Apprentissage automatique de profils de lecteurs
Mémoire de DEA

Laurent Candillier
Equipe GRAPPA, Lille 3
Laboratoire d'Informatique Fondamentale de Lille

Juin 2001

Remerciements à :
Isabelle Tellier, Rémi Gilleron, Fabien Torre,
Aurélien Lemay, Marc Tommasi, et François Denis.

Table des matières

Chapitre 1

Introduction

Le volume d'informations disponibles sur Internet ne cessant d'augmenter chaque jour, il survient aujourd'hui un problème de plus en plus important de *surcharge de données*, lors d'une recherche d'un utilisateur sur le réseau mondial. Il devient donc de plus en plus nécessaire de développer des outils permettant de filtrer cet ensemble important de données, pour cibler au mieux les réponses fournies aux utilisateurs, afin qu'elles soient plus proches de leurs attentes et de leurs goûts personnels.

L'*apprentissage automatique de profils de lecteurs* s'inscrit dans ce cadre de recherche. Le but de cette étude est de trouver la meilleure caractérisation possible de chaque internaute, qui permettra ensuite de lui recommander efficacement les pages qui seront susceptibles de l'intéresser.

Cette étude s'effectue en outre en collaboration avec la société Rosebud, qui gère des sites de revues telles "Le Point" ou "La Recherche", et qui serait intéressée par cette technologie pour être plus proche de ses clients. Dans ce cas, le contenu des pages est donc principalement de nature textuelle. Cependant, les données de Rosebud n'étant pas encore disponibles, pour le stage de DEA, nous avons mis l'accent sur les techniques n'utilisant pas explicitement le contenu des pages parcourues, mais plutôt l'ensemble des appréciations sur ces pages ; et les tests ont été effectués sur les données du site de l'équipe GRAPPA.

Plus précisément, la recherche de profils de lecteurs consiste donc en fait à utiliser au mieux les informations que nous pouvons obtenir sur les visiteurs des sites (des appréciations sur ce qu'ils ont lu, ou le contenu de ce qu'ils ont lu). L'idée est d'en dériver des profils de lecteurs (un ensemble de notes sur certaines pages, ou un ensemble de thèmes ou mots récurrents dans leurs lectures) et/ou des communautés de lecteurs. Ceux-ci seront ensuite utilisés pour aider les utilisateurs à sélectionner les pages qui sont susceptibles de

leur plaire, étant donné ce que nous savons de ce qu'ils ont apprécié jusqu'alors.

Cette problématique complexe rattache naturellement le projet à au moins deux domaines de recherche différents, exploitant des stratégies différentes. Le premier domaine est celui du *filtrage collaboratif* qui se fixe pour objectif de deviner l'évaluation qu'un utilisateur pourra avoir d'un service, en fonction de ses goûts personnels déjà identifiés et des évaluations déjà disponibles du même service par d'autres utilisateurs, dont les goûts ont été reconnus comme proches du premier. Dans ce cas, le contenu du service lui-même (dans notre cas : les textes) n'est pas analysé, seules importent les opinions qui ont été émises à leur encontre par leurs lecteurs précédents.
L'autre stratégie, qui relève cette fois de l'*extraction d'information*, consiste à prendre vraiment en compte le contenu des documents mis à disposition des internautes sur les sites qu'ils visitent. Elle peut se faire à différents niveaux de précision. Une première approche rudimentaire peut s'en tenir au balisage des textes (spécifiant par exemple leur auteur, leur date de parution ou la rubrique de la revue à laquelle ils se rattachent) et aux mots clés qui leur ont été associés par indexation, manuelle ou automatique. Une démarche plus précise mais plus coûteuse en calculs utilisera comme données l'ensemble des mots du textes, avec là-encore des variantes possibles : les mots peuvent être pris soit isolément soit par groupes, on peut ou non les normaliser... Enfin, une approche résolument linguistique pourra tenter de mettre en œuvre des outils comme les analyseurs syntaxiques ou les thesaurus. Ces choix possibles sont détaillés dans la suite de ce document.

Bien qu'utilisant des stratégies différentes, les deux approches se rattachent pourtant au même domaine informatique : la *classification*. Le problème de la classification en générale est de contruire une procédure permettant d'associer une classe à un objet. Ce problème se décline en deux variantes : l'approche *supervisée* et l'approche *non-supervisée*. Dans la première, on connaît les classes possibles et on dispose d'un ensemble d'objets déjà classés, servant d'ensemble d'apprentissage. Le problème est alors d'être capable d'associer à tout nouvel objet sa classe la plus adaptée, en se servant des exemples. Dans la seconde (la classification non-supervisée), les classes possibles ne sont pas connues à l'avance. Le but est donc de regrouper dans un même *cluster* (ou groupe) les objets considérés comme similaires, pour constituer les classes. Puis de la même façon que pour la classification supervisée, on regarde ensuite, pour tout nouvel objet, le cluster qui lui correspond le mieux, et on lui associe alors la classe de ce cluster, en l'y intégrant. Pour l'approche filtrage collaboratif, les classes sont alors les notes qu'un utilisateur attribue à un article. En fonction des notes déjà attribuées à un article par d'autres utilisateurs, et en fonction de ce qu'on connaît de l'utilisateur considéré, il s'agit en effet d'attribuer la note qui lui correspond le mieux.

Mais la nature numérique de cette classe permettra des traitements plus spécifiques. Pour l'approche analyse de contenu, les classes correspondent aux thématiques générales abordées par l'article considéré. En fonction des mots composant le document considéré, et des classes déjà attribuées à d'autres documents, il s'agit d'affecter au document considéré la classe des documents dont le contenu se rapproche le plus du sien.

La notion de profil peut elle aussi donner lieu à des interprétations variées. Ainsi, elle peut servir soit à caractériser des groupes d'utilisateurs, soit des goûts individuels. Elle peut également être codée par des structures de données diverses : vecteurs de notes associés à des descripteurs pour le filtrage collaboratif, ou vecteurs de thèmes ou mots préférés pour l'analyse de contenu, etc...

Dans le cadre d'un stage de DEA, il ne saurait être question d'aborder l'ensemble des questions posées par ce projet. Nous avons donc centré le stage de DEA sur les points suivants :

- constituer un cahier des charges du projet,
- étudier la génération de notes des utilisateurs sur les pages qu'ils parcourent,
- implémenter les méthodes de filtrage collaboratif selon *Pearson* et *Bayes*, et implémenter d'autres méthodes ne prenant pas en compte la description des utilisateurs, mais seulement l'ensemble des notes sur chaque page (*vote majoritaire* et *votes moyens*), analyser les résultats obtenus, et évaluer l'intérêt de chaque approche.

Organisation de la recherche

Nous proposons de découper la recherche en plusieurs étapes comme suit :

- Tout d'abord, nous travaillerons sur la façon d'évaluer le taux de satisfaction d'un utilisateur pour une page donnée. Celle-ci peut se faire avec ou sans la participation de l'utilisateur.
- La seconde étape sera l'étude des différentes méthodes de filtrage collaboratif.
- Ces deux étapes feront l'objet d'un premier chapitre de présentation des différentes méthodes de filtrage collaboratif, puis d'un deuxième chapitre présentant les implémentations, tests et analyse des résultats, que nous avons effectués lors du stage de DEA pour étudier et comparer ces méthodes.
- La troisième étape de notre recherche sera l'étude des différentes méthodes de classification de texte, et la recherche du niveau d'analyse linguistique le plus approprié à appliquer aux textes.
- La suivante sera l'étude de la combinaison de ces deux approches (filtrage collaboratif versus analyse de contenu).

– Ces deux autres étapes formeront alors un troisième chapitre dans notre rapport, que nous intitulerons “perspectives de recherche”.
– Et enfin la dernière étape de la recherche sera la formalisation de la notion de profil.
– Ces derniers points constitueront le thème de la recherche que nous proposons de poursuivre en thèse.

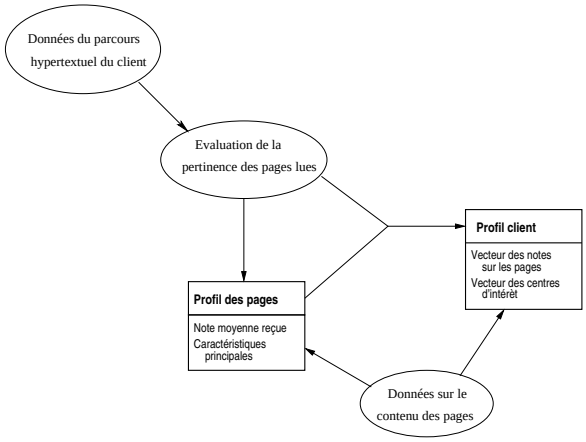

FIG. 1.1 – Schéma (très) général de la recherche de profils à partir de parcours hypertextuels de clients

Chapitre 2

Présentation des méthodes de filtrage collaboratif

La motivation du *Collaborative Filtering* (terme introduit en 1994) est d'étendre la notion de *bouche à oreille entre amis* à des milliers de personnes sur Internet : vos amis (quelques personnes) peuvent vous recommander ce qu'ils ont apprécié ; sur Internet des milliers d'individus sont susceptibles de vous donner leur avis.

Les objets pour lesquels on veut évaluer l'intérêt des internautes peuvent être de toute sorte : films, restaurants, jeux, blagues, articles, etc... Dans le cadre de la collaboration avec la société Rosebud, les objets pour lesquels nous tentons d'évaluer l'intérêt porté par les utilisateurs sont les pages des sites web qu'ils gèrent. C'est donc dans le cadre de l'évaluation de pages web que nous nous plaçons dans ce rapport.

La figure 2.1 présente l'organisation générale d'un système de recommandation. Typiquement, les étapes du système sont les suivantes :
- collecter les appréciations des utilisateurs sur les pages qu'ils parcourent,
- intégrer ces informations dans les profils d'utilisateur,
- et utiliser ceux-ci ensuite pour aider les utilisateurs dans leurs prochaines recherches.

Typiquement, ensuite, un utilisateur peut demander au système de :
- suggérer une page susceptible de lui plaire,
- lister les pages qu'il ne devrait pas apprécier,
- ou faire une prédiction sur l'appréciation qu'il donnerait sur une certaine page.

Chaque recommandation est accompagnée d'une mesure de confiance dé-

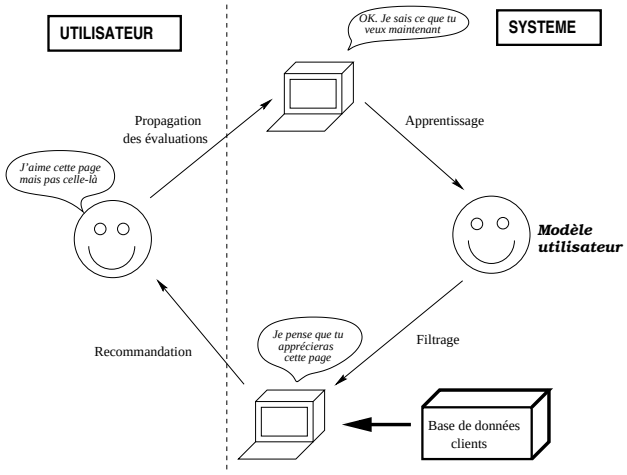

FIG. 2.1 – Schéma général d'un système de recommandation

pendant de facteurs tels que le nombre d'utilisateurs utilisé pour la recommandation, la consistance de la valeur fournie par ces utilisateurs, etc...

2.1 Première étape : l'évaluation des articles par les utilisateurs

Tout d'abord, la première notion à prendre en compte est celle du taux de satisfaction des utilisateurs pour les pages qu'ils parcourent. En effet, pour caractériser un profil d'utilisateur, la première chose à comprendre, c'est si ce qu'il a lu lui a plus ou moins plu, ou pas du tout. Pour cela, deux choix sont possibles : demander à l'utilisateur de noter lui-même les pages à partir d'une échelle de notes fixée (par exemple de 0-Nul à 7-génial), ou bien évaluer automatiquement cette note, grâce aux informations que l'on peut récolter à partir des données du protocole de communication Internet.

2.1.1 Évaluation avec investissement de l'utilisateur

L'idéal pour apprendre les goûts, et donc le profil d'un utilisateur, est que celui-ci nous donne son avis lui-même sur le plus de documents possible de la base de données. Pour cela, la plupart des systèmes de recommandation demandent à leurs nouveaux utilisateurs de participer à la définition de leur profil initial, en passant d'abord par une phase de notation, selon une échelle de notes fixée, d'articles qui leurs sont proposés. Pour cette méthode, un problème est alors de trouver l'échelle de notes la plus adaptée pour permettre une notation aussi précise que possible des articles par les utilisateurs. Une

autre problématique est celle du choix de l'échantillon : comment le générer ? propose-t-on le même à tous ? quelle quantité d'articles suggérer ? demande-t-on à l'utilisateur de noter chaque nouvel article qu'il lira dorénavant ?

Remarquons à ce niveau que la façon de noter des articles varie fortement selon l'utilisateur. Par exemple, certains ne notent que les articles qui leur plaisent, alors que d'autres utilisent l'entièreté de l'échelle proposée. Il faut donc bien comprendre ce que chaque note signifie pour chaque utilisateur. De plus, si on fournit la même liste d'articles à un utilisateur à différents moments, les notes attribuées aux articles peuvent déjà être légèrement différentes.

2.1.2 Évaluation sans participation de l'utilisateur

Si l'on veut éviter à l'utilisateur de s'investir dans la définition de son profil, au lieu de lui demander si une page lui a plu ou non, on peut le deviner, en se servant des informations que l'on peut obtenir lors de son passage sur le site. En effet, chaque fois qu'un internaute visite une page d'un site web, il laisse derrière lui des traces de son passage. Pour traiter ce problème, P.K. Chan [Cha99] propose de développer un *Page Interest Estimator*, pour prédire si la page proposée a été appréciée ou non. Pour cela, quatre sources générales d'informations ont été identifiées : *l'historique* et *le bookmark* du côté utilisateur ; *l'access log* et *le contenu des pages* du côté serveur :

1. Typiquement, *l'historique global* d'un browser maintient une marque du dernier moment où chaque page a été visitée. On peut donc utiliser cette donnée pour calculer combien de fois un utilisateur passe sur une page et depuis quand il n'est pas retourné la visiter. On conjecture alors qu'une plus haute fréquence de visites, et des visites plus récentes d'une URL, indique un intérêt plus fort de l'utilisateur pour l'URL concernée.

