Extraction de clusters à partir du treillis de concepts : application à la découverte de communautés d'intérêt pour améliorer l'accès à l'information

Nicolas Durand

Université de Caen / Basse-Normandie

Département d'Informatique
U.F.R. Sciences

École doctorale SIMEM

Extraction de clusters à partir du treillis de concepts : Application à la découverte de communautés d'intérêt pour améliorer l'accès à l'information

THÈSE

présentée et soutenue publiquement le 23 novembre 2004

pour l'obtention du

Doctorat de l'Université de Caen

Spécialité Informatique

(Arrêté du 25 avril 2002)

par

Nicolas DURAND

Composition du jury

Directeur :	M. Khaldoun Zreik	Professeur, Université de Caen
Rapporteurs :	M. Gilles Venturini	Professeur, École Polytechnique de l'Université de Tours
	M. Pascal Poncelet	Professeur, École des Mines d'Alès
Examinateurs :	M. Mohamed Quafafou	Professeur, Université d'Avignon
	M. Bruno Crémilleux	Maître de conférences, Université de Caen
	M. Luigi Lancieri	Ingénieur (Dr.), France Télécom R&D Caen

Mis en page avec la classe thloria.

Remerciements

Je tiens tout d'abord à exprimer ma reconnaissance à mon directeur de thèse, Khaldoun Zreik, pour la confiance qu'il m'a témoignée.

Cette thèse a été effectuée à France Télécom R&D Caen, et est le fruit d'une étroite collaboration entre l'université de Caen et France Télécom. J'ai été très heureux de travailler avec Luigi Lancieri à France Télécom R&D, et avec Bruno Crémilleux au GREYC. Nous avons pu concilier recherche et application. Je pense que nos collaborations n'en resteront pas là ...

Un grand merci à Amar Nezzari et Michel Dudet, mes responsables hiérarchiques à France Télécom R&D Caen pendant la quasi-totalité de ma thèse. Ils m'ont permis de réaliser ce travail dans les meilleures conditions.

Je remercie Pascal Poncelet et Gilles Venturini d'avoir rapporté cette thèse et contribué à son amélioration. Je remercie également Mohamed Quafafou de l'intérêt porté à ce travail et d'avoir accepté de faire partie du jury.

Je tiens à remercier François Rioult et Arnaud Soulet avec qui j'ai travaillé au GREYC. Le logiciel *MVMINER*, développé par François, m'a permis de réaliser une grande partie des expérimentations présentées dans cette thèse.

Je remercie toute l'équipe de France Télécom R&D dont je faisais partie : Samuel avec qui j'ai travaillé dès mon arrivée, Patrick mon collègue de bureau pendant plus de deux ans et demi, Cyril avec qui j'ai partagé le bureau les six premiers mois et qui m'a donné goût à LaTeX, et Anne, Karel, Nicolas, Cédric, Olivier, Pierre, Eric, Jean-François, Patrice et Yann, pour leur accueil et les bons moments passés ensemble (une pensée particulière pour les poyos).

Je tiens aussi à remercier Michelle Harel, Catherine Mondet, Didier Connan, Charlyne Marie, Françoise Quiquemelle et Virginie Carreau, de leurs aides pour les tâches administratives et les missions.

Je souhaite adresser un remerciement spécial à Gaëlle, Anne-Claire, Anne-Sophie, Stéphanie, Valérie, Olivier, Etham, Mathieu, Alexandre, Nicolas, Ludovic, Henrik, Gwenaël, Cyril, Jean-Raphaël, ...

Et enfin, merci aux membres de ma famille pour leur soutien.

Table des matières

Table des figures

Liste des tableaux

Notations

$\mathcal{B} = \{\mathcal{T}, \mathcal{I}, \mathcal{R}\}$	contexte de fouille de données
$\mathcal{T}$	ensemble de transactions
$\mathcal{I}$	ensemble d'items
$\mathcal{R}$	relation entre $\mathcal{T}$ et $\mathcal{I}$
n	nombre de transactions de $\mathcal{T}$
t	une transaction
m	nombre d'items de $\mathcal{I}$
$A, B, \ldots$	des items
X, Y, ABC	ensembles d'items (appelés motifs)
$\|X\|$	nombre d'items du motif X
$\mathcal{F}(X)$	fréquence du motif X
$minfr$	seuil minimum de fréquence
$\mathcal{FC}$	ensemble des motifs fermés fréquents
$\mathcal{F}'(t)$	fréquence de la transaction t dans $\mathcal{FC}$
$X \rightarrow Y$	une règle d'association
$conf(X \rightarrow Y)$	confiance de la règle $X \rightarrow Y$
$minconf$	seuil minimum de confiance
h et $h\prime$	opérateurs de fermeture de Galois
M	paramètre utilisé par ECCLAT pour sélectionner des clusters
$\mathcal{C}$	un clustering (ensemble de clusters)
k	nombre de clusters d'un clustering
c, C_i	des clusters
s_i	nombre d'éléments du cluster i
$\varphi()$	entropie d'un cluster
$E_{\mathcal{C}}$	entropie d'un clustering
$\mathcal{D}$	un document (texte, HTML, vidéo, ...)
$K_{\mathcal{D}}$	termes décrivant le document $\mathcal{D}$
$mincr$	seuil minimum de recommandation de documents
Q_i	séquence d'analyse de la régularité des accès utilisateurs
T	taille des séquences analysées
Δ	écartement entre deux séquences
$CQTS$	nombre de requêtes communes entre deux séquences
$cqmax$	valeur maximale de $CQTS$
P_w^i	pondération du terme w sur la période i
$\Delta P_w^{i,j}$	variation de la pondération du terme w entre les périodes i et j
S	seuil minimum d'accroissement

d	durée des périodes d'analyse
e	nombre de mots émergents
p	nombre de nouveaux mots émergents
N	nombre maximum de documents pré-chargés
$Tmax$	taille de l'espace alloué pour le pré-chargement

Première partie

Contexte des travaux

Introduction générale

Avec plus de 300 millions de connectés dans le monde en 2004[1], Internet est victime de son succès car le trafic induit est considérable. Les goulets d'étranglement et la surcharge des serveurs sont nombreux. Bien que les équipements réseaux et les débits aient augmenté, les informations circulant sur Internet ont aussi augmenté à la fois en nombre et en taille. La principale cause de cette hausse est l'orientation vers la vidéo et la musique. Cela a pour conséquence d'augmenter les temps de récupération d'information, et ainsi dégrader la qualité de service (QoS). A cela vient s'ajouter la difficulté de trouver l'information compte tenu du nombre très élevé de serveurs web. Les moteurs de recherche comme Voila, Google ou Altavista, tentent de pallier à ce problème en proposant des systèmes de recherche de documents parmi ceux indexés par les moteurs. Actuellement, 60% du Web public est indexé par les plus importants moteurs de recherche[2]. Bien que la recherche d'information soit facilitée, elle n'est pas pour autant évidente. Les utilisateurs ont beaucoup de difficultés à trouver les mots significatifs à écrire dans les requêtes. De plus, l'ordre des mots dans la requête peut avoir une influence sur les résultats. Dans la quasi-totalité des cas, les résultats sont "bruités". Par exemple, si un utilisateur recherche des informations sur les caches en pensant aux serveurs mandataires-caches, il obtient alors parmi les réponses un lien sur une galerie d'art dont le nom contient "cache", ou une entreprise de produits chimiques. Nous constatons donc deux problèmes qui ralentissent l'accès à l'information : la vitesse de transit, et, le temps de recherche d'information.

A l'heure actuelle, il existe des techniques de réplication permettant un meilleur accès à l'information, comme par exemple les miroirs, les CDN (Content Delivery Network) et les serveurs mandataires-caches (aussi appelés "proxies-caches"). Les systèmes de réplication tels que les serveurs mandataires-caches permettent d'accélérer la nagivation web. En effet, si un utilisateur dont le navigateur est paramétré pour utiliser un serveur mandataire, demande un objet qui se trouve être stocké dans le cache de ce serveur mandataire, alors ce dernier lui retourne aussitôt, car il n'est pas nécessaire de récupérer cet objet sur le serveur d'origine. Le temps de téléchargement se trouve alors fortement diminué, ainsi que la latence et le trafic vers les serveurs web. Notons que si un objet est présent dans le cache, cela signifie qu'à un instant donné, un utilisateur utilisant ce serveur mandataire a demandé cet objet, et que celui-ci s'est retrouvé stocké dans un espace disque dédié. Cet objet est présent dans le cache de façon temporaire. Son stockage ainsi que sa durée de vie dans le cache dépend de la politique de gestion de stockage du cache, et de la

[1] *http://www.journaldunet.com*
[2] *http://addnb.org/fr/docs/webinvisible.htm*

synergie d'intérêts liant les usagers du cache. Il est à noter que le contenu d'un serveur mandataire-cache est le résultat d'actions individuelles. Ces dernières années, beaucoup de travaux se sont axés sur l'étude et l'exploitation d'une forme d'intelligence collective induite par les accès effectués par les utilisateurs au niveau de serveurs mandataires-caches [Lan00a, LBBS01].

Pour améliorer à la fois l'accès et la recherche d'information, une nouvelle technique est apparue. Elle s'appelle le miroir actif autonome (MAA) et combine à la fois les techniques des miroirs, des caches, de réutilisation de l'information comme celle contenue dans les serveurs mandataires-caches de collectivités exploitant ainsi le phénomène d'intelligence collective. Le MAA dispose aussi d'un système de recherche de documents contenus dans son espace de stockage. Le projet CYRANO (SYstème de Réplication Actif persoNnalisé de flux audiOvisuels sur Internet) dirigé par France Télécom et auquel nous avons fortement contribué, correspond à une mise en œuvre d'une telle technique. Nous proposons d'étudier des systèmes de personnalisation dans le but d'améliorer les services offerts aux utilisateurs du système CYRANO.

Dans cette thèse, nous abordons plus spécifiquement les systèmes de recommandation de documents, et les méthodes de pré-chargement d'information dans les caches. Notre but est d'étudier les méthodes existantes selon les besoins de CYRANO, et de proposer de nouvelles approches pour améliorer l'accès à l'information. Nous verrons que l'apport principal de nos travaux consiste à former des communautés d'utilisateurs selon leurs intérêts communs grâce à une nouvelle méthode de découverte de clusters dans des données qualitatives, appelée ECCLAT. Nous utilisons alors ces clusters pour effectuer des recommandations de documents aux utilisateurs.

Ce mémoire est composé de trois parties. La première partie est consacrée à la présentation du contexte des travaux, des techniques de réplications, et du projet CYRANO (cf. chapitre 1).

Dans le chapitre 2, présent dans la partie II, nous effectuons un état de l'art des systèmes de recommandation de documents. Ces systèmes sont essentiellement constitués de deux familles. Les approches basées sur le contenu identifient la similarité entre les documents et les profils des utilisateurs. Parmi ces méthodes, certaines utilisent des méthodes de regroupement (aussi appelée "clustering") de documents. La deuxième famille correspond aux approches collaboratives. Ces méthodes identifient des utilisateurs pertinents qui ont des profils similaires, et recommandent les documents qu'ils apprécient aux autres utilisateurs ayant des profils similaires. Ainsi, chaque utilisateur bénéficie des documents consultés par les utilisateurs du même profil. A ces deux familles s'ajoutent les approches hybrides où le profil de chaque utilisateur est basé sur le contenu. La similarité entre les utilisateurs est détectée en se basant sur ces profils, et permet de mettre en œuvre des techniques collaboratives. Dans le processus de recommandation de ces systèmes hybrides, on trouve notamment une tâche de clustering des utilisateurs. Il apparaît que les systèmes hybrides utilisent mieux les informations et donc fournissent des recommandations plus précises (cf. section 2.3).

Comme nous venons de le voir, les méthodes de clustering représentent une tâche importante dans les systèmes de recommandation. Néanmoins, les méthodes existantes, dont nous présentons un état de l'art dans le chapitre 3, ne nous satisfont pas car elles ne permettent pas de capter un maximum d'informations sur les utilisateurs (ou les documents). Contrairement aux méthodes classiques formant une partition, nous pensons qu'il est plus intéressant d'obtenir un ensemble de clusters où plusieurs utilisateurs pourraient être présents dans plusieurs clusters. Nous avons alors des clusters "chevauchants" ce qui permet de capter les différents centres d'intérêt des utilisateurs, formant ainsi des communautés d'intérêt. Dans le cas de clusters de documents, cela correspondrait à différents points de vue. Le type de données est aussi à prendre en compte pour la recherche de clusters. De nombreuses informations issues du Web, comme par exemple les fichiers de trace de serveurs mandataires, contiennent des données qualitatives. De plus, il est nécessaire d'obtenir une description claire des clusters produits, de manière à faciliter leur interprétation et leur utilisation. La description d'un cluster peut être l'ensemble des propriétés partagées par les éléments du cluster. Nous verrons que parmi les méthodes existantes, il n'y en a pas qui satisfasse nos besoins c'est-à-dire traitant les données qualitatives, autorisant le chevauchement éventuel d'éléments entre les clusters, et donnant une description claire de chaque cluster obtenu. Néanmoins, certaines présentent des avantages et des idées intéressantes.

Nous proposons donc, dans le chapitre 4, une nouvelle méthode de découverte de clusters pour les données qualitatives, appelée Ecclat (Extraction of Clusters from Concept LATtice). Ecclat est basé sur de récents travaux de KDD (Knowledge Discovery in Databases) d'extraction de motifs fermés fréquents dans les bases de données composées de transactions. Une transaction est décrite par des propriétés appelées items. Un motif fermé correspond à un ensemble maximal (au sens de la longueur) d'items partagés par un ensemble de transactions. Un motif est dit fréquent lorsque le nombre de transactions contenant ce motif, est supérieur ou égal à un seuil minimal de fréquence fixé par l'utilisateur. La fermeture apparaît très importante dans la formation de cluster du fait de la maximalité. De même que la fréquence, qui donne de l'importance à certains motifs. Nous voyons alors l'ensemble des motifs fermés fréquents comme un réservoir de clusters potentiels. Un cluster correspond à un motif fermé fréquent (description du cluster) et des transactions contenant ce motif (éléments du cluster). Du point de vue de la classification conceptuelle, un tel cluster correspond à un concept du treillis de Galois qui serait fréquent.

Ce réservoir de clusters "candidats" étant important et difficilement utilisable, nous avons défini une mesure d'évaluation que nous appelons *Intérêt*. Cette mesure évalue l'erreur entre les items des transactions du cluster candidat et le motif fermé fréquent du cluster (*Homogénéité*). Elle évalue aussi la duplication des transactions correspondant aux motifs fermés fréquents générés (*Concentration*). Nous proposons ensuite une méthode de sélection privilégiant les clusters candidats ayant le meilleur *Intérêt*. Nous obtenons alors un ensemble de clusters se chevauchant, à partir desquels par ajustement de paramètres, un clustering approché est produit, ou même dans certains cas une partition.

A notre connaissance, Ecclat est la seule méthode de découverte de clusters chevauchants utilisant les motifs fermés fréquents. La première méthode de clustering formant

une partition et utilisant les motifs fréquents s'appelle *ARHP* (Association Rules Hypergraph Partitioning). Pour évaluer ECCLAT, nous utilisons des benchmarks de façon à comparer les résultats avec des méthodes existantes, des données médicales issues des Discovery Challenges PKDD 2002 et 2004, et des données provenant de fichiers de trace de serveurs mandataires de France Télécom R&D (cf section 4.3).

Dans le chapitre 5, nous décrivons un système de recommandation de documents s'appuyant sur des clusters d'utilisateurs. Ce système correspond à une approche hybride permettant de sélectionner et distribuer des documents aux utilisateurs. Nous utilisons les clusters chevauchants produits par ECCLAT sur des données issues de fichiers de trace de serveurs mandataires, et nous utilisons ce dernier pour évaluer les résultats. Nous comparons les résultats avec ceux obtenus en utilisant les clusters produits par ARHP (partition des utilisateurs). Nous montrons que les clusters produits avec ECCLAT donnent de meilleurs résultats qu'en utilisant le clustering obtenu avec ARHP (cf. section 5.3).

Le deuxième service que nous étudions de manière à faciliter l'accès à l'information, est le pré-chargement de documents (aussi appelé "prefetching") dans les serveurs mandataires-caches (cf. partie III). Ces méthodes consistent à stocker dans le cache de serveurs mandataires des documents susceptibles d'être consultés par la suite. Ainsi, lorsqu'un utilisateur fait une demande sur un document pré-chargé, ce dernier lui est aussitôt retourné. La qualité de service est donc améliorée : forte diminution du temps de téléchargement et du temps de latence. Dans le chapitre 6, nous effectuons un état de l'art des méthodes de pré-chargement d'information dans les caches de navigateurs clients et de serveurs mandataires. Dans toutes les méthodes présentées, les documents pré-chargés sont des documents déjà consultés par le passé (utilisation de l'historique de consultations) ou sont directement issus de documents consultés (par l'intermédiaire de liens hypertextes). Mais il n'y a pas de systèmes proposant de réels nouveaux documents, n'ayant qu'un lien thématique avec les documents précédemment consultés. Cela nous apparaît néanmoins une voie intéressante, et importante pour le système CYRANO, mais il est vrai assez ambitieuse.

