simil\_item}(i_j, i_l)|$ et $|f_{simil\_item}(i_k, i_l)|$ ne permettent pas de déduire une relation même conjointe entre les prémisses et la conclusion. En revanche, des similarités moyennes n'autorisent aucune simplification et sont conformes à l'hypothèse selon laquelle un seul prémisse ne doit pas expliquer l'intérêt pour la conclusion, mais l'attrait pour les deux prémisses le peut.

Une fois la sélection des triplets opérée selon le critère de l'équation 4.23, nous pouvons générer les règles viables. Pour déterminer l'existence d'une règle du type $< \mathring{I}_j, \mathring{I}_k > \Rightarrow \mathring{I}_l$, nous calculons les probabilités $P(\mathring{I}_j,\ \mathring{I}_k\ ,\ \mathring{I}_l)$, $P(\mathring{I}_j,\ \mathring{I}_l\ \mid\ \ddot{I}_k)$ et $P(\mathring{I}_k,\ \mathring{I}_l\ \mid\ \ddot{I}_j)$. Le symbole $\ddot{I}_j$ signifie que l'utilisateur n'a pas fourni de vote pour i_j. Le terme $P(\mathring{I}_j,\ \mathring{I}_k\ ,\ \mathring{I}_l)$ désigne alors la probabilité pour l'ensemble des utilisateurs du système d'aimer les trois items à la fois. $P(\mathring{I}_j,\ \mathring{I}_l\ \mid\ \ddot{I}_k)$ est la probabilité d'aimer i_j et i_l parmi les utilisateurs n'ayant pas voté pour i_k. Nous notons $\breve{I}_j$ la situation dans laquelle l'utilisateur considéré u_i a voté pour i_j quelle que soit la valeur de $v(u_i, i_j)$ et $N(\mathring{I}_j,\ \mathring{I}_l\ ,\ \ddot{I}_k)$ le nombre d'utilisateurs ayant aimé i_j et i_l sans avoir exprimé de vote pour i_k. Nous avons alors :

$$P(\mathring{I}_j,\ \mathring{I}_k\ ,\ \mathring{I}_l) = \frac{N(\mathring{I}_j,\ \mathring{I}_k\ ,\ \mathring{I}_l)}{N(\breve{I}_j,\ \breve{I}_k\ ,\ \breve{I}_l)} \tag{4.24}$$

$$P(\mathring{I}_j,\ \mathring{I}_l\ \mid\ \ddot{I}_k) = \frac{N(\mathring{I}_j,\ \ddot{I}_k\ ,\ \mathring{I}_l)}{N(\breve{I}_j,\ \ddot{I}_k\ ,\ \breve{I}_l)} \tag{4.25}$$

Nous générons une règle $< \mathring{I}_j, \mathring{I}_k > \Rightarrow \mathring{I}_l$ dès lors que la probabilité $P(\mathring{I}_j,\ \mathring{I}_k\ ,\ \mathring{I}_l)$ est très supérieure aux probabilités $P(\mathring{I}_j,\ \mathring{I}_l\ \mid\ \ddot{I}_k)$ et $P(\mathring{I}_k,\ \mathring{I}_l\ \mid\ \ddot{I}_j)$:

$$P(I_k,\ I_t\ ,\ I_w) \gg P(I_k,\ I_w\ \mid\ \ddot{I}_t) \tag{4.26}$$

$$P(I_k,\ I_t\ ,\ I_w) \gg P(I_t,\ I_w\ \mid\ \ddot{I}_k) \tag{4.27}$$

Dans notre modèle, l'écart de probabilité est jugé significatif lorsqu'il est supérieur ou égal à $0,3$.

[47] Dans le cadre de nos expérimentations, nous avons positionné ces valeurs de t_{min} et t_{max} respectivement à $0,4$ et $0,6$, afin d'avoir un nombre de similarités suffisant d'après la répartition affichée dans le tableau 2.2 (cf. supra, 2.2.1 GroupLens, p.39).

Une fois les règles générées à partir de la matrice des votes `M` et de la matrice item-item `S` sur le serveur, il est possible d'affiner certaines prédictions de l'utilisateur courant u_a en complément d'un algorithme de filtrage collaboratif traditionnel. Nous considérons qu'une règle est applicable lorsque les prémisses sont respectées par u_a et lorsque la prédiction porte sur l'item contenu dans la conclusion. Nous appelons AR_{u_a,i_l} l'ensemble des règles applicables pour l'utilisateur courant u_a et l'item i_l. Une règle est dite positive si la conclusion contient un terme $\mathring{I}_l$. Elle est négative dans le cas contraire. Nous notons $w(r, u_a, i_l)$ le poids associé à une règle applicable r et portant sur l'item i_l. Ce poids vaut 1 si la règle r est positive, -1 sinon. Au final, nous utilisons la formule suivante pour chacune des prédictions de u_a à affiner :

$$f_{predict}(u_a, i_l) = f_{predict\_trad}(u_a, i_l) + \frac{coef * \sum_{r \in AR_{u_a,i_l}} w(r, u_a, i_l)}{\sum_{r \in AR_{u_a,i_l}} |w(r, u_a, i_l)|} \tag{4.28}$$

$f_{predict\_trad}(u_a, i_l)$ représente la prédiction obtenue avec un algorithme de filtrage collaboratif traditionnel. `coef` est un cœfficient d'affinement que nous avons positionné à $0,1$ dans notre modèle.

Résultats et Discussion

Afin d'évaluer la valeur ajoutée de notre modèle, nous avons implanté un algorithme de filtrage collaboratif servant de support à l'algorithme `RIBA` et que nous appelerons `CIBA` dans cette section. `CIBA` utilise le cœfficient de corrélation de Pearson pour mesurer les similarités entre items et dispose d'une formule de prédictions proche de celle de `FRAC` :

$$f_{predict\_trad}(u_a, i_l) = max\Big(v_{min}, min\Big(\frac{\sum_{i_t \in R_a} f_{simil\_item}(i_l, i_t) \times (v(u_a, i_t) - \bar{i_t})}{\sum_{i_t \in R_a} |f_{simil\_item}(i_l, i_t)|} + \bar{i_k}, v_{max}\Big)\Big) \tag{4.29}$$

Nous considérons qu'une prédiction $f_{predict\_trad}(u_a, i_l)$ doit être affinée, dès lors que la moyenne en valeur absolue des similarités prises en compte dans le calcul est inférieure à la moyenne des similarités contenues dans la matrice item-item `S` :

$$\frac{\sum_{i_t \in R_a} |f_{simil\_item}(i_l, i_t)|}{\mid R_a \mid} < \frac{\sum_{j=1}^{m-1} \sum_{k=2}^{m} f_{simil\_item}(i_j, i_k)}{\mid R_s \mid} \tag{4.30}$$

avec $\mid R_a \mid$ le nombre d'items i_t dans R_a tels que $f_{simil\_item}(i_l, i_t) \neq 0$ et $\mid R_S \mid$ le nombre de similarités non nulles dans la matrice `S`.

Afin d'évaluer notre algorithme, nous avons utilisé le corpus de test `MovieLens` contenant 100.000 votes. Nous avons ensuite calculé les mesures d'erreur `MAE`, `HMAE1` et `HMAE2`. Les résultats sont présentés dans le tableau 4.13.

L'algorithme `RIBA` améliore donc les résultats de 6 à 8% d'après la mesure `HMAE1`. Cela signifie que l'utilisation de triplets améliore la qualité des prédictions élevées, c'est-à-dire portant sur les items qui seront suggérés par le système de recommandations. Nous remarquons que les mesures `MAE` et `HMAE2` restent stables, prouvant que le système n'affine pas trop de prédictions et qu'il ne fait pas empirer la qualité globale du système. Cette même qualité globale ne s'est au demeurant

pas vue accrue car le corpus de test utilisé ne permet pas de générer un grand nombre de règles négatives. Une trop grande majorité de règles positives implique un affinement des prédictions toujours à la hausse.

Ces expérimentations ont mis en lumière l'intérêt de l'utilisation de règles d'association portant sur des triplets. Notre objectif sera désormais d'intégrer ce modèle dans une architecture distribuée telle que `FRAC+` afin de respecter la contrainte de passage à l'échelle. En effet, la génération des règles reste encore assez coûteuse en temps de calcul dans `RIBA` (environ 1h de calcul sur le corpus `MovieLens`), bien que le temps nécessaire soit très inférieur à celui de l'algorithme d'apprentissage des structures dans le modèle `FSB` et permette d'ores et déjà une utilisation en contexte réel.

Datasets	U1.base		U2.base		U3.base		U4.base		U5.base	
Metrics	CIBA	RIBA	CIBA	RIBA	CIBA	RIBA	CIBA	RIBA	CIBA	RIBA
MAE	0.75	0.75	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74
HMAE1	**0.64**	**0.60**	**0.64**	**0.62**	**0.63**	**0.62**	**0.64**	**0.62**	**0.64**	**0.62**
HMAE2	0.64	0.65	0.62	0.62	0.65	0.65	0.64	0.64	0.63	0.63
Evaluation des prédictions uniquement dans les cas où il y avait des règles applicables										
Nombre de prédictions	2244		2481		2505		2316		2464	
MAE	0.75	0.75	0.80	0.80	0.78	0.78	0.79	0.79	0.79	0.79
HMAE1	**0.64**	**0.60**	**0.84**	**0.77**	**0.84**	**0.77**	**0.84**	**0.78**	**0.85**	**0.78**
HMAE2	0.64	0.65	0.72	0.71	0.65	0.63	0.66	0.68	0.67	0.66

TAB. 4.13 – Evaluation de la qualité des prédictions des algorithmes CIBA et RIBA.

4.6 Cadre applicatif

Les algorithmes FRAC et FSB ont fait l'objet d'une intégration réelle dans un contexte industriel. FRAC a, entre autres, été implanté dans le logiciel de diffusion de sites Web par satellites de la société ASTRA dans le cadre du projet ESA Sat'N'Surf mené en collaboration avec le centre de recherche public Henri Tudor et avec l'équipe Cortex au LORIA (Castagnos et Kassab, 2005b). Le modèle FSB a, quant à lui, été mis en place au sein de l'intranet documentaire du service Veille Technologique du Crédit Agricole S.A. (Boyer et al., 2006).

4.6.1 ASTRA

La société ASTRA, basée au Luxembourg, a mis au point un service de diffusion de sites Web par satellites baptisé Sat@once. Le tableau comparatif 4.14 permet de comprendre les attraits de ce système par rapport à l'Internet standard.

Tab. 4.14 – Analogie Internet/Sat@once.

Internet	Sat@once
- Payant ; - Communication bidirectionnelle entre 2 acteurs ; - Contenu privé.	- Gratuit (financé par la publicité) ; - Diffusion unidirectionnelle du serveur vers N clients ; - Contenu intéressant une large audience.
	- Nécessite une carte de réception satellite.

Pour simplifier, si on voit des similitudes entre Internet et le téléphone, alors on peut considérer Sat@once comme analogue à la radio.

La solution Sat@once peut, par exemple, être intéressante pour les mises à jour telles que *Microsoft Update IE*. En effet, ces opérations requièrent un grand nombre de serveurs avec l'Internet standard alors que le satellite permet la diffusion en une seule fois sans circulation inutile. Cette technologie est également intéressante dans les régions du monde où il n'est pas possible de déployer l'ADSL, en faisant bénéficier les usagers du haut débit satellitaire.

Présentation de Casablanca

Casablanca est à la fois le nom de l'application cliente développée par ASTRA et la désignation de la technologie utilisant le service Sat@once (cf. figure 4.11).

Il y avait environ 120.000 utilisateurs du service au moment du projet en 2005.

La vitesse de transmission est de l'ordre de 3,5 Mbit/s et est divisée en 3 canaux (l'appellation technique d'un canal est *webcaster*). Ces canaux correspondent aux 3 zones géographiques

Fig. 4.11 – Interface de Casablanca.

couvertes par le système d'ASTRA, à savoir :

- Italie (le moins important) ;
- Allemagne (le plus important) ;
- et Europe.

Chaque pays peut capter les 3 canaux mais un utilisateur allemand ne parlant qu'allemand prendra, par exemple, le canal par défaut.

La couverture satellitaire est donc européenne. Le service est gratuit et est financé par la publicité. Il permet de diffuser 150 sites Web (i.e. les items) faciles à visualiser avec un proxy (pas de `Flash`, etc.) sur les 3 canaux (tels que les sites du `Monde` ou du `Louvre` par exemple). Ces sites représentent un volume avoisinant les 3 Go en étant compressés (ou 15 Go en étant décompressés). La liste des 150 sites change toutes les semaines après intervention d'un opérateur. Initialement, le contenu du bouquet était déterminé manuellement. L'introduction du service de personnalisation a permis d'identifier plus facilement les items populaires. Le rafraîchissement des pages des sites se fait toutes les 2 ou 3 heures.

Fonctionnement de `Casablanca`

L'architecture technique de `Casablanca` est détaillée à la figure 4.12.

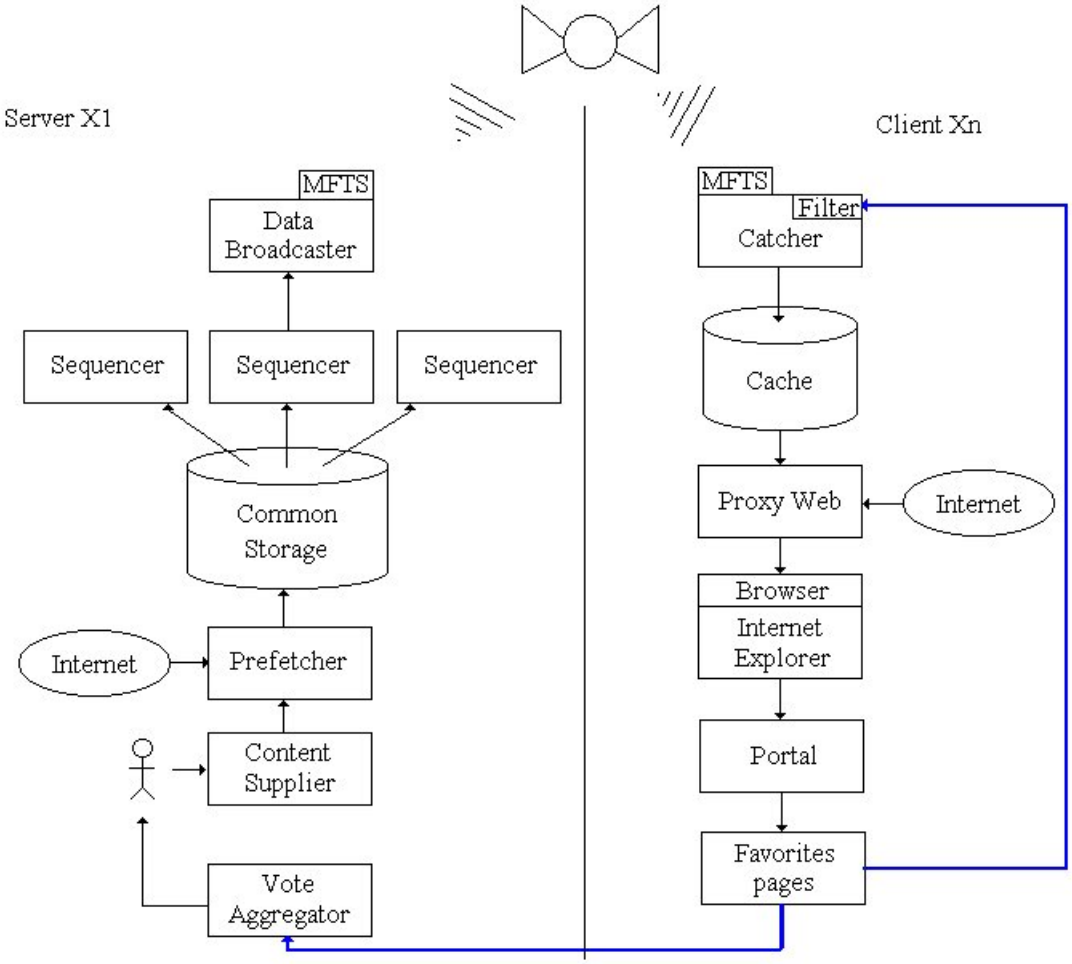


FIG. 4.12 – Architecture technique de `Casablanca`.

Les flèches en bleu sur le schéma 4.12 représentent les votes de l'utilisateur envoyés au serveur par une connexion Internet classique et constituant une entrée pour le filtre. Si l'utilisateur ne dispose pas d'un accès à Internet, il ne peut pas voter.