2. Chaque entrée dans un *access log* correspond à une requête http, qui contient typiquement l'adresse IP du client, une marque du moment de connexion, des méthodes d'accès, une URL, un protocole, un status, et une taille de fichier.

   Typiquement, une ligne de fichier de log se présente comme suit :

   216.221.34.111 - - [22/Apr/2001 :04 :04 :54 +0200] "GET /polys/access-1997/node2.html HTTP/1.1" 200 2856

   – 216.221.34.111 correspond au numéro IP du client,
   – 22/Apr/2001 :04 :04 :54 correspond au moment de connexion,
   – GET correspond à la méthode d'accès,

- /polys/access-1997/node2.html est l'URL de la page qui a été visitée,
- HTTP/1.1 correspond au protocole de communication,
- 200 est la valeur de retour (ici, tout s'est bien passé),
- et 2856 correspond à la quantité d'informations transférées (généralement la taille du fichier).

Grâce à ces informations, le temps passé sur chaque page peut être calculé. Cependant, le temps passé sur une page dépendant également de la longueur de cette page, l'intérêt d'un utilisateur pour une page sera approximé par le temps passé sur la page, normalisé par la taille de la page. Notons également que d'autres activités que le surf sur le web (par exemple répondre à un coup de téléphone) peuvent être inclues dans le temps passé sur une page ; mais on ne peut manifestement pas éviter ce problème. Pour réduire cette limitation, on peut par contre fixer une borne supérieure pour le temps (10 minutes par exemple).

3. On peut ensuite également supposer qu'une URL présente dans le *bookmark* d'un utilisateur est considérée comme très intéressante pour celui-ci.

4. Enfin, chaque page contient des *liens vers d'autres pages*. On suppose que si un utilisateur est intéréssé par une page, il va probablement visiter les liens référencés par celle-ci. Ainsi, on conjecture qu'un fort pourcentage de liens visités à partir d'une page dénote un intérêt spécial pour cette page.

Finalement, P.K. Chan définit le degré d'intérêt d'une page par :

$$Interest(Page) = Frequency(Page) \times$$
$$(1 + IsBookmark(Page) + Duration(Page) + Recency(Page) + LinkVisitPercent(Page)),$$

$Frequency(Page)$, la fréquence de visite de la page, est considérée comme l'information la plus discriminante quant à l'intérêt porté par un utilisateur sur une page. L'utilisation de "1" dans la parenthèse sert à éviter au terme entre parenthèse de s'annuler. Avec cette formule, l'intérêt pour une page est alors compris entre $Frequency(Page)$ et $5 \times Frequency(Page)$.

$$IsBookmark(Page) = \begin{cases} 1 & \text{si la page appartient au bookmark de l'utilisateur} \\ 0 & \text{sinon} \end{cases}$$

$$Duration(Page) = \frac{TotalDuration(Page)/Size(Page)}{max_{page \in VisitedPage} TotalDuration(Page)/Size(Page)},$$

$$Recency(Page) = \frac{Time(LastVisit) - Time(StartLog)}{Time(Now) - Time(StartLog)},$$

et

$$LinkVisitPercent(Page) = \frac{NumberOfLinksVisited(Page)}{NumberOfLinks(Page)}.$$

Puis il nous faudra ensuite normaliser ces valeurs pour qu'elles appartiennent à l'intervalle qu'on se sera fixé pour l'échelle de notes sur les articles.

On ne peut cependant pas supposer que toute page non visitée n'est pas intéressante, puisque l'utilisateur ne connait sûrement même pas son existence. Une approche pour identifier les pages inintéressantes serait de considérer les liens non suivis par l'utilisateur sur une page visitée, ou bien les pages quittées trop tôt (et il nous faudrait dans ce cas rechercher le seuil temporel le plus adapté pour séparer les pages intéressantes des autres).

Notons enfin que d'autres informations, contenues dans les *cookies* (fichiers utilisateur), peuvent être très instructives. Mais ce qui est écrit dans ces fichiers étant propre à chaque site, nous ne pourrons exploiter ces informations que dans un cadre précis, où l'on connait les traces laissées dans ces fichiers.

La société Rosebud avec laquelle nous travaillons serait intéressée par ce genre de méthode car ses responsables veulent éviter à leurs clients de s'investir dans la définition de leur profil. Pour plus de détails pour notre application, il nous faut pour l'instant attendre que Rosebud nous transmette les données qu'ils collectent à partir des différentes visites de leurs clients sur leurs sites. En attendant, les expérimentations ont donc été effectuées sur les fichiers de log du site de l'équipe GRAPPA.

2.1.3 Stockage des données

Les notes obtenues sont alors stockées dans la base de données utilisateur.

Typiquement, on représente ces données par une matrice de notes des utilisateurs sur les pages qu'ils ont parcourues.
La base de données utilisateur se présente alors sous la forme du tableau 2.1 suivant (que nous utiliserons comme exemple dans la suite de ce chapitre).

Formellement, cette matrice correspond à en un ensemble de votes $v_{i,j}$ des utilisateurs (i) sur les pages (j). Par exemple ici, on a $v_{3,3} = 6$.

	Article 1	Article 2	Article 3	Article 4	Article 5
Utilisateur 1			7	6	
Utilisateur 2			5	6	7
Utilisateur 3			6	6	7
Utilisateur 4	7	5			7
Utilisateur 5	7	6			7

TAB. 2.1 – Une matrice des notes attribuées aux articles par les utilisateurs

$\overrightarrow{v_i}$ correspond au vecteur décrivant l'utilisateur i (l'ensemble de ses votes, $\{v_{i,j}|i\}$). Dans notre exemple, on a $\overrightarrow{v_3} = (0, 0, 6, 6, 7)$.

On utilise la notation I_i pour désigner l'ensemble des pages pour lesquelles l'utilisateur i a voté. On a alors $I_3 = \{3, 4, 5\}$.

Et nous noterons $p_{i,j}$ l'estimation du vote de l'utilisateur i sur la page j, le but étant que $p_{i,j}$ soit le plus proche possible de $v_{i,j}$. Nous verrons alors dans la suite quelle est l'estimation $p_{3,3}$ du vote de l'utilisateur 3 sur la page 3, pour les méthodes présentées.

2.2 Les méthodes de filtrage collaboratif

Deux classes d'algorithmes de filtrage collaboratif, exploitant ensuite ces données pour aider les utilisateurs dans leurs futures recherches, ont été distinguées [BHK98] :

- les *algorithmes basés sur la mémoire* utilisent l'entièreté de la base de données utilisateur pour faire des prédictions,
- alors que les *algorithmes basés sur les modèles* utilisent la base de données utilisateur pour estimer ou apprendre un modèle, qui sera ensuite utilisé pour les prédictions.

2.2.1 Méthodes basées sur la mémoire

L'idée de base des systèmes basés sur la mémoire est la suivante :

- Le système maintient un profil utilisateur, c'est-à-dire un enregistrement des intérêts (aussi bien positifs que négatifs) de l'utilisateur pour certains articles.
- Puis il compare ce profil avec les profils des autres utilisateurs, et pèse chaque profil en fonction de son degré de similarité avec le profil de l'utilisateur considéré.
- Enfin, il considère un ensemble des profils les plus similaires (ou les plus opposés), et utilise l'information qu'ils contiennent pour recommander à l'utilisateur (ou mettre en garde contre) des articles qu'il n'a pas encore évalués.

Pour prédire la pertinence d'un article pour un utilisateur, on calcule donc la moyenne des notes données aux articles par les utilisateurs ayant les mêmes goûts (ou des goûts opposés), en utilisant des poids différents selon la mesure de similarité entre utilisateurs.

Pearson

Formellement, la base de données utilisateur consiste donc en un ensemble de votes $v_{i,j}$ correspondant au vote de l'utilisateur i sur l'article j. Si I_i désigne l'ensemble des articles pour lesquels l'utilisateur i a voté, alors on définit le vote moyen pour l'utilisateur i par :

$$\overline{v_i} = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

Soit a l'utilisateur considéré (actif). Alors l'estimation du vote de l'utilisateur actif pour l'article j, $p_{a,j}$, est le vote pondéré des autres utilisateurs :

$$p_{a,j} = \overline{v_a} + \frac{\sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v_i})}{\sum_{i=1}^{n} |w(a,i)|},$$

avec n le nombre d'utilisateurs de la base dont le poids n'est pas nul, et ayant notés l'article j. Les poids $w(a,i)$ peuvent refléter distance, corrélation, ou similarité entre chaque utilisateur i et l'utilisateur actif a.

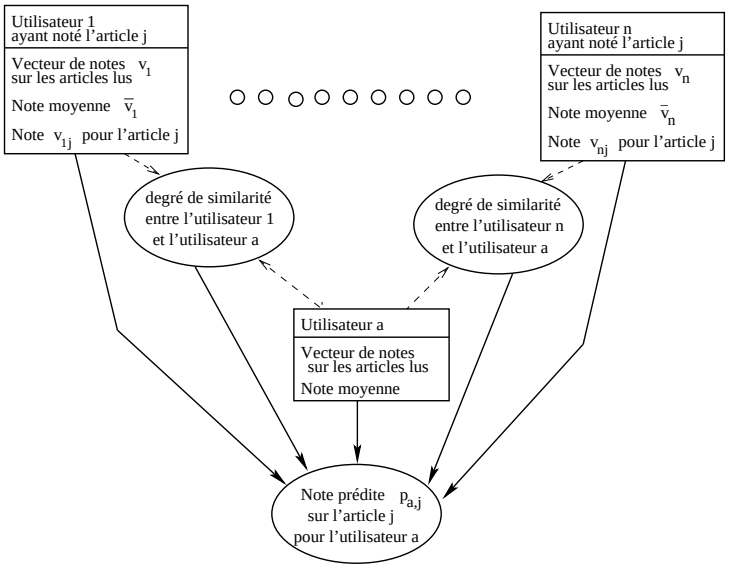

FIG. 2.2 – Recommander un article à un utilisateur

Pour affecter les poids entre utilisateurs, typiquement, la similarité entre utilisateurs est mesurée en représentant chaque utilisateur comme un vecteur de votes pour les articles de la base, puis en calculant le cosinus ($cos(\overrightarrow{u}, \overrightarrow{v})$) de l'angle formé par les deux vecteurs ($\overrightarrow{u}$ et $\overrightarrow{v}$) représentant les deux utilisateurs considérés :

$$cos(\overrightarrow{u}, \overrightarrow{v}) = \frac{\overrightarrow{u}.\overrightarrow{v}}{||\overrightarrow{u}||.||\overrightarrow{v}||} = \frac{\sum_i u_i.v_i}{\sqrt{\sum_i u_i^2 . \sum_i v_i^2}}$$

Une mesure possible pour le poids entre les utilisateurs a et i est alors le *coefficient de similarité*

$$w(a,i) = \frac{\sum_j v_{a,j} * v_{i,j}}{\sqrt{\sum_j v_{a,j}^2 \sum_j v_{i,j}^2}},$$

où la somme sur j concerne l'ensemble des articles pour lesquels les utilisateurs a et i ont tous deux affecté une note ($j \in I_a \cap I_i$).

Mais le *coefficient de corrélation de Pearson* s'est avéré être une mesure plus efficace, en considérant plutôt le cosinus de l'écart à la moyenne des deux utilisateurs considérés : il pose :

$$w(a,i) = cos(\overrightarrow{v_a} - \overline{v_a}, \overrightarrow{v_i} - \overline{v_i}) = \frac{(\overrightarrow{v_a} - \overline{v_a}).(\overrightarrow{v_i} - \overline{v_i})}{||\overrightarrow{v_a} - \overline{v_a}||.||\overrightarrow{v_i} - \overline{v_i}||}$$

d'où

$$w(a,i) = \frac{\sum_{j \in I_a \cap I_i}(v_{a,j} - \overline{v_a})(v_{i,j} - \overline{v_i})}{\sqrt{\sum_{j \in I_a \cap I_i}(v_{a,j} - \overline{v_a})^2 \sum_{j \in I_a \cap I_i}(v_{i,j} - \overline{v_i})^2}},$$

Pour observer l'utilisation de cette méthode selon Pearson, revenons maintenant à l'exemple que nous avons présenté précédemment, et voyons quelle est l'estimation de vote de l'utilisateur 3 sur l'article 3. Nous supposons donc cette note non attribuée, et allons voir quelle estimation de note l'algorithme va fournir.

Les utilisateurs que nous considérons sont alors ceux ayant noté l'article 3, et qui ont noté au moins un article en commun avec l'utilisateur 3. Dans notre cas, il s'agit des utilisateurs 1 et 2.

La première étape est alors le calcul des moyennes des utilisateurs concernés : $\overline{v_1} = 6.5$; $\overline{v_2} = 6$; $\overline{v_3} = 6.5$.

Puis on calcule les corrélations entre utilisateurs concernés. Pour la corrélation entre 1 et 3, on considère les articles appartenant à $I_1 \cap I_3 = \{4\}$, et on trouve $w(1,3) = 1$. Pour la corrélation entre 2 et 3, on considère les articles appartenant à $I_2 \cap I_3 = \{4,5\}$, et on trouve $w(2,3) = 1$.

Puis on peut enfin calculer l'estimation de vote de l'utilisateur 3 sur l'article 3, et on trouve $p_{3,3} = 6.25$, qu'on arrondi à 6, et qui correspond effectivement à la note réellement attribuée.

Nous détaillerons davantage les calculs de cette approche dans le chapitre suivant.

Bayes

Une autre approche, envisagée pendant le stage, est d'estimer les probabilités $P(v_{ij} = k|\overrightarrow{v_i})$ d'attribuer une note k à un article j par un utilisateur i, connaissant sa description $\overrightarrow{v_i}$ (vecteur de notes v_{ip} sur les articles parcourus p), et les informations sur les autres utilisateurs de la base.

L'idée ici est alors d'utiliser la règle de Bayes suivante :

$$P(v_{ij} = k|\overrightarrow{v_i}) = \frac{P(v_{ij} = k) * P(\overrightarrow{v_i}|v_{ij} = k)}{P(\overrightarrow{v_i})}$$

où $P(v_{ij} = k)$ correspond à $P(v_j = k)$, la probabilité d'attribuer la note k sur l'article considéré j.