Pour proposer un tel système, il est nécessaire de passer par l'intermédiaire d'un niveau supérieur à savoir le niveau thématique. Afin de valider l'utilisation de ce niveau, nous étudions la régularité des consultations des utilisateurs en se basant sur les URL consultées et des termes significatifs extraits de ces URL (cf. chapitre 7). Il en résulte que l'utilisation de ces termes pour caractériser les utilisateurs ne fait aucun doute : la régularité est meilleure. Nous proposons alors une méthode de détection de mots dits émergents sur une période de temps donnée pour un utilisateur. D'une façon simplifiée, un mot est émergent si le nombre de documents consultés sur la période analysée, ayant ce mot dans leurs descriptions, a augmenté par rapport à la période précédente.

Dans le chapitre 8, nous proposons un système de pré-chargement de documents, appelé PrefetchEm (Prefetching of information based on the detection of Emergences). PrefetchEm se base sur la détection de mots émergents pour trouver, via un moteur de recherche, des documents qui sont ensuite pré-chargés dans le cache. Ainsi nous obtenons de nouveaux documents portant sur les intérêts de l'utilisateur. Dans un but d'évaluation,

nous utilisons un serveur mandataire-cache Squid. Nous effectuons des tests en rejouant des fichiers de trace de serveurs mandataires pour simuler le fonctionnement du cache, à la fois sur un Squid "classique" et le Squid intégrant le système de pré-chargement. Ainsi, nous pouvons comparer les résultats obtenus en utilisant ou non le pré-chargement.

Dans la conclusion générale, nous dressons un bilan de nos travaux en terme d'études réalisées et de résultats obtenus. Nous discutons du pré-chargement et de la recommandation de documents qui sont, comme nous avons pu le constater, des méthodes assez proches dans le sens où il est nécessaire de connaître les intérêts et les goûts des utilisateurs. Cela nous amènera à l'intégration de nos travaux dans CYRANO. Enfin, nous présentons les différentes perspectives de nos travaux.

Chapitre 1

Personnalisation des systèmes d'accès à l'information sur le Web

Sommaire

Face au problème d'accès à l'information sur Internet, des techniques de réplication d'information ont vu le jour de façon à diminuer le trafic sur le réseau et améliorer la QoS (qualité de service). Dans ce chapitre, nous présentons les techniques de réplication, à savoir les miroirs, les serveurs mandataires-caches et les CDN (Content Delivery Network). Nous abordons ensuite une technique récente résultante des précédentes : le miroir actif autonome (MAA). Cela nous amènera à présenter le projet CYRANO (SYstème de Réplication Actif persoNnalisé de flux audiOvisuels sur Internet), dirigé par France Télécom R&D, qui est une mise en œuvre des MAA auxquels s'ajoutent des services de personnalisation. L'idée d'un tel système est de réduire l'espace global d'information en ne conservant que ce qui est proche des centres d'intérêts des utilisateurs du système, et de rapprocher ce sous-ensemble vers les utilisateurs en le stockant dans un espace disque dédié. Le projet CYRANO représente la motivation première et le cadre applicatif de nos travaux sur la recommandation de documents (cf. partie II) et le pré-chargement d'information (cf. partie III).

1.1 Les technologies de réplications

Dans cette section, nous présentons des techniques permettant de diminuer les temps d'accès à l'information. L'idée commune à ces techniques est de rapprocher les contenus près des utilisateurs en les stockant dans un espace disque. Cet espace correspond à un serveur web dans le cas d'un miroir. Dans les autres techniques présentées, cet espace s'appelle un *cache*.

1.1.1 Les miroirs

Un miroir est un site contenant strictement les mêmes informations qu'un site principal, ou une portion de ce site. Le plus souvent mis à jour automatiquement à une fréquence correcte, les sites miroirs sont souvent disposés dans des pays bien répartis dans les différents continents. La création de sites miroirs permet d'éviter la surcharge de connexions sur des sites trés fréquentés et donc d'améliorer la qualité de connexion. Par exemple, les serveurs proposant le téléchargement de distributions Linux proposent à l'utilisateur de choisir un miroir correspondant à son pays ou à un pays proche. Le temps de téléchargement est alors diminué.

1.1.2 Les serveurs mandataires-caches

Les serveurs mandataires-caches (aussi appelés "proxies-caches") servent de relais entre les utilisateurs et les serveurs web. Lorsqu'un utilisateur demande un document à un serveur (il effectue alors une requête), la demande est envoyée au serveur mandataire auquel il est relié. Ensuite, c'est le serveur mandataire qui effectue la requête auprès du serveur web. Le document est transmis au serveur mandataire qui le transmet ensuite à l'utilisateur.

Lorsque la fonctionnalité de cache est activée, le serveur mandataire stocke le document dans un espace disque dédié. Cet espace n'étant pas infini, il dispose d'une politique de remplacement de documents comme par exemple LRU (Least Recently Used) qui élimine le document le moins récemment consulté, ou LFU (Least Frequently Used) qui lui élimine le moins fréquemment consulté, ou des techniques plus évoluées basées sur la taille des documents, le nombre de consultation, la fréquence d'accès, etc. Si un utilisateur demande un document qui est dans le cache, alors il lui est directement transmis. Notons que les serveurs mandataires se trouvent généralement au sein d'un réseau local. Les temps de chargement sont donc très rapides.

Les caches peuvent aussi être placés au niveau des serveurs web pour alléger la charge de ces derniers ("reverse proxy"). Notons que les navigateurs web des utilisateurs disposent aussi d'une fonctionnalité de cache de fichiers. Les caches peuvent aussi être associés en groupes coopérants ou dans une hiérarchie. Par exemple, le serveur mandataire-cache de l'université de Caen est relié à un cache régional (VIKMAN) qui est lui même relié à un cache national (RENATER).

Le lecteur intéressé par les serveurs mandataires-caches peut se reporter, entre autres, à l'article de J. Wang [Wan99] et à la thèse de Nicolas Saillard [Sai03].

Dans des conditions optimales d'utilisation, les avantages des serveurs mandataires-caches sont nombreux : consommation réduite de bande passante, réduction de la charge des serveurs, réduction du temps de latence, plus de fiabilité (un document peut être dans le cache alors que le serveur d'origine n'est pas disponible), plus de sécurité et d'économie d'adresses IP (une seule connection externe, celle du serveur mandataire, utilisation possible d'un firewall pour éviter les intrusions).

Notons que chaque requête effectuée par un utilisateur est enregistrée dans un fichier de trace appelé *log*. Il est donc possible de savoir que tel utilisateur a consulté tel document à telle date. Remarquons que cela pose le problème de la mise en œuvre d'une politique de confidentialité pour les administrateurs de serveurs mandataires.

1.1.3 Les CDN

La philosophie du Content Delivery Network (CDN) est de bâtir le réseau de transport pour qu'il délivre chaque flux (Web, Real Audio, voix sur IP, ...) en fonction de ses contraintes de qualité de service propres. Pour délivrer le contenu à l'utilisateur avec une bonne qualité de service, il faut assurer cette qualité de bout en bout, et notamment au niveau des serveurs. Le CDN met à chaque niveau des ressources capables d'arbitrer les flux de données sur la base de leur "contenu" et de leurs contraintes particulières : Real Audio, flux sécurisés e-commerce (HTTPS), ..., et séparation des flux "business" et de flux "grand public" dans un même protocole (HTTP).

Les solutions émergentes en matière de CDN s'appuient sur des équipements d'aiguillage des flux : ainsi, ils savent prendre des décisions sur la base du type de flux, mais aussi du contenu du flux lui-même. Les systèmes reconnaissent les flux dynamiques, et aiguillent "intelligemment" les requêtes vers le cache pouvant fournir la réponse la plus rapide : cache dédié à une famille d'URL, cache le plus proche (minimise la traversée du réseau), ou cache le moins chargé (minimise le temps de traitement).

1.1.4 Bilan : vers le miroir actif autonome

La mise en place de systèmes de réplication permet de diminuer les temps de téléchargement. Néanmoins, les miroirs sont coûteux en terme de mise à jour des contenus et de dispersion sur l'ensemble du monde, et ils ne permettent que d'accélérer le téléchargement d'un ensemble particulier d'informations. Quand aux technologies liées aux CDN, elles ne sont pas très développées car elles ne sont pas très mûres et sont coûteuses dans leur mise en place. Les systèmes de réplication les plus répandus et efficaces dans une perspective d'adaptabilité à moindre coût, sont les serveurs mandataires-caches. Pour mesurer l'efficacité d'un cache, nous pouvons évaluer le nombre de fois qu'un document présent dans le cache a été demandé (nombre de *hits*). De nombreuses études ont montré des taux de hits entre 30 et 70% [Mar96]. Les caches semblent être une bonne solution pour accélérer l'accès à l'information, mais on peut noter des points non exploités au niveau du contenu du cache. En effet, il n'y a pas de prise en compte des goûts des utilisateurs dans la politique de remplacement des documents. De plus, il n'y a aucune visibilité sur les documents contenus dans le cache. Il semblerait intéressant d'avoir accès au contenu. Ainsi les utilisateurs pourraient consulter directement les documents cachés qui les intéressent

et que peut-être ils n'auraient jamais demandés. Le problème est qu'on ne maîtrise pas le temps de mémorisation dans le cache.

En se basant sur ces constats, un nouveau concept a été mis au point. Le miroir actif autonome (MAA) est un système récemment introduit [Lan00b] dont le but est de réutiliser le contenu provenant d'une source d'information (comme les caches de serveurs mandataires) en les présentant aux utilisateurs par l'intermédiaire d'une interface web. Un MAA dispose d'un espace de stockage et d'une politique de remplacement des objets tout comme un cache classique. Le projet CYRANO est une mise en œuvre des MAA, et dispose de fonctionnalités supplémentaires, comme une meilleure indexation des documents stockés, une meilleure prise en compte du profil des utilisateurs du système pour la constitution du contenu de l'espace de stockage, et un système de gestion optimale de la coopération entre plusieurs MAA.

1.2 Le projet CYRANO

Le projet CYRANO[3] [LCM02] correspond au contexte de nos travaux et justifie nos recherches et nos choix techniques. Il est axé sur les technologies de réplication coopérative, l'indexation multimédia, ainsi que la distribution de contenus sur Internet. Il est constitué de trois partenaires : France Télécom R&D (leader), l'école centrale de Lyon, et l'INRIA Rocquencourt, correspondant à trois axes :

- la caractérisation des centres d'intérêt des utilisateurs,
- la caractérisation des documents multimédia,
- la mise en relation de serveurs de proximité (éléments de CYRANO) selon divers modes de coopération (e.g. Peer to Peer).

Le système CYRANO correspond à la mise en œuvre et à une évolution du concept de miroir actif autonome [Lan00b], et a fait l'objet d'un dépôt logiciel [LLDS02].

Ce projet consiste à rapprocher vers les utilisateurs des contenus susceptibles de les intéresser, de façon à améliorer la recherche et l'accès à l'information. CYRANO utilise des techniques similaires aux serveurs mandataires-caches pour la gestion de l'espace de stockage et la politique de remplacement des documents, auxquelles s'ajoute la prise en compte de profils utilisateurs. L'indexation des documents par l'extraction de termes, permet aux utilisateurs d'effectuer des recherches dans l'ensemble des documents disponibles. Les consultations des documents permettent de calculer les profils utilisateurs. L'originalité majeure de CYRANO est la prise en compte de ces profils dans le contexte de cache.

1.2.1 Fondements du projet

Un point très important du projet CYRANO est d'exploiter les synergies latentes et les phénomènes d'intelligences collectives [Lan00a, LBBS01] au sein d'une communauté. Nous donnerons un exemple dans la section 1.2.2. Les regroupements communautaires telles que les associations ou les entreprises, ou les regroupements d'individus peuvent

[3] SYstème de Réplication Actif persoNnalisé de flux audiOvisuels sur Internet, projet du Réseau National de Recherche en Télécommunications, labélisé en 2000 pour une durée de deux ans et demi.

souvent être expliqués en terme d'intérêts partagés. Il est donc intéressant de capturer et d'exploiter les phénomènes de coopération implicite. Prenons un exemple simple de conséquence de ces phénomènes. Un document qui a été découvert et consulté par 30% d'une communauté a des chances d'intéresser une bonne majorité des 70% restant. Il y a un phénomène de filtrage implicite dans la même logique que le filtrage collaboratif présent dans les systèmes de recommandation (cf. chapitre 2).

L'idée est d'utiliser l'action naturelle des utilisateurs pour réduire un espace de données qui sera dès lors implicitement orienté vers leurs centres d'intérêt. De plus, l'information coûte cher à produire. Sa réutilisation [Lan01] permet de faire des économies, et d'en faire profiter à de nouveaux utilisateurs. Il existe deux stratégies de réutilisation : à la source de la consultation (recyclage du contenu lourd des caches), et à la source de diffusion (par exemple des programmes en provenance d'un satellite). CYRANO prend en compte les deux cas de réutilisation.

Si nous prenons le cas d'une entreprise, l'ensemble des employés partagent des intérêts communs dans le domaine d'activité de l'entreprise. Généralement, l'entreprise dispose d'un serveur mandataire-cache qui permet à l'ensemble de ses employés d'accéder à Internet. Il y a donc un enjeu important de réutiliser les contenus du cache. En effet, un employé peut être très intéressé par un document consulté par un de ses collègues mais ne jamais savoir que ce dernier l'a consulté et que le document est stocké dans le cache (donc très rapidement téléchargeable).

1.2.2 Architecture de CYRANO

Architecture générale

La figure 1.1 représente le contexte général d'utilisation et de fonctionnement du système CYRANO, dans le cadre d'un réseau local d'entreprise où chaque utilisateur accède à Internet par l'intermédiaire d'un serveur mandataire-cache (proxy-cache). Un serveur de proximité de CYRANO, appelé CyraNode, se trouve aussi sur le réseau et est relié au serveur mandataire-cache ainsi qu'à toute autre source de contenu comme par exemple un satellite.

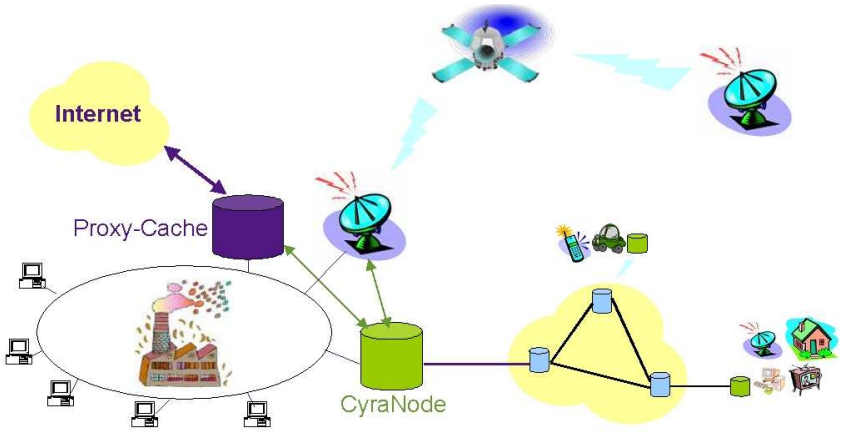

FIG. 1.1 – Schéma général du système CYRANO

Prenons un exemple : un utilisateur effectue une recherche sur Internet et consulte un document technique sur les réseaux IP. Ce document se trouve caché par le serveur mandataire-cache de l'entreprise. La CyraNode qui est relié au cache, détecte ce nouveau document et teste s'il correspond au profil des utilisateurs de la CyraNode. Supposons que le profil de la CyraNode soit très axé réseaux et programmation, le document est alors indexé et copié dans l'espace de stockage de la CyraNode.

Un autre utilisateur qui a le même intérêt que le premier concernant les réseaux IP, désire rechercher des documents sur ce sujet. Au lieu d'utiliser un moteur de recherche sur Internet, il se connecte à la CyraNode par l'intermédiaire d'une interface Web (cf. annexe A, figure A.4). Il entre des mots dans l'espace consacré à la recherche. Parmi les résultats, il trouve le précédent document qu'il pense intéressant, et le consulte. Le chargement est très rapide puisque le document est stocké dans la CyraNode, et le document ne transite que par le réseau local.

Au final, le deuxième utilisateur a bénéficié de la précédente consultation de document de son collègue. Dans ce cas, les résultats des recherches sur la CyraNode sont très faiblement bruitées contrairement à un moteur de recherche classique. Cela a donc permis d'améliorer l'accès à l'information et de faire gagner du temps au deuxième utilisateur.

Nous détaillons maintenant les différents modules mis en jeu : capture des contenus, interface utilisateur, gestion de l'espace de stockage, interactions entre les CyraNodes, et un module central gérant la connaissance de la CyraNode (notre principale contribution au projet).

Capture des contenus

Dans le cas de la récupération de contenus dans un cache, la CyraNode scrute en permanence l'état du cache. Lorsqu'un nouveau document est stocké dans le cache, la CyraNode extrait des termes de façon à évaluer s'il est proche des intérêts des utilisateurs. L'extraction des termes s'effectue selon le type du document. Pour les documents de type texte, comme par exemple les fichiers pdf ou postcript, les différentes étapes consistent à extraire le texte, puis détecter un ensemble de mots d'un dictionnaire les plus occurrents dans le document. Pour les vidéos, il y a une étape préliminaire utilisant un système de reconnaissance vocale pour extraire un texte correspondant à la bande sonore [HC02]. Ensuite, la détection de termes s'effectue comme pour un document de type texte[4]. Dans le cas d'une autre source d'information comme par exemple un satellite, le système enregistre les émissions d'une manière automatique (horaire et canal programmés). Les fichiers sont alors traités comme précédemment.