Côté Serveur. Le *content supplier* permet de lister les sites que l'on souhaite diffuser.

Le *prefetcher* est un module de bas niveau permettant de changer les sites diffusés et qui fait appel à d'autres outils :

- Un *crawler* (`Teleport pro`) rapatrie les sites sur le serveur. Le meilleur aspirateur Web était « Offline explorer » mais il y a eu échec de l'intégration dans `Casablanca` car il ne pouvait pas télécharger 3 sites Web en parallèle (un par canal) ;
- Un *profiler* analyse le contenu des sites ;
- Un *search engine* crée des fichiers d'index ;
- L'application utilise également `Swish` (moteur de recherche gratuit développé par `Berke-`

ley) ;
– Un outil de compression car certains sites ont jusqu'à 5000 répertoires et 10000 fichiers. L'outil utilisé est `WinRAR`.

Les sites sont ensuite placés dans un espace de stockage (archives accompagnées d'un fichier de description). Les *sequencer* permettent de connaître à l'avance les transmissions prévues. Il y en a un pour chaque canal. Le *data broadcaster* lit le programme et diffuse le contenu vers le satellite à l'aide d'un protocole de transport bas niveau des paquets `IP` (`MFTS` est basé sur la norme `TFTP`). Le *vote aggregator* fait la synthèse des sites les plus populaires pour aider l'opérateur à faire son choix de contenu à envoyer.

Côté Client. Le poste client (également appelé « terminal utilisateur ») reçoit les paquets diffusés (le contenu du site, ainsi qu'un fichier `XML` d'annonce d'un package avec la date d'envoi, le point d'entrée du site, la taille et le profil du site). Le *catcher* écoute les *packages* (chaque *package* correspond à un site Web donné). Le *filter* vérifie s'il va récupérer ou pas le *package* (selon s'il intéresse ou non l'utilisateur). Si le cache de l'utilisateur est plein, le *filter* refuse le site, à moins qu'il soit d'une priorité plus élevée qu'un site déjà contenu dans le cache (auquel cas, il supprime le package correspondant et accepte le site Web entrant). Initialement, le *filter* disposait d'une règle binaire (priorité haute si l'utilisateur final a voté pour ce site et basse sinon). Le *filter* effaçait donc un site de priorité basse contenu dans le cache pour faire de la place lorsqu'un site de priorité haute arrivait. Lorsqu'un site était parcouru par l'utilisateur, il était considéré comme de priorité maximale. Le système de personnalisation a permis de définir plus finement les priorités entre les différents packages.

Les sites vont ensuite être placés dans un cache. Le proxy Web permet d'assurer la transparence pour l'utilisateur final entre l'affichage des sites de `Casablanca` et l'utilisation de l'Internet standard (si l'utilisateur l'a activée). L'utilisateur accède au contenu du cache depuis un portail (fait en `HTML` et `ActiveX`). Les sites Web sont classés par catégorie et le portail inclut un moteur de recherche. Le système est conçu de telle sorte que, lors de la première utilisation, les clients peuvent voter pour des sites dans une liste ou proposer de nouveaux sites (si un nouveau site est proposé, l'opérateur le teste sur un client `sat@once` et le rajoute si l'apparence est acceptable. En effet, le client peut rencontrer des problèmes d'affichage avec certaines technologies telles que le `Flash`). Ces votes sont ensuite envoyés au *vote aggregator* côté Serveur.

Une version du client et du serveur voit le jour tous les 3 à 5 mois au terme d'un cycle de projet bien défini (*workshop*, *design*, développement, test, intégration sur des PCs sur un `LAN`, validation sur une copie conforme de la plate-forme de production et la phase de *packaging* pour la réalisation du *setup* et la mise en ligne sur le site Web). Une version du serveur est compatible avec la même version du client, ainsi qu'une version antérieure (afin que l'utilisateur ne soit pas obligé de télécharger la nouvelle version dès la mise à jour et rencontre des dysfonctionnements).

Détails du projet Sat'N'Surf

Les différents points entrants dans le cadre de ce projet sont énumérés ci-dessous :

- La construction automatique du côté du serveur du bouquet des sites ②[48] diffusés par satellite à partir des votes des utilisateurs ① et du modèle des documents ③ ;
- La création du modèle des documents ③ (collaboration entre les équipes `CORTEX`, `ORPAILLEUR` et `MAIA`). L'équipe `CORTEX` a proposé des méthodes permettant la structuration des documents. De notre côté, nous avons étudié les algorithmes mettant les ressources en corrélation à l'aide de calculs de similarité (Castagnos, 2004; Castagnos et Kassab, 2005a) ;
- La création de classes d'utilisateurs ④ (profils types) du côté du serveur. Une fois ces classes rapatriées sur un terminal, le système peut identifier l'utilisateur à un groupe en fonction de ses actions passées et prédire son intérêt pour des ressources non votées ;
- La modélisation des préférences de l'utilisateur courant sur son poste. Celle-ci repose sur les votes précédents de l'utilisateur et sur des critères implicites (temps passé sur une page, clics de l'utilisateur, fréquence de consultation, etc.). Toutes ces données ne quittent bien évidemment pas les terminaux afin de garantir la confidentialité des informations relatives aux utilisateurs ;
- Un système de cache intelligent ⑤ sur les terminaux. Ce filtre permet de ne fournir à l'utilisateur que les items susceptibles de l'intéresser à partir des profils types ④ (récupérés au niveau des terminaux), du modèle de documents ③, du profil local de l'utilisateur courant et du bouquet de sites ② diffusé.

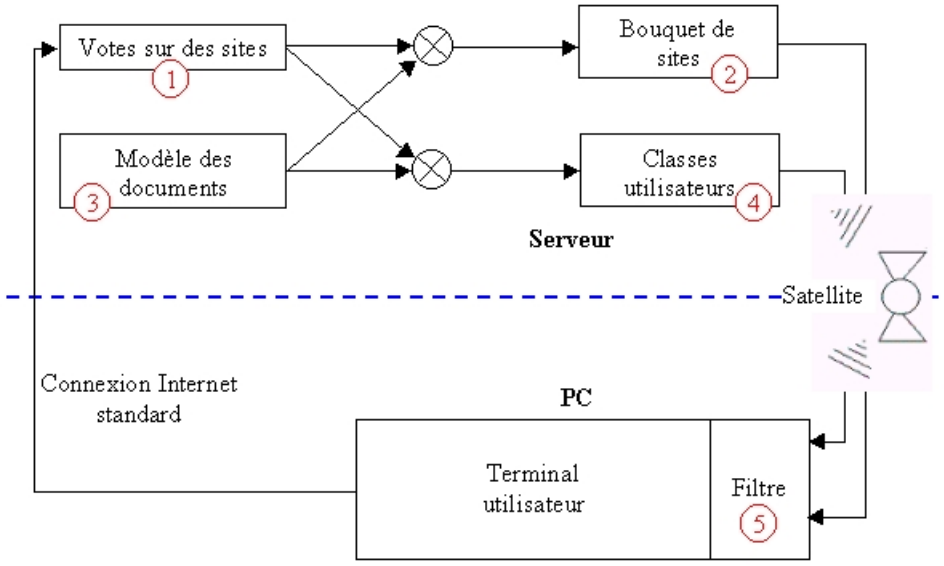


FIG. 4.13 – Positionnement du filtrage collaboratif dans l'architecture de `Casablanca`.

[48] les numéros entourés correspondent à ceux de la figure 4.13

Apports de FRAC dans Casablanca

La collaboration avec la société ASTRA a constitué un support concret dans le cadre de cette thèse. Ce partenariat a permis d'avoir un retour sur l'efficacité de l'algorithme FRAC.

Pour ASTRA, ce projet visait à améliorer leur produit afin d'accroître la satisfaction des usagers. En effet, parmi les 120.000 utilisateurs du système, seuls 50.000 disposaient d'une connexion internet standard et acceptaient de partager des votes. Les 70.000 autres personnes avaient un cache rempli aléatoirement, limitant de ce fait l'intérêt du système. L'ajout d'un service de personnalisation a contribué à cette amélioration. En effet, la constitution automatique du bouquet satellitaire et l'intégration d'un système de recommandations a permis au produit de la société ASTRA de fournir des ressources conformes aux attentes des utilisateurs.

Parmi les avantages notables du modèle FRAC au sein de Casablanca, nous noterons que :

- il n'a pas été nécessaire de modifier l'interface graphique. Cette contrainte était en effet imposée par la société, afin que les utilisateurs ne se sentent pas perdus avec une nouvelle interface ;
- l'algorithme a pu s'adapter facilement aux contraintes particulières de la connexion par satellites et aux traces récupérables au sein d'un tel système en affinant la modélisation des préférences de l'utilisateur courant. L'interface de Casablanca permettait par exemple de voter pour un ensemble d'items en cochant des cases dans une liste. Toutefois, ces votes n'étaient pas booléens puisqu'il n'est pas possible de déterminer si une case non cochée désigne un item non apprécié par u_a ou une ressource que u_a n'a pas pris le temps de cocher. Ces votes caractérisaient donc les préférences individuelles de façon peu précise. Par ailleurs, un démon http du côté client listait les actions de l'utilisateur au sein du système dans des fichiers log. Nous avons pu exploiter ces fichiers log et les votes non booléens (considérés comme un critère dans la formule de transformation 1.1 de la page 30) pour estimer l'intérêt de u_a pour chaque site Web ;
- la phase de modélisation des préférences, et donc la constitution du profil utilisateur, est réalisée côté client. Les profils types des communautés sont, quant à eux, calculés côté serveur puis envoyés aux postes clients via le satellite. De cette façon, même les utilisateurs sans connexion internet standard ou ne souhaitant pas partager leurs préférences peuvent bénéficier du service de personnalisation (le profil utilisateur stocké en local est comparé aux profils types reçus) ;
- le caractère décentralisé de ce modèle a permis de surmonter le problème du passage à l'échelle (le nombre d'utilisateurs constituait à l'époque une limite difficilement atteignable au sein d'un algorithme de filtrage collaboratif) ;
- la modélisation des préférences a pu se faire dans le respect de la vie privée des utilisateurs.

Processus de modélisation dans Casablanca

La modélisation des préférences de l'utilisateur courant se fait sur la base de critères implicites et explicites (Castagnos et al., 2005a). Nous avons en effet vu dans le chapitre 1 que les préférences explicites s'avèrent nécessaires mais insuffisantes. Dans ce modèle, les utilisateurs ont la possibilité de voter pour les ressources sur une échelle de valeurs arbitraire et/ou d'exprimer

leurs goûts en cochant des cases associées à des items. Par ailleurs, nous considérerons le cas où chaque poste client est équipé d'un `démon` collectant les actions de l'utilisateur au sein du portail. Ces actions sont retranscrites sous forme de fichiers log (cf. supra, 1.2 Typologie des données collectables, paragraphe "la liste d'actions effectuées", p.15).

Les actions sont conservées pendant une durée limitée et ne quittent pas les postes clients afin de préserver la caractère confidentiel de ces informations. Une fonction de transformation f_{transf} (cf. équation 1.1, p. 30) permet ensuite de combiner les critères implicites et explicites pour représenter les préférences individuelles sous une forme numérique plus respectueuse de la vie privée des utilisateurs.

Afin d'illustrer les possibilités de modélisation offertes par notre système, nous considérerons la situation où le système de recommandations d'`ASTRA` laissent les utilisateurs voter pour un site web (les items) en attribuant une valeur entière $v_{exp}(u_j, i_k)$ comprise entre 1 et 7[49], quels que soient $u_j \in U$ et $i_k \in I$. La valeur 0 est associée à l'absence de vote. Les utilisateurs ont également la possibilité de fournir une liste de leurs sites web favoris en cochant des cases. Nous pouvons alors prendre une hypothèse sur la prévalence des critères explicites par rapport aux critères implicites. Nous considérerons, pour l'exemple, qu'un vote explicite prédomine sur tout autre type de modélisation car il constitue une estimation précise des attentes de l'utilisateur courant vis-à-vis de cette ressource. La liste des favoris représente également un signe d'intérêt fort de la part de l'usager. Toutefois, il ne suffit pas à quantifier cet intérêt. Une case cochée signifie un fort attrait pour un item. Une case non cochée ne permet pas d'effectuer de conclusion par rapport à cet item. La liste des favoris est donc un des critères de la fonction de transformation avec un poids plus élevé que les critères implicites. Les fichiers log nous permettent également de recourir aux critères de récence (équation 1.2), de durée (équation 1.3) et de fréquence de consultation (équation 1.4). Le système fournit également des informations quant à la taille des sites web et au nombre de liens hypertexte dans chacun d'eux[50]. La formule de transformation revêt alors la forme suivante :

$$f_{transf}(u_a, i_k) = \begin{cases} v_{exp}(u_a, i_k) & \text{si } v_{exp}(u_a, i_k) \neq 0 \\ 1 + 2 \,.\, EstFavori(u_a, i_k) + Recence(u_a, i_k) & \\ \quad + 2 \,.\, Frequence(u_a, i_k) \,.\, Duree(u_a, i_k) & \\ \quad + PourcentageLiensVisites(u_a, i_k) & \text{sinon} \end{cases} \tag{4.31}$$

$$\text{Avec : } PourcentageLiensVisites(u_a, i_k) = \frac{\text{Nombre de liens de } i_k \text{ visités par } u_a}{\text{Nombre total de liens dans } i_k}$$

$$\text{Et : } EstFavori(u_a, i_k) = \begin{cases} 1 \text{ si la case associée à } i_k \text{ est cochée} \\ 0 \text{ sinon} \end{cases} \tag{4.32}$$

Les critères `Recence`, `Frequence` et `Duree` sont ceux définis dans le chapitre 1. Chaque critère prend une valeur maximale égale à 1. Les votes estimés sont donc bien compris entre 1 et 7.

[49] Dans la réalité, cette possibilité n'a pas été laissée aux usagers.

[50] Il s'agit d'une situation spécifique à la communication satellitaire où le site web a été aspiré puis compressé sur un serveur, avant d'être envoyé aux clients.

L'ensemble des votes estimés de l'utilisateur courant constitue son profil. L'utilisateur a la possibilité, s'il le souhaite, de partager ses préférences en envoyant anonymement son profil côté serveur. L'anonymat et la non duplication des données sont garantis par un identifiant propre à chaque utilisateur mais ne permettant pas de l'identifier. Celui-ci est généré par une fonction de hachage, conformément à notre procédé générique présenté en section 1.4.2.

La modélisation des préférences pourrait également se faire côté serveur, comme ce fût le cas pour le `Crédit Agricole S.A.`, en exploitant les votes contenus dans une base de données et les fichiers log du serveur. Il suffirait pour cela de remplacer les u_a par des u_j (u_j représentant un utilisateur lambda) au sein des formules de cette section. Une modélisation effectuée côté client est toutefois plus respectueuse de la vie privée des usagers.

4.6.2 Crédit Agricole S.A.

Face à la banalisation de l'offre bancaire notamment sur Internet, la personnalisation des services devient un des enjeux importants pour renforcer la fidélisation et l'usage des canaux (Boyer et al., 2006).

Parmi les fonctions mises à disposition sur les sites Internet bancaires (le raisonnement est aussi valable sur d'autres canaux), il est probable que certaines d'entre elles soient plus sollicitées que d'autres en fonctions de comportements spécifiques liés à de multiples critères propres au client et à sa sphère d'intérêt à un moment donné de la relation avec sa banque.

Notre collaboration avec le `Crédit Agricole` avait pour but de déterminer dans quelle mesure il est possible d'envisager une personnalisation de services qui repose sur le filtrage collaboratif et respecte les particularités et contraintes inhérentes au contexte bancaire.

L'étude a fait ressortir un manque de connaissances des clients sur les produits de la banque et assurance. Il est par conséquent nécessaire d'informer, de conseiller et de guider un client ou prospect. Un autre aspect constaté est que plus un client est pris en charge et plus il est attaché à sa banque.

Les technologies de personnalisation ne présentent pas seulement des intérêts pour les clients. Ils peuvent également aider les employés de la banque à mieux s'informer en les assistant lors de leurs navigations sur l'intranet documentaire. C'est dans cette optique que le modèle `FSB` a été implanté à titre de démonstrateur au sein de l'intranet du service Veille Technologique du `Crédit Agricole`. Cette plate-forme exploite le logiciel de gestion de contenu JCMS de la société `Jalios`. Il a donc fallu développer les connecteurs avec cette application. La généricité de notre modèle a permis à un stagiaire d'effectuer sous notre supervision cette intégration dans des délais relativement courts. Au niveau de l'interface, l'implantation de ce nouveau service s'est traduit par l'ajout d'un portlet proposant des items susceptibles d'intéresser les employés de la banque (cf. figure 4.14).