$P(\overrightarrow{v_i}|v_{ij} = k)$ correspond à $P(\overrightarrow{v_i}|v_j = k)$, la probabilité de la description $\overrightarrow{v_i}$ de l'utilisateur i sachant qu'on a attribué la note k sur l'article j.

et $P(\overrightarrow{v_i})$ est la probabilité de la description $\overrightarrow{v_i}$ de l'utilisateur i.

Ainsi, estimer la probabilité $P(v_{ij} = k|\overrightarrow{v_i})$ revient à calculer $P(\overrightarrow{v_i}|v_j = k)$ et $P(v_j = k)$, ce qui peut être fait en consultant la base de données utilisateur des notes déjà attribuées aux articles, puis à attribuer, pour l'utilisateur i sur l'article j, la note k qui maximise la probabilité $P(\overrightarrow{v_i}|v_j = k) * P(v_j = k)$.

Soient les notations suivantes :
$NbNotes(k, j)$ = nombre de notes égales à k pour l'article j.
$TotalNotes(j)$ = nombre total de notes attribuées à l'article j.
On pose alors

$$P(v_j = k) = \frac{NbNotes(k, j)}{TotalNotes(j)}$$

En ce qui concerne $P(\overrightarrow{v_i}|v_j = k)$, en faisant une hypothèse d'indépendance entre les notes attribuées aux articles, on pose

$$P(\overrightarrow{v_i}|v_j = k) = \prod_{p \in I_i} P(v_p = v_{ip}|v_j = k)$$

avec $P(v_p = v_{ip}|v_j = k)$ la probabilité d'attribuer la note v_{ip} sur l'article p sachant qu'on a attribué la note k sur l'article j.

Soient les notations suivantes :

$NbNotes(v_p = v_{ip}|v_j = k)$ = nombre de notes égales à v_{ip} pour l'article p, pour les utilisateurs ayant noté k sur l'article j,

$TotalNotes(p|v_j = k)$ = nombre total de notes attribuées à l'article p, pour les utilisateurs ayant noté k sur l'article j,

et MaxEchelle le maximum de l'échelle de notation utilisée.

Étant donné qu'il faut éviter d'avoir $P(v_p = v_{ip}|v_j = k) = 0$, qui annulerait directement $P(\overrightarrow{v_i}|v_j = k)$, on ne pose pas

$$P(v_p = v_{ip}|v_j = k) = \frac{NbNotes(v_p = v_{ip}|v_j = k)}{TotalNotes(p|v_j = k)}$$

On pose plutôt

$$P(v_p = v_{ip}|v_j = k) = \frac{1 + NbNotes(v_p = v_{ip}|v_j = k)}{MaxEchelle + TotalNotes(p|v_j = k)}$$

Ainsi, $P(v_p = v_{ip}|v_j = k)$ ne risquera pas d'annuler $P(\overrightarrow{v_i}|v_j = k)$, et on obtient bien $\sum_{n=1}^{MaxEchelle} P(v_p = n|v_j = k) = 1$:

$$\begin{aligned}
&\sum_{n=1}^{MaxEchelle} P(v_p = n|v_j = k) \\
&= \sum_{n=1}^{MaxEchelle} \frac{1 + NbNotes(v_p = n|v_j = k)}{MaxEchelle + TotalNotes(p|v_j = k)} \\
&= \frac{\sum_{n=1}^{MaxEchelle} 1 + \sum_{n=1}^{MaxEchelle} NbNotes(v_p = n|v_j = k)}{MaxEchelle + TotalNotes(p|v_j = k)} \\
&= \frac{MaxEchelle + TotalNotes(p|v_j = k)}{MaxEchelle + TotalNotes(p|v_j = k)} \\
&= 1
\end{aligned}$$

Finalement, par la méthode de Bayes, pour estimer la note attribuée par l'utilisateur i sur l'article j, on cherche donc k qui maximise

$$M_k = \frac{NbNotes(k, j)}{TotalNotes(j)} \prod_{p \in I_i} \left(\frac{1 + NbNotes(v_p = v_{ip}|v_j = k)}{MaxEchelle + TotalNotes(p|v_j = k)} \right)$$

Pour observer l'utilisation de cette méthode selon Bayes, revenons à notre exemple, et voyons quelle est l'estimation de vote de l'utilisateur 3 sur l'article 3. Nous supposons donc cette note non attribuée, et allons voir quelle estimation de note l'algorithme va fournir.

Dans un premier temps, on regarde les notes déjà attribuées à l'article considéré. Ici, deux notes sont envisageables : 5 et 7. À partir de là, pour chaque note k envisageable, on calcule la probabilité M_k de l'attribuer, étant donnée la description de l'utilisateur considéré :

$$M_5 = \frac{1}{32} \texttt{ et } M_7 = \frac{1}{56}$$

Puis on sélectionne la note de plus forte probabilité : $M_5 > M_7$, donc la note attribuée à l'article 3 pour l'utilisateur 3 sera 5 par la méthode de Bayes.

Nous détaillerons davantage les calculs de cette approche dans le chapitre suivant.

vote moyen pondéré de Bayes

Une autre utilisation de cette approche de Bayes a été envisagée : effectuer un vote moyen pondéré sur l'article j, en prenant en compte la description $\overrightarrow{v_i}$ de l'utilisateur i :

$$\texttt{On renvoie alors la note } \sum_{k=1}^{MaxEchelle} P(v_j = k | \overrightarrow{v_i}) * k$$

2.2.2 Méthodes basées sur les modèles

D'un point de vue probabiliste, la tâche du filtrage collaboratif peut être vue comme le calcul de la valeur la plus probable d'un vote, étant donné ce que nous savons à propos de l'utilisateur, et le modèle que l'on aura construit à partir de la base de données utilisateur.

Le modèle cluster

L'idée du modèle cluster est de regrouper (en clusters) les gens ayant les mêmes goûts, et de regrouper (en clusters) les articles portant sur les mêmes sujets, ou qui tendent à plaire aux mêmes personnes. Ainsi, pour prédire la note qu'un utilisateur donnera à un article, on pourra utiliser les avis des utilisateurs qui appartiennent à son groupe. En d'autres termes, on veut associer une classe (ou plusieurs) à chacun des utilisateurs, ainsi qu'à chacun des articles. Mais ces classes étant a priori inconnues, elles doivent être dérivées du processus d'estimation du modèle.

Typiquement, les systèmes de clustering se différencient par la *fonction objectif* choisie pour évaluer la qualité du clustering, et la *stratégie de contrôle* pour parcourir l'espace des clusters possibles. Mais tous suivent le principe général traditionnel en clustering qui consiste à maximiser la similarité des

observations à l'intérieur d'un cluster, et minimiser la similarité des observations entre clusters, pour arriver à une partition de la base aussi pertinente que possible.

En entrée, ce que nous avons, c'est un ensemble d'enregistrements de qui a apprécié quoi, représenté par une matrice d'appréciations des articles par les utilisateurs (cf. exemple tableau 2.2).

	Article 1	Article 2	Article 3	Article 4	Article 5
Utilisateur 1			7	6	
Utilisateur 2			5	6	7
Utilisateur 3			6	6	7
Utilisateur 4	7	5			7
Utilisateur 5	7	6			7

TAB. 2.2 – Une matrice des notes attribuées aux articles par les utilisateurs

L'idée du clustering est alors de former des blocs les plus pertinents et significatifs possibles, à partir de cette matrice, et former ainsi des clusters d'utilisateurs et des clusters d'articles, tels que les utilisateurs considérés comme similaires tendent à noter de la même façon les articles considérés comme similaires (cf. exemple tableau 2.3).

	Article 1	Article 2	Article 3	Article 4	Article 5
Utilisateur 1			7	6	
Utilisateur 2			5	6	7
Utilisateur 3			6	6	7
Utilisateur 4	7	5			7
Utilisateur 5	7	6			7

TAB. 2.3 – Partition de la matrice précédente articles/utilisateurs

Puis la partition est évaluée en estimant, pour chaque bloc formé, la probabilité que les utilisateurs d'un même cluster k apprécient les articles d'un même cluster l, le but étant que ces probabilités soient les plus proches possible de 1, c'est à dire que tout le monde soit du même avis dans le groupe.

Un bloc B_c est en fait caractérisé par k, l'ensemble des lignes de la matrice faisant partie de ce bloc (le cluster des utilisateurs considéré), et l, l'ensemble des colonnes de la matrice faisant partie de ce bloc (le cluster des articles considéré), et il est défini par l'ensemble des notes $\{v_{ij} | i \in k, j \in l\}$, et $|B_c|$

le nombre d'éléments du bloc. Pour évaluer la pertinence d'un bloc B_c (cf. exemple tableau 2.4), une méthode simple que nous avons générer est la suivante : tout d'abord, on calcule la moyenne M_c des notes attribuées de ce bloc ; puis pour chaque élément v_{ij} tel que $i \in k$, $j \in l$, et $v_{ij} \neq 0$, on calcule son écart à la moyenne $e_{ij} = |v_{ij} - M_c|$; on somme ensuite cet ensemble des écarts à la moyenne ; et puis on normalise pour obtenir un réel compris entre 0 et 1 ; l'évaluation de la pertinence P_c du bloc peut alors être donnée par :

$$P_c = 1 - 2 * \frac{\sum_i \sum_j e_{ij}}{MaxEchelle * |B_c|}$$

Ainsi, plus les notes d'un bloc sont proches les unes des autres, plus la probabilité P_c sera proche de 1, et inversément, plus les notes d'un bloc sont éloignées les unes des autres, et plus la probabilité P_c sera proche de 0.

	Cluster d'articles 1	Cluster d'articles 2	Cluster d'articles 3
Cluster d'utilisateurs 1	1	0.90	1
Cluster d'utilisateurs 2	0.86	1	1

TAB. 2.4 – Evaluation de la partition précédente

Ensuite, pour évaluer la partition générale de la matrice M, partitionnée en blocs B_c, on pose

$$P = \frac{\sum_c |B_c| P_c}{|M|}$$

L'idée de l'utilisation d'une pondération ici est de favoriser les "gros clusters pertinents".

Dans notre cas, la pertinence de la matrice est alors $P = 0.91$

Après cette présentation générale du clustering, rentrons maintenant dans les détails des approches existantes. Comme nous l'avons énoncé précédemment, les systèmes de clustering sont définis par la *fonction objectif* utilisée pour évaluer la qualité du clustering, et la *stratégie de contrôle* pour parcourir l'espace des clusters possibles.

Voyons d'abord les différentes stratégies de contrôle qu'on peut utiliser pour parcourir l'espace des clusters possibles :

- Le modèle EM [UF98] :

  Il s'agit d'un modèle statistique qui permet de gérer les affectations de classes comme des paramètres cachés du modèle. L'idée est de se servir du modèle courant pour affecter des classes aux objets, puis, en fonction de ces nouvelles affectations et des informations qu'elles peuvent apporter, de réévaluer les paramètres du modèle, et ainsi améliorer

celui-ci à chaque étape.

Typiquement, on définit 3 ensembles de paramètres pour le modèle :
- P_k = probabilité qu'un utilisateur appartienne à la classe k.
- P_l = probabilité qu'un article appartienne à la classe l.
- et P_{kl} = probabilité qu'un utilisateur de la classe k apprécie un article de la classe l.

Puis on utilise l'approche *EM*, qui consiste en une succession de deux phases, pour améliorer la partition au fur et à mesure :

1. *Expectation* :
   trouver les classes attendues pour chaque personne et chaque article, en fonction des paramètres courants du modèle.
2. *Maximisation* :
   calculer les nouveaux paramètres du modèle, c'est à dire trouver les P_k, P_l, et P_{kl} les plus probables.

- Le clustering répété [UF98] :

  L'idée est ici de regrouper (en clusters) les personnes et les articles de façon séparée : dans une première phase, les personnes sont regroupées en clusters en se basant sur les articles, et les articles sont regroupés en clusters en se basant sur les personnes ; et dans un deuxième temps, les personnes sont regroupées en clusters en fonctions des clusters d'articles, et les articles en fonction des clusters de personnes. On répète ces 2 phases jusqu'à satisfaction du clustering obtenu, ou bien un nombre de fois fixé.

- Le clustering hiérarchique [Fis95] :

  L'idée ici est d'utiliser un arbre de clusters : on associe à la racine tous les utilisateurs et la répartition de leurs notes sur les articles de la base. Puis plus on descend dans l'arbre, plus les clusters sont spécifiques à un certain groupe d'utilisateurs similaires. C'est-à-dire que plus on descend dans l'arbre, plus les utilisateurs sont d'accord pour attribuer une certaine note à un article donné. À chaque nœud est donc associé la liste des utilisateurs intégrant le cluster formé, la répartition des notes de ces utilisateurs sur les articles de la base, et une probabilité d'atteindre ce nœud étant donné l'état du parent. Pour former l'arbre, on suit toujours le même principe qui consiste à maximiser la similarité des observations à l'intérieur d'un cluster, et minimiser la similarité des observations entre clusters (cf. exemple figure 2.3)

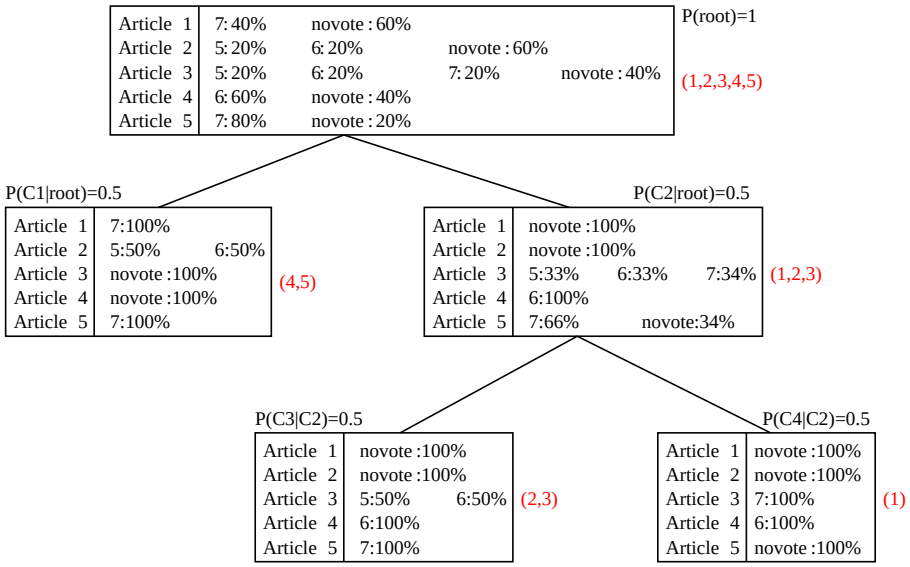

FIG. 2.3 – Arbre de clustering hiérarchique

Dans un premier temps, on utilise donc une stratégie simple pour former une partition initiale, puis on utilise ensuite plusieurs stratégies de contrôle pour lancer une optimisation itérative du modèle courant. Se définir une fonction objectif va alors permettre d'évaluer la pertinence de la partition effectuée. Voyons donc maintenant quelques fonctions objectif qu'on peut utiliser pour cela :

1. Soient V_{ij} les notes attribuées par les utilisateurs (i) sur les articles (j), et C_n les clusters d'utilisateurs formés. Alors une mesure de la Qualité d'un Cluster peut être donnée par :

$$QC(C_n) = P(C_n) \sum_j \sum_k [P(V_{ij} = k | i \in C_n)^2 - P(V_{ij} = k)^2]$$

   L'idée ici est de favoriser les clusters qui vont accroître la prédictabilité, c'est à dire $P(V_{ij} = k | i \in C_n) > P(V_{ij} = k)$.