Interface utilisateur

L'interface utilisateur se présente sous la forme de pages web, permettant de rechercher et visualiser des documents stockés dans la CyraNode. Différentes captures d'écran sont

[4]L'indexation des vidéos et des contenus audio a été approfondie par l'Ecole Centrale de Lyon. L'extraction de termes à partir de textes, ainsi que l'intégration dans la CyraNode, a été réalisée par France Télécom R&D Caen.

présentées en annexe A. Un utilisateur se connecte en s'authentifiant avec un pseudo et un mot de passe (cf. figure A.1). Ensuite, une page d'accueil apparaît contenant les dernières informations sur le système (derniers documents récupérés, statistiques, ...), ainsi que des éléments de personnalisation comme par exemple des boutons de recherche rapide sur des thèmes choisis par l'utilisateur et/ou correspondant à son profil (cf. figure A.2). Un utilisateur connecté a la possibilité d'effectuer des recherches parmi les documents stockés dans le cache de CYRANO (cf. figure A.4). Lorsqu'un utilisateur consulte un document (cf. figure A.5), cette consultation est notifiée au système de façon à raffiner par la suite ses centres d'intérêt et calculer son profil. Ce module a été réalisé par France Télécom R&D Caen.

Gestion de l'espace de stockage

L'espace de stockage d'une CyraNode étant limité, une politique de gestion des documents contenu dans l'espace disque a été mise en œuvre[5]. Le contenu de l'espace disque d'une CyraNode se constitue en fonction de son profil. Ce profil correspond à un ensemble de mots. Les documents contenus dans une CyraNode sont gérés par l'intermédiaire d'un calcul de priorités. La priorité d'un document est déterminée en fonction du nombre de thèmes communs entre celui-ci et le profil de la CyraNode. Lorsque l'espace de stockage des documents de la CyraNode est plein, un nouveau document est stocké si sa priorité est supérieure à celles de documents ayant les plus petites priorités. Si la priorité du document est suffisamment importante, et il est possible de libérer assez d'espace disque, alors le système efface des fichiers de priorité inférieure. Remarquons que pour éviter un phénomène de stagnation des documents dans la CyraNode, un mécanisme décroît les priorités des documents au cours du temps. Notons aussi le parallèle avec la gestion de l'espace de stockage des serveurs mandataires-caches. La différence est une prise en compte du profil des utilisateurs, par opposition à la politique plus simpliste des serveurs mandataires-caches (LFU, LRU, cf. section 1.1.2).

Interactions entre les nœuds de CYRANO

Plusieurs CyraNodes peuvent être réliées afin de bénéficier du contenu des autres (cf. partie droite de la figure 1.1). Un système de médiation permet de gérer et centraliser les informations nécessaires. La coopération entre les CyraNodes est prise en charge par Cymes (SYstème de MEdiation de CYRANO)[6] [JM01, LFP03]. Il s'agit de modules placés dans le réseau Internet qui dialoguent avec les CyraNodes et assurent la gestion de flux de données entre elles, et le cas échéant avec des fournisseurs de contenus professionnels qui ne disposent pas d'interface directe avec les utilisateurs finaux. Grâce à Cymes, les CyraNodes constituent les nœuds d'un réseau actif. Ainsi structuré, le réseau Cyrano peut mettre en œuvre des modes de collaboration variés telles que Peer to Peer, tout en gardant son interopérabilité avec le réseau Internet. Nous ne détaillons pas dans ce document le système Cymes. Le lecteur intéressé peut se reporter aux pages web consacrées à ce système : *www−sor.inria.fr/projects/wise/cymes/*.

[5]La gestion de l'espace de stockage a été développé à France Télécom R&D Caen.

[6]Cymes est développé à l'INRIA de Rocquencourt.

Gestion de la personnalisation de la CyraNode

Le rôle de ce module[7] est de centraliser les connaissances nécessaires au fonctionnement de la CyraNode, ce qui le rend très important. Toutes les informations sont stockées dans une base de données. Toutes opérations effectuées sur cette base sont le résultat d'appels à des fonctions d'une bibliothèque spécialement développée. Ainsi, chaque module appelle les fonctions qui lui sont dédiées. Par exemple, le module de capture de contenu appelle la fonction permettant d'insérer les informations correspondant à un nouveau document stocké dans la CyraNode, tel que le nom ou les termes extraits. Le module de gestion de l'espace de stockage indique à la base où trouver un nouveau document stocké. Parmi les autres informations contenues dans la base de données, nous trouvons notamment les pseudos et mots de passe des utilisateurs de la CyraNode. Tout cela permet à l'interface utilisateur d'effectuer les authentifications pour les connexions, d'effectuer des recherches de document (par appel à une fonction de la bibliothèque), d'obtenir le chemin d'accès d'un document dans l'espace de stockage de la CyraNode. Nous avons aussi ajouté un mécanisme permettant à l'administrateur de la CyraNode de poser des filtres sur certains documents ou types de documents, de manière à interdire la visualisation de ces documents par les autres utilisateurs.

Un point important de ce module, est la gestion du cœur de la personnalisation. En effet, en stockant les documents, les termes les caractérisant et les consultations par les utilisateurs, le module met à disposition des autres modules des fonctions de personnalisation. Par exemple, l'interface peut obtenir les termes les plus consultés (par l'intermédiaire des documents) pour un utilisateur donné, de façon à présenter à cet utilisateur une page d'accueil personnalisée avec des boutons de recherche rapide (cf. annexe A, figure A.3). Ces boutons correspondent à une combinaison des termes les plus consultés et les plus demandés dans les recherches par l'utilisateur. Le module met aussi à disposition la fonction de calcul du profil de la CyraNode. Cette fonction est utilisée par le module de gestion de l'espace de stockage. Le profil de la CyraNode correspond au profil fusionné de ses utilisateurs : un ensemble de termes les plus consultés par tous les utilisateurs de la CyraNode. Tous les utilisateurs n'ont pas forcément les mêmes centres d'intérêt. Il en ressort donc le ou les centres d'intérêt majoritaires.

Comme nous venons de le voir, ce module est central à la CyraNode. Il est mis en jeu dans la quasi-totalité des interactions entre les autres modules. Il doit être capable de fournir toutes les informations nécessaires pour effectuer des tâches de personnalisation, comme la recommandation de documents aux utilisateurs (cf. chapitre 5) et le préchargement d'information (cf. chapitre 8).

1.3 Conclusion

En résumé, CYRANO est un réseau de serveurs de proximité (CyraNodes) qui permettent de capter le profil des utilisateurs, de récupèrer et gérer seuls leur contenu en

[7] La gestion de la personnalisation de la CyraNode a été développé à France Télécom R&D Caen, et correspond à notre principale contribution au projet CYRANO.

fonction des profils utilisateurs, et de présenter le contenu aux utilisateurs du système. Les avantages sont les suivants : optimisation du coût et du temps de constitution du contenu, rapidité d'accès à l'information, et réduction du temps de recherche.

Des études [LBBS01] ont montré que le niveau de réutilisation d'information est 9 fois plus important que celui d'un serveur mandataire-cache seul sur la même période. De plus, en ce qui concerne la pertinence de l'information correspondant à des requêtes utilisateurs volontairement ambiguës, les tests ont montré une amélioration notable des réponses. Comparé à un moteur de recherche généraliste sur le Web, la recherche sur le miroir autonome a donné un niveau de satisfaction client 4,5 fois plus important [Lan04].

Notre principale contribution au projet CYRANO[8] a concerné le module de gestion de la personnalisation de la CyraNode (tant au niveau de la conception que du développement), et la création de "briques blanches" comme la découverte de clusters et la gestion de la personnalisation de serveurs mandataires. Nous présentons ces deux briques respectivement dans les chapitres 4 et 8. Nous avons aussi pris part à la conception des modules développés par France Télécom R&D. Pendant plusieurs mois, nous avons réalisé la maintenance de la plate-forme située à France Télécom R&D Caen et géré l'évolution des développements des différents modules.

CYRANO améliore l'accès à l'information en mettant en œuvre une réplication sélective. Néanmoins, il manque des fonctionnalités qui amélioreraient aussi l'accès à l'information comme la recommandation de documents aux utilisateurs et le pré-chargement d'information. En effet, il serait utile de recommander des documents aux utilisateurs en fonction de leurs centres d'intérêt, et ainsi leur faciliter la sélection de l'information. Le pré-chargement d'information serait utile dans le processus de constitution du contenu de la CyraNode. Cette dernière utilise les serveurs mandataires-caches ou une source d'information comme le satellite pour constituer son contenu, mais elle ne dispose pas de processus interne de pré-chargement allant rechercher de l'information sur le Web par exemple.

Dans le reste du mémoire, nous allons étudier ces deux services facilitant l'accès à l'information. La partie II est consacrée aux systèmes de recommandation de documents. La partie III est dédiée aux méthodes de pré-chargement d'information dans les caches.

[8] Au niveau des trois partenaires, le projet CYRANO a induit 7 brevets et 32 publications internationales dont 7 dans des revues. En ce qui concerne France Télécom, CYRANO se traduit par 2 brevets, 2 dépôts de logiciel, et 6 publications internationales. CYRANO a aussi permis la mise en place d'une plate-forme de suivi des usages et des comportements d'utilisateurs au sein de France Télécom R&D.

Les perspectives de ces travaux sont nombreuses. D'une part, il est possible de ré-utiliser les briques blanches dans d'autres contextes. D'autre part, la plate-forme mise en place offre un cadre d'étude du comportement et des communautés d'utilisateurs. France Télécom va étudier plus en détails d'autres points comme la communication entre les CyraNodes et la migration automatique des contenus, et l'approfondissement des phénoménes d'intelligences collectives. Ces travaux sont poursuivis dans le cadre de deux thèses.

Deuxième partie

Nouvelle méthode de regroupement d'utilisateurs et application à la recommandation de documents

Chapitre 2

Les systèmes de recommandation

Sommaire

Dans ce chapitre, nous proposons un état de l'art des systèmes de recommandation de documents. Il n'est pas exhaustif, mais donne un apercu des systèmes existants. Les méthodes présentées sont les plus populaires. Notre objectif est d'évaluer leur utilisation dans le cadre du projet CYRANO (cf. section 1.2). Nous constatons que beaucoup de ces systèmes utilisent des méthodes de clustering pour regrouper des utilisateurs ou des documents similaires. Nous formulons quelques critiques, ainsi que nos besoins en terme de système de recommandation et de méthode de regroupement d'utilisateurs.

Les systèmes de recommandations sont assimilés aux systèmes de filtrage d'information car le principe et les techniques sont très proches. Cela correspond à sélectionner et filtrer des documents susceptibles d'intéresser l'utilisateur. Ces systèmes sont segmentés en trois catégories : le filtrage par le contenu, le filtrage par collaboration, et les approches hybrides.

2.1 Filtrage par le contenu

Le premier type d'approche correspond aux systèmes de filtrage par le contenu. Ces méthodes identifient et fournissent des documents intéressants aux utilisateurs en se basant sur la similarité entre les documents et les profils.

Parmi ce type d'approche, nous pouvons citer le système *Syskill&Webert* [PMB96] qui construit un classifieur bayésien à partir d'une base d'apprentissage contenant des pages web notées par l'utilisateur. Le classifieur est ensuite utilisé pour déterminer si une page est susceptible d'intéresser l'utilisateur. Le système *SiteHelper* [NW97] effectue des recommandations seulement sur les documents d'un site web, et n'est pas autonome.

Dans *Amalthaea* [Mou97], l'utilisateur dispose d'un agent évolutif qui permet d'une part de créer et modifier son profil, et d'autre part de filtrer et rechercher des documents

parmi des sources d'information (cf. figure 2.1). Ce système utilise des clusters pour représenter les intérêts de l'utilisateur. Le système est capable de collecter sur le Web, et aussi de filtrer à partir d'un flot d'information (comme des articles de presse) des documents intéressants pour l'utilisateur. Nous remarquons donc deux points intéressants dans ce système. D'une part, l'utilisation de clusters pour capter les différents centres d'intérêt de l'utilisateur, et d'autre part la capacité à aller chercher ou de filtrer de l'information intéressante pour l'utilisateur. Mais ce système est à base d'agents logiciels dédiés à un individu, ce qui peut entrainer des problèmes de performances dans un système tel que CYRANO où le nombre d'utilisateurs peut être très élevé.

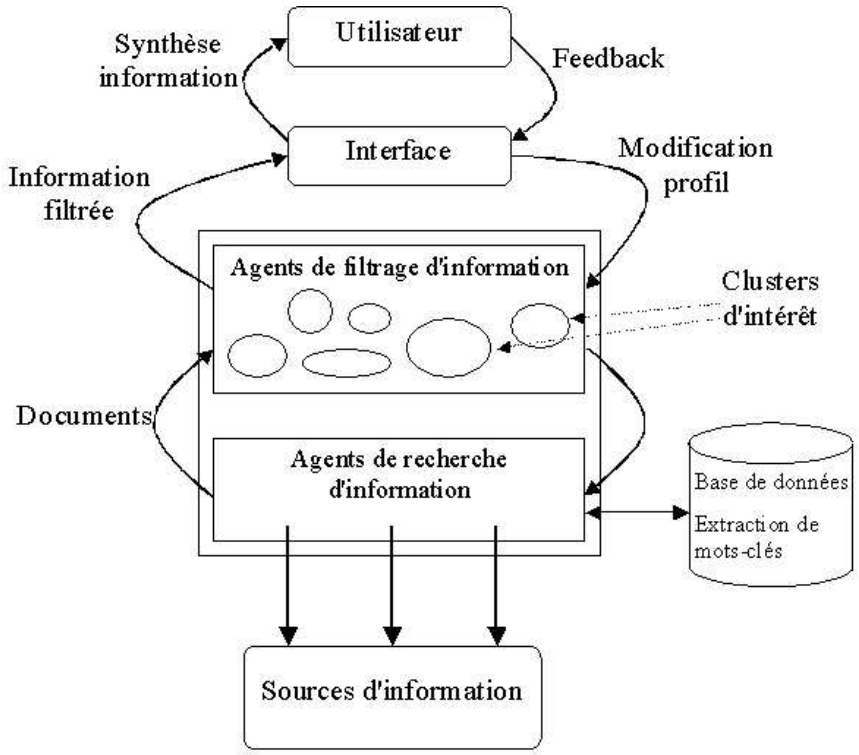

FIG. 2.1 – Architecture générale du système de recommandation *Amalthaea*

Les trois systèmes que nous venons de présenter demandent une forte implication des utilisateurs qui doivent en effet spécifier et gérer leurs profils. Cela peut être en défaveur du système de recommandation, car le temps et les efforts demandés à un utilisateur pour la gestion de son profil, peuvent l'inciter à s'orienter vers un système plus simple d'utilisation. De plus, le temps perdu supprime les avantages à utiliser un tel système au lieu d'un moteur de recherche. Nous pensons donc qu'il est préférable de minimiser au maximum les contraintes imposées aux utilisateurs, en préférant un système transparent qui détecte automatiquement leurs profils.

Letizia [Lie95] est un agent coté client qui recherche sur le Web des pages similaires à celles déjà consultées par l'utilisateur ou contenues dans ses bookmarks. Il fonctionne donc de manière autonome en utilisant l'historique de navigation. La mise en œuvre de *Letizia* dans CYRANO ne serait pas possible, car c'est un agent individuel. Il faudrait donc un

agent pour chacun des utilisateurs de CYRANO, ce qui conduirait à des problèmes de performance du système.

WebWatcher [JFM97] utilise les fichiers de trace d'un serveur mandataire pour effectuer des recommandations. Il suggère des pages similaires à celle en cours de consultation par un utilisateur, en se basant sur la structure des liens hypertextes. Or les documents présents dans une CyraNode ne sont pas a priori liés entre eux. Il n'est donc pas envisageable d'utiliser un tel système.

Mobasher et al. [MCS99] ont proposé un système de recommandation basée sur le clustering d'URL à partir de fichiers de trace d'un serveur web (cf. figure 2.2).

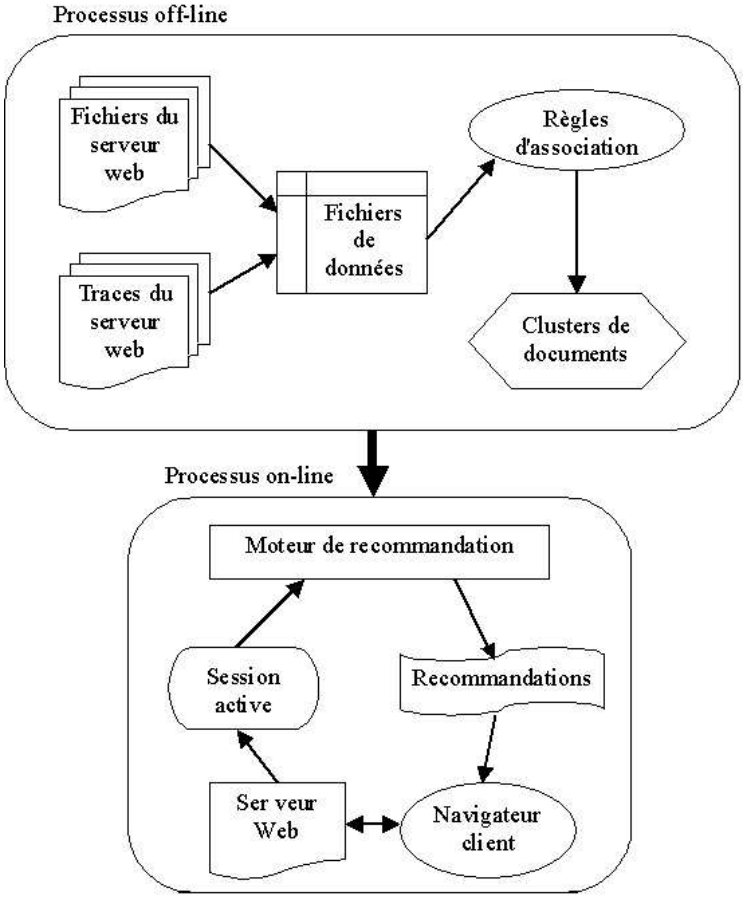

FIG. 2.2 – Architecture générale du système de recommandation de Mobasher et al.