Dans ce contexte applicatif, l'analyse des usages (fichiers log) et la modélisation des préférences se fait au niveau du serveur. Les utilisateurs se connectent à la plate-forme via un portail, peuvent accéder aux différents items et émettre un avis pour chaque item (notes de 1 à 5, commentaires textuels, etc.) comme en atteste la figure 4.15.

Au moment de la rédaction de cette thèse, l'évaluation de la valeur ajoutée de cette tech-

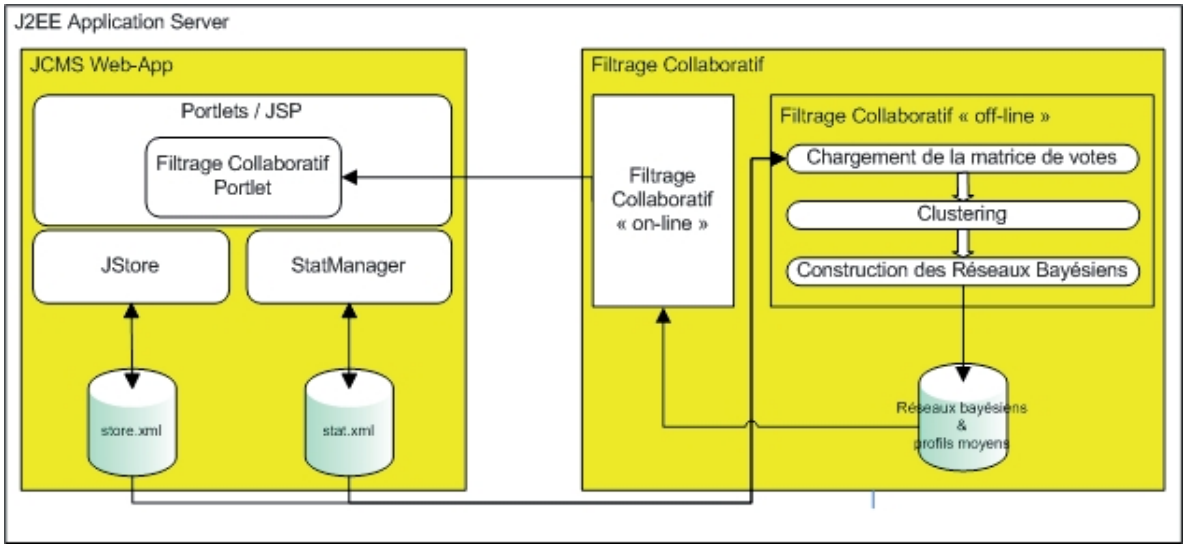


FIG. 4.14 – Architecture technique de JCMS.

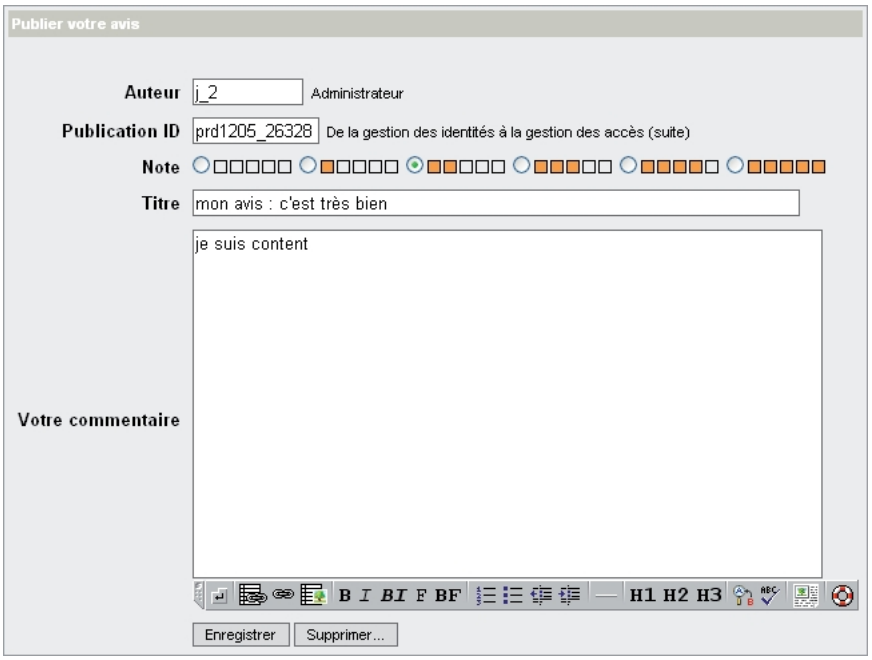


FIG. 4.15 – Impression écran de la publication d'avis.

nologie dans le secteur bancaire est encore en cours mais les premiers résultats et les pistes fonctionnelles sont prometteurs.

Chapitre 5

Filtrage collaboratif sur des grilles de calcul

5.1 AURA

Dans ce chapitre, nous nous positionnons dans le contexte d'une architecture pair-à-pair, toujours dans l'optique de résoudre le problème du passage à l'échelle dans les algorithmes de filtrage collaboratif. Ce type d'architecture devient de plus en plus répandu dans les systèmes de recherche et d'accès à l'information car il constitue une méthode efficace de partage de ressources (Miller et al., 2004). Par ailleurs, en plus des applications grand public les plus connues (partage de fichiers et messagerie instantanée), les technologies pair-à-pair peuvent résoudre un ensemble de problèmes liés à l'utilisation optimale des ressources disponibles dans les réseaux et des entités composant ces derniers. L'objectif est donc d'exploiter les capacités de la grille de calculs, c'est-à-dire la puissance de calcul (processeurs, mémoire, etc.) de milliers d'ordinateurs.

La distribution complète du filtrage collaboratif en terme de calculs permet de réduire considérablement le temps nécessaire à la génération des prédictions. Dans la suite de ce chapitre, nous choisirons comme cadre applicatif une plate-forme documentaire pair-à-pair capable de fournir des recommandations aux utilisateurs. Ce type d'architecture peut par exemple être utilisé dans les entreprises comme alternative aux banques de données centralisées : plutôt que d'ajouter périodiquement des documents sur un serveur (au risque que les items accessibles par tous ne correspondent pas aux dernières versions des documents), les employés peuvent les partager directement depuis leur poste. Il s'agit donc d'un autre mode d'organisation possible de l'information. Chaque ordinateur connecté à la plate-forme est appelé un pair. Nous verrons dans un premier temps comment appliquer notre méthode générique de modélisation des préférences dans ce type de scénario (Castagnos et Boyer, 2007a). Puis, nous présenterons une façon d'exploiter ces profils au sein de notre algorithme de filtrage collaboratif distribué AURA (Castagnos et Boyer, 2007b).

5.1.1 Modélisation des préférences

Chaque utilisateur évolue dans un environnement dynamique dont il n'a qu'une connaissance partielle. Il peut partager des ressources avec les autres usagers ou accéder à de nouveaux documents via la plate-forme documentaire. Les items disponibles sont hétérogènes (documents texte, vidéos, fichiers audio, sites web, blogs, etc.) et transitoires dans la mesure où leur durée de vie sur la plate-forme est limitée. Afin de reprendre les notations introduites dans le chapitre 1, nous appellerons I_t la liste des m' items disponibles à l'instant t sur la plate-forme.

Chaque item est considéré comme une unité élémentaire et insécable. On pose pour contrainte que le système ne requiert pas de connaissance *a priori* sur le contenu de ces items. Au demeurant, un identifiant id_j décrivant de façon unique chaque item i_j est nécessaire et suffisant pour le localiser et le distinguer des autres ressources. En effet, lorsque des utilisateurs partagent des documents sur une plate-forme pair-à-pair comme `KaZaA`[51] ou `eMule`[52], ils ont la possibilité de renommer les fichiers. Cet identifiant unique permet au système d'éviter la duplication des items dans le catalogue I_t. De cette façon, un item peut être téléchargé par fragments depuis plusieurs machines en parallèle, même si les fichiers correspondant à cet item ne portent pas le même nom sur les différents ordinateurs. L'identifiant unique garantit également qu'un utilisateur ne peut pas fournir deux votes différents au même instant pour un même item. La table de correspondance item-identifiant $T_{i/id}$ est stockée sur un serveur et contient la liste de tous les couples $< i_j, id_j >$.

L'ensemble des n utilisateurs du service de personnalisation est noté U. Chaque usager doit s'abonner à se service. L'ensemble des n' abonnés à un instant t est noté U_t. Il dispose ensuite d'un *login* et d'un mot de passe pour se connecter à la plate-forme. Comme explicité dans la section 1.4.2, une fonction de hachage permet d'attribuer à un utilisateur un identifiant unique à partir de son *login* et de son adresse IP. De cette façon, plusieurs utilisateurs peuvent se connecter sur un même ordinateur sans que leurs profils respectifs n'interfèrent entre eux. Nous affinons donc la définition du terme "pair" en associant ce terme à un utilisateur sur un ordinateur donné du réseau. La génération des identifiants des utilisateurs (ID) ne nécessite pas de disposer d'un serveur spécifique. Cette opération peut s'effectuer sur les différents pairs. Il n'est pas possible de retrouver le nom ou l'adresse IP d'un utilisateur à partir de son ID. Le module de communication utilise une adresse IP multicast pour diffuser sur le réseau les paquets contenant les ID des destinataires. Cette procédure garantit l'anonymat de chaque utilisateur au cours des échanges.

Les actions des utilisateurs au sein de la plate-forme sont listées localement dans des fichiers log respectant la syntaxe CLF présentée dans (Masseglia et al., 2004). Les utilisateurs ont également la possibilité de renseigner explicitement leurs préférences. Prenons l'exemple d'une interface où l'utilisateur peut voter pour chaque item sur une échelle de votes. Lors de l'application de la fonction de transformation de la section 1.4.2, nous pouvons donc appliquer les critères de récence et de durée de consultation, ainsi que l'appartenance à la liste des favoris d'après les votes explicites. Si l'on considère que chacun de ces critères a un poids équivalent, sauf les votes explicites entrant deux fois plus en ligne de compte (car il s'agit d'un signal fort), la fonction de

[51] http ://www.kazaa.com/fr/index.htm
[52] http ://www.emule-project.net/home/perl/general.cgi ?l=13

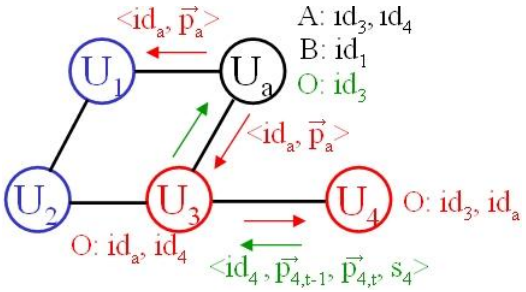


FIG. 5.1 – Exemple d'exécution de l'algorithme AURA.

transformation devient alors :

$$f_{transf}(u_a, i_k) = v_{min} + \frac{recence(u_a, i_k) + duree(u_a, i_k) + 2 \times \frac{v(u_a, i_k)}{v_{max}}}{4} * (v_{max} - v_{min}) \quad (5.1)$$

Cette fonction permet de construire et de mettre à jour les profils utilisateurs au cours du temps. Ces profils sont stockés localement sur les pairs pour respecter la vie privée de leurs propriétaires respectifs. Dans la suite de ce chapitre, nous désignerons ces profils par le terme "profils personnels".

5.1.2 Filtrage collaboratif distribué

Nous avons conçu un algorithme de filtrage collaboratif basé sur les utilisateurs et adapté à ce type d'architecture totalement distribué (Castagnos et Boyer, 2007b). Cette distribution transparaît en terme de subdivision des calculs sur les différents pairs, mais également en terme de répartition des profils et des items sur les différents postes. Notre algorithme s'appelle `AURA`, pour *Adaptive User-Based Recommender Algorithm*.

Au sein de notre système, chaque pair gère plusieurs structures de données : un profil personnel, un profil public, un profil de groupe et des listes d'ID (une liste "A" pour les ID des pairs appartenant au groupe de l'utilisateur courant, une liste "B" pour les ID de ceux dont le cœfficient de corrélation (distance) excède un seuil explicité plus bas, une liste "C" pour les ID mis sur liste noire [53], une liste "O" pour les ID des pairs ayant ajoutés le profil public de l'utilisateur courant à leur profil de groupe). Un exemple d'exécution de l'algorithme `AURA` est présenté dans la figure 5.1. Sur ce schéma, les ronds correspondent à des pairs connectés au système. A côté de chaque pair, nous retrouvons les listes A, B et O contenant des identifiants d'utilisateurs. Les notations du type $< id_a, \ldots >$ désigne des messages échangés à travers le réseau.

Nous définissons le profil public $\vec{p_a}$ comme la partie du profil personnel $\vec{u_a}$ que l'utilisateur courant accepte de partager avec autrui. Ce sont donc des profils publics qui pourront être échangés à travers le réseau.

Le système construit également un profil de groupe. Ce dernier représente les préférences d'une communauté d'intérêts virtuelle centrée sur l'utilisateur courant (cf. figure 5.2).

[53] Notre modèle prévoit de laisser l'utilisateur courant bloquer les utilisateurs qu'il juge malveillant. Il est également possible de rajouter un filtre sécuritaire automatique détectant les utilisateurs malveillants.

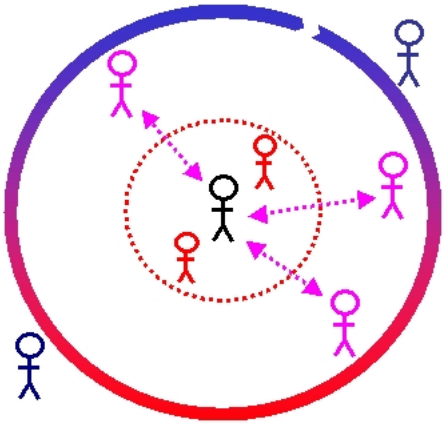


FIG. 5.2 – Communauté virtuelle centrée sur u_a.

Afin de construire cette communauté, le pair de l'utilisateur courant demande le profil public de tous les pairs qu'il peut atteindre sur le réseau. Puis, pour chacun de ces profils, il calcule le degré de similarité avec le profil personnel de l'utilisateur courant. Si le résultat est inférieur à un seuil fixé s spécifique à chaque utilisateur, l'ID du pair en question est ajouté à la liste "A" et le profil correspondant est inclus dans le profil de groupe de l'utilisateur courant, en appliquant la procédure de l'équation 5.1.

Procedure AjoutAuProfilDeGroupe(profil public de u_n)
$W = W + |f_{simil}(u_a, u_n)|$
pour chaque item i **faire**
$f_{predict}(u_a, i) = f_{predict}(u_a, i) * (W - |f_{simil}(u_a, u_n)|)$
$f_{predict}(u_a, i) = (f_{predict}(u_a, i) + f_{simil}(u_a, u_n) * v(u_n, i))/W$
fin pour

Avec : $f_{predict}(u_a, i)$ le vote pour l'item i dans le profil de groupe ;
$v(u_n, i)$ le vote de l'utilisateur u_n pour l'item i ;
W la somme des $|f_{simil}(u_a, u_i)|$, qui est sauvegardée ;
$f_{simil}(u_a, u_n)$ le cœfficient de corrélation entre l'utilisateur courant u_a et u_n.

TAB. 5.1 – Ajout d'un profil public au profil de groupe.

La mesure de similarité entre les profils peut être calculée à l'aide du cœfficient de corrélation de Pearson, du cœfficient de Pearson forcé, du vecteur cosine ou des moindres carrés. Par défaut, notre algorithme AURA Castagnos et Boyer (2007a) utilise la métrique de Pearson.

Si le profil personnel de l'utilisateur courant est modifié, et donc potentiellement son profil public p_a, l'utilisateur courant doit contacter tous les pairs dont l'ID est dans sa liste "O". Chacun de ces pairs recalcule la mesure de similarité. Si celle-ci excède le seuil, le profil p_a est supprimé du profil de groupe en appliquant la procédure de l'équation 5.2. Sinon p_a doit être mis à jour dans le profil de groupe, c'est-à-dire que le pair doit supprimer l'ancien profil et ajouter le nouveau. Périodiquement, le profil mis à jour est renvoyé à tous les pairs et non pas seulement aux pairs de la liste "O" afin de détecter des opportunités d'ajout dans les groupes d'autres pairs.