   Une mesure de la Qualité d'une Partition peut alors être :

$$QP(\{C_1, C_2, \cdots, C_N\}) = \frac{\sum_{n=1}^{N} |C_n| QC(C_n)}{\sum_n |C_n|}$$

   De nouveau, l'idée ici est de favoriser les "gros clusters pertinents".

2. Une autre idée est de minimiser la distance moyenne entre éléments d'un même cluster tout en maximisant la distance entre clusters :

Soient N utilisateurs $\overrightarrow{v_i}$ appartenant à un même cluster.
On définit d'abord le centroïd $\overrightarrow{X_0}$ du cluster par :

$$\overrightarrow{X_0} = \frac{\sum_{i=1}^{N} \overrightarrow{v_i}}{N}$$

Puis deux mesures sont proposées ici pour définir la distance entre éléments d'un même cluster :

(a) Le radius R correspond à la distance moyenne entre points membres du cluster et son centroïd :

$$R = \sqrt{\frac{\sum_{i=1}^{N} (\overrightarrow{v_i} - \overrightarrow{X_0})^2}{N}}$$

(b) Et le diamètre D correspond à la distance moyenne entre paires de membres du cluster :

$$D = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\overrightarrow{v_i} - \overrightarrow{v_j})^2}{N(N-1)}}$$

Ces deux mesures doivent alors être minimisées pour maximiser la similarité des observations à l'intérieur d'un cluster.

Et ensuite, deux méthodes sont proposées ici pour définir la distance entre deux clusters et mesurer leur proximité :

(a) Etant donné les centro$\overrightarrow{X_{0_2}}$ de deux clusters, la distance euclidienne D_0 entre centroïds est donnée par :

$$D_0 = \sqrt{(\overrightarrow{X_{0_1}} - \overrightarrow{X_{0_2}})^2}$$

(b) Et D_1, la distance de Manhattan entre centroïds est donnée par :

$$D_1 = |\overrightarrow{X_{0_1}} - \overrightarrow{X_{0_2}}| = \sum_{i=1}^{d} |\overrightarrow{X_{0_1}}^{(i)} - \overrightarrow{X_{0_2}}^{(i)}|,$$

avec d la dimension des vecteurs.

Ces deux mesures doivent alors être maximisées pour minimiser la similarité des observations entre clusters.

Le modèle *réseau de Bayes*

Un réseau de Bayes [Hec96] est un graphe acyclique dirigé qui représente une distribution de probabilités de dépendance entre un ensemble de variables. Chaque nœud dans le graphe représente une variable, et chaque arc une dépendance directe entre variables. Ainsi, chaque variable est indépendante de ses non-descendants dans le graphe, étant donné l'état de ses parents.

Par exemple, si on représente un classifieur naif de Bayes sous la forme d'un réseau de Bayes, on obtient une structure simple telle que décrite dans la figure 2.4. En effet, chaque attribut (les feuilles du réseau) est indépendant des autres attributs, étant donné l'état de la variable de classe (la racine du réseau).

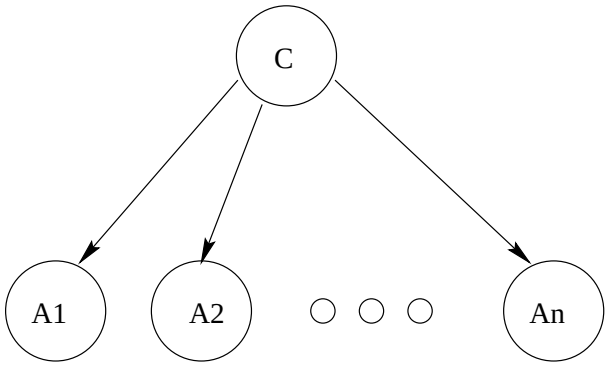

FIG. 2.4 – La structure d'un réseau de Bayes naif

Si on dispose d'un ensemble d'apprentissage, l'idée pour l'algorithme d'apprentissage est alors d'effectuer une recherche parmi les différentes structures de modèle possible, en terme de dépendance pour chaque article, pour que dans le réseau résultant, chaque article ait un ensemble d'articles parents qui soient les meilleurs prédicteurs de ses votes.

Une idée pour notre application est alors d'associer un réseau de Bayes à chaque article de la base. À chaque feuille de l'arbre est associée une probabilité d'attribuer une note à l'article, étant donné l'état des parents identifiés. Pour prédire l'estimation de note d'un client sur un article, on se déplace alors dans le réseau de Bayes correspondant, selon les notes que l'utilisateur considéré a donné aux articles parents présents dans le réseau, et on attribue, pour l'article considéré, la note la plus probable.

Formellement, on obtient :

Soit $U = \{X_1, ..., X_n\}$ un ensemble fini de variables aléatoires discrètes, où chaque X_i prend ses valeurs dans un ensemble fini $Val(X_i)$. Soit P_B une distribution de probabilités sur les variables de U. Un réseau de Bayes B pour U est un couple $< G, \Theta >$. Le graphe G code les hypothèses d'indépendance suivantes : chaque variable X_i est indépendante de ses non-descendants, étant donné l'état de ses parents dans G. Et Θ représente l'ensemble des paramètres qui quantifient le réseau. Il contient les paramètres $\theta_{x_i|\prod_{x_i}} = P_B(x_i|\prod_{x_i})$ pour chaque valeur possible x_i de X_i, et $\prod_{x_i}$ de $\prod_{X_i}$, où $\prod_{X_i}$ est l'ensemble des parents de X_i dans G.

Un réseau bayesien B définit donc une unique distribution de probabilités sur U, donnée par :

$$P_B(X_1, ..., X_n) = \prod_{i=1}^{n} P_B(X_i|\prod X_i) = \prod_{i=1}^{n} \theta_{X_i|\prod_{X_i}}$$

Par exemple, soit $U = \{A_1, ..., A_n, C\}$, où les variables A_i sont les attributs et C est la variable de classe. Soit la structure de graphe où la variable de classe est la racine du graphe (c'est à dire $\prod_C = \emptyset$), et chaque attribut a la variable de classe comme son unique parent (c'est à dire $\prod_{A_i} = \{C\}$ pour tout $1 \leq i \leq n$). Pour ce type de structure de graphe, on obtient :

$$P_B(A_1, ..., A_n, C) = P_B(C) \prod_{i=1}^{n} P_B(A_i|C),$$

$P_B(C)$ et $P_B(A_i|C)$ étant calculés à partir de la base de données utilisateur, servant d'ensemble test de votes de clients. De la définition de probabilité conditionnelle, on obtient :

$$P_B(C|A_1, ..., A_n) = \alpha P_B(C) \prod_{i=1}^{n} P_B(A_i|C),$$

où α est une constante de normalisation. Ceci est en fait la définition de Bayes naif connue dans la littérature.

Le problème d'apprentissage d'un réseau bayesien peut alors être formulé comme suit : étant donné un ensemble test $D = \{u_1, ..., u_n\}$ d'instances de U, trouver un réseau B qui correspond le mieux à D. De nouveau, il nous faut une fonction de score pour évaluer nos réseaux.

2.2.3 Les limites de cette approche

Les points sensibles du filtrage collaboratif sont :

- l'échelle de notation proposée : parfois une échelle de notation allant de 1 à 7 permet de meilleures prédictions qu'une échelle de 1 à 10 ;
- l'obtention des notes de la part des utilisateurs : ce problème a été discuté précédemment ;
- le taux de champs renseignés dans la base de profils : souvent, on se retrouve confronté au problème que peu (ou pas) d'utilisateurs ont noté un article donné, ou qu'un utilisateur donné a évalué très peu d'articles de la base ;
- le nombre de personnes comparables par la mesure de similarité entre profils : parfois, on se retrouve confronté au problème qu'un utilisateur n'est comparable avec aucun autre ;
- dépasser une masse critique avant laquelle les recommandations ne sont pas pertinentes ;
- présenter la recommandation pour qu'elle soit acceptée par l'utilisateur : utiliser une mesure de confiance, ou une présentation graphique, etc. ;
- gérer le fait qu'il existe un grand nombre de documents dans la base, mais peu de documents effectivement lus, et donc peu de documents sur lesquels on reçoit un retour (recherche sur les techniques de classification semi-supervisée) ;
- un système collaboratif doit faire mieux que la procédure par défaut qui consiste à prédire systématiquement la réponse majoritaire.

Chapitre 3

Implémentations, tests et analyse

Concernant l'évaluation des notes attribuées aux articles par les utilisateurs, deux méthodes ont été implémentées : celle à partir de *fichiers de log*, et celle par développement d'un *modèle de répartition des notes*. Mais les *fichiers de log* de *Rosebud* n'étant pas encore disponibles, nous avons travaillé sur ceux du site de l'équipe GRAPPA. Ainsi, nous avons pu analyser les conséquences de l'utilisation des matrices creuses résultantes sur nos algorithmes.

Ensuite, deux méthodes de filtrage collaboratif ont été implémentées : celle selon *Pearson* et celle selon *Bayes*. Quelques variantes ont également été testées : le *vote moyen pondéré selon Bayes*, ou *Pearson* avec sélection de quelques corrélations les plus fortes entre utilisateurs. De plus, d'autres méthodes, plus classiques, ont été mises en place pour comparer les résultats obtenus avec et sans prise en compte de la description des utilisateurs.

Puis une série de tests a été organisée, pour comparer les résultats des différents algorithmes face à différents types de données en entrée, et observer la contribution des informations utilisées par chaque méthode dans la précision des estimations effectuées.

3.1 Les critères d'évaluation

L'erreur absolue de chaque note estimée doit être minimisée.

Ainsi, si, pour l'utilisateur actif a, les v_{aj} sont les notes réelles attribuées aux articles j de l'ensemble test P_a (de taille m_a), et les p_{aj} sont les valeurs estimées pour les mêmes articles, alors on calcule l'erreur moyenne dans

l'estimation des notes comme suit :

$$MoyError(a) = \frac{1}{m_a} \sum_{j \in P_a} |p_{aj} - v_{aj}|,$$

C'est cette valeur qu'il faut chercher à minimiser.

Et deux autres mesures pourront également être utilisées pour comparer les méthodes :

- PrctMauvaises : le pourcentage de mauvaises prédictions effectuées :

$$PrctMauvaises(a) = \frac{1}{m_a} |\{j \in P_a | p_{aj} \neq v_{aj}\}|$$

- et MaxError : l'erreur maximale commise dans l'estimation :

$$MaxError(a) = Max_{j \in P_a} |p_{aj} - v_{aj}|$$

Les étapes des algorithmes d'estimation des notes sont les suivantes :

Pour chaque note de la base de données utilisateur,

1. stocker la note en mémoire,
2. supprimer la note de la base de données,
3. calculer la prédiction de note, l'erreur de prédiction, et la stocker,
4. et réintégrer la note dans la base de données.

puis calculer la moyenne des erreurs de prédiction, le pourcentage de mauvaises prédictions effectuées, et l'erreur maximale commise dans l'estimation.

3.2 Méthodes comparatives

Dans un souci d'analyse, pour comparer les résultats des méthodes énoncées au chapitre précédent avec d'autres, nous avons implémenté quelques méthodes plus classiques ne prenant pas en compte la description des utilisateurs :

- Attribuer la note majoritaire sur l'article j :
  `On renvoie la note` N `telle que` $P(N, j) = Max_k P(k, j)$

- Attribuer la note moyenne sur l'article j :
  `On renvoie la note` $\frac{SommeNotes[j]}{TotalNotes[j]}$

– Ou effectuer un vote moyen pondéré sur l'article j :
`On renvoie la note` $\sum_{n=1}^{MaxEchelle} P(n,j) * n$

Ainsi, on pourra observer la contribution de l'utilisation de la description des utilisateurs dans les résultats.

3.3 Tests, résultats et analyse des résultats

Deux séries de tests ont été effectuées.

La première est celle effectuée sur les fichiers de log de l'équipe GRAPPA :
- Dans un premier temps, nous expliquerons la méthode utilisée pour engendrer la matrice de notes à partir de ces fichiers de log.
- Ensuite, nous testerons l'utilisation de différentes échelles de notation et choisirons celle qui paraîtra la plus adaptée.
- Puis nous comparerons les résultats obtenus par les différentes méthodes présentées.
- Et enfin, dans un dernier temps, nous testerons l'algorithme de Pearson, en ne considérant qu'un nombre fixé des corrélations les plus fortes, pour vérifier s'il est nécessaire ou non de prendre en compte toutes les corrélations existantes entre les utilisateurs.

La seconde série de tests sera ensuite effectuée sur les matrices engendrées par notre modèle de répartition des notes, pour observer les différences de résultats des différentes méthodes en fonction du type de données fournies en entrée de l'algorithme.
- Tout d'abord, nous justifierons l'utilisation d'un modèle de répartition des notes, et présenterons le générateur implémenté.
- Puis nous ferons tourner nos algorithmes sur une matrice simple générée par celui-ci, pour détailler les étapes de chaque algorithme et expliquer plus en détails leur fonctionnement.
- Ensuite, nous effectuerons plusieurs comparaisons entre les méthodes, en fonction du type de données qu'on leur fournit en entrée.
- Et enfin, une dernière partie nous servira à observer l'influence de la répartition des notes dans les résultats obtenus.