Le système est composé d'un processus off-line de formation de clusters d'URL en utilisant une méthode appelée ARHP (Association Rules Hypergraph Partioning). Nous présenterons cette méthode en détail dans la section 3.3.2. Cet algorithme regroupe des URL qui sont fréquemment consultées ensemble dans les sessions des utilisateurs, en utilisant des règles d'association (cf. section 3.1). Cela permet au système d'obtenir des clusters qui captent potentiellement les intérêts communs des utilisateurs. Les recommandations sont calculées dans un processus on-line. Le système détermine les URL qui peuvent potentiellement intéresser l'utilisateur par matching des URL de la session cou-

rante de l'utilisateur avec celles des clusters. Par exemple, si un utilisateur consulte une URL présente dans un cluster, le système lui recommande les autres URL du cluster. Les expérimentations de cette méthode se sont effectuées sur un serveur web d'une université où l'ensemble des documents est connu. Ce point est important, car si on se place dans le contexte de CYRANO, les documents présents dans l'espace de stockage sont en constante évolution. Cela est dû à la politique de remplacement des documents dans l'espace de stockage de la CyraNode (cf. section 1.2.2). Par conséquent, il serait difficile et coûteux de mettre en œuvre une telle technique dans CYRANO. Il en ressort qu'un système basé sur le clustering de documents, au sens strict, est difficilement envisageable dans notre cas.

2.2 Filtrage collaboratif

Le deuxième type d'approche correspond aux systèmes de filtrage par collaboration. Ces systèmes identifient les utilisateurs pertinents qui ont des profils similaires, et fournissent les documents qu'ils apprécient aux autres utilisateurs similaires. Au lieu d'utiliser une similarité entre les documents et les profils, cette approche mesure la similarité entre les profils utilisateurs.

Parmi les méthodes de ce type, nous pouvons citer *Tapestry* [GNOT92] et *Grouplens* [KMM+97] qui permettent de commenter les documents des forums et de bénéficier de ceux recommandés par les autres utilisateurs. Nous ne nous attardons pas sur ces deux systèmes, car ils ne sont pas autonomes et demandent une forte implication des utilisateurs.

Parmi les systèmes collaboratifs basés sur le clustering d'utilisateurs, nous avons *RecTree* [CHW01]. Il construit un arbre à partir d'un ensemble de données quantitatives sur des utilisateurs. Chaque utilisateur est décrit par un ensemble de valeurs continues : notes données à des films. L'algorithme partionne les données en clusters d'utilisateurs similaires en scindant récursivement l'ensemble des données en deux clusters fils. Les coupures sont choisies de manière à maximiser la similarité intra-cluster et à minimiser la similarité inter-cluster (principe du clustering, cf. chapitre 3). Pour cela, *RecTree* utilise l'algorithme de clustering K-Means (cf. section 3.2.1). Chaque nœud de l'arbre correspond donc à un cluster d'utilisateurs. *RecTree* génère des recommandations pour un utilisateur en le localisant dans un cluster correspondant à une feuille de l'arbre, et en appliquant une méthode basée sur la déviation pondérée pour identifier les films à lui recommander. Remarquons que dans ce système, les utilisateurs doivent noter les films qu'ils ont vus. Les utilisateurs sont donc très sollicités. Or dans le cadre de CYRANO, il serait plus adapté d'avoir un système transparent pour les utilisateurs. Nous nous tournons donc vers des approches autonomes.

Parmi les méthodes autonomes, nous pouvons citer celles basées sur le regroupement et l'association d'items (livres, journaux, émissions télévisés,...). Wilson et al. [WSO02] utilisent les associations fréquentes contenant deux items (ici des programmes télévisés). Il n'y a pas ici de clustering d'utilisateurs, mais des regroupements de deux programmes télévisés fréquemment vus ensemble par les utilisateurs. Le profil d'un utilisateur est constitué par les programmes qu'il a vu. Les associations précédemment trouvées sont alors utilisées pour déterminer la similitude entre deux profils d'utilisateurs et effectuer

des recommandations de programmes. D'une façon plus abstraite, si A, B, C et D sont des programmes télévisés, un utilisateur a comme profil $\{A, B\}$, un autre utilisateur, le profil $\{C, D\}$, et il existe une relation entre B et C, alors les deux utilisateurs peuvent être comparés (cf. figure 2.3). Comme il existe un lien entre les deux utilisateurs, le système recommande le programme D au premier utilisateur car ce dernier ne l'a pas déjà regardé. Nous pourrions imaginer un tel système dans CYRANO, où les items serait les documents. Mais nous avons le problème de la durée de vie des documents dans CYRANO. Dans le cas d'un programe télévisé, il n'y a pas de problème car une émission est réccurente (journalière, hebdomadaire, ...) et sa programmation à l'antenne est rarement changée avant un an. Dans CYRANO, cela impliquerait une mise à jour constante des associations de documents, ce qui serait coûteux en temps de calcul.

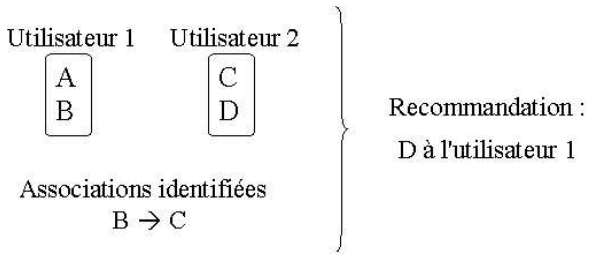

FIG. 2.3 – Exemple de recommandation du système de Wilson et al.

Nous pouvons remarquer l'utilisation des notions d'association d'items et de fréquence (cf. section 3.1) qui nous parraissent très intéressantes pour associer des utilisateurs. Nous présenterons ces notions dans le chapitre 3. Notons qu'elles sont aussi utilisées par certaines méthodes de clustering.

2.3 Approches hybrides

Récemment, de nouvelles méthodes dites hybrides sont apparues. Elles sont à la fois collaboratives et basées sur le contenu. Pazzani [Paz99] a montré par des expérimentations que les systèmes hybrides utilisent mieux les informations et donc fournissent des recommandations plus précises. Pazzani parle de *collaboration via le contenu*, où le profil de chaque utilisateur est basé sur le contenu des documents, et est utilisé pour détecter la similarité parmi les utilisateurs.

La collaboration via le contenu est similaire à peu de choses près à l'approche mis en œuvre dans *Fab* [Bal97], où plusieurs agents (un par utilisateur) collectent des documents et les placent dans un répertoire commun pour que les agents de sélection les recommandent par la suite. *Fab* tire avantage de l'intérêt partagé par les utilisateurs, sans perdre le bénéfice des représentations fournies par l'analyse du contenu.

Parmi les systèmes existants, nous pouvons citer *OTS* (Online Thesis System) [WCC01]. Il permet à un groupe d'utilisateurs de consulter, par l'intermédiaire d'une interface, des articles fournis par un serveur de publication (cf. figure 2.4).

Nous remarquons une certaine analogie dans le fonctionnement avec CYRANO, puisque les utilisateurs peuvent consulter un ensemble de documents via une interface web, et

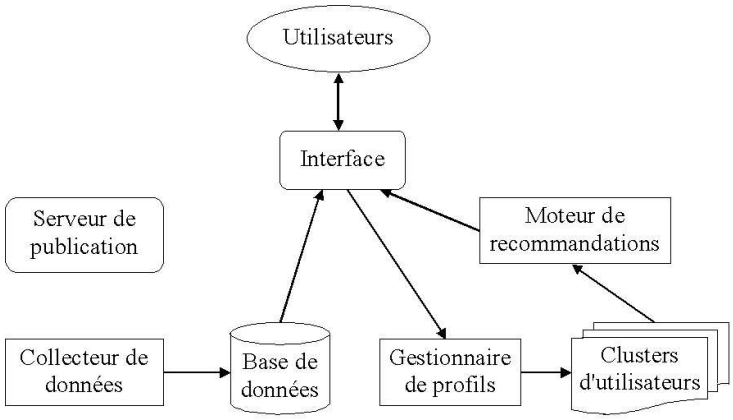

FIG. 2.4 – Architecture générale du système *OTS*

que les consultations sont archivées. *OTS* est un système hybride de type collaboration via le contenu, car les recommandations sont basées sur des clusters d'utilisateurs, et le clustering est effectuée selon les différents contenus des documents consultés. Le système est autonome, et identifie les intérêts communs des utilisateurs. Un profil utilisateur est composé d'un ensemble de catégories préférées de documents, et d'un ensemble de paires de catégories. Une paire de catégories traduit le fait qu'elles ont été fréquemment consultées ensemble. Le profil d'un utilisateur est identifié en analysant son historique de consultations des documents, et en utilisant des techniques de découverte de règles d'association (cf. section 3.1). En effet, une paire de catégories comme nous l'avons évoqué précédemment, correspond à une association fréquente de longueur 2. Pour regrouper les utilisateurs, le système utilise des types de profils pré-définis. Il y a 6 types de profils. Par exemple, le profil de type 2 correspond à des utilisateurs qui n'ont pas de catégories préférées, mais qui consultent souvent des articles de deux catégories données. Le profil de type 3 correspond aux utilisateurs qui sont intéressés par telles catégories, et ne consultent pas d'articles d'autres catégories. Les utilisateurs sont classés en deux étapes dans des clusters. Premièrement, ils sont répartis dans les 6 types de profils pré-définis. Remarquons que la méthode utilisée pour répartir les clusters dans ces types de profils, n'est pas clairement expliquée. Ensuite, dans chacun des 6 clusters, les utilisateurs les plus similaires sont regroupés. La similitude est basée sur le calcul de la distance euclidienne entre les profils utilisateurs. Un profil utilisateur correspond en pratique à une matrice carrée binaire des catégories de documents.

Les recommandations pour un utilisateur s'effectuent sur les documents que les autres utilisateurs du groupe, dont il fait partie, ont déjà consulté et ces documents correspondent à ses intérêts. Prenons l'exemple de la figure 2.5, où les trois utilisateurs font partie d'un même cluster. Pour chaque utilisateur, nous avons le profil et les documents consultés par catégorie. Par exemple, les catégories préférées de l'utilisateur 2 sont `b` et `c`. Ses associations sont `(b,c)` et `(c,d)`. Il a consulté le document `p8` de la catégorie `b`, `p3` de la catégorie `c`, et `p4` de la catégorie `d`. Pour générer les recommandations de l'utilisateur 1, *OTS* identifie les chevauchements d'intérêt : `(b,c)` avec l'utilisateur 2, et `(c,e)` avec l'utilisateur 3. Les documents recommandés à l'utilisateur 1 sont alors : `p8` (venant de l'utilisateur 2), `p9` et `p11` (venant de l'utilisateur 3).

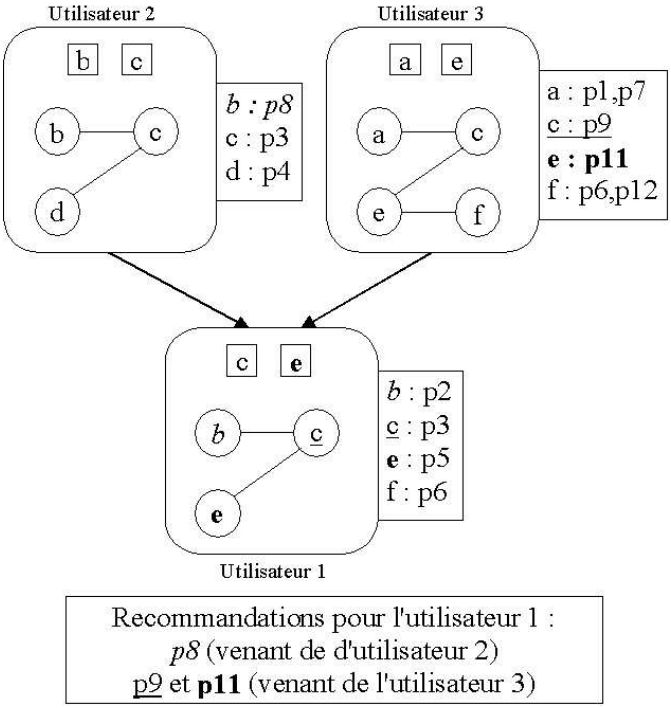

FIG. 2.5 – Exemple de recommandations avec *OTS*

Le système est bien hybride car il s'appuit sur des clusters d'utilisateurs (aspect collaboratif) et sur les catégories de documents (le contenu) pour effectuer des recommandations. Remarquons que les données manipulées sont qualitatives. Chaque utilisateur est décrit par les catégories des documents qu'il a consulté. Le cadre d'expérimentation est limité aux articles scientifiques du domaine informatique, ce qui permet de limiter le nombre de catégories.

Pour une utilisation plus générale comme CYRANO, le nombre de catégories est difficilement calculable et peut être très élevé. L'utilisation d'une matrice pour représenter le profil d'un utilisateur serait alors très gourmand en mémoire et en performance, d'autant plus que de nombreux utilisateurs sont possibles, ce qui conduit à gérer autant de matrices.

Néanmoins, notons que *OTS* propose des idées intéressantes comme le regroupement d'utilisateurs selon leurs intérêts communs détectés à un niveau supérieur (les catégories). De plus, le système utilise des associations fréquentes pour formuler les profils utilisateurs. Cela semble être une voie très intéressante comme nous l'avions déjà notée pour les approches autonomes collaboratives.

2.4 Bilan

L'état de l'art que nous avons effectué sur les systèmes de recommandation, nous a permis de mettre en évidence nos besoins en terme de méthode de recommandation, étant donné que nous n'avons pas identifié de méthodes utilisables dans le cadre de CYRANO.

Cela nous a aussi amené à exprimer nos besoins en terme de méthode de regroupement d'utilisateurs.

Parmi les différents systèmes que nous avons évoqués, les systèmes hybrides de type *collaboration via le contenu* [Paz99] sont plus performants que les systèmes basés sur le contenu ou collaboratifs. Ils utilisent mieux l'information et fournissent des recommandations plus précises. Il est donc intéressant de mettre en place une approche hybride. Néanmoins, les systèmes de collaboration via le contenu, effectuent leurs recommandations uniquement sur les documents déjà consultés par au moins un utilisateur. Dans un système comme CYRANO où il y a une arrivée et un renouvellement de documents, il serait aussi intéressant d'être en mesure de filtrer les documents d'un flux, correspondant aux nouveaux documents stockés dans l'espace disque de CYRANO. De plus, il est important d'avoir une approche autonome, ne nécessitant pas l'intervention de l'utilisateur.

Concernant le type de données mises en jeu, nous avons pu remarquer qu'il y a deux points de vue : les méthodes travaillant avec des données quantitatives (des commentaires ou notes sur les documents) comme c'est le cas pour *Syskill&Webert* [PMB96] et *RecTree* [CHW01], ou des données qualitatives (directement les documents consultés ou les catégories) comme dans le système de Mobasher et al. [MCS99], l'approche de Wilson et al., et *OTS* [WCC01]. D'une façon générale, les systèmes autonomes tendent à utiliser des données qualitatives.

L'étude de ces méthodes permet de faire ressortir une idée intéressante, qui est de transformer les données en une structure directement utilisable pour générer des recommandations, comme par exemple un arbre dans *RecTree*, un ensemble d'associations d'items dans le système de Wilson et al., ou un ensemble de clusters pour Mobasher et al. et *OTS*. L'idée commune de ces systèmes est le regroupement : d'items pour Wilson et al., de documents pour Mobasher et al., et d'utilisateurs pour *RecTree* et *OTS*.

Le système *OTS* semble très intéressant. L'idée générale est la consultation d'un ensemble de documents par des utilisateurs et l'archivage de ces consultations. L'historique des consultations est alors utilisé pour extraire les profils des utilisateurs et ensuite procéder aux recommandations. Pour déterminer le profil des utilisateurs, ce système utilise les catégories des documents consultés par les utilisateurs. Par analogie, dans CYRANO, nous pourrions utiliser les termes caractérisant les documents stockés. *OTS* utilise aussi les associations fréquentes d'items, ici des catégories, pour calculer les profils. Ces associations ne sont que de longueur 2, ce qui est regrettable car le système peut passer à côté d'informations supplémentaires. Un utilisateur peut très bien consulter fréquemment des documents inter-disciplinaires correspondant à trois catégories. Dans le cas de CYRANO, il serait très limitatif de se baser sur des associations de deux termes de documents. Néanmoins, nous pouvons en tirer un point très intéressant, qui est de regrouper des utilisateurs selon leurs intérêts communs identifiés par des associations fréquentes d'items. Nous présenterons la notion d'association d'items dans la section 3.1.