Procedure RetirerDuProfilDeGroupe(ancien profil de u_n)
$W = W - |f_{simil}(u_a, u_n)|$
pour chaque item i **faire**
$f_{predict}(u_a, i) = f_{predict}(u_a, i) * (W + |f_{simil}(u_a, u_n)|)$
$f_{predict}(u_a, i) = (f_{predict}(u_a, i) - f_{simil}(u_a, u_n) * v(u_n, i))/W$
fin pour

TAB. 5.2 – Retirer le profil public du profil de groupe.

Par convention, nous utilisons la notation $< id, \vec{p} >$ pour désigner les paquets envoyés par les nouveaux inscrits sur le service de personnalisation. $< id, \vec{p}, s >$ correspond au paquet d'un pair ayant déjà été connecté et envoyant des données à un nouvel arrivant. Il n'est pas nécessaire de spécifier la valeur du seuil de corrélation minimum s dans le paquet d'un nouvel arrivant, dans la mesure où s est initialisé avec une valeur par défaut égale à 0. Enfin, $< id, \vec{p_{t-1}}, \vec{p_t}, s >$ désigne un paquet de mise à jour, contenant l'ancien profil public à l'instant $(t-1)$ et le nouveau à l'instant t. Dans chacun de ces paquets, le premier élément contient l'identifiant de l'expéditeur du message.

La figure 5.3 fournit un exemple de fonctionnement du système. Afin de simplifier la notation des paquets, nous n'inclurons pas les identifiants des destinataires des messages dans cet exemple.

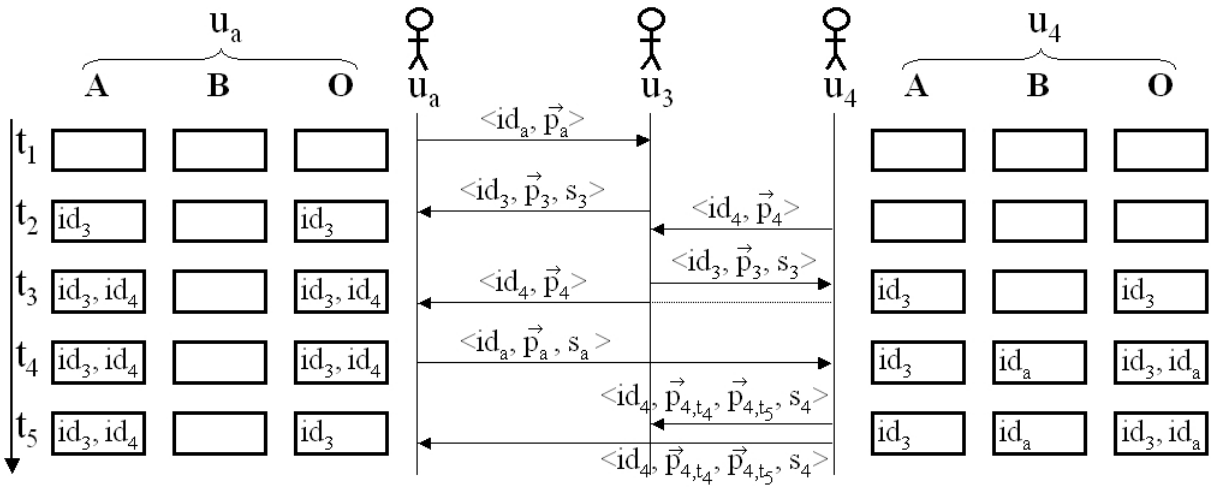


FIG. 5.3 – Exemple d'interactions entre les utilisateurs.

Dans cet exemple, nous considérons 3 des 5 utilisateurs de la figure 5.1. La figure 5.3 explicite les registres de l'utilisateur courant u_a et de l'utilisateur u_4.

1. Au temps t_1, l'utilisateur courant u_a tente de contacter pour la première fois les autres pairs en envoyant son profil public et son identifiant à ses voisins. Il s'agit du paquet $< id_a, \vec{p_a} >$.
2. u_3 reçoit ce paquet et envoie une réponse au temps t_2. u_a calcule alors la similarité entre les profils publics $\vec{p_3}$ et $\vec{p_a}$. Dans la mesure où le cœfficient de Pearson est nécessairement

supérieur au seuil par défaut utilisé à l'initialisation, u_a ajoute l'identifiant id_3 à sa liste "A". Si la similarité calculée est supérieure au seuil s_3 de l'utilisateur u_3, alors u_a ajoute id_3 à sa liste "O". Parallèlement, les pairs atteints ajoutent $\vec{p_a}$ dans leur profil de groupe et l'identifiant id_a dans leur liste "A". C'est le cas de u_3 pour lequel la similarité avec u_a est supérieure à son seuil. Au temps t_2, u_4 se connecte pour la première fois sur la plate-forme et envoie un paquet à u_3.

3. Au temps t_3, u_3 répond à u_4 et fait suivre le paquet de u_4 aux pairs qu'il connaît déjà. u_a le reçoit et ajoute id_4 à sa liste "A". Il ajoute également id_4 à sa liste "O", puisque u_4 est un nouveau venu et dispose d'un seuil par défaut.

4. Au temps t_4, u_a envoie son profil public à u_4. Au même moment, u_4 change la valeur de son seuil et considère que u_a est trop loin dans l'espace de représentation utilisateurs/ressources. En d'autres termes, le cœfficient de corrélation entre les profils de u_a et u_4 devient inférieur au seuil fixé. u_4 ajoute alors id_a à sa liste "B". Par le biais du paquet $< id_a, \vec{p_a}, s_a >$, le pair de u_4 sait qu'il doit compléter la liste "O" avec l'identifiant id_a.

5. Enfin, au temps t_5, u_4 met à jour son profil public. Par la suite, il informe les pairs dont les identifiants sont dans sa liste "O" de cette mise à jour par l'intermédiaire du paquet $< id_4, p_{\vec{4,t_4}}, p_{\vec{4,t_5}}, s_4 >$. $p_{\vec{4,t_4}}$ et $p_{\vec{4,t_5}}$ sont respectivement l'ancien et le nouveau profil public de u_4. A la réception de ce message, u_a met à jour sa liste "O" en supprimant id_4 puisque s_4 est un seuil trop élevé au vu de la similarité existante.

5.1.3 Seuil de corrélation minimum adaptatif

Comme expliqué dans la section précédente, l'utilisateur courant peut définir indirectement un seuil de corrélation minimum que les autres pairs doivent atteindre afin de devenir membres de sa communauté. Il s'agit du cercle de confiance de la figure 5.2. Concrètement, un seuil de corrélation élevé signifie que les utilisateurs pris en compte dans le calcul des prédictions sont très proches de l'utilisateur courant. Les recommandations seront par conséquent extrêmement proches de ses propres préférences. A l'inverse, un seuil de corrélation faible illustre le souhait de l'utilisateur courant de rester ouvert à des recommandations généralistes en intégrant les préférences d'utilisateurs plus distants. De cette façon, l'utilisateur courant évite les recommandations sclérosées en acceptant des suggestions atypiques ou qui lui sont inconnues (nouveautés). Dans l'interface graphique, une barre de suivi (*trackbar* en anglais) permet à l'utilisateur courant de définir ce degré de personnalisation souhaité. Ce réglage permet à l'algorithme `AURA` de savoir dans quelle mesure il peut modifier le seuil. Ce seuil désigne le minimum que peuvent prendre en valeur absolue les cœfficients de corrélation de Pearson pris en compte dans le calcul du profil de groupe. Par exemple, si le système fixe un seuil à "0,1", seuls les profils publics des pairs u_i dont le cœfficient de corrélation $|f_{simil}(u_a, u_i)|$ est supérieur à 0,1 seront inclus dans le profil de groupe de l'utilisateur courant.

La valeur du seuil par défaut est 0, ce qui signifie que nous prenons tous les pairs en compte. Le pas de variation par défaut du seuil est de 0,1. Il est toutefois possible d'adapter ce pas à la densité de la population.

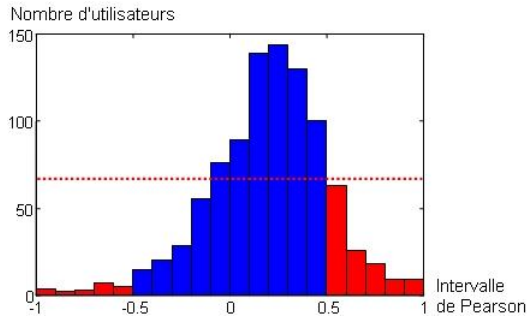


FIG. 5.4 – Seuil adaptatif basé sur la densité.

Comme le montre la figure 5.4, nous divisons l'intervalle des valeurs possibles du cœfficient de Pearson [-1 ;+1] en sous-intervalles. Pour chaque sous-ensemble, nous comptons le nombre de pairs ayant eu des échanges avec l'utilisateur courant et dont le cœfficient de corrélation est contenu dans l'intervalle de valeurs correspondant. Ainsi, quand un utilisateur envoie un paquet à u_a, le cœfficient de corrélation de Pearson est calculé afin de savoir si le profil de groupe de l'utilisateur courant doit être mis à jour en fonction de la valeur du seuil. Parallèlement, nous mettons à jour les valeurs correspondantes dans l'histogramme reflétant la distribution de la population. Par exemple, si u_a reçoit un paquet de mise à jour et que le cœfficient de Pearson passe de 0,71 à 0,89, nous décrémentons le registre de l'intervalle [0,7 ;0,8[et nous incrémentons le registre de l'intervalle [0,8 ;0,9[. De cette façon, nous connaissons à chaque instant la densité de population pour chaque intervalle.

Quand le nombre total d'utilisateurs dont le cœfficient de Pearson est supérieur en valeur absolue à la somme de la valeur actuelle du seuil et du pas de variation (i.e. seuil + 0,1) excède une certaine limite (ligne en pointillée sur la figure 5.4 symbolisant le nombre d'utilisateurs à atteindre), le système est à même d'augmenter le seuil d'un pas de variation. Il réalise cette opération progressivement jusqu'à atteindre le seuil souhaité par l'utilisateur courant. Dans le cadre de nos expérimentations, nous avons la valeur du seuil maximum à 0,2 pour les utilisateurs désirant avoir un haut degré de nouveauté et 0,9 pour ceux qui attendent des préférences très proches de leurs préférences. Ces valeurs ont été choisies arbitrairement. Lors d'une utilisation en situation réelle, il est souhaitable de réaliser une estimation de la répartition de la population sur l'intervalle de Pearson avant de déterminer les seuils idéaux et le pas de variation, et ainsi optimiser l'utilisation de notre seuil adaptatif en fonction du contexte.

5.2 Résultats

5.2.1 Qualité des prédictions

Afin d'évaluer la qualité des prédictions de l'algorithme `AURA`, nous avons utilisé le corpus `MovieLens` contenant 100.000 votes réels (cf. supra, 2.2.1 GroupLens, p.37). Nous avons utilisé la métrique d'erreur MAE et nous avons confronté nos résultats à l'algorithme `PocketLens` (Miller et al., 2004) présenté en section 3.6.2.

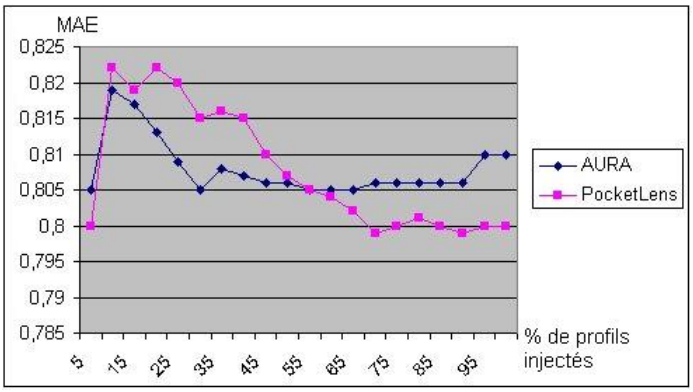


FIG. 5.5 – Evolution des MAE lors de l'injection progressive des profils.

Les résultats sont présentés sur la figure 5.5. Il s'agit de l'erreur moyenne obtenue sur les 5 jeux de tests de `MovieLens` en injectant progressivement les profils des utilisateurs dans le calcul des prédictions. Cet ajout progressif nous a permis de simuler la découverte de nouveaux pairs à travers le réseau. En effet, les profils publics ne sont échangés que lorsque les pairs sont connectés au service et il faut donc du temps pour découvrir l'ensemble des utilisateurs. Les tests prouvent que la qualité des recommandations reste à peu près constante en dépit du fait que le système ne dispose pas de tous les profils au début des calculs.

Pour cette évaluation, nous n'avons pas utilisé le seuil adaptatif afin que les résultats entre `PocketLens` et `AURA` soient comparables. Les expérimentations montrent que nous obtenons d'aussi bons résultats que l'algorithme pair-à-pair très performant `PocketLens`.

5.2.2 Intérêt du seuil adaptatif

Le but du seuil adaptatif est d'influencer le degré de personnalisation du système. Lorsque le système augmente le seuil, les suggestions sont plus proches des attentes des utilisateurs mais le risque est plus fort d'avoir des recommandations sclérosées. Afin de faire ressortir ce phénomène, nous avons généré une matrice de votes de 1.000 utilisateurs et 1.000 items à l'aide de notre outil de génération de matrices (cf. supra, 2.2.1 Génération de corpus, p.40). Les votes suivent une loi gaussienne et nous pouvons voir la distribution des utilisateurs en fonction du cœfficient de Pearson dans la partie gauche de la figure 5.6. Nous avons retiré aléatoirement 20% des votes et avons appliqué l'algorithme `AURA`. Puis nous avons calculé la mesure de rappel déterminant le nombre moyen d'items déjà votés par chaque utilisateur figure dans son top-10 des recommandations.

Nos tests confirment que lorsque le système augmente le seuil, la mesure de rappel s'accroît. Les recommandations sont donc effectivement plus proches des attentes des utilisateurs mais témoignent d'une moins grande ouverture à la nouveauté.

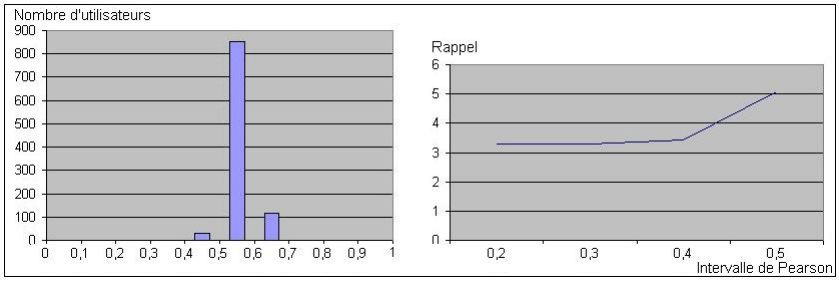


FIG. 5.6 – Sur la gauche, la distribution des utilisateurs sur l'intervalle de corrélation de Pearson. Sur la droite, la mesure de rappel lorsque le seuil adaptatif augmente.

5.2.3 Temps de calcul

Enfin, nous avons évalué notre algorithme en terme de temps de calcul. Nous nous sommes comparés à deux approches centralisées pour illustrer l'intérêt de l'approche distribuée : l'algorithme `Item-Item` (Sarwar et al., 2001) (cf. supra, p.65) et l'algorithme basé sur la mémoire `CorrCF` (Resnick et al., 1994) (cf. supra, p.53).

Afin de déterminer les temps de calcul de ces algorithmes, nous avons généré des profils publics aléatoires avec différents nombres d'items (cf. supra, 2.2.1 Génération de corpus, p.40). Dans cette simulation, les votes de chaque utilisateur suivent une distribution gaussienne centrée sur le milieu de l'échelle de votes $[v_{min}; v_{max}]$. Par ailleurs, le corpus ainsi généré ne comprend qu'un pourcent de données manquantes. Ce pourcentage n'est certes pas réaliste mais il permet d'augmenter potentiellement le temps de calcul et donc d'éprouver les algorithmes. Dans la mesure où les algorithmes `Item-Item` et `CorrCF` sont centralisés, nous avons dans un premier temps aggrégé les profils dans une matrice de votes pour ne pas les pénaliser en incluant le temps d'aggrégation dans le temps de calcul des prédictions.

Les temps de calcul nécessaires aux trois algorithmes pour prédire l'ensemble des votes inconnus dans les profils des utilisateurs sont présentés dans le tableau 5.3. Pour des raisons matérielles, les délais annoncés pour l'algorithme `AURA` n'incluent le temps requis pour propager les profils publics à travers le réseau. De plus, il est délicat d'estimer le délai supplémentaire occasionné, puisque la communication entre les pairs suppose une connexion simultanée des utilisateurs associés. Rappelons toutefois que notre algorithme pair-à-pair n'a pas besoin de disposer de l'ensemble des profils pour fournir des recommandations, contrairement aux deux autres algorithmes. Par ailleurs, les temps obtenus pour les algorithmes `Item-Item` et `CorrCF` n'intègrent pas non plus le temps de rapatriement des votes sur le serveur et leur aggrégation dans une matrice.