Et à partir de là, nous analyserons les résultats obtenus, et observerons les points forts, points faibles et intérêts de chaque approche.

3.3.1 Tests sur *fichiers de log*

Génération des notes à partir des *fichiers de log* de l'équipe GRAPPA

Typiquement, une ligne d'un fichier de log se présente comme suit :

216.221.34.111 - - [22/Apr/2001 :04 :04 :54 +0200] "GET /polys/access-1997/node2.html HTTP/1.1" 200 2856

Entre autres, 216.221.34.111 correspond au numéro IP du client, 22/Apr/2001 :04 :04 :54 correspond au moment de connexion, et /polys/access-1997/node2.html est l'URL de la page qui a été visitée. Ce sont ces trois informations que nous allons utiliser pour estimer l'intérêt que l'utilisateur a eu pour la page.

Pour chaque ligne du fichier de log, on récupére donc ces informations (numIP, URL, date et heure de connexion), puis pour chaque client (identifié par numIP), pour chaque page visitée (URL), on calcule le temps moyen de visite de la page par le client (stockage dans la double table de hachage meanTimeVisit{IP}{URL}), le nombre de visites de la page par le client (nbVisit{IP}{URL}), et le temps passé depuis la dernière visite (lastVisit{IP}{URL}), ainsi que leurs maxima (dans les tables de hachage MaxTimeVisit{IP}, MaxNbVisit{IP} et MaxLastVisit{IP}).

Les notes s'expriment ensuite en fonction de ces valeurs :

$$Note\{IP\}\{URL\} = f\left(\frac{meanTimeVisit\{IP\}\{URL\}}{MaxTimeVisit\{IP\}}, \frac{nbVisit\{IP\}\{URL\}}{MaxNbVisit\{IP\}}, \frac{lastVisit\{IP\}\{URL\}}{MaxLastVisit\{IP\}}\right)$$

Notons que le nombre de visites d'un client donné pour une page donnée a été borné (à 20) : on considère qu'à partir d'une certaine fréquence de visites, le client est très intéressé. De même, le temps de visite d'une page par un client a été borné (à 10 minutes) : on considère qu'à partir d'un certain temps de visite, le client est très intéressé.

Finalement, par exemple pour une échelle de 0 à 7, on peut poser :

$$Note\{IP\}\{URL\} = arrondi\left(3\times\frac{meanTimeVisit\{IP\}\{URL\}}{MaxTimeVisit\{IP\}}+3\times\frac{nbVisit\{IP\}\{URL\}}{MaxNbVisit\{IP\}}+\frac{lastVisit\{IP\}\{URL\}}{MaxLastVisit\{IP\}}\right)$$

Pour une échelle de 0 à 20, on peut poser :

$$Note\{IP\}\{URL\} =$$
$$arrondi\left(8\times\frac{meanTimeVisit\{IP\}\{URL\}}{MaxTimeVisit\{IP\}}+8\times\frac{nbVisit\{IP\}\{URL\}}{MaxNbVisit\{IP\}}+4\times\frac{lastVisit\{IP\}\{URL\}}{MaxLastVisit\{IP\}}\right)$$
...etc...

On considère alors ici le temps moyen de visite et la fréquence de visites sur une page comme des informations plus discriminantes que le temps depuis la dernière visite.

En ce qui concerne la fonction arrondi, pour bien faire la différence entre une page non visitée et une page très peu appréciée, nous forçons les notes attribuées à 1 minimum, alors que 0 servira de note par défaut pour toute page non visitée. La fonction arrondi d'un réel r est donc la suivante :

```
if ( r > (float)MaxEchelle - 1.5 ) return MaxEchelle;
else if ( r - int(r) > 0.5) return int(r) + 2;
else return int(r) + 1;
```

Tests d'échelles de notation

Le premier paramètre que nous devons fixer est alors MaxEchelle. C'est pourquoi l'utilisation de différentes échelles de notes a été testée : les échelles [0,7], [0,10], [0,15], et [0,20].

Le fichier de log utilisé pour les tests contient 60498 lignes. Cela représente 3377 utilisateurs, 7890 pages, et 46983 notes (soit 0.176% des notes de la matrice).

Les quatre premiers tableaux (3.1, 3.2, 3.3 et 3.4) présentent la répartition des notes dans l'intervalle de notation utilisée, selon l'échelle de notation utilisée.

Et les deux tableaux suivants (3.5 et 3.6) présentent les résultats des erreurs de prédictions des algorithmes de Pearson et de Bayes selon l'échelle de notation utilisée.

La répartition des notes influe beaucoup sur les résultats. C'est ce qui explique que l'échelle de notation [0,15] est celle qui donne les moins bons résultats. Nous détaillerons ce point dans la partie suivante.

Mais si les répartitions des notes sont similaires, une échelle de notation plus réduite fournit de meilleurs résultats. Cependant, une échelle trop réduite ne serait pas pertinente car elle permetttrait à peine de distinguer les

$\%notes \in [1,2]$	24.45%
$\%notes \in [2,3]$	74.08%
$\%notes \in [3,4]$	0.97%
$\%notes \in [4,5]$	0.20%
$\%notes \in [5,6]$	0.13%
$\%notes \in [6,7]$	0.17%

TAB. 3.1 – Répartition des notes pour l'échelle de notation [0,7]

$\%notes \in [1,2]$	7.41%
$\%notes \in [2,3]$	85.93%
$\%notes \in [3,4]$	5.60%
$\%notes \in [4,6]$	0.71%
$\%notes \in [6,8]$	0.20%
$\%notes \in [8,10]$	0.15%

TAB. 3.2 – Répartition des notes pour l'échelle de notation [0,10]

$\%notes \in [2,3]$	53.35%
$\%notes \in [3,6]$	45.78%
$\%notes \in [6,9]$	0.57%
$\%notes \in [9,12]$	0.16%
$\%notes \in [12,15]$	0.14%

TAB. 3.3 – Répartition des notes pour l'échelle de notation [0,15]

$\%notes \in [2,4]$	86.91%
$\%notes \in [4,6]$	11.76%
$\%notes \in [6,9]$	0.85%
$\%notes \in [9,13]$	0.21%
$\%notes \in [13,17]$	0.16%
$\%notes \in [17,20]$	0.11%

TAB. 3.4 – Répartition des notes pour l'échelle de notation [0,20]

	$Notes \in [0,7]$	$Notes \in [0,10]$	$Notes \in [0,15]$	$Notes \in [0,20]$
MoyError	0.071	0.095	0.183	0.097
PrctMauvaises	6.33%	7.98%	15.07%	8.10%
MaxError	5	7	12	7

TAB. 3.5 – Erreurs de prediction de Pearson selon l'échelle de notes utilisée

	$Notes \in [0,7]$	$Notes \in [0,10]$	$Notes \in [0,15]$	$Notes \in [0,20]$
MoyError	0.111	0.122	0.203	0.122
PrctMauvaises	9.80%	8.55%	14.65%	9.00%
MaxError	6	8	11	8

TAB. 3.6 – Erreurs de prediction de Bayes selon l'échelle de notes utilisée

notes. En effet, dans ce cas, presque toutes les notes seraient égales (et surtout avec des notes dérivées de fichiers de log). L'échelle [0..7] est celle que nous utiliserons désormais dans la suite des tests.

Tests de coefficients pour chacune des informations récoltées

Les paramètres suivants à déterminer sont les coefficients à attribuer à chacune des informations récoltées (temps moyen de visite, nombre de visite, et temps depuis la dernière visite), pour estimer les notes attribuées aux pages par les clients.

Pour cela, nous avons créer un programme (TestCoeffsLog.pl) estimant les erreurs de chaque méthode en fonction de ces coefficients, et nous avons trouver que les meilleurs coefficients à attribuer à chacune des informations sont les suivants : $\frac{3}{7}$ pour le temps moyen de visite et le nombre de visite, et $\frac{1}{7}$ pour le temps passé depuis la dernière visite.

Comparaison des méthodes

La table 3.7 présente les résultats des différents algorithmes présentés (Pearson, Bayes, vote moyen pondéré de Bayes, vote majoritaire, vote moyen, et vote moyen pondéré) face à la matrice engendrée depuis le fichier de log de l'équipe GRAPPA.

	PrctMauvaises	MaxError	MoyError
Pearson	10.10%	6	0.111
Bayes	10.59%	6	0.141
Vote Moyen pondéré de Bayes	10.85%	5	0.142
Vote Majoritaire	20.57%	5	0.223
Vote Moyen pondéré	21.09%	5	0.227
Vote Moyen	37.13%	6	0.417

TAB. 3.7 – Comparaison des méthodes

On remarque que les méthodes prenant en compte la description des utilisateurs et leurs corrélations donnent significativement de meilleurs résultats

que les autres.

Tests de sélection d'utilisateurs les plus similaires

Comme nous l'avons énoncé précédemment, nous avons testé l'algorithme de Pearson, en ne considérant pas toutes les corrélations existantes entre utilisateurs, mais seulement un nombre fixé des corrélations les plus fortes, pour vérifier si prendre uniquement les avis des utilisateurs les plus similaires (ou les plus opposés) peut améliorer les résultats, au lieu de tenir compte également des avis d'utilisateurs qui ne paraissent pas ou peu comparables. Pour cela, nous n'avons conservé que les corrélations entre utilisateurs dont les valeurs absolues étaient maximales.

Les algorithmes *Select n* sélectionnent, pour chaque utilisateur, les n utilisateurs les plus similaires.

Les algorithmes *Best n* sélectionnent, pour chaque utilisateur, au maximum n utilisateurs ayant la corrélation la plus forte avec celui-ci.

Le tableau 3.8 présente les résultats de ces tests.

	MoyError
Select 1	0.132
Best 2	0.126
Select 5	0.124
Best 10	0.121
Select 15	0.120
Select 20	0.119
Pearson	0.111

TAB. 3.8 – Moyennes d'erreurs de prediction des algos de Pearson selon le nombre d'utilisateurs utilisés pour l'estimation

Ainsi, on remarque que ne prendre en compte que l'avis des utilisateurs les plus similaires (ou les plus opposés) ne conduit pas à de meilleurs résultats.

3.3.2 Tests avec générateur aléatoire de notes

Développement d'un modèle de répartition des notes

Les matrices de notes dérivées des fichiers de log sont particulières car elles sont très creuses, ce qui signifie que beaucoup de notes ne sont pas disponibles. En effet, étant donné que nous travaillons sur une grande quantité d'articles (plus de 7000 lors de nos tests), et de nombreux utilisateurs (plus de 3000 lors de nos tests), finalement, en proportion, peu d'utilisateurs

ont noté un article donné, et peu d'articles ont été visités par un utilisateur donné. De plus, la majorité des notes reste proche de 1 (la notation minimale sur une page ; à distinguer de 0, correspondant à une page non visitée), et peu se rapprochent du maximum de l'échelle de notes fixée, ce qui crée une disproportion dans la répartition des notes. Enfin, ne disposant pas encore des données réelles de Rosebud, nous avons travaillé sur les fichiers de log du site de l'équipe GRAPPA, qui typiquement contiennent moins d'informations.

Pour analyser les différentes méthodes de filtrage collaboratif implémentées, et tester leur robustesse face à différents types de matrices de notes, nous avons donc créé un générateur aléatoire de notes. Celui-ci a pour but de générer un ensemble de notes pour des utilisateurs sur des articles, à partir d'informations fournies en entrée du modèle de répartition fixé. Ainsi, en faisant varier quelques paramètres du modèle, nous pourrons observer les points forts, points faibles et intérêt de chaque approche selon le type de données fournies en entrée.

Cette partie se décompose comme suit :
- Tout d'abord, nous présenterons le générateur que nous avons implémenté, et développerons un exemple simple de son utilisation.
- Puis nous utiliserons cet exemple pour détailler les étapes des algorithmes que nous avons présentés (méthodes comparatives, de Pearson et de Bayes) et leur fonctionnement.
- Ensuite, une série de tests sera menée pour observer la robustesse de chaque approche face à différents types de matrice en entrée de leur algorithme.
- Et enfin, nous analyserons les résultats obtenus et verrons quels enseignements on peut en tirer.

Présentation du générateur implémenté

En entrée de notre générateur sont fournies les informations suivantes :
- le nombre d'utilisateurs de la base,
- le nombre d'articles de la base,
- le pourcentage de notes remplissant la matrice de notes,
- le nombre de groupes d'utilisateurs similaires (ayant les mêmes goûts),
- le nombre de groupes d'articles similaires (portant sur les mêmes sujets ou qui tendent à plaire aux mêmes personnes),
- l'échelle de notation utilisée,
- et l'écart type de notes entre utilisateurs du même groupe.

Dans un premier temps, on affecte un groupe à chaque utilisateur, et un groupe à chaque article (aléatoirement). Le but de ceci est de former des

clusters implicites, et d'observer si les différentes méthodes vont permettre de retrouver et exploiter ces informations de similarité entre utilisateurs.

Puis, pour chaque couple (groupe d'utilisateurs, groupe d'articles), on affecte une note moyenne attribuée aux groupes d'articles par les groupes d'utilisateurs (aléatoirement).

Et enfin, en fonction du pourcentage désiré de notes remplissant la matrice, on affecte les notes attribuées aux articles par les utilisateurs, selon une loi normale de distribution : pour un utilisateur donné et un article donné, la note moyenne attribuée est donnée par celle associée au couple (groupe de l'utilisateur considéré, groupe de l'article considéré), et la variation typique entre cette moyenne et la note réellement attribuée est donnée par l'écart type fixé.

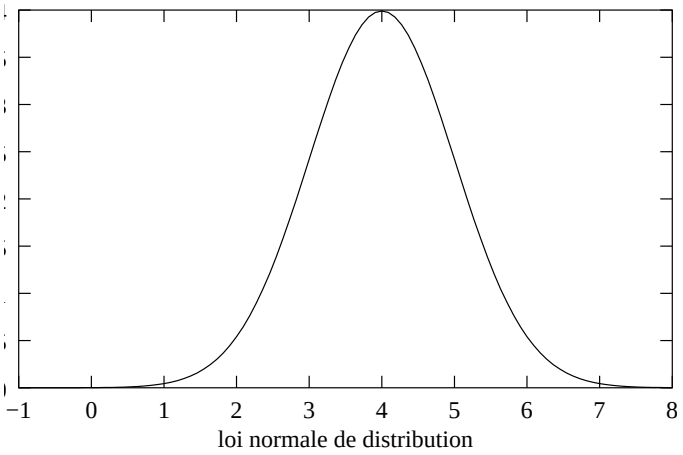

FIG. 3.1 – Loi normale de distribution : moyenne 4, écart type 1

Dans l'exemple de la figure 3.1, la majorité des notes attribuées sera 4, puis plus on s'éloigne de 4, plus la probabilité que la note soit attribuée sera faible.