Nous avons aussi remarqué un point très important soulevé par le système *Amalthaea* : la capacité de capter les différents centres d'intérêt de l'utilisateur. *Amalthaea* construit des clusters correspondant aux différents centres d'intérêt d'un utilisateur. Cela est réalisé par un agent. Il faut donc un agent par utilisateur, ce qui n'est pas envisageable pour un système comme CYRANO, qui doit gérer un nombre important d'utilisateurs. Il paraît

néanmoins important que la méthode de regroupement d'utilisateurs que nous devrons utiliser, soit capable de capter les différents centres d'intérêt des utilisateurs.

Pour réaliser cela, nous pensons qu'il est nécessaire d'autoriser le chevauchement d'utilisateurs entre les clusters. Si on considère qu'un cluster correspond à un ou plusieurs centres d'intérêt, il n'y a aucune raison pour que cet utilisateur n'en ait pas d'autres. Par exemple, un utilisateur peut être présent dans un cluster correspondant à la `chasse` et la `pêche`, et aussi être présent dans un cluster correspondant à la `programmation informatique`. Il est donc nécessaire de rechercher des clusters "chevauchants".

En résumé, l'étude de plusieurs systèmes de recommandation nous montre le besoin d'avoir dans le cadre de CYRANO, un système *hybride* basé sur une méthode de *clustering* d'utilisateurs. Pour cela, la technique de clustering doit être capable de gérer un volume important de données, ces dernières étant *qualitatives*. Chaque utilisateur est décrit par les termes caractérisant les documents qu'il a consultés. La méthode de clustering doit aussi fournir une *explication* sur les clusters formés de façon à faciliter le processus de recommandation. Elle doit aussi capturer les différents intérêts communs des utilisateurs, en autorisant le *chevauchement* d'utilisateurs entre les clusters formés. Nous recherchons alors à former des communautés d'utilisateurs ayant des intérêts communs (e.g. communautés d'intérêt).

Dans le chapitre 3, nous étudions les méthodes de clustering. Nous insisterons sur celles correspondantes à nos besoins, et utilisant les notions d'association et de fréquence. En effet, ces notions nous ont paru très intéressantes lors de notre état de l'art des méthodes de recommandation. Dans le chapitre 4, nous proposons une nouvelle méthode de découverte de clusters, apellée ECCLAT. Dans le chapitre 5, nous présentons notre système de recommandation.

Chapitre 3

La classification non-supervisée

Sommaire

Dans le chapitre précédent, nous avons formulé des besoins en terme de méthode de regroupement d'utilisateurs de façon à former des communautés d'intérêt. Dans ce chapitre, nous réalisons un tour d'horizon des méthodes existantes de regroupement d'éléments, aussi appelées classification non-supervisée ou clustering, dans le but d'identifier des méthodes compatibles avec nos besoins.

Le clustering est une méthode non-supervisée, car elle ne nécessite pas d'avoir identifié au préalable des classes correspondant aux différents clusters. Ce point est important car nous n'avons pas de connaissances sur les futures communautés d'utilisateurs que nous voulons former. L'objectif du clustering est de former des clusters tels que chaque cluster soit homogène (les éléments de chaque cluster doivent être le plus similaires possible), et les clusters doivent être hétérogènes entre eux (deux clusters doivent être le plus dissimilaires possible). La plupart des méthodes utilisent une mesure pour calculer la qualité d'un clustering en se basant sur une distance intra-cluster et une distance inter-clusters. Les

algorithmes cherchent alors à obtenir un clustering de qualité optimale, correspondant à une distance intra-cluster faible et une distance inter-clusters élevée.

Rappelons que nous voulons traiter des quantités importantes de données qualitatives, obtenir des explications sur les clusters obtenus, et autoriser un chevauchement possible d'éléments entre les clusters. En ce qui nous concerne, chaque cluster correspond à une communauté d'utilisateurs ayant des intérêts communs. De plus, le nombre de communautés n'étant pas connu à l'avance, le nombre de clusters ne peut pas être a priori fixé.

Traditionnellement, les techniques de clustering sont divisées en trois familles : les méthodes de partitionnement, les approches hiérarchiques, et la classification conceptuelle. Une voie plus récente provenant de la fouille de données propose des méthodes de partitionnement et hiérarchiques à partir de la notion de motif qui est une association de données qualitatives. Cette caractéristique est particulièrement intéressante pour nos applications. Nous avons alors isolé ces méthodes par rapport aux autres que nous qualifions de "classiques". Notons que nous trouvons parmi toutes ces méthodes, des approches de bi-partitionnement effectuant un regroupement à la fois des éléments et des attributs. Un des avantages de ces méthodes est de fournir une description des clusters obtenus.

Nous commençons ce chapitre par la présentation de la notion de motif car elle nous sera utile pour présenter plusieurs méthodes. Nous donnons un exemple simple de jeu de données que nous utiliserons tout au long de ce mémoire. Ensuite, nous réalisons un tour d'horizon des méthodes classiques de clustering, puis nous présentons plus en détails des techniques basées sur les motifs et la notion de fréquence. Enfin, nous abordons les méthodes utilisées en classification conceptuelle qui, comme nous le verrons, présentent des idées intéressantes comme la découverte simultanée des éléments et de la description de chaque cluster. Notons que plusieurs méthodes présentées seront utilisées par la suite lors d'études comparatives (cf. section 4.3).

3.1 Préliminaires : motifs

Nous introduisons les notions de motif et de fréquence provenant du domaine de la fouille de données[9] car elles nous seront utiles pour présenter plusieurs méthodes de clustering, puis ensuite ECCLAT, la méthode de découverte de clusters que nous proposons (cf. chapitre 4).

Les données sont contenues dans une base de données transactionnelle. Cette dernière contient un ensemble d'enregistrements appelés **transactions**. Chaque transaction est décrite par des propriétés appelées **items**. Nous notons $t_1, t_2, \ldots$, les transactions, et $A, B, \ldots$, les items. Le tableau 3.1 présente un exemple de base de données transactionnelle contenant 8 transactions décrites par 9 items. Cette base peut correspondre par exemple, dans le cas d'un magasin, à des clients décrits par les produits qu'ils ont achetés, ou dans le cas de données issues du Web, à des utilisateurs décrits par les mots-clés des documents qu'ils ont consultés.

Nous donnons à présent un ensemble de définitions précisant les données et les notions utilisées.

[9] En anglais : Data Mining

Id.	Items
t_1	$A\ B\ C$
t_2	$A\ B\ C$
t_3	$A\ B\ C$
t_4	$D\ E$
t_5	$D\ E\ H$
t_6	$A\ D\ E\ F\ G\ H$
t_7	$A\ F\ G\ I$
t_8	$H\ I$

TAB. 3.1 – Un exemple de base de données transactionnelle

Définition 1 (Contexte de fouille de données) *Soient $\mathcal{B} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ un contexte de fouille de données, $\mathcal{T}$ un ensemble de transactions, $\mathcal{I}$ un ensemble d'items, et $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ une relation binaire entre les transactions et les items. Chaque couple $(t, i) \in \mathcal{R}$ montre le fait que la transaction t est associée à l'item i.*

Définition 2 (Motif) *X est un motif si $X \subseteq \mathcal{I}$.*

Nous utilisons une notation simplifiée pour les motifs. Par exemple, si $\{A, B\}$ est un motif, nous le notons AB.

Définition 3 (Transaction supportant un motif) *Soient t une transaction et X un motif, t supporte X si et seulement si $\forall i \in X \ \ (t, i) \in \mathcal{R}$.*

Définition 4 (Fréquence d'un motif) *Soit X un motif, sa fréquence notée $\mathcal{F}(X)$, correspond au nombre de transactions de $\mathcal{T}$ supportant X.*

$$\mathcal{F}(X) = |\{t \in \mathcal{T} \mid \forall i \in X \ \ (t, i) \in \mathcal{R}\}|$$

Définition 5 (Motif fréquent) *Soient X un motif et $minfr$ un seuil minimal de fréquence, X est fréquent si $\mathcal{F}(X) \geq minfr$.*

Sur notre exemple, le motif ABC est supporté par les transactions t_1, t_2 et t_3. $\mathcal{F}(ABC)$ est donc égale à 3. Nous pouvons aussi exprimer la fréquence de manière relative, nous obtenons alors 37,5% ($100 \times 3/8$). Si nous fixons $minfr$ à 2, ABC est un motif fréquent.

Il existe plusieurs stratégies de découverte de motifs fréquents : les algorithmes par niveaux (ou "en largeur") comme *APRIORI* [AS94, AMS+96], et les algorithmes en profondeur comme *MAX-MINER* [Bay98]. Nous ne détaillons pas ces algorithmes dans ce mémoire, néanmoins le lecteur intéressé peut se reporter, entre autres, à la thèse de doctorat de Nicolas Pasquier [Pas00].

Beaucoup de travaux portent sur la réduction de l'espace de recherche de façon à améliorer les temps de calcul. Parmi ces travaux, nous trouvons ceux utilisant des représentations condensées de motifs. L'objectif est d'obtenir un ensemble plus restreint de motifs tout en conservant les informations nécessaires pour en déduire tous les motifs

fréquents. Les motifs fermés [PBTL99] (cf. chapitre 4), et les motifs libres [BJ01] sont des représentations condensées de motifs qualifiées d'exactes car elles permettent de déduire avec exactitude toutes les fréquences des motifs fréquents. Récemment, est apparue la notion de motifs δ-libres [BBR03] introduisant une approximation sur les fréquences des motifs.

L'utilisation la plus connue des motifs fréquents est la découverte de règles d'association [AIS93]. Les algorithmes génèrent les règles d'association ayant au moins une fréquence minimale et une confiance miminale (notée *minconf*). La fréquence d'une règle d'association correspond à la fréquence du motif utilisé pour la générer. Dans notre exemple, nous avons la règle $AB \rightarrow C$, basée sur le motif ABC. La fréquence de $AB \rightarrow C$ correspond à la fréquence de ABC qui est égale à 3. La confiance de cette règle est égale à 100% car elle est vérifiée dans tous les cas : à chaque fois que A et B sont présents dans une transaction, C l'est aussi.

Les travaux autour des règles d'associations sont très nombreux. Ils portent notamment sur la définition de critères de qualité des motifs [TK00], et la recherche de règles d'association intéressantes [BA99], dans le but d'améliorer la présentation des résultats à l'utilisateur. Les règles d'association sont aussi utilisées pour le traitement des valeurs manquantes dans les bases de données [RC98, RC99].

De récents travaux utilisent les notions de motifs et de fréquence pour effectuer d'autres tâches classiques de fouille de données, comme le clustering. Nous allons présenter ces travaux dans la section 3.3. Avant cela, nous abordons les méthodes dites "classiques" qui n'utilisent pas ces notions, mais qui permettent d'introduire les idées mises en œuvre, et ainsi d'avoir une connaissance suffisante sur les diverses techniques utilisées.

3.2 Les approches classiques

Les techniques de clustering sont constituées de méthodes de partitionnement répartissant les éléments dans des clusters, et de méthodes hiérarchiques générant plusieurs partitions d'éléments. Nous présentons plusieurs de ces algorithmes pour illustrer les idées et les techniques mises en œuvre.

3.2.1 Les méthodes de partitionnement

Le principe des méthodes de partitionnement est de rechercher une partition des éléments en k clusters de façon à optimiser un critère donné (souvent nommé "fonction objectif"). Certaines méthodes font exception et proposent des idées intéressantes comme par exemple la description des éléments des clusters sous forme de probabilité (*Fuzzy c-Medoids*), ou la recherche simultanée des éléments et des descriptions des clusters (*Bi-Clust*).

Les algorithmes de partitionnement sont de divers types. Certains sont basés sur la notion de centroïde, ou la notion de médoïde. Nous trouvons aussi des approches probabilistes, des méthodes basées sur la densité ou utilisant des grilles, des méthodes basées sur des modèles, et enfin des algorithmes de bi-partitionnement.

Centroïdes

L'algorithme de partitionnement le plus connu s'appelle *K-Means* [Mac67]. Cet algorithme utilise des données quantitatives, et est fondé sur la minimisation de la dissimilarité intra-cluster. Chaque cluster y est représenté par son centre, appelé centroïde. Il partitionne les éléments en k sous-ensembles non vides. L'algorithme choisit au hasard k clusters, puis calcule les centroïdes de la partition courante, et assigne chaque élément avec le centroïde le plus proche. L'algorithme réitére ces opérations tant qu'il y a des affectations possibles. *K-Means* est relativement efficace en temps d'exécution, car sa complexité est linéaire. Par contre, il est nécessaire de définir une distance entre les éléments, et de spécifier le nombre de clusters résultats. De plus, la sélection au hasard des centroïdes initiaux, signifie que deux exécutions de l'algorithme ne donnent pas les mêmes partitions. La version utilisant des données qualitatives s'appelle *K-Modes* [Hua99].

Médoïdes

Au lieu de représenter un cluster par son centre, certaines méthodes utilisent un des éléments du cluster comme élément représentatif qui est appelé médoïde. Parmi ce type de méthodes, nous pouvons citer celle des *nuées dynamiques* [Did71], *PAM* (Partition Around Medoids) [KR90], *CLARA* (Clustering LARge Application) [KR90] qui utilise *PAM* sur de multiples échantillons de données, et *CLARANS* (Clustering Larger Applications based on RANdomized Search) [NH94] qui utilise un graphe où les sommets représentent des solutions potentielles. Ces méthodes utilisant des données quantitatives, nous ne les détaillons pas dans ce mémoire.

Néanmoins, nous trouvons des algorithmes basés sur les médoïdes qui traitent les données qualitatives, comme par exemple, celui proposé par Krishnapuram et al. [Kri01]. Cet algorithme, appelé *Fuzzy c-Medoids*, permet de gérer l'appartenance multiple d'éléments à plusieurs clusters par une approche probabiliste. Ainsi, parmi les résultats nous pouvons trouver qu'un élément appartient à 65% à tel cluster, et 35% à tel autre cluster. De ce fait, *Fuzzy c-Medoids* n'est pas une "réelle" méthode de partitionnement, mais les techniques mises en œuvre sont identiques. La fonction objectif est basée sur la sélection de médoïdes de façon à ce que la dissimilarité intra-cluster soit minimisée. Cet algorithme est très intéressant du fait du chevauchement d'éléments entre les clusters. Mais le nombre de clusters doit être fixé, et il n'y a pas de descriptions claires des clusters obtenus.

Pôles

La notion de pôle a été introduite par l'algorithme *PoBOC* (Pole-Based Overlapping Clusters) [CMV04]. Cette notion est assez proche de celle de centroïde et de médoïde. Cela correspond à un ensemble d'éléments qui est utilisé comme base pour former les clusters finaux. La recherche des pôles s'effectue dans un graphe de similarité que l'algorithme construit à partir d'une matrice donnée en paramètre. Cette matrice indique les similarités entre les éléments à classer. Les sommets du graphe correspondent aux éléments. Une arête est présente entre deux sommets si la similarité entre ces deux sommets est supérieure ou égale au maximum entre la similarité moyenne du premier sommet avec les autres, et la similarité moyenne du deuxième avec les autres. Un pôle est un ensemble de sommets

tel que le sous-graphe formé soit complet. La recherche d'une clique de taille maximale étant un problème NP-complet, l'algorithme utilise l'heuristique d'approximation "Best in" [BBPP99]. Une fois les pôles identifiés, il construit une matrice d'appartenance de chaque élément à chacun des pôles. L'appartenance d'un élément à un pôle est donné par la moyenne des similarités de cet élément avec chacun des éléments du pôle considéré. Enfin, chaque élément est affecté au pôle le plus proche. L'affectation à un autre pôle est fonction des valeurs d'appartenance aux pôles immédiatement plus proche et immédiatement plus éloigné. Si la moyenne de ces deux valeurs est inférieure ou égale à la valeur d'appartenance au pôle considéré, alors l'élément est affecté à ce pôle. La dernière étape de l'algorithme organise les clusters obtenus en une hiérarchie, ce qui facilite la visualisation des résultats.

Il est intéressant de noter que cet algorithme permet d'obtenir des clusters non-disjoints où un élément peut appartenir à plusieurs clusters. De plus, le nombre de clusters n'est pas fixé. Il est déterminé automatiquement lors de la recherche des pôles. En cela, cette méthode est intéressante. Néanmoins, les clusters ne sont pas décrits clairement. De plus, les données utilisées sont quantitatives ou qualitatives, sous réserve d'utiliser ou de définir une mesure de similarité adéquate, ce qui est un point délicat.

Densité

Les méthodes basées sur la densité découvrent des éléments denses séparés par des régions de faibles densités considérées comme du bruit. Elles sont spécialisées pour le clustering de données quantitatives spatiales 2D et 3D, au delà, la notion de densité n'est plus utilisable. Parmi ces méthodes nous pouvons citer *DBSCAN* (Density Based Spatial Clustering of Applications with Noise) [EKSX96] et *OPTICS* (Ordering Points to Identifying the Clustering Structure).

Grilles

Les algorithmes de clustering basés sur les grilles, comme par exemple *STING* (STatistical INformation Grid approach) [WYM97] ou *WaveCluster* (multi-resolution clustering approach using wavelet method) [SCZ98], utilisent des grilles pour découper l'espace de données. Elles sont assez proches des méthodes basées sur la densité, avec parfois les mêmes limitations. Par exemple, *STING* est adaptée aux données quantitatives spatiales, et la densité est analysée dans chaque case de la grille.