Les résultats mettent par exemple en évidence le fait qu'il est désormais possible de prédire les votes de 10.000 utilisateurs sur 100 ressources en moins d'une demi-seconde contre une minute et 22 secondes pour `Item-Item`, et 7 heures et demi pour `CorrCF`. Bien évidemment, un tel écart entre `AURA` et les deux autres algorithmes est dû au fait que nous utilisons les capacités de calcul d'autant d'ordinateurs que d'utilisateurs. En d'autres termes, le temps de calcul des prédictions pour 10.000 utilisateurs est égal au temps d'intégration de 10.000 profils publics dans le profil de groupe d'un seul usager.

Afin de ne pas surcharger le tableau 5.3, nous n'avons pas affiché les temps de calcul de `PocketLens`. En effet, ce dernier obtient les mêmes temps que notre algorithme `AURA`.

Items	100			150			1000		
Users	AURA	CorrCF	It-It	AURA	CorrCF	It-It	AURA	CorrCF	It-It
200	0”01	2”60	2”14	0”01	3”17	2”71	0”07	11”09	52”74
400	0”02	6”09	3”87	0”02	7”62	5”29	0”12	32”24	1’22”
600	0”02	11”78	5”59	0”03	15”21	7”34	0”18	1’04”	2’05”
800	0”03	19”98	7”23	0”04	25”67	10”53	0”27	1’52”	2’33”
1,000	0”03	30”22	8”56	0”05	40”68	12”84	0”30	3’06”	3’25”
1,400	0”04	1’00”	11”50	0”06	1’17”	18”10	0”42	6’04”	4’29”
10,000	**0”31**	**7 :30’**	**1’22”**	0”48	-	2’05”	1”90	-	49’28”
100,000	3”04	-	-	-	-	-	-	-	-

TAB. 5.3 – Temps de calcul de trois algorithmes de filtrage collaboratif.

5.3 Discussion

Dans ce chapitre, nous avons proposé un modèle de filtrage collaboratif adapté aux architectures pair-à-pair et présentant l'avantage d'être totalement distribué en terme de calculs et de répartition des profils à travers le réseau. La distribution des calculs a permis de s'attaquer à l'épineux problème du passage à l'échelle en exploitant les capacités des grilles de calcul. Ainsi, l'algorithme `AURA` est capable d'intégrer les profils de 100.000 utilisateurs en seulement 3 secondes, après réception de ces derniers sur le pair concerné. Mais la véritable force de cet algorithme réside dans le fait qu'il est *anytime*. Il calcule en effet les prédictions incrémentalement en ajoutant l'un après l'autre les profils publics dans le profil de groupe. Cette solution serait plus coûteuse qu'un algorithme de clustering au sein d'une architecture centralisée. Dans une architecture distribuée au contraire, elle permet de casser les calculs et de paralléliser les traitements requis par cet algorithme. La complexité algorithmique pour la construction des communautés tous utilisateurs confondus est en $O(n^2)$ puisqu'il s'agit de comparer le profil de chaque utilisateur avec les autres profils soit $n \times n \times r$ opérations avec r le nombre de ressources communément évaluées. Les algorithmes de *clustering* ont bien souvent une complexité quasi-linéaire O(n.log(n)). En parallélisant les calculs sur n pairs, nous obtenons une complexité linéaire $O(n)$. Avec `AURA`, il n'est donc plus nécessaire d'effectuer de calculs hors ligne. Les prédictions sont récupérables immédiatement, même dans le cas où tous les profils publics n'ont pu être pris en compte. La qualité des recommandations reste au demeurant constante comme le prouve les mesures d'erreur affichées en figure 5.5.

Les résultats obtenus en terme de qualité des prédictions sont équivalents à ceux de l'algorithme `PocketLens`. `AURA` améliore même la mesure de `MAE` de 0,04 par rapport à `FRAC` (Castagnos et Boyer, 2006a) et de 0,02 par rapport à `Item-Item` (Sarwar et al., 2001). Cela est dû au fait que l'utilisateur courant est désormais au centre de sa communauté, alors qu'il n'était au préalable qu'un membre de sa communauté plus ou moins éloigné du barycentre dans l'espace de représentation utilisateurs/ressources.

Les temps de calcul entre `AURA` et `PocketLens` sont similaires pour deux raisons. Tout d'abord, `PocketLens` effectue également ses calculs en accédant aux profils des utilisateurs l'un après l'autre. Par ailleurs, le corpus généré ne comprend qu'un pourcent de votes manquants. Or, cet algorithme ne gagne en temps de calcul que dans le cas où il y a peu de votes, puisque les prédictions reposent uniquement sur les votes connus de l'utilisateur courant. Autrement dit, les calculs se limitent aux colonnes de la matrice de votes `M` pour lesquelles l'utilisateur courant a renseigné une valeur. Réduire le nombre de votes dans le corpus avantage l'algorithme `PocketLens` mais le gain de temps occasionné est négligeable au vu des performances obtenues par `AURA`. Par ailleurs, la réduction du nombre de votes pénalise plus rapidement la qualité des recommandations et le taux de couverture de `PocketLens` par rapport à `AURA`. En effet, la connaissance amoindrie des préférences de l'utilisateur est compensée dans notre algorithme par l'expérience des autres usagers, en particulier dans les cas où le nombre d'utilisateurs est très supérieur au nombre d'items. Il est donc possible d'obtenir des résultats performants avec un algorithme de filtrage collaboratif basé sur les utilisateurs. Cela démontre une fois de plus que le choix entre un algorithme basé sur les items et un algorithme basé sur les usagers doit se faire en fonction du contexte d'utilisation. Un modèle basé sur les items est pertinent si le

nombre de ressources est faible et si l'ensemble d'items `I` est stable dans le temps. Dans le cas contraire, il est préférable de recourir à une approche basée sur les utilisateurs. En effet, si un item disparaît du catalogue, les votes portant sur cet item ne seront plus exploitables dans un algorithme basé sur les items, puisqu'il ne sera plus possible de connaître sa similarité avec les nouveaux items apparaissant au fur et à mesure. Dans une approche basée sur les utilisateurs au contraire, les votes de l'item indisponible continueront à être exploitables puisqu'il s'agira de calculer la similarité entre des usagers en fonction de leur expérience passée commune.

Le principe de l'algorithme `AURA` peut sembler très similaire à celui de l'algorithme `PocketLens`. Il présente pourtant trois différences majeures :

1. D'autre part, notre algorithme construit les profils de groupe incrémentalement tout en intégrant la composante temporelle, c'est-à-dire l'évolution des goûts des utilisateurs au cours du temps. Contrairement à `PocketLens`, il n'est pas nécessaire de réinitialiser les communautés avant chaque phase de prédictions, notamment en cas de modification de certains votes. Cette considération nous permet également de distinguer notre modèle de l'algorithme de Berkovsky et al. (2006).
2. L'occupation mémoire de l'algorithme `AURA` est plus faible que celle de `PocketLens` puisqu'il ne s'agit plus de stocker une matrice de similarités, mais uniquement un nombre restreint et fixe de vecteurs (profil privé, profil public, listes d'identifiants et profil de groupe).
3. Enfin, le trafic réseau est potentiellement moins important avec `AURA`, puisque les profils ne sont échangés qu'en cas de première connexion ou de mise à jour avec une fréquence d'envoi plus élevée pour les utilisateurs suffisamment proches de u_a. Dans `PocketLens`, les envois se font périodiquement pour l'ensemble des utilisateurs à destination de tous les pairs.

Notons que la distribution des calculs ne s'est pas faite au détriment de la portabilité. Les opérations effectuées sur chaque pair sont en effet très peu coûteux en ressources, aussi bien pour l'utilisation du processeur que de la mémoire. De ce fait, il est possible de recourir à notre système de personnalisation depuis divers supports matériels tels qu'un ordinateur portable, un téléphone ou un PDA. La portabilité est également assurée par le fait que notre modèle a été implanté en `Java`.

La répartition des profils a quant à elle permis de renforcer la confiance des utilisateurs. Chaque usager est désormais pleinement maître de l'utilisation qui est faite de son profil. Chaque profil privé est stocké sur la machine de son propriétaire et ce dernier a la possibilité de définir la partie publique de son profil. Le système fait donc preuve d'une grande transparence lors du processus de modélisation des préférences et en amont des collaborations entre pairs.

Notre système accorde également une grande importance à la préservation des données personnelles. A l'instar de `PocketLens` mais à la différence de l'algorithme de Berkovsky et al. (2006), notre algorithme ne requiert pas de stocker les profils provenant des autres pairs dans une matrice sur le poste de l'utilisateur courant. De ce fait, il n'est pas possible à l'utilisateur courant d'accéder aux votes d'autrui, mais uniquement aux votes moyens de sa communauté. Par ailleurs, les profils de l'utilisateur courant, qu'ils soient publics ou privés, sont stockés uniquement en local avec la possibilité de crypter leur contenu pour empêcher toute personne malveillante de consulter les fichiers de sauvegarde. Les profils de groupe n'intègrent que les pré-

férences moyennes de la communauté, et non pas les préférences individuelles des membres de la communauté. Enfin, notre modèle est pleinement compatible avec la méthode de modélisation proposée dans le chapitre 1 (cf. supra, 1.4.2 Proposition d'un procédé générique de modélisation, p.28). La modélisation des préférences est donc respectueuse de la vie privée et permet de réduire le nombre de données manquantes, puisqu'elle repose partiellement sur des critères implicites.

Lors de l'élaboration de ce modèle, nous nous sommes également intéressé au problème des recommandations sclérosées. Dans ce contexte, nous avons imaginé un seuil adaptatif permettant aux utilisateurs de spécifier le niveau de personnalisation souhaité. Toutefois, les exigences des utilisateurs ne doivent pas se faire au détriment de la qualité des recommandations. A l'initialisation du profil, les recommandations sont forcément mauvaises et le système peut se permettre de prendre en compte l'ensemble des usagers pour lesquels u_a reçoit les profils publics. L'algorithme intègre donc au départ les préférences des utilisateurs dont la corrélation avec l'utilisateur courant en valeur absolue est supérieure ou égale à 0. Puis, plus le temps passe, plus le système acquiert de l'information relative aux préférences de l'utilisateur courant. Il peut donc se montrer plus sélectif dans le choix des profils prenant part à la phase de prédictions. Nos expérimentations montrent qu'une sélection des voisins les plus proches permet d'engendrer des recommandations plus proches des attentes de l'utilisateur. Au demeurant, si le nombre de voisins répondant au critère de similarité n'est pas suffisant, le nombre de données manquantes risque de s'accroître considérant, pénalisant d'autant la qualité des recommandations. Pour cette raison, notre seuil adaptatif n'augmente progressivement vers la valeur souhaitée qu'à la condition que le nombre de voisins reste suffisant.

La principale faiblesse de l'algorithme `AURA` réside dans le trafic réseau générée. Bien que le nombre de messages échangés soit moins important qu'avec l'algorithme `PocketLens`, il reste assez important, notamment en raison de la stratégie de diffusion assez proche des architectures en découverte aléatoire et transitive transverse présentées dans l'état de l'art (cf. supra, 3.6.2 PocketLens, p.72). Dans cette topologie, les pairs se connectent aléatoirement sur une branche du réseau. Il n'y a donc pas d'optimisation en fonction de la proximité avec leurs principaux interlocuteurs. Lorsqu'un message est envoyé, il se propage à travers le réseau jusqu'à atteindre l'ensemble de ses destinataires, en transitant par la plupart des pairs du réseau. Il serait possible de réduire ce phénomène de propagation en introduisant des super-pairs. Ces derniers, sortes de serveurs connectés en permanence et responsables d'un sous-ensemble de pairs, se chargeraient de réguler le trafic en se répartissant les pairs afin de maximiser le nombre de messages à destination des pairs sous la responsabilité d'un même super-pair, et de minimiser le nombre de messages entre pairs assignés à des super-pairs différents (cf. figure 5.7).

Les super-pairs constitueraient alors une sorte de colonne vertébrale du réseau, contenant la liste des identifiants des pairs sous leur responsabilité. Ces pairs seraient donc répartis en groupes et les super-pairs comptabiliseraient le nombre de messages intra- et inter-groupes pour chaque pair. Si un pair envoie un plus grand nombre de messages vers un autre groupe de pairs qu'à destination du groupe dont il fait parti, il pourra alors être automatiquement réassigné à cet autre groupe. Sur la figure 5.7, l'utilisateur u_9 pourrait par exemple être réassigné au super-pair X.

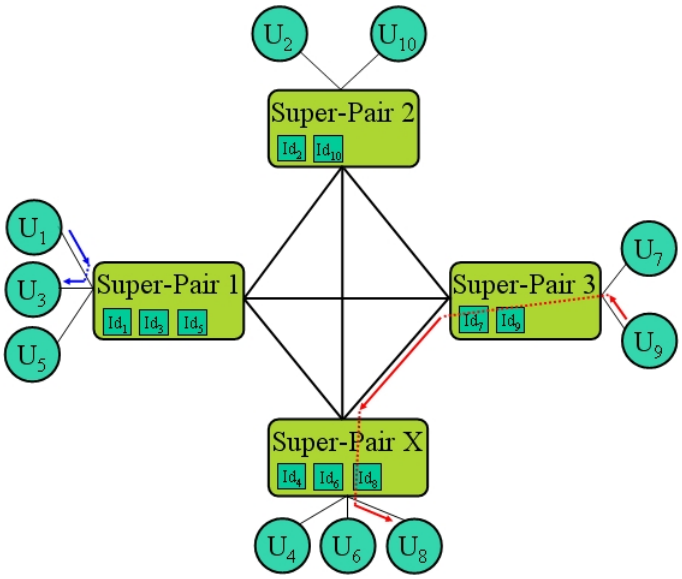


FIG. 5.7 – Architecture du réseau avec des super-pairs se répartissant les pairs.

5.4 Cadre applicatif : la plate-forme SofoS

Afin de mener à bien nos expérimentations, nous avons conçu une plate-forme pair-à-pair de partage et d'accès à des documents de divers formats. Cette plate-forme s'appelle `SofoS` pour "*Sharing Our Files On the System*" (cf. figure 5.8). Elle intègre un service de personnalisation reposant sur l'algorithme `AURA`. Nous avons développé `SofoS` par briques avec l'objectif de le faire évoluer et d'en faire une base pour d'autres algorithmes pouvant s'avérer complémentaires avec `AURA`. Notre plate-forme repose sur la technologie `JXTA`[54]. Ce choix a été motivé par le fait que cette technologie permet entre autres choses de déployer le système sur tous types de supports matériels et qu'il est grandement utilisé par les membres de la communauté scientifique. `JXTA` gère notamment tous les protocoles de communication bas niveau pour les réseaux pair-à-pair, nous permettant de nous concentrer sur nos modèles lors de la phase de conception.

De façon plus générale, `Sun Microsystem` a proposé le *framework* pair-à-pair `Open Source JXTA` pour répondre à 3 objectifs principaux :

- l'interopérabilité à travers différents systèmes et communautés pair-à-pair ;
- une plate-forme indépendante du système d'exploitation (Windows, Unix, Solaris, ...), du langage de programmation (C/C++, Java, Perl, Messages du protocole en XML) et du moyen de transport (TCP/IP, HTTP, Bluetooth) ;
- l'ubiquité en restant indépendant du type de machine (PDA, PC, téléphone mobile, ...).