Exemple de génération de notes

En fixant le nombre d'utilisateurs à 6, le nombre de pages à 10, le pourcentage de notes remplissant la matrice à 60%, le nombre de groupes d'utilisateurs à 3, le nombre de groupes de pages à 3, l'échelle de notation à 7, et l'écart type de notes entre utilisateurs du même groupe à 1, un résultat possible est le suivant :

- Affectation des groupes d'utilisateurs :
  le groupe 1 est constitué des utilisateurs 1 et 5,
  le groupe 2 est constitué de l'utilisateur 6,

et le groupe 3 est constitué des utilisateurs 2, 3 et 4.

- Affectation des groupes de pages :
  le groupe 1 est constitué des pages 3, 4, 6 et 7
  le groupe 2 est constitué des pages 8 et 9
  et le groupe 3 est constitué des pages 1, 2, 5, et 10

- Affectation des notes moyennes de groupes :
  Nous obtenons la matrice de la table 3.9.

	page group 1	page group 2	page group 3
user group 1	7	1	6
user group 2	3	7	1
user group 3	2	1	3

TAB. 3.9 – Matrice des notes (groupes d'utilisateurs, groupes de pages)

- Affectation des notes :
  Nous obtenons la matrice de la table 3.10.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
user 1	7	6	7	0	5	7	7	1	1	0
user 5	5	0	0	0	5	0	7	1	0	6
user 6	0	1	4	0	1	4	2	6	6	1
user 2	0	3	2	2	3	0	0	1	1	3
user 3	3	4	1	3	0	2	1	1	2	3
user 4	0	0	1	4	0	2	3	0	0	1

TAB. 3.10 – Matrice de notes (utilisateurs, pages)

Et la table 3.11 présente la matrice réorganisée, en fonction des groupes formés.

	P3	P4	P6	P7	P8	P9	P1	P2	P5	P10
user 1	7	0	7	7	1	1	7	6	5	0
user 5	0	0	0	7	1	0	5	0	5	6
user 6	4	0	4	2	6	6	0	1	1	1
user 2	2	2	0	0	1	1	0	3	3	3
user 3	1	3	2	1	1	2	3	4	0	3
user 4	1	4	2	3	0	0	0	0	0	1

TAB. 3.11 – Matrice réorganisée de notes (utilisateurs, pages)

Les méthodes comparatives sur l'exemple

Pour présenter les implémentations, nous les ferons tourner sur l'exemple de la table 3.10 que nous venons de présenter, et nous verrons quels sont les résultats de l'estimation de note de l'utilisateur 5 sur l'article 1 pour chaque méthode présentée (la note réelle étant 5).

Commençons par les estimations fournies par les méthodes comparatives : une fois la note de l'utilisateur 5 sur l'article 1 supprimée,
le vote majoritaire sur l'article 1 renvoie soit 3, soit 7 ;
le vote moyen sur l'article 1 renvoie $\frac{3+7}{2} = 5$;
et le vote moyen pondéré sur l'article 1 renvoie $\frac{1}{2} * 3 + \frac{1}{2} * 7 = 5$.

Étapes de l'algorithme d'estimation des notes par Pearson

1. Première étape : pour chaque utilisateur, calculer la moyenne des notes qu'il a attribuées.
2. Deuxième étape : calculer les corrélations entre utilisateurs.
3. Troisième étape : estimer les notes prédites :
   Pour chaque note qu'un client a attribuée à une page,
   - sauvegarder la note,
   - supprimer la note de la base (c'est-à-dire la faire passer à 0),
   - calculer la nouvelle moyenne de notes du client,
   - calculer les nouvelles corrélations avec les autres clients ayant également noté l'article considéré,
   - lancer l'algorithme pour estimer la note prédite,
   - calculer l'erreur commise dans l'estimation,
   - stocker le résultat, pour calculer ensuite la moyenne des erreurs de prédiction, le pourcentage de mauvaises prédictions effectuées, et l'erreur maximale commise dans l'estimation,
   - et réintégrer la note dans la base.

Remarquons que pour un client ayant noté un seul article, on ne pourra pas faire de prédiction sur sa note, car une fois celle-ci supprimée, nous n'avons plus aucune information sur le client, et donc plus aucun élément de comparaison. En effet, on se retrouve alors avec un client dont la description est vide, et manifestement, dans un tel cas, on ne peut utiliser cette description pour l'estimation. Pour nos tests, nous avons choisi d'ignorer ces cas, mais dans l'avenir, à l'arrivée d'un nouvel utilisateur, nous utiliserons sûrement alors un vote majoritaire ou un vote moyen.

De même, si un client est le seul à avoir noté un article, on ne pourra prédire pour lui aucune note sur cet article car on ne dispose dans ce cas d'aucun avis extérieur sur l'article.

De plus, si on utilise le coefficient de corrélation de Pearson, et si un client a toujours attribué la même note à tous les articles qu'il a parcourus, il ne pourra participer aux estimations de notes car son écart à la moyenne sera toujours nul.

Enfin, notons que certains clients ne sont comparables avec aucun autre.

Dans de tels cas, la note attribuée par défaut est la note moyenne de l'utilisateur considéré.

Pearson sur l'exemple

Le tableau 3.12 présente les résultats des deux premières étapes de l'algorithme de Pearson : les moyennes de chaque utilisateur, et les corrélations entre utilisateurs.

	user 1	user 2	user 3	user 4	user 5	user 6	moyenne user
user 1		0.78	0.2	-0.24	0.93	-0.67	5.12
user 2			0.7	-0.51	0.83	-0.99	2.14
user 3				0.19	0.33	-0.62	2.22
user 4					0.09	0.08	2.2
user 5						-0.83	4.8
user 6							3.12

TAB. 3.12 – Tableau de données de Pearson : corrélations et moyennes

Le tableau 3.13 présente les résultats d'estimation des notes par Pearson.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
user 1	5	6	4	0	6	4	5	3	4	0
user 2	0	4	2	2	3	0	0	1	1	3
user 3	3	3	2	2	0	2	4	3	1	3
user 4	0	0	3	2	0	2	1	0	0	4
user 5	6	0	0	0	6	0	4	6	0	5
user 6	0	2	3	0	3	2	3	5	4	3

TAB. 3.13 – Estimations de notes par Pearson

Et le tableau 3.14 présente les erreurs d'estimation des notes par Pearson.

La moyenne d'erreurs de predictions de Pearson dans ce cas est de 1.38

Pour illustrer le fonctionnement de cet algorihtme, nous allons détailler le calcul d'estimation de note de l'utilisateur 5 sur l'article 1 :

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
user 1	-2	0	-3	0	+1	-3	-2	+2	+3	0
user 2	0	+1	0	0	0	0	0	0	0	0
user 3	0	-1	+1	-1	0	0	+3	+2	-1	0
user 4	0	0	+2	-2	0	0	-2	0	0	+3
user 5	+1	0	0	0	+1	0	-3	+5	0	-1
user 6	0	+1	-1	0	+2	-2	+1	-1	-2	+2

TAB. 3.14 – Erreurs d'estimation de notes par Pearson

1. nouvelle moyenne de l'utilisateur 5 : 4.75
2. Notes des autres utilisateurs sur l'article 1 :
   7 pour l'utilisateur 1
   3 pour l'utilisateur 3
3. nouvelles corrélations entre l'utilisateur 5 et ceux ayant noté l'article 1 :
   0.99 avec 1
   0.33 avec 3
4. moyennes des utilisateurs corrélés :
   5.12 pour 1
   2.22 pour 3
5. l'estimation de note de l'utilisateur 5 sur l'article 1 est alors donnée par :

$$4.75 + \frac{0.99 * (7 - 5.12) + 0.33 * (3 - 2.22)}{0.99 + 0.33} = 6.355$$

La note prédite par Pearson est donc 6.

Étapes de l'algorithme d'estimation des notes par Bayes

Pour chaque note qu'un client a attribuée à une page,
- sauvegarder la note,
- supprimer la note de la base (c'est-à-dire la faire passer à 0),
- pour chaque note possible sur l'article considéré (attribuée par au moins un autre utilisateur), estimer la probabilité de l'attribuer connaissant la description de l'utilisateur considéré,
- sélectionner la note de plus forte probabilité,
- calculer l'erreur commise dans l'estimation,
- stocker le résultat, pour calculer ensuite la moyenne des erreurs de prédiction, le pourcentage de mauvaises prédictions effectuées, et l'erreur maximale commise dans l'estimation,
- et réintégrer la note dans la base.

De nouveau, si on se retrouve dans le cas où l'utilisateur n'a noté qu'un article, ou qu'il est le seul à avoir noté un article, on ne pourra prédire aucune note pour lui. Pour les tests, nous ignorerons ces cas.

Bayes sur l'exemple

Le tableau 3.15 présente les résultats d'estimation des notes par Bayes.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
user 1	5	3	2	0	5	2	7	1	1	0
user 2	0	6	1	3	5	0	0	1	2	6
user 3	5	3	2	4	0	2	3	1	1	1
user 4	0	0	4	3	0	4	1	0	0	3
user 5	7	0	0	0	5	0	7	1	0	3
user 6	0	3	1	0	5	2	3	1	1	1

TAB. 3.15 – Estimations de notes par Bayes

Et le tableau 3.16 présente les erreurs d'estimation des notes par Bayes.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
user 1	-2	-3	-5	0	0	-5	0	0	0	0
user 2	0	+3	-1	+1	+2	0	0	0	+1	+3
user 3	+2	-1	+1	+1	0	0	+2	0	-1	-2
user 4	0	0	+3	-1	0	+2	-2	0	0	+2
user 5	+2	0	0	0	0	0	0	0	0	-3
user 6	0	+2	-3	0	+4	-2	+1	-5	-5	0

TAB. 3.16 – Erreurs d'estimations de notes par Bayes

La moyenne d'erreurs de predictions de Bayes dans ce cas est de 1.74

Pour illustrer le fonctionnement de cet algorihtme, nous allons détailler le calcul d'estimation de note de l'utilisateur 5 sur l'article 1 :

1. Notes possibles sur l'article 1 : 3 ou 7
2. Nouvelle description de l'utilisateur 5 : (0,0,0,0,5,0,7,1,0,6)
3. Estimation de la probabilité de prédire 3 sur l'article 1, connaissant la description de l'utilisateur 5 : $P(v_1 = 3|(0,0,0,0,5,0,7,1,0,6))$:
   - $P(v_1 = 3) = \frac{1}{2}$
   - $P((0,0,0,0,5,0,7,1,0,6)|v_1 = 3) = P(v_5 = 5|v_1 = 3) * P(v_7 = 7|v_1 = 3) * P(v_8 = 1|v_1 = 3) * P(v_{10} = 6|v_1 = 3)$
   - $P(v_5 = 5|v_1 = 3) = \frac{1+0}{7+0} = \frac{1}{7}$

- $P(v_7 = 7|v_1 = 3) = \frac{1+0}{7+1} = \frac{1}{8}$
- $P(v_8 = 1|v_1 = 3) = \frac{1+1}{7+1} = \frac{1}{4}$
- $P(v_{10} = 6|v_1 = 3) = \frac{1+0}{7+1} = \frac{1}{8}$
- donc $P((0,0,0,0,5,0,7,1,0,6)|v_1 = 3) = \frac{1}{7} * \frac{1}{8} * \frac{1}{4} * \frac{1}{8} = \frac{1}{1792}$
- et donc $P(v_1 = 3|(0,0,0,0,5,0,7,1,0,6)) = \frac{1}{2} * \frac{1}{1792} = \frac{1}{3584}$

4. Estimation de la probabilité de prédire 7 sur l'article 1, connaissant la description de l'utilisateur 5 : $P(v_1 = 7|(0,0,0,0,5,0,7,1,0,6))$:
   - $P(v_1 = 7) = \frac{1}{2}$
   - $P((0,0,0,0,5,0,7,1,0,6)|v_1 = 7) = P(v_5 = 5|v_1 = 7) * P(v_7 = 7|v_1 = 7) * P(v_8 = 1|v_1 = 7) * P(v_{10} = 6|v_1 = 7)$
   - $P(v_5 = 5|v_1 = 7) = \frac{1+1}{7+1} = \frac{1}{4}$
   - $P(v_7 = 7|v_1 = 7) = \frac{1+1}{7+1} = \frac{1}{4}$
   - $P(v_8 = 1|v_1 = 7) = \frac{1+1}{7+1} = \frac{1}{4}$
   - $P(v_{10} = 6|v_1 = 7) = \frac{1+0}{7+0} = \frac{1}{7}$
   - donc $P((0,0,0,0,5,0,7,1,0,6)|v_1 = 7) = \frac{1}{4} * \frac{1}{4} * \frac{1}{4} * \frac{1}{7} = \frac{1}{448}$
   - et donc $P(v_1 = 7|(0,0,0,0,5,0,7,1,0,6)) = \frac{1}{2} * \frac{1}{448} = \frac{1}{896}$
5. Sélection de la note de plus forte probabilité :
   $P(v_1 = 7|(0,0,0,0,5,0,7,1,0,6)) > P(v_1 = 3|(0,0,0,0,5,0,7,1,0,6))$

La note prédite par Bayes est donc 7.

Comparaison des méthodes

Les tests effectués ici ont été menés sur des matrices de notes contenant 200 utilisateurs, 500 pages, 25 groupes de pages similaires, 25 groupes d'utilisateurs similaires, une échelle de notes de 0 à 7, et un écart type de 1 séparant les notes entre utilisateurs du même groupe. Nous avons fait varier le pourcentage de notes remplissant la matrice pour observer la robustesse des différentes approches face à différents types de matrices.

D'autres tailles de matrices ont été envisagées pour les tests. Des matrices de petite taille ne sont pas significatives car les résultats dépendent alors fortement de l'échantillon de notes généré. Par contre, utiliser des matrices de plus grande taille ne modifie pas de façon significative les résultats, mais nécessite plus de temps en exécution et d'espace mémoire. La taille des matrices que nous proposons d'utiliser paraît par conséquent bien adaptée pour nos tests.