Modèles

Le plus populaire des algorithmes de clustering provenant des statistiques est sans doute *EM* (Expectation-Maximization) [DLR77]. Partant d'un modèle initial de clustering pour les données, il affine itérativement le modèle pour tendre vers le meilleur en maximisant la vraisemblance. Ceci est équivalent à minimiser la distance intra-cluster. L'algorithme suppose que l'ensemble des données est stockable en mémoire centrale, ce qui n'est pas envisageable en pratique sur de grands jeux de données.

Parmi les méthodes basées sur des modèles, nous pouvons aussi citer *AutoClass* [CS95] qui se base sur les statistiques bayésiennes pour calculer la probabilité que les éléments

appartiennent à chaque cluster. L'algorithme fonctionne avec des données quantitatives ou qualitatives. Le but de la méthode est de trouver la classification qui est la plus probable à avoir engendré les éléments. On obtient alors du chevauchement d'éléments entre les clusters, ce qui est intéressant de notre point de vue. Néanmoins, cet algorithme suppose l'indépendance des attributs, ce qui peut être génant quand on traite des documents où les attributs sont les mots. Rappelons que nous voulons regrouper des utilisateurs en se basant sur le contenu des documents qu'ils ont consultés et donc des mots. De plus, *AutoClass* fonctionne d'une façon moins optimisée sur des données clairesemées, et les résultats ne sont pas faciles à utiliser.

Plusieurs algorithmes sont basés sur les cartes de Kohonen. *SOM* (Self-Organizing Maps) [Koh95], souvent appelés "réseaux de Kohonen", est basé sur un réseau de neurones à une couche. Les données sont représentées par un nombre réduit de neurones. Les neurones sont disposés sur une carte. Chaque élément à classer est associé à un neurone de façon à ce que les éléments les plus similaires soient les plus proches sur cette carte. La carte finale est une projection non linéaire de la densité de données. *SOM* est essentiellement un outil d'analyse exploratoire et de visualisation.

Des méthodes comme *COD* (Clustering with Obstructed Distance) [THH01] sont basées sur la définition de contraintes. *COD* effectue un clustering en utilisant une taxonomie de contraintes incluant des contraintes sur les éléments à classer, et d'autres paramètres comme le nombre de clusters.

Certains algorithmes s'inspirent du comportement d'insectes pour modéliser les données et former des clusters. Ainsi, *AntClust* [LMV02] forme une partition d'éléments en se basant sur le système de reconnaissance des fourmis. En effet, les fourmis d'une colonnie ont une même odeur et sont capables de se reconnaître. L'idée est que chaque élément à classer est une fourmi. Les fourmis vont alors former des nids au fur et à mesure qu'elles vont se reconnaître. Les colonnies reconstituées forment alors les clusters. Le point important est la construction et l'apprentissage d'une odeur propre à chaque nid, ainsi que la détection des intrus. *AntClust* utilise pour cela une mesure de similarité entre deux éléments. Les données utilisées peuvent être qualitatives ou quantitatives. La mesure de similarité utilisée correspond à la moyenne des similarités des attributs de même type. Si le type est qualitatif, la similarité vaut 1 si les attributs sont identiques, 0 sinon. Si les attributs sont quantitatifs, la similarité est basée sur le calcul d'une distance. Cette mesure est utilisée pour définir un "template" correspondant à un seuil d'acception d'une fourmi à un nid, et fait parti d'un processus d'apprentissage de l'appartenance d'une fourmi à un nid. *AntClust* simule pendant un nombre fixé d'itérations et pour chaque fourmi, une rencontre avec une autre fourmi choisie aléatoirement. L'issue de ces rencontres est déterminée par un ensemble de règles comportementales qui font évoluer les indications d'appartenance aux nids, et les templates. Parmi les règles comportementales, on trouve par exemple la création d'un nouveau nid, et l'ajout d'une fourmi à un nid existant. Il est intéressant de noter qu'il n'est pas nécessaire de fixer le nombre de clusters. Néanmoins, il faut fixer le nombre d'itérations et définir le seuil initial d'acceptation (template initial) basé sur une mesure de similarité. De façon expérimentale, 100000 itérations semblent être suffisantes pour que l'algorithme converge. Remarquons qu'il n'y a pas de description des clusters obtenus.

Bi-partitionnement

Nous avons jusqu'ici présenté des algorithmes de partitionnement d'éléments, mais il existe aussi des méthodes de bi-partitionnement produisant une partition des éléments et aussi une partition des attributs. Parmi ces algorithmes, nous pouvons citer *Bi-Clust* [RR02] qui traite des données qualitatives. Il produit un couple de partitions constitué d'une partition des transactions et d'une partition des items. Ces deux partitions sont liées par les données originales. L'ajustement d'une partition à l'autre permet de déterminer automatiquement le nombre de classes des partitions. *Bi-Clust* utilise une fonction objectif basée sur la mesure τ_S de Goodman et Kruskal. Cependant, il est intéressant de noter qu'il n'est pas nécessaire de fixer le nombre de clusters, et qu'on obtient une description des clusters sous forme d'une conjonction d'items. En effet, la description d'un cluster de transactions correspond au cluster d'items dans l'autre partition. Du fait de la création simultanée des clusters et des descriptions, nous remarquons une analogie avec les méthodes de classification conceptuelle (cf. section 3.5). Sur notre exemple (cf. tableau 3.1), *Bi-Clust* produit deux clusters pour les transactions, et bien entendu deux clusters pour les items. Le tableau 3.2 illustre les deux clusters obtenus : $\{t_1, t_2, t_3 : A,B,C\}$ et $\{t_4, t_5, t_6, t_7, t_8 : D, E, F, G, H, I\}$.

Id.	Items									
t_1	A	B	C							
t_2	A	B	C							
t_3	A	B	C							
t_4				D	E					
t_5				D	E				H	
t_6	*A*			D	E	F	G	H		
t_7	*A*					F	G			I
t_8									H	I

TAB. 3.2 – Le résultat de *Bi-Clust* sur l'exemple

Le lecteur intéressé par les algorithmes de bi-partitionnement peut se reporter, entre autres, au mémoire de thèse de doctorat de Céline Robardet [Rob02].

3.2.2 Les méthodes hiérarchiques

Les algorithmes hiérarchiques de clustering produisent une hiérarchie où un niveau correspond à une partition des éléments. Il existe deux stratégies de clustering hiérarchiques. Dans les approches "ascendantes" (*CAH* [DE84]), chaque élément correspond à un cluster, formant ainsi la partition initiale. Puis les clusters les plus similaires sont fusionnés deux à deux jusqu'à obtenir un cluster contenant tous les éléments. Les approches "divisives" procédent de manière inverse en scindant en deux les clusters.

La hiérarchie formée se représente sous la forme d'un dendogramme. La figure 3.1 représente un dendogramme sur notre exemple du tableau 3.1.

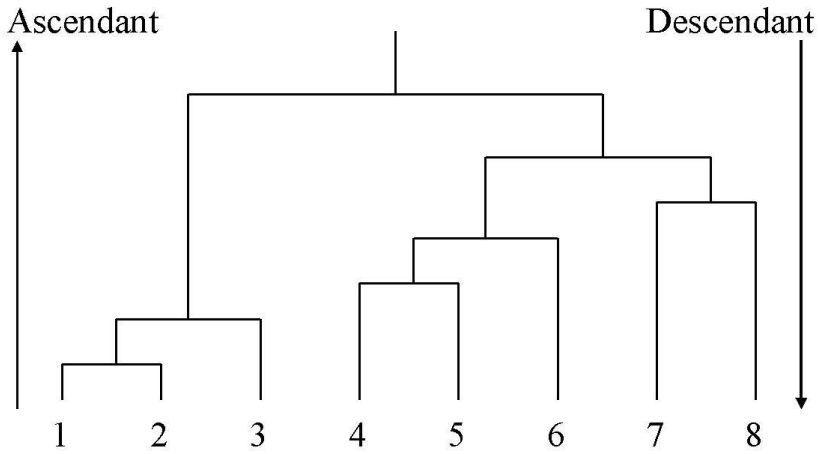

FIG. 3.1 – Exemple de dendogramme

Le clustering résultat correspond à la partition contenant k clusters où k est fixé par l'utilisateur. Une autre possibilité est de sélectionner la partition ayant la meilleure qualité au sens de la distance intra-cluster et inter-clusters. Sur l'exemple de la figure 3.1, avec k est égal à 3, le clustering obtenu est : {1, 2, 3} {4, 5, 6} {7, 8}.

Sur des données quantitatives, nous pouvons citer deux standards : *AGNES* (AGglomerative NESting) et *DIANA* (DIvisive ANAlysis) [KR90]. *AGNES* est une méthode ascendante qui utilise une matrice de dissimilarité. *DIANA* est un algorithme de clustering descendant. C'est en quelque sorte l'inverse d'*AGNES*, néanmoins la technique est assez différente, car pour un cluster contenant n éléménts, il y a $2^{n-1} - 1$ possibilités de le scinder en deux. *DIANA* utilise alors une heuristique pour scinder itérativement le plus gros cluster. Remarquons que cela peut être un inconvénient. Un gros cluster ne veut pas dire qu'il n'est pas homogène.

Plusieurs méthodes sont basées sur l'optimisation d'algorithmes existants de clustering. *CURE* (Clustering Using REpresentatives) [GRS98] est une méthode hiérarchique ascendante. Initialement un échantillon aléatoire est partitionné. Les parties sont ensuite fusionnées en utilisant un algorithme hiérarchique ascendant comme *AGNES*. Ensuite, *CURE* regroupe les clusters partiels. Lorsque deux clusters sont fusionnés, leurs représentants sont tassés vers le centroïde du nouveau cluster créé, selon un facteur de tassement spécifié par l'utilisateur. Au final, les éléments n'étant pas présents dans l'échantillon, sont rangés dans les clusters les plus proches. Notons que *CURE* est très sensible au choix du facteur de tassement et de la taille de l'échantillon aléatoire.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [ZRL96] est une méthode qui propose une structure pour utiliser un algorithme existant de clustering. Il peut être vu comme un algorithme hiérarchique descendant. Il est composé de deux phases, et utilise une structure appelée *CF-Tree* (Clustering Feature Tree). Les deux phases sont la construction du *CF-Tree* en scannant chaque élément des données, et l'application d'un algorithme de clustering pour classer les feuilles de l'arbre. Remarquons que l'ordre des données influence les résultats.

Concernant le traitement de données qualitatives, nous citons *ROCK* (RObust Clustering using linKs) [GRS99] qui utilise une notion appelée "lien" pour mesurer la similarité entre deux transactions. Il n'y a pas de distance définie. La similarité entre deux transactions est évaluée avec le coefficient de Jaccard qui correspond au ratio entre le nombre d'items communs et le nombre d'items distincts entre les deux transactions. Nous utiliserons ce coefficient lors d'un exemple dans le chapitre 5. Soient deux transactions t_1 et t_2, alors la similarité, notée $Sim(t_1, t_2)$, est égale à

$$\frac{|t_1 \cap t_2|}{|t_1 \cup t_2|}$$

Si la similarité entre deux transactions est supérieure ou égale à une seuil fixé par l'utilisateur, alors les transactions sont dites voisines. Le nombre de liens entre deux transactions correspond alors au nombre de voisins communs. L'idée principale est que plus le nombre de liens entre deux transactions est élevé, plus il est probable que ces deux transactions soient dans le même cluster. Si nous reprenons notre exemple, nous avons les similarités suivantes :

$$Sim(t_1, t_2) = Sim(t_1, t_3) = \frac{|ABC|}{|ABC|} = 1$$

$$Sim(t_1, t_4) = Sim(t_1, t_5) = Sim(t_1, t_8) = 0$$

$$Sim(t_1, t_6) = \frac{|A|}{|ABCDEFGH|} = \frac{1}{8}$$

$$Sim(t_1, t_7) = \frac{1}{6}$$

$$Sim(t_2, t_3) = 1$$

$$Sim(t_2, t_4) = Sim(t_2, t_5) = Sim(t_2, t_8) = 0$$

$$Sim(t_2, t_6) = \frac{1}{8}$$

$$Sim(t_2, t_7) = \frac{1}{6}$$

Si nous fixons un seuil de similarité égal à $1/6$, le nombre de liens pour les transactions t_1 et t_2 est égal à 5. En effet, cela correspond au nombre de paires où interviennent t_1 ou t_2, et donnant une similarité supérieure ou égale à $1/6$. Nous avons 5 liens : $t_1 - t_2$, $t_1 - t_3$, $t_1 - t_7$, $t_2 - t_3$, $t_2 - t_7$.

Initialement, *ROCK* sélectionne un échantillon de transactions et effectue un clustering hiérarchique ascendant en utilisant une mesure de similarité basée sur les liens. Tous les éléments qui ne sont pas dans l'échantillon sont ensuite rangés dans les clusters les plus proches. Ce point est discutable, car les résultats dépendent donc de l'échantillon de départ. De plus, il est difficile de concevoir qu'un simple échantillon puisse être représentatif de l'ensemble des données.

A partir de données qualitatives, *CHAMELEON* [KHK99] construit le graphe des k plus proches voisins où un sommet représente un élément à classer et le poids d'une arête représente la similarité entre les deux sommets. Une arête est présente entre deux sommets x et y, si y fait parti des k éléments les plus similaires à x, et inversement. Ensuite, le graphe est partitionné avec l'algorithme *hMETIS* [KK99], en supprimant les arêtes ayant les plus faibles poids. Nous reparlerons de cet algorithme lors de la présentation de l'algorithme *ARHP* (Association Rules Hypergraph Partitionning) (cf. section 3.3.2). Puis, dans une dernière phase, il utilise un algorithme hiérarchique ascendant pour fusionner les parties précédemment obtenues. Les principaux inconvénients de *CHAMELEON* est le nombre de paramètres et la difficulté à les fixer, comme par exemple la condition d'arrêt du partitionnement du graphe.

3.2.3 Bilan

Dans la quasi-totalité des méthodes que nous venons de présenter, il est nécessaire de fixer le nombre de clusters désirés. Les rares méthodes ne nécessitant pas un tel paramètre, utilisent un critère pour arrêter le processus de clustering. Les méthodes qu'elles soient de partitionnement ou hiérarchiques, sont basées sur une mesure de similarité entre les éléments à classer pour évaluer la distance intra-cluster et inter-clusters. Ce point est délicat et pose problème pour les données qualitatives malgré l'existence de mesures de similarité comme le coefficient de Jaccard utilisé par *ROCK*, ou le Matching, ou Dice, ou Cosine [FB92]. Ces méthodes restent dépendantes de la mesure mise en œuvre.

En ce qui concerne les contraintes que nous nous sommes fixées, remarquons que les méthodes telles que *Fuzzy c-Medoids* et *AutoClass* autorisent l'appartenance multiple d'éléments à plusieurs clusters. Mais le nombre de clusters doit être fixé. Seul *PoBOC* formant aussi des clusters non-disjoints, ne nécessite pas de connaitre le nombre de clusters. Cependant, ces méthodes ne permettent pas d'obtenir une description claire des clusters. L'algorithme *Bi-Clust* recherche simultanément les éléments des clusters et de la description des clusters dans des données qualitatives. Cette recherche simultanée permet de déterminer le nombre de clusters résultats. Mais on obtient une partition stricte des éléments, il n'y a pas de chevauchement possible.

Nous présentons maintenant des méthodes basées sur les motifs et la notion de fréquence. Ces méthodes utilisent des techniques que venons d'évoquer dans cette section, et offrent d'autres possibilités pour évaluer la similarité intra-cluster.

3.3 Les méthodes basées sur les motifs et la notion de fréquence

Le foisonnement de méthodes de clustering adaptées aux données qualitatives a fait émerger des travaux utilisant des motifs et/ou la notion de fréquence. Ils tentent de parer au problème de la définition de mesure de similarité entre transactions en utilisant ces deux notions.

Nous identifions trois types de méthodes : celles regroupant les transactions, celles regroupant les items, et des méthodes hybrides. Elles correspondent aussi bien à des

méthodes de partitionnement, à des méthodes hiérarchiques, ou des méthodes basées sur la densité et les grilles (cf. section 3.2.1). Certaines définissent une mesure de similarité en se basant sur la fréquence, d'autres ne se préoccupent pas d'une telle définition et contournent ce point en utilisant directement les motifs fréquents. Remarquons qu'il faut néanmoins fixer un seuil de fréquence.

3.3.1 Regroupement de transactions

TrK-Means

TrK-Means (Transactional K-Means) [GGM02] est un algorithme de partitionnement pour les données transactionnelles, basé sur *K-Means*. Il utilise le coefficient de Jaccard pour évaluer la distance entre deux transactions. Pour déterminer le centroïde d'un cluster, *TrK-Means* utilise un seuil de fréquence d'items. Le centroïde d'un cluster contient des items fréquents. *TrK-Means* permet d'avoir des temps d'exécution rapides grâce à la complexité linéraire de *K-Means*. Les centroïdes des clusters obtenus peuvent être utilisés pour décrire les résultats. Mais cette description n'est pas exacte, dans le sens où les items du centroïde d'un cluster ne sont pas exactement partagés par tous les éléments du cluster. De plus, il faut fixer le nombre de clusters souhaités, et on obtient une partition stricte des transactions.