`JXTA` possède une architecture à trois couches (cf. figure 5.9) :

- le noyau (`JXTA Core`) permet aux applications de communiquer ;
- la couche des services (`JXTA Services`) propose différents services optionnels comme la recherche, le partage de fichier, l'authentification, etc. ;
- la couche des applications (`JXTA Applications`) est la couche la plus haute gérant no-

[54] http ://www.jxta.org/

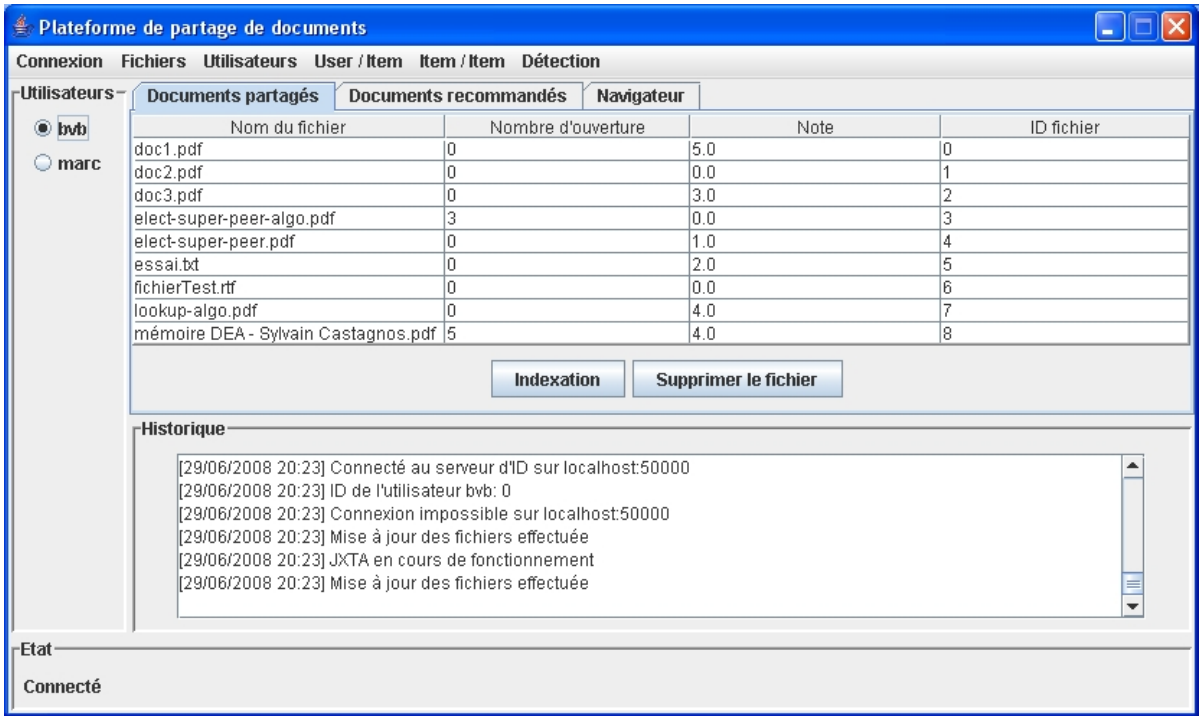


FIG. 5.8 – Interface de SofoS.

tamment les messages instantanés et les e-mails.

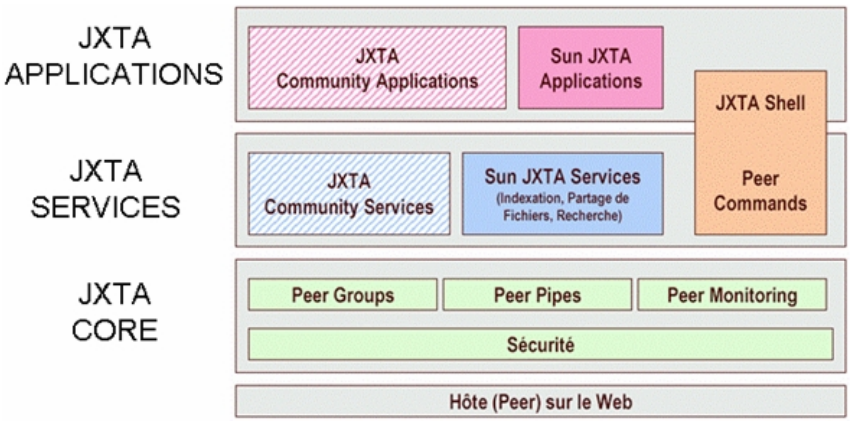


FIG. 5.9 – Les différentes couches de JXTA.

`JXTA` crée un réseau virtuel afin de permettre à l'utilisateur de faire abstraction de la configuration du réseau physique (cf. figure 5.10).

Il est prévu que notre plate-forme `SofoS` serve notamment de socle au projet `Cérès`[55] intitulé "Comportement et Intelligence Artificielle" (`CIA`). Ce projet vise à développer capable de gérer les aspects sécuritaires et d'identifier les utilisateurs malveillants au sein d'un service de

[55]Initiative de mutualisation de la valorisation des universités lorraines soutenue par l'`ANR`.

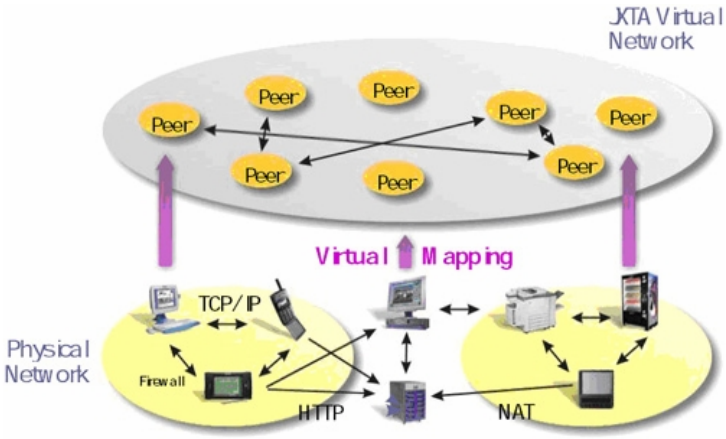


FIG. 5.10 – Réseau virtuel créé par JXTA.

personnalisation.

Conclusion et Perspectives

Bilan

Avec l'avènement des nouvelles technologies de l'information et de la communication (NTIC), et en particulier l'apparition de l'Internet, l'Homme est entré dans l'ère de l'information. Les NTIC revêtent une dimension sociétale, puisqu'il est désormais possible à tout un chacun de s'exprimer, d'échanger, de partager ou de consulter des ressources depuis n'importe quel point du globe. Les sources d'information sont devenues quasi-illimitées.

L'objectif premier de ces nouvelles technologies était un accès facilité à la connaissance, ainsi qu'un gain de temps significatif pour les usagers en quête d'une information. Cet objectif a été partiellement atteint avec la démocratisation d'Internet au milieu des années 90. Toutefois, Internet a fini par être victime de son succès : le nombre d'internautes et la profusion d'items accessibles ont paradoxalement rendu l'accès à l'information plus difficile. Comment trouver rapidement une ressource pertinente dans un contexte de recherche spécifique ? Comment se tenir globalement informé face à tant de profusion ? Comment faire le tri entre informations utiles et informations superfétatoires ou caduques ? Comment élargir ses horizons en découvrant de nouvelles sources d'information ? Autant de questions auxquelles se sont trouvés confrontés les internautes qui ont pu vérifier le vieil adage selon lequel trop d'informations tue l'information.

Dans ce contexte de surabondance d'items, de nombreuses méthodes sont venues enrichir les systèmes de recherche et d'accès à l'information dans le but d'améliorer les interactions homme/machine. Parmi ces méthodes, on retrouve l'emploi de services de personnalisation et de systèmes de recommandations. Le but est d'assister les usagers en sélectionnant une liste d'items pertinents et en proposant de nouvelles ressources potentiellement intéressantes à l'aulne de leurs préférences individuelles. Cette approche est particulièrement intéressante, lorsqu'elle est utilisée dans un environnement à faible risque. Dans le cas de la recherche de livres ou de films par exemple, les réponses proposées ne présentent pas un caractère vital et le système dispose d'un droit à l'erreur. Au demeurant, les utilisateurs sont le plus souvent exigeants et il est parfois préférable de s'abstenir de répondre plutôt que de les mécontenter, si l'on souhaite les fidéliser.

Le fonctionnement d'un système de recommandations s'articule en deux temps. La première phase consiste à modéliser les préférences de chaque utilisateur, c'est-à-dire à collecter des données brutes et à les rendre légalement et techniquement exploitables. La deuxième étape vise à exploiter les modèles de préférences générés préalablement pour suggérer des items pertinents.

Dans ce manuscrit, nous avons dans un premier temps cherché à établir une typologie aussi exhaustive que possible de données brutes pouvant caractériser des interactions entre un utilisateur et un système de recommandations. Cette liste a servi de support à une discussion sur les différents modes possibles de collecte de ces données. De nombreuses solutions d'acquisition des données ou de modélisation des préférences existent. Elles se démarquent bien souvent les unes des autres par le contexte applicatif dans lequel elles s'inscrivent et le degré d'implication attendu de l'utilisateur courant. Pour notre part, nous avons posé pour hypothèse qu'il était préférable de solliciter l'utilisateur le moins possible lors du processus de modélisation et de le décharger au maximum en lui fournissant les ressources attendues et en lui suggérant de nouveaux items. A ces suppositions sont venues s'ajouter les contraintes légales et éthiques en matière de respect de la vie privée. **C'est dans ce contexte que nous avons proposé une méthode de modélisation des préférences pouvant s'adapter à tout type de système de recommandations à base de filtrage collaboratif.**

Le filtrage collaboratif est une famille d'algorithmes issue de l'Intelligence Artificielle numérique permettant de faire du filtrage pertinent, de la diffusion ciblée, de créer des communautés d'utilisateurs ou de classer des items relativement aux usages. **Il s'agit de techniques d'apprentissage non supervisé exploitant l'expérience acquise collectivement pour combler les vides dans les modèles individuels.** L'aspect collaboratif de ces algorithmes est transparent pour l'utilisateur.

Le premier défi auquel nous avons été confrontés, afin de rendre cette technologie applicable en situation réelle, est le problème du passage à l'échelle. L'approche envisagée pour relever ce défi est la distribution des procédés de filtrage collaboratif. Nous avons donc étudié l'apport d'une répartition des calculs au sein d'architectures client/serveur et pair-à-pair. **Nos recherches ont abouti à la création de deux modèles (`FRAC` et `AURA`) capables de gérer plusieurs centaines de milliers d'utilisateurs et plusieurs milliers d'items tout en fournissant des recommandations en temps réel.**

Il est rapidement apparu que le passage à l'échelle n'était pas le seul enjeu à intégrer lors de l'élaboration d'un algorithme de filtrage collaboratif. Outre le respect de la vie privée précédemment mentionné, il s'agit de minimiser l'impact des données manquantes, de garantir la qualité des recommandations, d'éviter les suggestions sclérosées, d'assurer la confiance de l'utilisateur, de proposer un système portable, d'intégrer les aspects sécuritaires, de prendre en compte le contexte et de compenser le problème du démarrage à froid.

En implantant notre méthode de modélisation des préférences au sein de nos algorithmes et en positionnant les profils sur les postes clients plutôt que sur un serveur, **nous proposons des modèles respectueux de la vie privée des usagers. Cette méthode de modélisation permet par ailleurs de réduire la quantité de données manquantes**, puisqu'elle complète les profils de votes explicites sur la base de critères implicites. Nos expérimentations ont montré que **la distribution du filtrage collaboratif, tant en terme de calculs que de contenu, n'altère pas la qualité des recommandations.** Les erreurs mesurées restent comparables aux algorithmes performants de la littérature.

Lors de l'élaboration de notre modèle pair-à-pair, nous nous sommes également intéressés aux problèmes de portabilité, de confiance et de recommandations sclérosées. **Notre algorithme**

AURA est portable dans le sens où il ne requiert pas de gros calculs et ne nécessite donc pas de matériel spécifique. Il peut fonctionner indifféremment sur un serveur, un ordinateur personnel, un PDA ou un téléphone portable[56]. **L'utilisation d'un seuil adaptatif définissant dynamiquement la taille de la sphère de voisinage pour chaque individu a permis d'étudier le problème des recommandations sclérosées.** En effet, plus le seuil est bas, plus les suggestions témoignent d'une volonté d'ouverture à la nouveauté. A l'inverse, plus le seuil est élevé, plus les recommandations seront proches des attentes des utilisateurs. Toutefois, ce seuil est conçu pour augmenter progressivement, à la condition expresse qu'il y ait suffisamment d'utilisateurs dans le voisinage. Un nombre trop faible de voisins risquerait de susciter la sclérose des résultats. Enfin, **le problème de confiance a été abordé en considérant que la confiance des utilisateurs sera d'autant plus grande s'ils ont accès à tout instant au contenu de leurs profils.** Ainsi, les usagers ont non seulement la possibilité de visualiser les estimations du système concernant leurs préférences individuelles, mais ils peuvent également y apporter des modifications et définir la partie publique du profil qui pourra être échangée anonymement avec les autres pairs.

Afin de clore ce bilan des travaux réalisés, soulignons que nos modèles ont fait l'objet d'une intégration dans un contexte industriel réel, notamment au sein du `Crédit Agricole S.A.`, de `SES ASTRA` et de la société nouvellement créée `Sailendra S.A.S.` (cf. infra, figure 1).

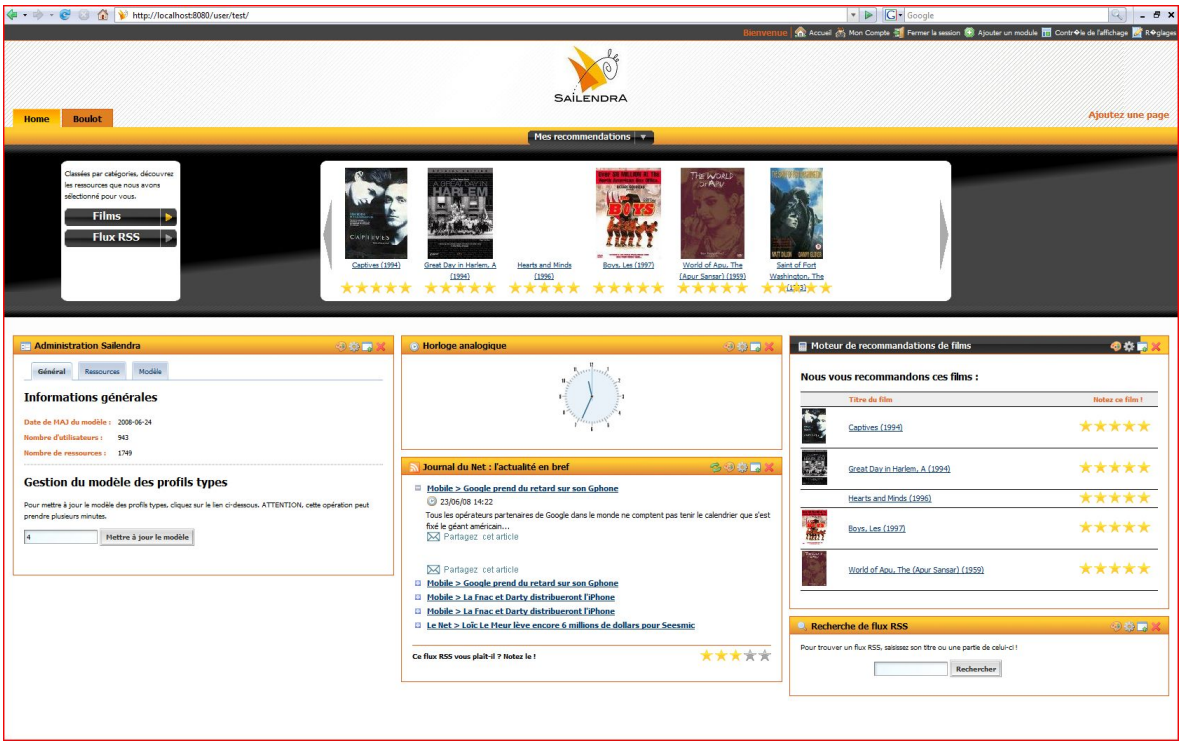


FIG. 1 – Interface du moteur de recommandations de Sailendra.

[56]Cette portabilité est également en grande partie rendue possible par l'utilisation du langage de programmation par objets `Java`.

Perspectives

Le filtrage collaboratif constitue un moyen efficace d'assister les utilisateurs lors de leurs interactions avec un système de recherche et d'accès à l'information. En effet, il permet de personnaliser l'accès à des sources d'information et de proposer des services à haute valeur ajoutée, tels que la création de communautés virtuelles, la mise en relation, le marketing ciblé, etc. Elle facilite l'accès rapide et pertinent à l'information en s'adaptant aux utilisateurs et en présentant un comportement proactif vis-à-vis des attentes et besoins des usagers. **Toutefois, comme toute famille d'algorithmes, elle comporte des limites.** Ainsi, le problème du démarrage à froid reste un des enjeux majeurs encore irrésolus au sein d'algorithmes de filtrage collaboratif. Ces derniers sont également victimes d'une nouvelle génération de pirates informatiques dont le but est d'influencer les usagers en injectant de faux profils dans le système de recommandations. Ces deux constats ne sont que des exemples des nombres défis qu'il reste à relever.