Tout d'abord, la figure 3.2 présente les résultats des principales méthodes présentées (Bayes, Pearson, vote majoritaire et vote moyen) en fonction du pourcentage de notes remplissant la matrice.

La figure 3.3 présente les résultats des différentes versions de Bayes en fonction du pourcentage de notes remplissant la matrice.

La figure 3.4 présente les résultats des différentes versions de Pearson en fonction du pourcentage de notes remplissant la matrice.

Et enfin, la figure 3.5 présente la comparaison des résultats des deux meilleures méthodes identifiées (Pearson et vote moyen pondéré de Bayes) en fonction du pourcentage de notes remplissant la matrice.

Chaque donnée calculée correspond à une moyenne des résultats sur dix tirages.

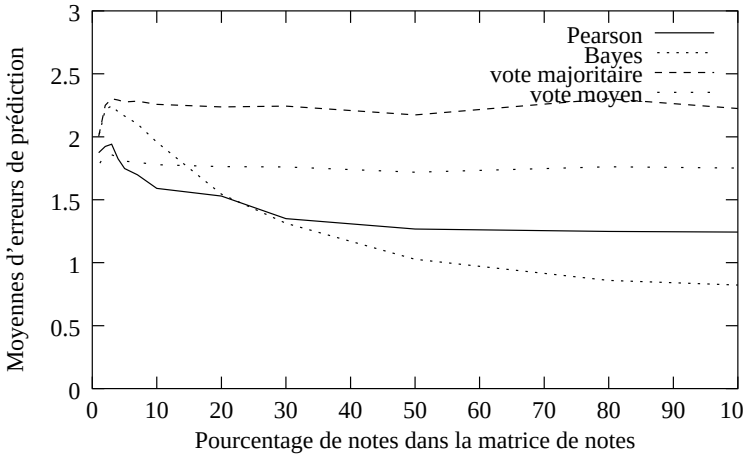

FIG. 3.2 – Comparaison de Bayes, Pearson, vote majoritaire et vote moyen

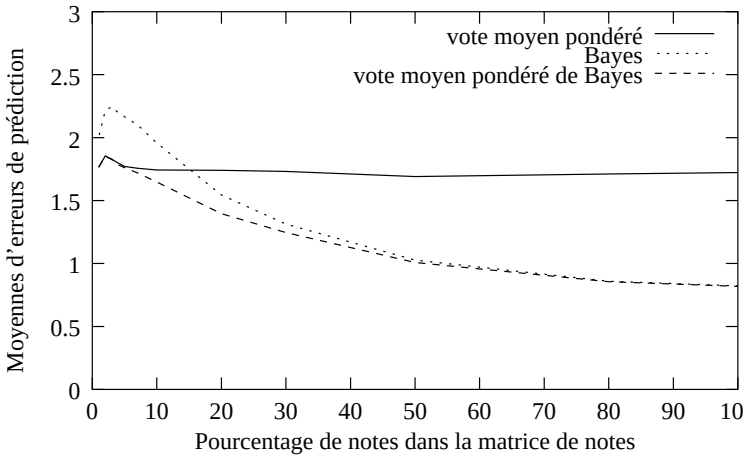

FIG. 3.3 – Comparaison des méthodes de Bayes

Influence de la répartition des notes

Le tableau 3.17 présente les résultats des tests effectués en faisant varier la répartition générale des notes de la matrice. Pour cela, nous avons légè-

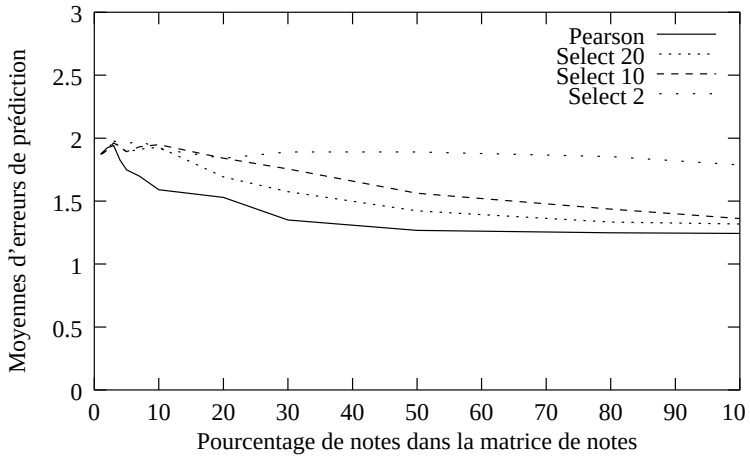

FIG. 3.4 – Comparaison des méthodes de Pearson

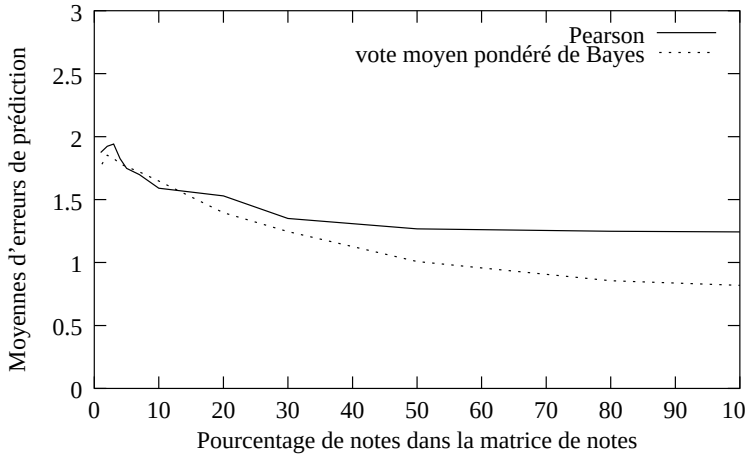

FIG. 3.5 – Comparaison des meilleures méthodes de Bayes et Pearson

rement modifié le générateur implémenté, en le forçant à prendre certaines valeurs de notes. Ainsi, les paramètres présentés précédemment restent fixes : le nombre d'utilisateurs à 200, le nombre d'articles à 500, le nombres de groupes d'utilisateurs et d'articles similaires à 25, le pourcentage de notes remplissant la matrice à 50%, l'échelle de notation à 7, et l'écart type de notes entre utilisateurs du même groupe à 1. Seule change la répartition des notes dans l'intervalle de notes proposé.

Ces résultats nous apportent peu d'information supplémentaire quant au choix de la méthode la plus adaptée à notre problème. Par contre, elle peut fournir des éléments de réflexion quant au choix de l'échantillon et de l'échelle de notation à fournir aux utilisateurs qui participent à la définition de leur profil. En effet, on peut par exemple remarquer ici qu'il vaut mieux avoir un grand nombre d'appréciations positives sur des articles, et quelques appréciations négatives (ou inversément), car nous obtiendrons ainsi de meilleurs résultats que si on a une répartition équitable des notes dans l'intervalle de notation proposé.

3.3.3 Analyse des résultats

Prendre en compte la description des utilisateurs améliore les résultats. En effet, les méthodes de Bayes et de Pearson donnent presque toujours de meilleurs résultats qu'un simple vote majoritaire ou un vote moyen, et un vote moyen pondéré de Bayes est toujours meilleur qu'un vote moyen pondéré classique.

On aurait pu penser que ne prendre en compte que l'avis des utilisateurs les plus similaires ou les plus opposés, au lieu de prendre également en compte les utilisateurs paraissant peu (ou pas) comparables, aurait pu améliorer les résultats. Il n'en est rien. En effet, il a été constaté que Pearson donne presque toujours de meilleurs résultats si on prend en compte toutes les corrélations existantes entre utilisateurs.

Entre un vote moyen pondéré de Bayes et la méthode directe de Bayes, le vote moyen pondéré de Bayes donne presque toujours de meilleurs résultats. On peut penser que cela vient du fait qu'un vote moyen pondéré sera plus souple qu'un simple choix direct de la note existante la plus probable.

En ce qui concerne la différence entre la méthode de Bayes et celle de Pearson, tout dépend du type de données en entrée de l'algorithme : alors que la méthode de Pearson reste plutôt stable quelle que soit la quantité de données en entrée, pour la méthode de Bayes, plus on fournit de données, mieux elle se comporte. Ceci s'explique par le fait que la méthode de Pearson va considérer deux utilisateurs comme corrélés dès qu'ils ont tous deux

69% notes 1	Pearson => 0.423	58% notes 1	Pearson => 0.665
24% notes 2	Bayes => 0.383	22% notes 2	Bayes => 0.636
6% notes 3	vote majoritaire => 0.383	7% notes 3	vote majoritaire => 0.914
1% notes 4	vote pondéré Bayes => 0.383	4% notes 4	vote pondéré Bayes => 0.634
0% notes 5	vote moyen => 0.383	3% notes 5	vote moyen => 0.970
0% notes 6	vote moyen pondéré => 0.394	3% notes 6	vote moyen pondéré => 1.078
0% notes 7		3% notes 7	
45% notes 1	Pearson => 0.824	32% notes 1	Pearson => 0.911
27% notes 2	Bayes => 0.828	26% notes 2	Bayes => 1.108
14% notes 3	vote majoritaire => 1.154	21% notes 3	vote majoritaire => 1.409
5% notes 4	vote pondéré Bayes => 0.826	10% notes 4	vote pondéré Bayes => 1.103
2% notes 5	vote moyen => 1.063	5% notes 5	vote moyen => 1.167
3% notes 6	vote moyen pondéré => 1.076	3% notes 6	vote moyen pondéré => 1.210
4% notes 7		3% notes 7	
19% notes 1	Pearson => 1.176	15% notes 1	Pearson => 1.283
16% notes 2	Bayes => 1.105	13% notes 2	Bayes => 1.023
17% notes 3	vote majoritaire => 1.964	13% notes 3	vote majoritaire => 2.293
16% notes 4	vote pondéré Bayes => 1.083	14% notes 4	vote pondéré Bayes => 1.006
13% notes 5	vote moyen => 1.608	15% notes 5	vote moyen => 1.758
10% notes 6	vote moyen pondéré => 1.593	14% notes 6	vote moyen pondéré => 1.721
9% notes 7		16% notes 7	
22% notes 1	Pearson => 1.472	33% notes 1	Pearson => 1.592
15% notes 2	Bayes => 0.973	12% notes 2	Bayes => 0.763
9% notes 3	vote majoritaire => 2.762	4% notes 3	vote majoritaire => 2.715
6% notes 4	vote pondéré Bayes => 0.960	2% notes 4	vote pondéré Bayes => 0.757
9% notes 5	vote moyen => 2.142	4% notes 5	vote moyen => 2.486
15% notes 6	vote moyen pondéré => 2.117	12% notes 6	vote moyen pondéré => 2.485
24% notes 7		33% notes 7	
4% notes 1	Pearson => 1.010	3% notes 1	Pearson => 0.920
4% notes 2	Bayes => 1.108	3% notes 2	Bayes => 1.079
10% notes 3	vote majoritaire => 1.591	5% notes 3	vote majoritaire => 1.400
18% notes 4	vote pondéré Bayes => 1.096	11% notes 4	vote pondéré Bayes => 1.073
21% notes 5	vote moyen => 1.405	22% notes 5	vote moyen => 1.317
20% notes 6	vote moyen pondéré => 1.314	26% notes 6	vote moyen pondéré => 1.206
23% notes 7		30% notes 7	
1% notes 1	Pearson => 0.760	0% notes 1	Pearson => 0.423
2% notes 2	Bayes => 0.812	0% notes 2	Bayes => 0.385
2% notes 3	vote majoritaire => 0.971	0% notes 3	vote majoritaire => 0.385
5% notes 4	vote pondéré Bayes => 0.811	1% notes 4	vote pondéré Bayes => 0.385
15% notes 5	vote moyen => 1.109	6% notes 5	vote moyen => 0.764
29% notes 6	vote moyen pondéré => 0.902	24% notes 6	vote moyen pondéré => 0.395
46% notes 7		69% notes 7	

TAB. 3.17 – Résultats des algos selon la répartition des notes de la matrice

noté au moins un article en commun, alors que la méthode de Bayes attend pour cela que les deux utilisateurs aient attribué la même note à l'article considéré. Par conséquent, la méthode de Bayes étant plus contraignante, elle donnera de bons résultats si elle dispose de suffisamment d'éléments de comparaison, mais elle sera beaucoup moins stable si elle en dispose de peu. Au contraire, la méthode de Pearson pourra donner de bons résultats même si elle dispose de peu d'informations en entrée (comme c'est le cas dans les matrices de notes dérivées de fichiers de log) car elle pèse chaque information reçue en fonction du degré de similarité avec l'utilisateur fournissant cette information.

Finalement, on constate que le vote moyen pondéré de Bayes est souvent meilleur que Pearson, sauf pour des matrices creuses. Or, dans le cas de notes dérivées de fichiers de log, il s'agit effectivement de matrices creuses. Le choix de la méthode dépendera donc du type de données que nous avons en entrée de l'algorithme. Remarquons par ailleurs que Pearson possède un autre avantage sur Bayes : c'est qu'on peut le combiner plus facilement avec d'autres méthodes (cf. chapitre suivant).

Chapitre 4

Perspectives de recherche

4.1 Prendre en compte l'information textuelle

Jusque là, nous avons constaté que prendre en compte la description des utilisateurs améliore les résultats. De la même façon, on peut penser que prendre en compte la description des articles aidera également. L'une des perspectives de recherche que nous envisageons est donc d'utiliser le contenu des articles lus par les utilisateurs pour aider à la caractérisation de ceux-ci.

Nous nous retrouvons de nouveau ici dans un cadre de classification, tel que nous l'avons présenté au début de ce rapport. En classification de textes, les objets considérés sont donc des textes, et les classes caractérisent d'une manière ou d'une autre leur contenu. Mais la nature particulière des objets textuels oblige à des traitements spécifiques.

4.1.1 Les différents niveaux d'analyse possible

Différents niveaux d'analyse linguistique sont possibles pour la classification. On peut décider de ne pas rentrer en profondeur dans le texte, et de ne considérer que certaines informations superficielles comme le titre ou les mots clés associés au document ; ou on peut considérer le texte comme un sac de mots (approche *bag of words*) en supposant tous les mots indépendants les uns des autres ; ou bien on peut considérer des *n-grammes*, c'est-à-dire conserver les groupes de n mots ; enfin on peut également faire appel à des informations syntaxiques (données par des grammaires) ou sémantiques (données par exemple par un *thesaurus* : dictionnaire hiérarchique décrivant des relations sémantiques entre termes), pour cibler les concepts plus généraux émanant du texte.