Méthode de Wang et al.

Wang et al. [WCL99] proposent un algorithme hiérarchique descendant de clustering de transactions. Cet algorithme utilise la notion de fréquence pour définir des mesures intra-cluster et inter-clusters. Le coût intra-cluster pour un clustering est défini comme le nombre total d'items non fréquents pour chaque cluster. Par exemple, pour un seuil de fréquence égal à 50%, le coût intra-cluster d'un cluster contenant 4 transactions correspond au nombre d'items présents dans moins de 2 de ces transactions. Le coût inter-clusters mesure la duplication d'items fréquents parmi les clusters. Ainsi, le but est de trouver un clustering de transactions ayant un coût global minimum. Dans une phase préliminaire, l'algorithme lit chaque transaction dans les données et l'ajoute dans un cluster de façon à minimiser le coût global. Ensuite, tant qu'il y a des changements possibles, il déplace une transaction d'un cluster dans un autre de manière à minimiser le coût.

Cluster C_1=$\{t_1, t_2, t_3\}$	
$minfr$	1
Items fréquents	A, B, C
Items non fréquents	aucun

Cluster C_2=$\{t_4, t_5, t_6, t_7, t_8\}$	
$minfr$	2
Items fréquents	A, D, E, F, G, H, I
Items non fréquents	aucun

TAB. 3.3 – Clustering obtenu sur l'exemple avec la méthode de Wang et al.

Si nous reprenons notre exemple (cf. tableau 3.1) avec $minfr$ égal à 40%, considérons la partition composée d'un seul cluster $C_1 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$. Son coût intra-cluster est égal à 3 car les items F, G et H ne sont pas fréquents. Le coût inter-clusters est nul.

Par contre si nous avons la partition suivante : $C_1 = \{t_1, t_2, t_3\}$ $C_2 = \{t_4, t_5, t_6, t_7, t_8\}$, le coût intra-cluster est égal à 0 car il n'y a pas d'items non fréquents (cf. tableau 3.3). Le coût inter-clusters est égal à 1 $(10 - 9)$. En effet, il y a 10 items fréquents au total dans les clusters et l'item A est présent deux fois. Le coût de cette partition est donc égal à 1. Sur cet exemple, cette partition obtient le meilleur coût global. Remarquons que dans cette méthode, le nombre de clusters n'a pas besoin d'être fixé. La description des clusters peut être obtenue par la suite en utilisant les items fréquents de chaque cluster. Cette méthode n'utilise pas de distance. Par contre, cette méthode forme une partition stricte des transactions, il n'y a pas de chevauchement possible de transactions.

CLIQUE

CLIQUE (CLustering In QUEst) [AGGR98] est un algorithme hybride à la fois basé sur les grilles et sur la densité. Il propose une méthode de construction de clusters dans l'ensemble des sous-espaces de données. Les données quantitatives sont discrétisées en intervalles de taille égale. Par exemple, un attribut "âge" allant de 0 à 80 ans, peut être discrétisé en quatre items : "âge≤20", "20<âge≤40", "40<âge≤60" et "âge>60". *CLIQUE* est utilisable sur de grandes quantités de données, et propose une description des clusters sous la forme d'une conjonction d'items. L'espace des données transformées est partitionné en unités rectangulaires. Le pourcentage de transactions contenues dans une unité correspond à sa "sélectivité" (analogie avec la fréquence). Une unité est dite dense lorsque sa sélectivité est supérieure ou égale à un seuil fixé. Deux unités sont dites connectées, si elles ont une face commune : si les unités sont de dimension m, alors elles partagent leurs $m - 1$ dimensions. Un cluster est défini comme un ensemble maximum d'unités denses connectées. *CLIQUE* utilise une approche par niveau, comme *APRIORI*. Initialement, les unités denses de dimension 1 sont calculées. Ensuite, les unités denses de dimension 2 sont déterminées en utilisant celles de dimension 1, et ainsi de suite. Une fois que les unités denses sont identifées, l'algorithme regroupe les unités denses dans le même sous-espace, détermine le nombre de transactions pour chaque sous-espace, et supprime les sous-espaces qui sont inférieurs à un seuil de recouvrement. Le recouvrement de sous-espaces correspond à la somme des sélectivités de ces sous-espaces. Ensuite, les clusters sont identifiés en déterminant les composantes connectées. Enfin, l'algorithme détermine la description minimale de chaque cluster sous forme d'une conjonction d'items. Remarquons que certains paramètres sont difficiles à fixer, comme par exemple la taille des intervalles pour la discrétisation. *CLIQUE* voit un cluster comme un ensemble maximum d'unités denses connectées. Il semble qu'un ensemble maximum d'unités denses soit un motif fréquent maximal. Cela montre qu'il serait peut-être utile d'effectuer une recherche des motifs fréquents maximaux pour déterminer des clusters.

3.3.2 Regroupement d'items

CACTUS

CACTUS (CAtegorical ClusTering Using Summaries) [GGR99] est une méthode de clustering basée sur les attributs. Un attribut $\mathcal{A}$ est décrit par un ensemble de valeurs ap-

pelé domaine. Un attribut associé à une de ces valeurs correspond à un item. L'algorithme est composé de trois phases : récapitulation, clustering, et validation. L'idée centrale est qu'un résumé des données est suffisant pour calculer un ensemble de clusters candidats, lesquels sont évalués pour déterminer un ensemble de clusters. La phase de récapitulation consiste à calculer un résumé inter-attributs et un résumé intra-attribut. Le résumé inter-attributs correspond à un ensemble de paires d'attributs fortement connectés. Cela s'apparente à des motifs fréquents de longueur 2. Le résumé intra-attribut d'un attribut $\mathcal{A}$ se calcule par rapport à un autre attribut, par exemple $\mathcal{B}$. Pour une paire de valeurs de $\mathcal{A}$, cela correspond au nombre de valeurs prises par l'attribut $\mathcal{B}$ présentes dans les paires du résumé inter-attributs faisant intervenir les deux valeurs de $\mathcal{A}$. Reprenons notre exemple du tableau 3.1. Supposons que $\{A, D, E\}$ soient des valeurs prises par un attribut $\mathcal{A}$ et $\{F, G, H, I\}$ des valeurs prises par un attribut $\mathcal{B}$. Avec un seuil minimum fixé à 2, le résumé inter-attributs contient alors les motifs fréquents de longueur 2 : DH, AF, AG, EH (cf. figure 3.2). Le résumé intra-attribut de $\mathcal{A}$ par rapport à $\mathcal{B}$ est alors : $(A, D : 3)$ $(A, E : 3)$ $(D, E : 1)$. Pour chaque paire de valeurs de $\mathcal{A}$, on compte le nombre de valeurs de $\mathcal{B}$ associées aux items de la paire. Par exemple, pour la paire $(A,\ D)$, il y a trois valeurs d'attribut de $\mathcal{B}$. Par contre, il n'y a qu'une valeur de $\mathcal{B}$ pour la paire $(D,\ E)$, qui est H.

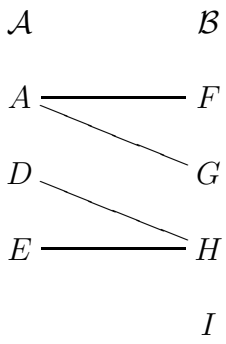

FIG. 3.2 – Exemple de résumé inter-clusters utilisé par *CACTUS*

La phase de clustering s'effectue en deux temps. Premièrement, chaque attribut est analysé pour calculer toutes les projections de cluster. Pour un attribut, sa projection est un sous-ensemble de valeurs de cet attribut, fortement connecté aux valeurs des autres attributs. Dans un second temps, *CACTUS* synthétise les clusters candidats sur les ensembles d'attributs à partir des projections de clusters sur les attributs. Les clusters candidats sont ensuite déterminés par niveau : sur une paire d'attributs, puis étendus à un ensemble de trois attributs, et ainsi de suite. L'étape de validation consiste à parcourir les données pour déterminer le support de chacun des clusters candidats. Seuls les clusters ayant une fréquence supérieure ou égale au seuil minimum sont gardés. Cela permet entre autres d'éliminer les clusters correspondant à un motif n'existant pas dans les données. L'inconvénient de cette méthode est qu'elle suppose que la cardinalité du domaine des attributs est relativement petit. Sinon, les temps de calcul sont très élevés. La description des clusters se présente sous la forme d'une conjonction d'items. Une transaction appartient à un seul cluster car l'ensemble des clusters correspond à des régions disjointes. Notons que sans le dire explicitement, cette méthode recherche des motifs fréquents pour définir des clusters.

Méthode de Ronkainen

Ronkainen [Ron98] utilise la notion de fréquence pour définir de mesure de similarité entre items, et utilise une approche hiérarchique ascendante pour effectuer un clustering des items. La mesure de similarité utilisée s'apparente au coefficient de Jaccard. Soient A et B deux items (ou ensembles d'items), la distance entre A et B est définie de la façon suivante :

$$d(A,B) = 1 - \frac{\mathcal{F}(AB)}{\mathcal{F}(A) + \mathcal{F}(B) - \mathcal{F}(AB)}$$

Rappelons que $\mathcal{F}(A)$ est la fréquence de l'item A dans les données. La matrice de distances entre les items est alors calculée, et est utilisée par un algorithme hiérarchique ascendant. Pour chaque partition, l'algorithme évalue sa qualité basée sur la dissimilarité inter-clusters et similarité intra-cluster. Au final, l'algorithme fournit la partition ayant la qualité la plus élevée. Il n'est donc pas nécessaire de fixer le nombre de clusters. Les clusters correspondent à des ensembles d'items. Il est donc nécessaire par la suite de ranger les transactions dans ces clusters. Ceci est une tâche délicate qui n'est pas évoquée dans les travaux de Ronkainen. Nous reviendrons sur ce point dans le paragraphe "Rangement des transactions". Sur notre exemple, l'algorithme trouve alors 5 clusters d'items : $\{ABC\}$ $\{DE\}$ $\{FG\}$ $\{H\}$ $\{I\}$.

ARHP

ARHP (Association Rules Hypergraph Partitioning) [HKKM97, HKKM98] est basé sur le partitionnement d'un hypergraphe représentant les motifs fréquents d'items présents dans une base de données. Il produit un clustering d'items, en trois phases. La première consiste à rechercher les motifs fréquents avec par exemple l'algorithme *APRIORI*, ce qui permet de capturer implicitement la similarité parmi les items. Remarquons que cet ensemble est généralement très important. Ensuite, l'hypergraphe correspondant est construit. Les sommets de l'hypergraphe correspondent aux items, et les hyper-arêtes forment les motifs fréquents. Chaque arête a un poids qui correspond à la moyenne des confiances des règles essentielles, c'est-à-dire ayant un seul item en conclusion, que l'on peut générer avec les items du motif fréquent. La dernière étape consiste à partitionner l'hypergraphe en k parties (avec *hMETIS* [KK99]), où k est fixé par l'utilisateur. L'idée du partitionnement est la suivante : on coupe l'hypergraphe en deux (ce qui revient à éliminer des motifs fréquents), de telle sorte que l'hyper-arête coupée ait un poids minimum, ensuite on réitère sur les morceaux restants jusqu'à ce qu'on ait k parties. Remarquons que minimiser le poids de l'hyper-arête coupée correspond à minimiser les relations qui ressortent violées en coupant les motifs. Ensuite, l'algorithme élimine les "mauvais" clusters selon un critère d'affinement calculant le rapport entre le poids des arêtes qui sont dans la partition et les poids des arêtes impliquant les sommets dans cette partition. Si un cluster a un affinement inférieur à un seuil fixé, alors il est éliminé. Chaque partition est examinée pour filtrer les sommets qui ne sont pas fortement connectés avec le reste des sommets de la partition. Pour chaque sommet d'une partition, on calcule la connectivité de ce sommet en mesurant le pourcentage d'arêtes associées avec le sommet. Si cette connectivité est inférieure à un seuil fixé, alors le sommet est éliminé.

Si nous reprenons notre exemple, avec $minfr$ égal à 30%, nous avons 12 motifs fréquents de longueur supérieure ou égale à 2. I est le seul motif fréquent de longueur 1. La figure 3.3 représente l'hypergraphe de ces motifs. Le nombre indiqué à côté d'une hyper-arête est son poids (confiance exprimée en pourcentage).

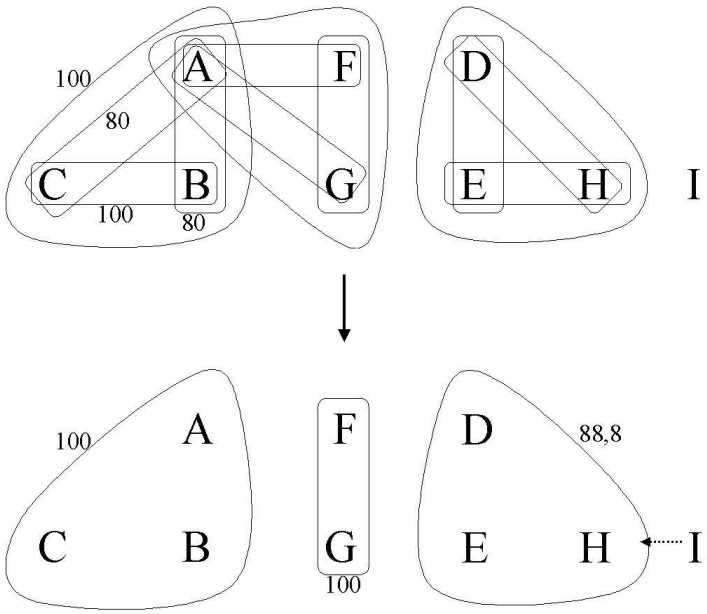

FIG. 3.3 – *ARHP* : Hypergraphes obtenus sur l'exemple

Avec k fixé à 3, nous obtenons les clusters suivants : $\{A, B, C\}$ $\{D, E, H, I\}$ $\{F, G\}$ (cf. figure 3.3). Remarquons que le cluster $\{D, E, H, I\}$ ne représente pas un motif existant dans les données. Nous verrons dans le paragraphe "Rangement des transactions" que ce point est un problème pour classer les transactions si nous voulons obtenir du chevauchement. Par contre, si nous fixons k à 4, nous obtenons les clusters suivants : $\{A, B, C\}$ $\{D, E, H\}$ $\{F, G\}$ $\{I\}$. Nous remarquons que dans ce cas, les clusters d'items correspondent à des motifs, sans que cela soit explicitement demandé. Il n'y a aucun moyen de spécifier à l'algorithme que nous voulons des motifs existants dans les données.

Nous remarquons aussi au niveau du principe de la suppression des hyper-arêtes de plus faible poids, que les motifs fréquents dont la moyenne des confiances des règles essentielles est égale à 100% sont donc très privilégiés. Ils ne seront éliminés qu'en dernier recours. De plus, si l'algorithme doit choisir entre deux hyper-arêtes de même poids, voire aussi de poids maximum, on ne sait pas ce que décide l'algorithme. Il garde peut-être celle contenant le plus d'items.

Mis à part le fait que les paramètres comme le nombre de clusters et le seuil d'affinement soient difficiles à fixer, nous considérons le problème de la création de motifs inexistants comme le principal inconvénient de cette méthode. En effet, cela aura des répercussions sur le rangement des transactions.

Rangement des transactions

Les méthodes de regroupement d'items, comme celle de Ronkainen et *ARHP*, ne rangent pas les transactions dans les clusters d'items obtenus. Si nous voulons utiliser ces clusters, il faut connaître leurs transactions. Nous présentons dans ce paragraphe, deux possibilités de rangement. La première est présentée dans *ARHP* mais elle peut s'appliquer à n'importe quelle méthode. Elle consiste à placer une transaction dans le cluster optimisant le score défini par le pourcentage d'items du cluster communs avec les items de la transaction. Nous appelons cette approche "rangement au mieux". La seconde range une transaction dans un cluster si elle supporte tous les items du cluster. Dans ce cas, nous avons une possible appartenance multiple de la transaction à différents clusters, et donc du chevauchement. Nous notons cette approche "rangement exact". Remarquons que le post-traitement nécessaire pour classer les transactions est coûteux pour une quantité importante de données.

Reprenons les résultats obtenus avec la méthode de Ronkainen. Le rangement au mieux fournit les clusters suivants : $\{A, B, C : t_1, t_2, t_3\}$ $\{D, E : t_4, t_5, t_6\}$ $\{F, G : t_7\}$ $\{H : t_8\}$ $\{I :\}$ (cf. tableau 3.4). Nous remarquons qu'il n'y a pas de transaction dans le cluster correspondant à l'item I. Avec le rangement exact, nous obtenons : $\{A, B, C : t_1, t_2, t_3\}$ $\{D, E : t_4, t_5, t_6\}$ $\{F, G : t_6, t_7\}$ $\{H : t_5, t_6, t_8\}$ $\{I : t_7, t_8\}$. Nous remarquons que dans le cluster $\{F, G\}$, les transactions t_6 et t_7 ont aussi en commun l'item A. Or celui-ci est déjà présent dans un autre cluster, il n'est donc pas répété, ce qui nous paraît être une perte d'information pour la description du cluster $\{F, G\}$.