Bien souvent, les recherches menées dans le domaine du filtrage collaboratif, ou plus globalement des systèmes de recommandations, s'attachent à résoudre un problème en particulier parmi les enjeux auxquels doivent faire face ce genre de systèmes. Certains s'intéressent à la mobilité des usagers en intègrant par exemple les déplacements et les co-présences des utilisateurs en un même lieu dans le calcul des prédictions (de Spindler et al., 2008). D'autres étudient les moyens d'améliorer la qualité des recommandations (Ampazis, 2008), de renforcer la confiance des usagers en justifiant les recommandations du système (Roth-Berghofer et Richter, 2008), etc. Dans le cadre de cette thèse, le défi principal qu'il s'est agi de relever est celui du passage à l'échelle. L'approche envisagée a été la distribution des calculs et nous avons rapidement constaté que de nouveaux enjeux venaient se greffer à l'objectif initial. Si nos algorithmes ont permis de repousser les limites en terme de nombre d'utilisateurs et de ressources, ils ont également intégré d'autres contraintes comme le respect de la vie privée ou la gestion des données manquantes. Nous pensons qu'un système de recommandations doit faire face conjointement à tous les problèmes énoncés en introduction de ce manuscrit, afin d'être pleinement exploitable et opérationnel en situation réelle. **Notre sentiment est que cet état ne pourra être atteint par l'emploi du filtrage collaboratif seul, mais sera rendu possible en combinant les avantages des différentes méthodes de filtrage existantes, des techniques d'Intelligence Artificielle, ainsi que des méthodologies de conception des interfaces et de modélisation des préférences et des comportements.** Le cadre idéal étant introduit, la sous-section suivante vise à présenter les objectifs à court terme et à plus longue échéance.

Nos perspectives à court terme consistent à poursuivre nos travaux dans le domaine de la génération automatique de règles d'associations (cf. supra, 4.5.2 RIBA, p.104). Il s'agira tout d'abord de modifier le modèle `RIBA` pour le rendre applicable à un contexte d'application distribué. L'objectif est d'améliorer la qualité de prédictions des modèles que nous avons proposés dans cette thèse tout en préservant leurs forces, à savoir :

- un temps de calcul faible permettant un passage à grande échelle ;
- la préservation de la vie privée des utilisateurs en utilisant uniquement des données non intrusives et peu volumineuses lors de la phase de prédictions côté client (comme par exemple la liste des règles générées, mais pas l'ensemble des profils des utilisateurs) ;
- etc.

Nous pourrions également étendre le concept des règles d'association de `RIBA` à des N-uplets d'items. Les règles pourraient alors contenir un plus grand nombre de prémisses, mais aussi plusieurs conclusions.

Sur le moyen et long terme, il serait intéressant d'étudier les apports de la combinaison du filtrage collaboratif avec différentes méthodes lors de l'élaboration de systèmes de recommandation. Une application, construite par briques facilement assemblables, servirait alors de support à des expérimentations joignant par exemple **filtrage collaboratif** et :

- **apprentissage par renforcement** ;

  L'emploi de techniques d'apprentissage par renforcement peut s'avérer utile à la détection d'utilisateurs malveillants au sein du système. Par "utilisateurs malveillants", nous entendons ici l'ensemble des usagers envoyant des votes erronés à travers le réseau afin d'influencer les autres utilisateurs ou afin de faire s'effondrer la qualité du système. Nous pourrions imaginer un algorithme dans lequel la confiance est une fonction de récompense. Gagner la confiance d'un individu est quelque chose qui prend du temps en ayant un comportement viable, mais qui se perd également très facilement dès lors qu'un comportement anormal est détecté. Le système tiendrait alors d'autant plus compte des préférences d'un individu que le degré de confiance serait élevé.

- **filtrage par contenu, navigation sociale et filtrage basé sur le contexte** ;

  Le démarrage à froid reste un des problèmes majeurs du filtrage collaboratif. Cette approche nécessite en effet de disposer d'un nombre de préférences conséquent avant de devenir efficace. La combinaison du filtrage collaboratif avec du filtrage par contenu ou de la navigation sociale permettrait, entre autres, de faire face à ce problème. A l'inverse, le filtrage collaboratif s'avère efficace et complémentaire de ces approches sur le long terme, car il permet d'élargir le panel des recommandations en bénéficiant de l'expérience d'une communauté. L'introduction du filtrage basé sur le contexte permettrait d'accroître la satisfaction des utilisateurs en affinant la liste des recommandations en fonction de la situation immédiate constatée (caractéristiques techniques de la machine, moment de la journée, etc.).

- **méthodes à base de grammaires et modèles markoviens** ;

  La combinaison du filtrage collaboratif avec des méthodes à base de grammaires et de modèles markoviens permettrait d'introduire une notion d'ordre dans la liste des recommandations. En effet, les algorithmes de filtrage collaboratif sont capables d'identifier les items potentiellement intéressants pour chaque utilisateur, mais sont incapables de détecter une quelconque notion de priorité dans les recommandations. Afin d'être appréciés des usagers, les items doivent parfois être consultés dans un ordre donné. Nous pourrions par exemple considérer des séquences de consultation d'items, en complément des votes fournis, et considérer chaque consultation comme

une lettre dans un mot. Nous disposerions alors des premières lettres d'un mot et chercherions les quelques lettres suivantes pour terminer efficacement la séquence de consultation, sur le principe des algorithmes de complétion automatique (avec utilisation d'arbres ternaires par exemple).

– **ontologies** ;

Les algorithmes de filtrage collaboratif sont bien souvent capables de fournir des recommandations pertinentes, mais peinent à expliquer aux utilisateurs les raisons poussant le système à promouvoir ces items. Une combinaison avec des ontologies permettrait de renforcer la confiance des usagers en leur expliquant à l'aide de concepts pourquoi les items recommandés peuvent s'avérer pertinents.

– **modélisation du comportement** ;

La méthode de modélisation générique des préférences que nous avons proposé dans le chapitre 1 peut faire l'objet d'études supplémentaires. Nous pourrions par exemple cherché à déterminer automatiquement les poids appropriés à chaque critère en fonction du cadre applicatif, des utilisateurs du service et du contexte. Il serait également utile d'intégrer de nouveaux critères. Nous pourrions par exemple mesurer l'attention de l'utilisateur à l'aide d'un oculomètre et introduire ce paramètre dans le calcul des votes estimés.

– **définition des interfaces**.

Un système de recommandations, même très performant, peinera à accroître la satisfaction des usagers s'il n'est pas couplé à une interface intuitive, explicite et mettant convenablement en avant les items suggérés. La définition des interfaces et le mode de représentation des données en sortie du système sont donc des facteurs importants à prendre en compte lors de l'élaboration d'un système de recommandations.

Ceci constitue bien évidemment une liste non exhaustive des combinaisons possibles, l'objectif étant d'améliorer progressivement le système sans perdre les avantages précédemment acquis. Notre souhait est de traiter en priorité les aspects liés à la sécurité, à la modélisation du comportement et la définition des interfaces car ces trois points constituent à notre sens la pierre d'angle d'un système de recommandation fiable et performant.

Table des figures

Liste des tableaux

Bibliographie

C.C. Aggarwal, J.L. Wolf, K-L. Wu et P.S. Yu. Horting hatches an egg : a new graph-theoretic approach to collaborative filtering. Dans *In Knowledge Discovery and Data Mining*, pages 201–212, 1999.

R. Agrawal et R. Srikant. Mining sequential patterns. Dans *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, March 1995.

N. Ampazis. Collaborative filtering via concept decomposition on the netflix dataset. Dans *ECAI'08 Workshop on Recommender Systems*, pages 26–30, Patras, Greece, 2008.

L. Baltrunas et F. Ricci. Dynamic item weighting and selection for collaborative filtering. Dans *Workshop PriCKL'07, in conjunction with the 18th European Conference on Machine Learning (ECML) and the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, Warsaw, Poland, September 2007.

J. Basilico et T. Hofmann. Unifying collaborative and content-based filtering. Dans *ICML '04 : Proceedings of the twenty-first international conference on Machine learning*, Banff, Alberta, Canada, 2004.

R. Bekkerman, S. Zilberstein et J. Allan. Web page clustering using heuristic search in the web graph. Dans *Proceedings of IJCAI-07, the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007.

S. Berkovsky, Y. Eytani, T. Kuflik et F. Ricci. Hierarchical neighborhood topology for privacy enhanced collaborative filtering. Dans *in CHI 2006 Workshop on Privacy-Enhanced Personalization (PEP2006)*, Montreal, Canada, April 2006.

V. Bénard. La satisfaction des utilisateurs de services en ligne. Dans *9ème salon des solutions pour la qualité et le management (SisQual 2002), session "qualité des services en ligne et leur influence sur la management par la qualité"*, Paris, France, Décembre 2002.

D. Bonnefoy, M. Bouzid, N. Lhuillier et K. Mercer. "more like this" or "not for me" : Delivering personalised recommendations in multi-user environments. Dans *Proceedings of the 11th International Conference on User Modeling (UM 2007)*, pages 87–96, Corfu, Greece, June 2007.

M. Bouzeghoub et D. Kostadinov. Personnalisation de l'information : Aperçu de l'état de l'art et définition d'un modèle flexible de profils. Dans *COnférence en Recherche d'Infomations et Applications (CORIA'05)*, Grenoble, France, Mars 2005.

A. Boyer. Du verbe à l'action : récit d'un parcours en intelligence artificielle. Dans *HDR en informatique de l'Université Nancy 2*, Nancy, France, Décembre 2004.

A. Boyer et A. Brun. Natural language processing for usage based indexing of web resources. Dans *in proc. of the 29th European Conference on Information Retrieval (ECIR 2007)*, pages 517–524, Rome, Italy, April 2007.

A. Boyer, A. Brun, A. Roussalany et A. Lelu. Projet d'équipe kiwi (knowledge, information and web intelligence). `http://kiwi.loria.fr/`, 2008.

A. Boyer, S. Castagnos, Y. Bertrand-Pierron, J. Anneheim et J-P. Blanchard. Le filtrage collaboratif : Pistes d'applications dans le domaine bancaire et présentation de la technologie. Rapport technique, Dossiers de la veille technologique du Crédit Agricole S.A., volume 27, Décembre 2006.

P.S. Bradley et U.M. Fayyad. Refining initial points for k-means clustering. Dans *In proc. of the 15th International Conference on Machine Learning (ICML98)*, pages 91–99, Madison, Wisconsin, USA, July 1998.

J.S. Breese, D. Heckerman et C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Dans *Proceedings of the fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, San Francisco, CA, July 1998.

R. Burke. Hybrid recommender systems : Survey and experiments. *Journal of Personalization Research, User Modeling and User-Adapted Interaction*, 12(4), 2002.

R. Burke. Research issues in recommender systems. Keynote Lecture in the Workshop on Recommender Systems (ECAI 2006), 2006.

R. Burke et B. Mobasher. Trust and bias in multi-agent recommender systems. Dans *Workshop on Multi-Agent Information Retrieval and Recommender Systems, in conjunction with the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, Edinburgh, Scotland, July 2005.

R. Burke, B. Mobasher, C. Williams et R. Bhaumik. Classification features for attack detection in collaborative recommender systems. Dans *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Philadelphia, USA, August 2006.

L. Candillier, F. Meyer et M. Boullé. Comparing state-of-the-art collaborative filtering systems. Dans *5th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'2007)*, Leipzig, Germany, July 2007.

J. Canny. Collaborative filtering with privacy. Dans *IEEE Symposium on Security and Privacy*, pages 45–57, Oakland, CA, May 2002a.

J. Canny. Collaborative filtering with privacy via factor analysis. Dans *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, Tampere, Finland, August 2002b.

J. Casademont, F. Perdrix, M. Einhoff, J. Paradells, G. Dummer et A. Boyer. Elin : A framework to deliver media content in an efficient way based in mpeg standards. Dans *In the IEEE International Conference on Web Services (ICWS'2005)*, pages 841–842, Orlando, Florida, USA, July 2005a.

J. Casademont, F. Perdrix, M. Einhoff, J. Paradells, G. Dummer et A. Boyer. Elin : A newspaper universal multimedia access platform based on mpeg standards. Dans *In the Fifth IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2005)*, pages 27–32, Benidorm, Spain, September 2005b.

S. Castagnos. Des agents collaboratifs pour un filtrage intelligent. Rapport technique, Mémoire de DEA (Ecole IAEM, LORIA), Juin 2004.

S. Castagnos et A. Boyer. A client/server user-based collaborative filtering algorithm : Model and implementation. Dans *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI2006)*, pages 617–621, Riva del Garda, Italy, August 2006a.

S. Castagnos et A. Boyer. Frac+ : A distributed collaborative filtering model for client/server architectures. Dans *2nd Conference on Web Information Systems and Technologies (Webist 2006)*, pages 435–440, Setùbal, Portugal, April 2006b.

S. Castagnos et A. Boyer. Modeling preferences in a distributed recommender system. Dans *Proceedings of the 11th International Conference on User Modeling (UM 2007)*, pages 400–404, Corfu, Greece, June 2007a.

S. Castagnos et A. Boyer. Modélisation des préférences pour un filtrage collaboratif distribué. Dans *Atelier "Intelligence Artificielle et Web Intelligence" (AFIA 2007)*, Grenoble, France, Juillet 2007b.

S. Castagnos et A. Boyer. Personalized communities in a distributed recommender system. Dans *Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*, pages 343–355, Rome, Italy, April 2007c.

S. Castagnos et A. Boyer. Privacy concerns when modeling users in collaborative filtering recommender systems. *Book chapter in "Social and Human Elements of Information Security : Emerging Trends and Countermeasures"*, 2008.

S. Castagnos, A. Boyer et F. Charpillet. A distributed information filtering : Stakes and solution for satellite broadcasting. Dans *Proceedings of WebIST05*, pages 299–304, Miami, USA, May 2005a.

S. Castagnos, A. Boyer et F. Charpillet. Vers un filtrage collaboratif distribué : le modèle rsb. Dans *3èmes journées francophones sur les Modèles Formels de l'Interaction (MFI 2005)*, pages 260–268, Caen, France, Mai 2005b.

S. Castagnos, A. Boyer et J-C. Lamirel. Proposal of valuation methods for the filtering algorithms. Rapport technique, Technical Report for ESA Sat'N'Surf Project (LORIA, CRPHT, SES ASTRA), August 2005c.

S. Castagnos, A. Brun et A. Boyer. Probabilistic association rules for item-based recommender systems. Dans *4th European Starting AI Researcher Symposium (STAIRS 2008), in conjunction with the 18th European Conference on Artificial Intelligence (ECAI 2008)*, Patras, Greece, July 2008a.

S. Castagnos, A. Brun et A. Boyer. Probabilistic reinforcement rules for item-based recommender systems. Dans *18th European Conference on Artificial Intelligence (ECAI 2008)*, Patras, Greece, July 2008b.

S. Castagnos et R. Kassab. Bibliography and state-of-the-art report on filtering and profiling techniques. Rapport technique, Technical Report for ESA Sat'N'Surf Project (LORIA, CRPHT, SES ASTRA), March 2005a.

S. Castagnos et R. Kassab. Selection of profiling, filtering and content analysis techniques. Rapport technique, Technical Report for ESA Sat'N'Surf Project (LORIA, CRPHT, SES ASTRA), March 2005b.

Continuity Central. Data storage, security and backup practices of us medium-sized businesses explored. `http://www.continuitycentral.com/news03878.htm`, April 2008.

P. Chan. A non-invasive learning approach to building web user profiles. Dans *Workshop on Web usage analysis and user profiling, Fifth International Conference on Knowledge Discovery and Data Mining*, San Diego, USA, August 1999.

R. Charton. Des agents intelligents dans un environnement de communication multimédia : vers la conception de services adaptatifs. Thèse de Doctorat de l'Université Henri Poincaré, Nancy, 2003.

S.H.S. Chee, J. Han et K. Wang. Rectree : An efficient collaborative filtering method. Dans *In Proceedings 2001 Int. Conf. on Data Warehouse and Knowledge Discovery (DaWaK'01)*, Munich, Germany, September 2001.

P. Cheeseman et J. Stutz. Bayesian classification (autoclass) : Theory and results. Dans *Advances in Knowledge Discovery and Data Mining*, pages 153–180. 1996.

L. Chen et P. Pu. Evaluating critiquing-based recommender agents. Dans *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI 2006)*, Boston, Massachusetts, USA, July 2006.

L. Chen et P. Pu. Hybrid critiquing-based recommender systems. Dans *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI 2007)*, Honolulu, Hawaii, USA, January 2007.