4.1.2 Un exemple : l'approche bag-of-words

Le preprocessing

La première étape dans la catégorisation d'un texte ([AE99]) est de transformer le document qu'on veut analyser en une représentation convenable pour l'algorithme d'apprentissage et la tâche de classification. Il s'agit donc généralement (surtout pour l'approche "bag-of-words"), dans un premier temps, de *normaliser* le texte, c'est-à-dire de se débarrasser des caractères spéciaux, puis d'utiliser une stoplist (ou liste de mots vides) pour supprimer tous les mots qui ne sont pas porteurs de sens (pronoms, prépositions, conjonctions, articles, etc.). Puis, sur les mots qui restent, on effectue une *lemmatisation*, c'est-à-dire une analyse des formes morphologiques des mots, pour ne garder que leur racine (la forme sous laquelle on les cherche dans un dictionnaire). Enfin, on regroupe les mots identiques, en les comptant ou non, selon l'algorithme qu'on veut développer.

Représentation d'un document

La représentation la plus utilisée pour stocker ensuite en mémoire le document analysé est le *modèle vectoriel*, c'est-à-dire qu'un document est représenté par le vecteur des mots qui apparaissent le plus souvent dans le document. Chaque utilisateur est alors associé à un ensemble de documents, et donc à une matrice $A = (a_{ik})$ des mots par documents. Selon l'algorithme choisi, on associera à chaque mot composant le vecteur son poids dans le document (son nombre d'occurences, ou le fait qu'il soit mis en valeur, etc.) (*word frequency weighting*), ou bien on ne s'intéressera qu'à l'absence ou la présence d'un mot (algorithme de *Bayes*). C'est à partir de ce vecteur engendré qu'on lance l'algorithme de classification.

4.1.3 La notion de profil

Une fois la classification thématique de document effectuée, on peut envisager d'intègrer les classes et mots majoritaires à un profil d'utilisateur.

La représentation la plus simple d'un profil d'utilisateur serait alors un vecteur des thèmes et mots les plus fréquents dans les lectures de celui-ci.

Mais une autre représentation possible d'un profil peut être la suivante : celui-ci serait tout d'abord constitué d'un *profil initial* rempli par le client lui-même ; puis d'un *profil thématique à long-terme* [WIY00], représenté sous la forme de vecteurs (un positif pour les textes considérés comme pertinents et un négatif pour les non-pertinents) des mots et classes les plus fréquents dans l'ensemble des lectures du client ; et enfin d'un *profil thématique à court-terme*, représenté sous la forme de vecteurs (positifs et négatifs) des mots et classes les plus fréquents dans les N dernières lectures du client.

Avec cette représentation, l'utilisateur ne serait donc pas enfermé dans un profil figé, mais l'ensemble des thèmes abordés serait toujours conservé en mémoire.

Enfin, si l'on veut intégrer le profil utilisateur dans un groupe d'utilisateurs similaires, il s'agit encore ici de classification, et on se retrouve donc de nouveau confronté au choix entre la *classification non-supervisée* (faire du clustering sur les profils utilisateurs pour former des groupes d'intérêt), et la *classification supervisée* (en caractérisant les groupes attendus a priori).

4.1.4 Limites de cette approche

Les points sensibles de l'analyse basée sur le contenu sont les suivants :
- Quel niveau d'analyse appliquer au texte ?
- Quelles sont les informations les plus discriminantes du texte ?
- Comment caractériser les classes auxquelles sont associés les textes (ensemble de classes fixé à l'avance/déterminé par clustering) ?
- Les textes appartiennent-ils à une seule classe ou à plusieurs (classification monoclasse/multiclasse) ?
- Comment valider les procédures de classification ?
- Quel est le lien entre les classes de textes et les profils ? Quelle représentation adopter pour un profil ?

4.2 Filtrage collaboratif versus analyse de contenu

À la base, nous avions des informations sur le parcours hypertextuel de nos clients. Dans un premier temps, nous en avons déduit quelles sont les pages qui ont plu, quelles sont celles qui ont déplu, et quelles sont celles qui n'ont pas été visitées. Ceci nous a permis de former un premier profil pour nos clients. Pour cela, chaque utilisateur est associé à un vecteur de notes sur les pages qu'il a parcourues. C'est cette représentation de profil qui est utilisée pour le filtrage collaboratif.

Par contre, la méthode d'analyse de contenu ne s'arrête pas là. Pour chaque page identifiée, une analyse en profondeur du texte va permettre d'en extraire les caractéristiques essentielles, pour tenter de mieux comprendre ce qui plait ou déplait dans la page au client, et former un autre profil en associant en plus à chaque utilisateur un vecteur des thèmes (et éventuellement des mots) les plus fréquemment rencontrés dans ses lectures, et constituer ainsi une mémoire des centres d'intérêts du client.

Chacune des approches possède ses limitations.

Celles de l'analyse basée sur le contenu sont les suivantes :

- Avant d'analyser le contenu, les articles doivent, dans un premier temps, être mis sous une forme convenable pour l'algorithme. Cependant, des nouvelles technologies telles le son, les photos, ou la vidéo, etc. ne peuvent être analysées automatiquement pour obtenir des informations suffisamment pertinentes.
- Etant donné que relativement peu de textes ont des vecteurs vraiment similaires, le système recommande souvent ce que l'utilisateur a déjà vu auparavant.
- À part si le système utilise un niveau d'analyse linguistique très poussé, il ne peut filtrer des articles en se basant sur leur qualité, style ou point de vue. Par exemple, si on se restreint à une approche bag-of-words, et si deux articles contiennent les mêmes termes, on ne pourra distinguer entre un bien écrit et un autre mal écrit.

Au contraire, le filtrage collaboratif n'utilisant aucune information concernant le contenu des articles (texte, image, musique, etc.), les articles filtrés n'ont pas besoin d'être préalablement analysés grammaticalement. De plus, ce système étant basé sur l'évaluation des objets par les utilisateurs, les articles recommandés peuvent alors être très différents de ceux que l'utilisateur a déjà vus auparavant, et les recommandations sont basées sur une évaluation implicite de la qualité des articles, perçue par les utilisateurs.

Mais par contre, trois limites sont présentes pour cette approche :
- La première est que la recommandation à un client dépend des autres clients.
- La seconde est qu'il faut associer à chaque client la liste des textes lus.
- Et la troisième est le fait que le système repose sur l'hypothèse implicite que toutes les caractéristiques d'un document sont de la même importance pour l'évaluation de ce document par un utilisateur. Or, deux utilisateurs peuvent donner tous deux une bonne (ou inversément une mauvaise) note à un même document pour des raisons complètement différentes.

4.3 Combinaison des méthodes

Remarquant que ces deux systèmes paraissent complémentaires, on peut penser que combiner les deux méthodes pourrait être très bénéfique. Le profil pourrait alors consister en des groupes d'utilisateurs considérés comme similaires pour un thème donné. Ou bien on peut utiliser les profils thématiques des utilisateurs pour les comparer dans la phase de calcul des corrélations entre utilisateurs du filtrage collaboratif. Ou encore, on peut utiliser le *boosting* [SSS98], qui consiste typiquement à utiliser les résultats de différentes techniques, et à les combiner.

4.3.1 Filtrage collaboratif via le contenu

Le filtrage collaboratif tel que nous l'avons présenté précédemment utilise l'ensemble des votes des utilisateurs sur les articles pour leur attribuer un degré de similarité. La matrice de données utilisée ressemble alors au tableau 4.1 suivant :

	Article 1	Article 2	Article 3	Article 4	Article 5 considéré
Utilisateur 1			7	6	? ? ?
Utilisateur 2			5	6	1
Utilisateur 3			6	6	3
Utilisateur 4	7	5			6
Utilisateur 5	7	6			4

TAB. 4.1 – Filtrage collaboratif basé sur les notes

Ici, on cherche la note que l'utilisateur 1 va attribuer à l'article 5, en fonction des notes que les autres utilisateurs ont attribué à l'article, et de la similarité entre leurs vecteurs de notes sur les autres articles.

L'idée du *filtrage collaboratif via le contenu* est d'utiliser les profils thématiques des clients, formés à partir de l'analyse du contenu des articles qu'ils ont lus, pour comparer les utilisateurs dans la phase de calcul des corrélations entre utilisateurs du filtrage collaboratif. La matrice de données utilisée ressemble alors au tableau 4.2 suivant :

	Mot 1	Mot 2	Mot 3	Mot 4	Mot 5	Article 5 considéré
Utilisateur 1	2.5	0	0.2	0	0	? ? ?
Utilisateur 2	1.1	0	1.1	1.5	0	1
Utilisateur 3	1.5	0	3.5	1.5	0.5	3
Utilisateur 4	1.1	1.1	2.1	2.0	2.5	6
Utilisateur 5	1.1	2.2	0	0	3.5	4

TAB. 4.2 – Filtrage collaboratif basé sur le contenu

Cette fois, on cherche la note que l'utilisateur 1 va attribuer à l'article 5, en fonction des notes que les autres utilisateurs ont attribué à l'article, et de la similarité entre leurs vecteurs de mots, représentant l'ensemble des thèmes identifiés comme préférés (récurrents) dans leurs lectures.

Michael Pazzani [Paz97] a obtenu ainsi de meilleurs résultats, et explique

cela par le fait d'utiliser la combinaison de deux systèmes dont les atouts paraissent complémentaires.

Une autre alternative possible, si nous possédons des informations personnelles sur les clients, est le *filtrage collaboratif démographique*. Celui-ci consiste à utiliser des informations personnelles sur les clients, du type âge, sexe, Catégorie Socio-Professionnelle, lieu d'habitation, etc., pour les comparer dans la phase de calcul des corrélations entre utilisateurs du filtrage collaboratif. La matrice de données utilisée ressemble alors au tableau 4.3 suivant :

	âge	sexe	CSP	Code habitation	Article 5 considéré
Utilisateur 1	22	M	1	59	? ? ?
Utilisateur 2	35	M	2	62	1
Utilisateur 3	18	F	1	59	3
Utilisateur 4	87	F	3	95	6
Utilisateur 5	29	M	1	62	4

TAB. 4.3 – Filtrage collaboratif démographique

4.3.2 Feature-Guided Automated Collaborative Filtering

L'idée qui a été développée ici par Y. Lashkari [Las99] est de combiner l'utilisation des caractéristiques essentielles des documents avec les évaluations des documents par les clients, c'est à dire de combiner les deux méthodes, qui paraissent complémentaires, de filtrage collaboratif et d'analyse de contenu.

Ceci permettrait de réduire le nombre d'articles à considérer pour l'algorithme de prédiction, et permettrait également d'obtenir de meilleures prédictions, puisque seuls les articles ayant des caractéristiques proches de celles du client considéré seront pris en compte pour calculer les prédictions.

Le système consiste donc en deux composantes principales : un *agent personnel d'information* pour chaque utilisateur contrôle continuellement les intérêts de l'utilisateur, recommande des documents potentiellement intéressants, et collecte et propage l'évaluation des documents par l'utilisateur ; et un *serveur de filtrage collaboratif automatique basé sur les caractéristiques (FGACF)* collecte les évaluations de plusieurs agents utilisateurs pour recommander ensuite de nouveaux documents à ces utilisateurs.

En fait, les agents utilisateurs aident à structurer l'ensemble des documents favoris d'un utilisateur. Ils apprennent les intérêts de l'utilisateur, collectent les évaluations des documents, effectuent des recommandations de documents, et peuvent aider à localiser un type particulier de documents.

Le serveur FGACF, lui, s'occupe de collecter les évaluations des agents utilisateurs. De plus, il contient un module de traitement des documents qui permet d'extraire des caractéristiques simples d'un document, qui seront utilisées pour déterminer une partition de l'espace des documents.

L'algorithme ACF peut être guidé par les caractéristiques des documents de deux façons : soit des clusters sont automatiquement formés à l'aide des corrélations entre utilisateurs, puis analysés pour trouver leurs points communs ; soit on utilise les caractéristiques du document pour déterminer une partition de l'espace des documents. On suppose que les deux formes de partitions sont utiles. À vérifier lors de la recherche en thèse...

Chapitre 5

Conclusions

Les approches présentées dans cette étude donnent des résultats très différents selon le but de l'utilisation du profil utilisateur.

Le *filtrage collaboratif* peut s'avérer très utile pour conseiller les utilisateurs sur certains textes qu'ils n'ont pas lus, et dont on sait qu'ils intéressent son groupe.

Déterminer un *profil thématique* pour chaque utilisateur permet par contre d'évaluer si un texte est pertinent ou non pour cet utilisateur, en fonction de ses centres d'intérêts identifiés.

Or, ces deux aspects du profil paraissent importants. Connaître les goûts thématiques personnels paraît la meilleure méthode pour fournir une aide spécialisée au client, mais associer les clients entre eux permet de faire bénéficier chacun des opinions des autres.

Le choix de la méthode à adopter dépendra essentiellement du degré de précision qu'on veut obtenir pour les profils, et de l'utilisation qu'on veut en faire.

Bibliographie

[AE99] Kjersti Aas and Line Eikvil. Text categorisation : a survey. report, 1999.

[BHK98] John Breese, David Heckerman, and Karl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *uncertainty in Artificial Intelligence*, 1998.

[Cha99] Philip Chan. A non-invasive learning approach to building user profiles. *Web Usage Analysis and User Profiling*, 1999.

[Fis95] Doug Fisher. Iterative optimization and simplification of hierarchical clusterings. *Artificial Intelligence Research*, 1995.

[Hec96] David Heckerman. A tutorial on learning with bayesian networks. *Microsoft Research*, 1996.

[Las99] Yezdi Lashkari. Feature-guided automated collaborative filtering. *Recommender Systems*, 1999.

[Paz97] Michael Pazzani. A framework for collaborative, content-based and demographic filtering. *Machine Learning*, 1997.

[SSS98] Robert Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. *Research and Development in Information Retrieval*, 1998.

[UF98] Lyle Ungar and Dean Foster. Clustering methods for collaborative filtering. *Recommendation Systems*, 1998.

[WIY00] Dwi Widyantoro, Thomas Ioerger, and John Yen. Learning user interest dynamics with a three-descriptor representation. *American Society of Information Science*, 2000.