D. Chickering, D. Geiger et D. Heckerman. Learning bayesian networks : Search methods and experimental results. Dans *In proceedings of Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128, 1995.

D.M. Chickering et D. Heckerman. Efficient approximations for the marginal likelihood of bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

Inc. ChoiceStream. 2006 choicestream personalization survey. Rapport technique, Février 2006.

C.K. Chow et C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

G. Cooper et E. Hersovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

A. Cordier, B. Fuchs, J. Lieber et A. Mille. Interactive knowledge acquisition in case based reasoning. Dans *Workshop "Knowledge Discovery and Similarity in Case-Based Reasoning" (ICCBR'07)*, pages 85–94, Belfast, Northern Ireland, August 2007.

L.F. Cranor. Hey, that's personal ! Dans *Invited talk at the International User Modeling Conference (UM05)*, 2005.

R. Cöster et M. Svensson. Incremental collaborative filtering for mobile devices. Dans *2005 ACM Symposium on Applied Computing (SAC'05)*, San Fe, New Mexico, USA, March 2005.

E. Cutrell, D. Robbins, S. Dumais et R. Sarin. Fast, flexible filtering with phlat – personal search and organization made easy. Dans *Proc. of the Conference for Human-Computer Interaction (CHI 2006)*, Montréal, Canada, April 2006.

A. de Spindler, S. Leone, M. Grossniklaus et M. Norrie. Harnessing facebook for the evaluation of recommender systems based on physical copresence. Dans *ECAI'08 Workshop on Recommender Systems*, pages 20–25, Patras, Greece, 2008.

A. Dempster, N. Laird et D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B39:1–38, 1977.

Y. Ding, X. Li et M.E. Orlowska. Recency-based collaborative filtering. Dans *In Proc. Seventeenth Australasian Database Conference (ADC2006)*, pages 99–107, Hobart, Australia, 2006.

C. Fauré, S. Delprat, J-F. Boulicaut et A. Mille. Iterative bayesian network implementation by using annotated association rules. Dans *In Proc. 15th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW'06)*, pages 326–333, Podebrady, Czech Republic, 2006a.

C. Fauré, S. Delprat, A. Mille et J-F. Boulicaut. Construction itérative d'un modèle de connaissance par l'exploitation de règles d'asssociation. Dans *Dans Actes des 17ème Journées Francophones Ingénierie des Connaissances (IC'06)*, pages 1–10, Nantes, France, 2006b.

O. François et P. Leray. Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux bayésiens. Dans *In Proceedings of RJCIA03, plate-forme AFIA03*, pages 167–180, 2003.

N. Friedman. Learning belief bayesian networks in the presence of missing values and hidden variables. Dans *In ML '97*, 1997.

N. Friedman. The bayesian structural em algorithm. Dans In Gregory F. Cooper et Serafín Moral editors, éditeurs, *Proceedings of the fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138, San Francisco, CA, July 1998. Morgan Kaufmann Publishers.

Y. Fu, K. Sandhu et M. Shih. A generalization-based approach to clustering of web usage sessions. Dans *Proceedings of the 1999 KDD Workshop on Web Mining*, volume 1836, pages 21–38, San Diego, CA, 2000. Springer-Verlag.

K.Z. Gajos, K. Everitt, D.S. Tan, M. Czerwinski et D.S. Weld. Predictability and accuracy in adaptive user interfaces. Dans *CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1271–1274, Florence, Italy, 2008.

D. Goldberg, D. Nichols, B. Oki et D. Terry. Using collaborative filtering to weave an information tapestry. Dans *Communications of the ACM, Special Issue on Information Filtering*, volume 35(12), pages 61–70. ACM Press, 1992.

K. Goldberg, T. Roeder, D. Huptan et C. Perkins. Eigentaste : a constant time collaborative filtering algorithm. Rapport technique M00/41, IEOR and EECS Departments, UC Berkeley, August 2000.

P. Han, B. Xie, F. Yang, J. Wang et R. Shen. A novel distributed collaborative filtering algorithm and its implementation on p2p overlay network. Dans *Proc. of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD04)*, Sydney, Australia, May 2004.

J.L. Herlocker, J.A. Konstan, A. Borchers et J. Riedl. An algorithmic framework for performing collaborative filtering. Dans *In Proceedings 1999 Conference of Research and Development in Information Retrieval*, pages 230–237, Berkeley, CA, August 1999.

J.L. Herlocker, J.A. Konstan, L.G. Terveen et J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

K. Hoashi, K. Matsumoto, N. Inoue et K. Hashimoto. Query expansion based on predictive algorithms for collaborative filtering. Dans *SIGIR '01 : Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, Louisiana, United States, 2001.

E. Horvitz et A. Klein. Utility-based abstraction and categorization. Dans *Proc. of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI93)*, pages 128–135, Providence, Washington, DC, USA, July 1993.

C. Huang et A. Darwiche. Inference in belief networks : a procedural guide. *International Journal of Approximate Reasoning*, 1994.

H. Jacquet et L. Drouot. Rapport de synthèse du projet e-veille. Rapport technique, Erdyn consultants, Février 2007.

R. Kassab, J-C. Lamirel et E. Nauer. Une nouvelle approche pour la modélisation du profil de l'utilisateur dans les systèmes de filtrage d'information : le modèle de filtre détecteur de nouveauté. Dans *2ème COnférence en Recherche d'Infomations et Applications (CORIA'2005)*, pages 185–200, Grenoble, France, Mars 2005.

T. Kurki, S. Jokela, R. Sulonen et M. Turpeinen. Agents in delivering personalized content based on semantic meta-data. Dans *Proceedings of the 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace*, pages 84–93, 1999.

S. Lauritzen et D. Speigelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Royal statistical society B*, 50:157–224, 1988.

I. Lavallée. *Complexité et algorithmique avancée : une introduction.* Hermann, 2008.

D.B. Leake. *Case-Based Reasoning : Experiences, Lessons and Future Directions.* AAAI Press/MIT Press, 1996.

M. Lethielleux. *Statistique descriptive (4ème édition).* Dunod, 2005.

R. Lhoste. Archimaj, une plate-forme multi-agents pour la recherche d'informations. Mémoire de DRT. Université Nancy 2, 25 Novembre, 2003.

B. Liu. Web content mining. WWW-2005 Tutorial, Chiba, Japan, May 2005.

N. Lumineau. Tour d'horizon du filtrage collaboratif. Rapport technique, LIP6, Paris, France, Décembre 2003.

P. Lyman et H.R. Varian. How much information ? `http://www.sims.berkeley.edu/howmuch-info-2003/`, 2003.

F. Masseglia, D. Tanasa et B. Trousse. Diviser pour découvrir. une méthode d'analyse du comportement de tous les utilisateurs d'un site web. *RSTI - Ingénierie des systèmes d'information (ISI)*, 9(1):61–83, 2004.

P. Melville, R.J. Mooney et R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. Dans *Proceedings of the Eighteenth National Conference on Artificial Intelligence(AAAI-2002)*, pages 187–192, Edmonton, Canada, July 2002.

D.R. Millen, J. Feinberg et B. Kerr. Dogear : Social bookmarking in the enterprise. Dans *Proc. of the Conference for Human-Computer Interaction (CHI 2006)*, Montréal, Canada, April 2006.

B.N. Miller, J.A. Konstan et J. Riedl. Pocketlens : Toward a personal recommender system. *ACM Transactions on Information Systems*, 22, July 2004.

K. Miyahara et M.J. Pazzani. Collaborative filtering with the simple bayesian classifier. Dans *In Proceedings of the sixth Pacific Rim International Conference on Artificial Intelligence*, Australia, 2000.

B. Mobasher, H. Dai, T. Luo et M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1):61–82, 2002.

M. Montaner, B. López et J.L. De La Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19:285–330, 2003.

P. Naïm, P-H. Wuillemin, P. Leray, O. Pourret et A. Becker. *Réseaux bayésiens*. Eyrolles (2ème édition), 2004.

A-T. Nguyen, N. Denos et C. Berrut. Modèle d'espaces de communautés basé sur la théorie des ensembles d'approximation dans un système de filtrage hybride. Dans *COnférence en Recherche d'Information et Application (CORIA 2006)*, Lyon, France, mars 2006.

J.W. Payne, J.R. Bettman et E.J. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, 1993.

D.M. Pennock, E. Horvitz et C.L. Giles. Social choice theory and recommender systems : Analysis of the axiomatic foundations of collaborative filtering. Dans *AAAI/IAAI*, pages 729–734, 2000a.

D.M. Pennock, E. Horvitz, S. Lawrence et C.L. Giles. Collaborative filtering by personality diagnosis : a hybrid memory- and model-based approach. Dans *Proceedings of UAI-2000*, San Francisco, USA, 2000b.

S. Perugini, M.A. Gonçalves et E.A. Fox. A connection-centric survey of recommender systems research. Dans *Journal of Intelligent Information Systems*, volume 23, 2003.

H. Polat et W. Du. Svd-based collaborative filtering with privacy. Dans *Proc. of ACM Symposium on Applied Computing*, Cyprus, 2004.

F. Rasheed, Y-K. Lee et S. Lee. Context summarization and garbage collecting context. Dans *Proceedings of the International Conference on Computational Science and its Applications (ICCSA 2005)*, Singapore, May 2005.

S. Ratnasamy, P. Francis, M. Handley, R. Karp et S. Shenker. A scalable content addressable network. Dans *In Proceedings of the 2001 ACM SIGCOMM Conference*, San Diego, CA, USA, August 2001.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm et J. Riedl. Grouplens : An open architecture for collaborative filtering of netnews. Dans *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

T.R. Roth-Berghofer et M.M. Richter. On explanation. *Künstliche Intelligenz*, 22(2):5–7, May 2008.

A. Rowstron et P. Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218, 2001.

S. Russell et P. Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, 1995.

G. Salton et M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

B.M. Sarwar, G. Karypis, J.A. Konstan et J. Reidl. Item-based collaborative filtering recommendation algorithms. Dans *World Wide Web*, pages 285–295, 2001.

A.I. Schein, A. Popescul, L.H. Ungar et D.M. Pennock. Methods and metrics for cold-start recommendations. Dans *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, page 253–260, New York City, New York, 2002.

V. Schickel et B. Faltings. Using an ontologcial a-priori score to infer user's preferences. Dans *Workshop on Recommender Systems, in Conjunction with the 17th European Conference on Artificial Intelligence (ECAI 2006)*, Riva del Garda, Italy, August 2006.

G. Schwarz. *Estimating the dimension of a model*, volume 6 de *Annals of Statistics*. 1978.

U. Shardanand et P. Maes. Social information filtering : Algorithms for automating "word of mouth". Dans *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

H.A. Simon. *Economic Analysis and Public Policy*, volume 1 and 2 de *Models of Bounded Rationality*. 1982.

B. Smyth, E. Balfe, O. Boydell, K. Bradley, P. Briggs, M. Coyle et J. Freyne. A live-user evaluation of collaborative web search. Dans *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, page 1419–1424, Edinburgh, Scotland, 2005.

P. Spirtes, C. Glymour et R. Scheines. *Causation, Prediction and Search*. The MIT Press, 2nd edition, 2000.

I. Stoica, R. Morris, D. Karger, F. Kaashoek et H. Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. Dans *In Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, San Diego, CA, USA, August 2001.

P. Thurrott. Windows live preview. Review on the Paul Thurrott's SuperSite for Windows and for the Windows IT Pro Magazine. `http://www.winsupersite.com/reviews/windowslive_preview.asp`, 2006.

B. Trousse, M. Jaczynski et R. Kanawati. Une approche fondée sur le raisonnement à partir de cas pour l'aide à la navigation sur le web. Dans *Proceedings of Hypertexte and Hypermedia : Products, Tools and Methods (H2PTM'99)*, pages 13–26, Paris, France, August 1999.

L. Ungar et D. Foster. Clustering methods for collaborative filtering. Dans *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California, 1998.

E. Vildjiounaite et V. Kyllönen. Learning context-dependency of user interests in smart homes : Comparison between svm and cbr. Dans *Artificial Intelligence and Soft Computing*, pages 52–57, 2006.

W3C. Logging control in w3c httpd. http ://www.w3c.org/Daemon/User/Config/Logging.html, 1995.

Y. Wang et A. Kobsa. Impacts of privacy laws and regulations on personalized systems. Dans *In CHI 2006 Workshop on Privacy-Enhanced Personalization (PEP 2006)*, Montreal, Canada, April 2006.

A. Westin. Privacy and freedom. Talk at the Atheneum. New York, USA, 1967.

B. White. The implications of web 2.0 on web information systems. Keynote Lecture in the International Conference on Web Information Systems and Technologies (WebIST 2006), 2006.

N. Wingfield et J. Pereira. Amazon uses faux suggestions to promote new clothing store. Dans *Wall Street Journal*, December 2002.

X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z-H. Zhou, M. Steinbach, D.J. Hand et D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, 2007. ISSN 0219-1377.

J. Zaslow. If tivo thinks you are gay, here's how to set it straight : What you buy affects recommendations on amazon.com, too ; why the cartoons ? The Wallstreet Journal, November 2002. URL `http://online.wsj.com/article_email/0,,SB1038261936872356908,00.html`.

I. Zitouni, K. Smaïli et J-P. Haton. Statistical language modeling based on variable-length sequences. *Computer Speech and Language*, 17(1):27–41, 2003.

Résumé

Internet constitue un environnement évolutif déstructuré et quasi-infini proposant des documents hétérogènes notamment à travers le Web et les intranets d'entreprises. La recherche et l'accès à cette profusion de documents nécessite d'assister l'utilisateur. Cependant, les outils actuels d'accès à l'information atteignent leur limite et ne garantissent plus d'identifier les ressources les plus pertinentes (également appelées "items") dans un temps raisonnable. La problématique consiste à "apprendre l'utilisateur courant". La connaissance de ce dernier permet au système de fournir des items susceptibles de les intéresser ou de répondre à un critère d'utilité. Il s'agit alors de collecter des données brutes pour caractériser une information de haut niveau, à savoir la connaissance de l'utilisateur. L'emploi de l'Intelligence Artificielle permet d'identifier les données nécessaires et suffisantes à l'apprentissage supervisé en situation de l'utilisateur courant.

Toutefois, les modèles utilisateurs souffrent d'un grand nombre de données manquantes. Notre approche consiste à exploiter collaborativement les données relatives à une population pour pallier le manque d'information inhérent à chaque utilisateur. L'emploi de techniques de filtrage collaboratif permet ainsi de bénéficier de l'expérience et des interactions au sein d'une population pour améliorer les services et prédire les futurs agissements d'un individu. Nous sommes partis du constat que, dans les approches centralisées, le nombre d'individus pris en compte dans la recherche des plus proches voisins ne peut excéder quelques milliers de candidats. Nos travaux nous ont donc conduit à distribuer le processus de filtrage sous plusieurs formes tant en terme de contenu que de calculs. L'objectif de cette thèse est de montrer comment il est possible d'assurer le passage à l'échelle, et faire face aux problèmes sous-jacents pouvant résulter de cette approche distribuée.

Mots-clés: Filtrage collaboratif distribué, Modélisation stochastique des préférences, Systèmes de recommandations, Apprentissage non-supervisé

Abstract

Internet constitutes an unstructured, almost infinite, and evolutive environment supplying heterogeneous documents (also called "items" or "resources") through the Web and a set of associated services. As there is a huge number of documents on Internet, the search and access to data involve assisting the active user. However, the usual search engines come up to their limit. Search engines provide too many results to ensure that the active user will identify the most relevant items in a reasonable time. Supplying the active user with his/her concerns requires efficient data collection methods to model accurately user's preferences in a stochastic context. However, it is not always possible to fully model the user in time because it is a lengthy process. Collaborative filtering algorithms bypass this difficulty by exploiting knowledge about a similar population to complete missing data in the active user's model and to do preference elicitation. They amount to identifying the active user to a set of persons having the same tastes, based on his/her preferences and his/her past actions. This kind of algorithms considers that users who liked the same items have the same topics of interest. Thus, it is possible to predict the relevancy of data for the active user by taking advantage of experiences of a similar population.

In centralized collaborative filtering approaches, finding the closest neighbors among several thousands of candidates in real time without offline computations may be unrealistic. The scientific problem consequently consists, within the context of my thesis, in finding a way to distribute the collaborative filtering process in order to deal with scalability and other underlying constraints.

Keywords: Distributed Collaborative Filtering, Stochastic Modeling of Preferences, Recommender Systems, Unsupervised Learning

---

Laboratoire Lorrain de Recherche en Informatique et Applications

Equipe MAIA/KIWI

Campus Scientifique, B.P.239 - 54506 Vandœuvre-lès-Nancy Cedex