Cluster d'items	Transactions	
	rangement au mieux	rangement exact
A, B, C	t_1 t_2 t_3	t_1 t_2 t_3
D, E	t_4 t_5 t_6	t_4 t_5 t_6
F, G	t_7	t_6 t_7
H	t_8	t_5 t_6 t_8
I		t_7 t_8

TAB. 3.4 – Clustering obtenu sur l'exemple avec la méthode de Ronkainen

Le rangement au mieux avec les résultats obtenus avec *ARHP* (k=3), produit les clusters suivants : $\{A, B, C : t_1, t_2, t_3\}$ $\{D, E, H, I : t_4, t_5, t_8\}$ $\{F, G : t_6, t_7\}$ (cf. tableau 3.5). Si nous voulons effectuer un rangement exact des transactions, le cluster $\{D, E, H, I\}$ pose un problème, car aucune transaction supporte tous ses items. $DEHI$ n'est pas un motif existant dans les données. Il correspond à deux motifs distincts : DEH et I.

Par contre, si nous fixons k à 4 et utilisons le rangement exact, nous obtenons les clusters suivants : $\{A, B, C : t_1, t_2, t_3\}$ $\{D, E, H : t_5, t_6\}$ $\{F, G : t_6, t_7\}$ $\{I : t_7, t_8\}$ (cf. tableau 3.6). Nous remarquons que dans ce cas, les clusters d'items correspondent à des motifs, sans que cela soit explicitement demandé. Nous avons bien du chevauchement de transactions : t_7 est commune au cluster $\{F, G\}$ et au cluster $\{I\}$. Mais nous observons

Cluster d'items	Transactions	
	rangement au mieux	rangement exact
A, B, C	$t_1 \quad t_2 \quad t_3$	$t_1 \quad t_2 \quad t_3$
D, E, H, I	$t_4 \quad t_5 \quad t_8$	
F, G	$t_6 \quad t_7$	$t_6 \quad t_7$

TAB. 3.5 – Clustering obtenu sur l'exemple avec $ARHP$, k=3

que la transaction t_4, contenant les items D et E, n'est pas classée, car il n'y a pas de cluster qui lui correspond. Autre remarque, les transactions t_6 et t_7 rangées dans le cluster $\{F, G\}$, partagent toutes l'item A. Or comme $ARHP$ forme une partition des items, A n'est pas répété. Cela signifie qu'il n'y a pas de chevauchement entre clusters au niveau des items, et on perd l'information que A est aussi présent pour toutes les transactions du cluster $\{F, G\}$. Cette notion de maximalité des items partagés entre des transactions est un point que nous pensons très important pour le clustering (cf. chapitre 4).

Cluster d'items	Transactions	
	rangement au mieux	rangement exact
A, B, C	$t_1 \quad t_2 \quad t_3$	$t_1 \quad t_2 \quad t_3$
D, E, H	$t_4 \quad t_5$	$t_5 \quad t_6$
F, G	$t_6 \quad t_7$	$t_6 \quad t_7$
I	t_8	$t_7 \quad t_8$

TAB. 3.6 – Clustering obtenu sur l'exemple avec $ARHP$, k=4

3.3.3 Méthodes hybrides

Ping-Pong [OKN01] est un algorithme de clustering de matrices binaires correspondant à des données qualitatives. L'idée est d'envoyer et de propager des marqueurs alternativement entre des transactions et des items, d'où le nom *Ping-Pong*. Les marqueurs reçus au niveau d'un item ou d'une transaction permettent d'en déterminer la fréquence. L'algorithme itère la propagation des marqueurs partant des transactions jusqu'aux items, puis effectue un élagage des items, puis une propagation des marqueurs des items jusqu'aux transactions, et un élagage des transactions, et ainsi de suite jusqu'à convergence. Au final, on obtient un ensemble stabilisé de transactions et d'items. L'élagage est réalisé en comparant le nombre de marqueurs reçus au niveau des items et des transactions avec un seuil minimum. Une exécution de *Ping-Pong* à partir d'une transaction produit une sous-matrice. Une sous-matrice ayant une densité supérieure ou égale au seuil minimum spécifié, correspond à un cluster. Pour obtenir plusieurs clusters, il faut l'exécuter sur les autres transactions. Les clusters n'étant pas disjoints, il est possible d'avoir des chevauchements de transactions et aussi d'items. Néanmoins, le nombre de clusters résultats peut être élevé, le maximum étant le nombre de transactions, ce qui ne facilite pas leur utilisation.

Id.	A	B	C	D	E	F	G	H	I
t_1	1	1	1	0	0	0	0	0	0
t_2	1	1	1	0	0	0	0	0	0
t_3	1	1	1	0	0	0	0	0	0
t_4	0	0	0	1	1	0	0	0	0
t_5	0	0	0	1	1	0	0	1	0
t_6	1	0	0	1	1	1	1	1	0
t_7	1	0	0	0	0	1	1	0	1
t_8	0	0	0	0	0	0	0	1	1

TAB. 3.7 – Matrice binaire correspondant à l'exemple

Reprenons notre exemple, ce dernier peut se présenter d'une façon binaire (cf. tableau 3.7). Fixons le nombre minimum de marqueurs à 2, et la densité minimum à 80%. Partant de la transaction 1, le cluster formé correspond à $\{ABC : t_1, t_2, t_3\}$ (cf. tableau 3.8). Sa densité est égale à 100% car les items A, B et C sont tous vérifiés par les transactions t_1, t_2 et t_3. Nous obtenons le même cluster partant des transactions t_2 et t_3. Pour la transaction t_4 ou t_5, le cluster formé est $\{DEH : t_4, t_5, t_6\}$ ayant une densité égale à 88,8% (8 cases à "1" sur les 9 cases). La transaction t_6 forme le cluster $\{ADEFGH : t_4, t_5, t_6, t_7\}$ qui est en fait éliminé car sa densité vaut 58,3%. Pour la transaction t_7, le cluster produit est $\{AFG : t_6, t_7\}$ de densité 100%. La transaction t_8 ne donne aucun cluster, car le nombre de marqueurs propagés est insuffisant.

Id.	A	B	C
t_1	1	1	1
t_2	1	1	1
t_3	1	1	1

Densité=100%

Id.	D	E	H
t_4	1	1	0
t_5	1	1	1
t_6	1	1	1

Densité=88,8%

Id.	A	D	E	F	G	H
t_4	0	1	1	0	0	0
t_5	0	1	1	0	0	1
t_6	1	1	1	1	1	1
t_7	1	0	0	1	1	0

Densité=58,3% → éliminée

Id.	A	F	G
t_6	1	1	1
t_7	1	1	1

Densité=100%

TAB. 3.8 – Sous-matrices produites par *Ping-Pong* sur l'exemple

Le seuil minimum de marqueurs s'apparente à la notion de fréquence, à la différence qu'il s'applique aussi bien sur les transactions que sur les items. Le résultat de l'algorithme correspond à un ensemble de transactions en relation avec un ensemble d'items en utilisant ce seuil. La différence avec la recherche de motifs fréquents est qu'une transaction d'un cluster résultat peut ne pas vérifier un item partagé par les autres transactions du cluster. Cela correspond alors à une exception qui peut être utilisée par la suite pour effectuer des prédictions. Le seuil de densité permet de limiter ces exceptions. Si ce seuil est égal à 100%, c'est-à-dire que l'on ne tolère aucune exception, on obtient des clusters où tous les items sont vérifiés par les transactions. Remarquons que dans ce cas, le mécanisme d'"aller-retour" entre les transactions et les items se rapproche de la connexion de Galois utilisée en classification conceptuelle (cf. section 3.5). Un cluster (sous-matrice dense) de

densité égale à 100% est un concept du treillis de Galois. Mais *Ping-Pong* ne permet pas de trouver tous ces concepts, comme par exemple, le concept $\{A : t_1, t_2, t_3, t_6, t_7\}$.

3.3.4 Bilan

Nous avons vu que la fréquence des items est utilisée pour déterminer la similarité entre les transactions, comme dans la méthode de Wang et al. L'avantage est qu'il n'y a pas de définition d'une distance entre les éléments. Mais ce type de méthode effectue un clustering strict des transactions. Il n'y a aucun chevauchement possible. Seules les méthodes de regroupement d'items permettent un chevauchement possible de transactions entre les clusters. Néanmoins, ces méthodes formant des clusters d'items, le possible chevauchement des transactions est obtenu en les rangeant dans les clusters lors d'une étape de post-traitement. Nous avons remarqué que ce point pose problème pour beaucoup de ces méthodes. La méthode de Ronkainen associée à un rangement des transactions dans les clusters où les items sont tous vérifiés, nous a en effet permis d'obtenir des chevauchements possibles de transactions entre les clusters. Mais les méthodes de regroupement d'items forment des partitions, de ce fait les items ne sont pas répétés dans plusieurs clusters. Il n'est pas possible d'obtenir du chevauchement d'items, ce qui dans certains cas induit une perte d'information au niveau de la description des clusters car celle-ci n'est pas maximale. Remarquons que le post-traitement nécessaire pour classer les transactions peut être coûteux pour une quantité importante de données.

Il paraît donc important de former à la fois la description utilisant les items, et les transactions des clusters. Nous avons remarqué que beaucoup de méthodes se basent sur les associations fréquentes d'items, autrement dit, les motifs fréquents. En effet, *CLIQUE* utilise une notion appelée unité dense maximale, ce qui peut être comparé aux motifs fréquents maximaux. *CACTUS* utilise sous une certaine forme les motifs fréquents. *ARHP* est une des rares méthodes qui utilisent directement les motifs fréquents. Notons que dans ce cas, il n'est pas nécessaire de définir une mesure de similarité, car cela est implicite. Remarquons que si on veut bénéficier de tous les avantages des motifs fréquents, il est préférable d'utiliser directement les motifs. On ne se préoccupe alors pas du rangement des transactions. Ce point pouvant être réalisé lors de la découverte des motifs. Notons qu'un motif associé à l'ensemble des transactions qui le supporte, forme un *bi-set*. Il semble donc être très intéressant de s'orienter vers de telles approches. Le principal inconvénient de l'utilisation des motifs fréquents est leur nombre qui est très élevé. De plus, beaucoup d'entre eux ne sont pas intéressants.

3.4 Aide à l'interprétation des clusters et notion de *bi-sets*

L'interprétation des clusters de manière à les exploiter, est une tâche difficile mais importante pour les utiliser. Il est donc indispensable que les résultats d'un algorithme de clustering aident l'utilisateur en présentant une description de chacun des clusters obtenus. Une formalisation de cette idée est apparue récemment avec la notion de *bi-sets* dans le cadre de travaux en bio-informatique [BRBR04]. Un bi-set est composé de deux

ensembles : l'un est constitué d'éléments, et l'autre représente des propriétés décrivant ces éléments. Ils sont notamment utilisés pour représenter des ensembles de gènes associés à des ensembles de situations. On parle alors dans ce cas de module de transcription.

Le fait de considérer un bi-set comme un cluster, permet de formaliser l'objectif de rechercher simultanément les clusters d'éléments et leurs descriptions. Les algorithmes de clustering réalisant cela sont très rares. Quasiment toutes les méthodes que nous avons vues dans la section précédente ne forment pas directement des bi-sets complets. Ils forment seulement soit les éléments (regroupement de transactions), soit les descriptions (regroupement d'items). Un algorithme tel que *Bi-Clust* produit des bi-sets, car chaque ensemble de la partition des items est lié à un ensemble de la partition des transactions. Remarquons que *Ping-Pong* est aussi un algorithme produisant des bi-sets avec d'éventuelles erreurs sur les descriptions.

Un autre point important que nous avons précédemment identifié, est l'intérêt d'obtenir une description maximale de chaque cluster. Ainsi, *Ping-Pong* permet d'identifier des ensembles maximaux d'items pour des transactions, en ajoutant des possibles exceptions. Remarquons que l'idée est proche de la connexion de Galois qui est utilisée en classification conceptuelle pour former le treillis de concepts.

Les méthodes de classification conceptuelle permettent en effet d'obtenir à la fois les éléments et les descriptions des clusters. Nous allons donc les aborder maintenant.

3.5 La classification conceptuelle

Les méthodes de classification conceptuelle relevant de l'apprentissage [Mit82] traitent les données qualitatives. Les mesures de similarité sont basées sur la probabilité de chevauchement entre transactions : plus les transactions partagent des items, plus elles auront de chances d'être regroupées dans un même cluster. Le mesure la plus simple est le rapport entre le nombre d'items similaires et le nombre total d'items. Le principal avantage est la capacité du système d'apprentissage à donner une description symbolique des clusters découverts. Il existe des approches probabilistes comme *CLUSTER* et *COBWEB*, et des approches déterministes comme *GALOIS* formant une hiérarchie de concepts formels [Wil89].

3.5.1 Hiérarchies de concepts

CLUSTER, proposé par Michalski et Stepp [MS83], construit une hiérarchie de concepts et leurs descriptions sous forme de conjonction d'items. Le principe utilisé est similaire aux méthodes de partitionnement vues précédemment dans la section 3.2.1. L'idée générale est la suivante. L'algorithme sélectionne k noyaux, puis détermine pour chaque noyau une description le recouvrant et ne recouvrant aucun des autres noyaux. Chacune des descriptions est appliquée aux transactions restant de façon à former k classes. Il détermine ensuite à partir des classes obtenues, k nouveaux noyaux qui seront utilisés pour générer une nouvelle classification. Il répète cela tant que le critère de qualité de la classification n'est pas atteint. Ce critère peut inclure par exemple, le nombre d'items par concept, le poids des différents items, le nombre de transactions par concept, ou les caractéristiques

géométriques des transactions. L'utilisateur doit fournir plusieurs paramètres dont le critère de qualité, et de ce fait, la méthode est "guidée". Remarquons que cette méthode n'est pas incrémentale.

COBWEB est un algorithme incrémental proposé par Fisher [Fis87]. La hiérarchie de concepts est construite au fur et à mesure que les transactions sont ajoutées. Les concepts sont organisés sous forme d'une hiérarchie de spécialisation. Les concepts les plus spécifiques se trouvent en bas de la hiérarchie. Le concept racine décrit toutes les transactions contenues dans les données. L'insertion d'une nouvelle transaction entraîne la mise à jour de la hiérarchie. Cela est réalisé en utilisant une "mesure de prédictivité" qui repose sur l'estimation des probabilités d'apparition des différents couples attribut-valeur. Cette mesure permet d'évaluer la similarité intra-cluster et la similarité inter-clusters à l'aide de probabilités. *COBWEB* utilise un ensemble de distribution de données pour décrire les concepts et non une conjonction d'items. Par exemple, la distribution pour un attribut *couleur* d'un concept peut être $(bleu : 0,6; vert : 0,4)$. Pour ce concept, l'attribut *couleur* a une probabilité égale à 0,6 d'être *bleu*. Remarquons qu'il n'est pas nécessaire de fixer le nombre de concepts. Néanmoins, *COBWEB* modifie la hiérarchie au niveau local lors de l'insertion d'une nouvelle transaction. Les hiérarchies produites dépendent donc de l'ordre d'insertion des transactions. Ce point est particulièrement gênant.

Il existe aussi d'autres algorithmes que nous ne détaillons pas dans ce mémoire, comme par exemple, *CLASSIT* [GLF89] qui est un successeur de *COBWEB* traitant les données quantitatives, *UNIMEM* [Leb87], *CYRUS* [Kol83], *WITT* [HB89], *KBG* [Bis92].

3.5.2 Treillis de concepts

Etant donné un contexte de fouille de données $\mathcal{B}$, il y a un unique ensemble ordonné qui décrit la structure du treillis en définissant le regroupement et les relations parmi les transactions et les items. Cette structure est connue sous le nom de treillis de concepts ou treillis de Galois [GMA95, Wil82]. Chaque élément du treillis est un couple (T, I) composé d'un ensemble de transactions (appelé extension) et d'un ensemble d'items (appelé intension). Chaque couple (appelé *concept* par Wille [Wil82]) doit être un couple complet selon $\mathcal{R}$, ce qui signifie que les deux propriétés suivantes (notées f et g) sont satisfaites. Pour $T \subseteq \mathcal{T}$ et $I \subseteq \mathcal{I}$, nous avons :

$$f(T) = \{i \in \mathcal{I} \mid \forall t \in T \ \ (t, i) \in \mathcal{R}\}$$
$$g(I) = \{t \in \mathcal{T} \mid \forall i \in I \ \ (t, i) \in \mathcal{R}\}$$

$f(T)$ associe à T, les items communs à toutes les transactions t appartenant à T, et $g(I)$ associe à I, les transactions en relation avec tous les items i appartenant à I. En d'autres mots, T est le plus grand ensemble de transactions décrites par les items trouvés de I, et symétriquement, I est le plus grand ensemble d'items communs pour les transactions supportant I. Par exemple, dans le tableau 3.1, le couple composé des transactions t_1, t_2 et t_3 d'un côté et des items ABC de l'autre, est un concept du treillis, tandis qu'il n'y a pas de couples composés des transactions t_1, t_2 (puisque la transaction t_3 partage les mêmes items que les transactions t_1 et t_2). L'idée de maximalité étendant les ensembles est formalisée par la notion mathématique de *fermeture* dans les ensembles ordonnés et les opérateurs $h = f \circ g$ et $h\prime = g \circ f$ sont les opérateurs de fermeture de Galois. Ils sont aussi appelés connexion de Galois.

Le diagramme de Hasse associé au treillis représente une relation de généralisation/spécialisation entre les concepts. La figure 3.4 représente le treillis de concepts obtenu sur notre exemple.

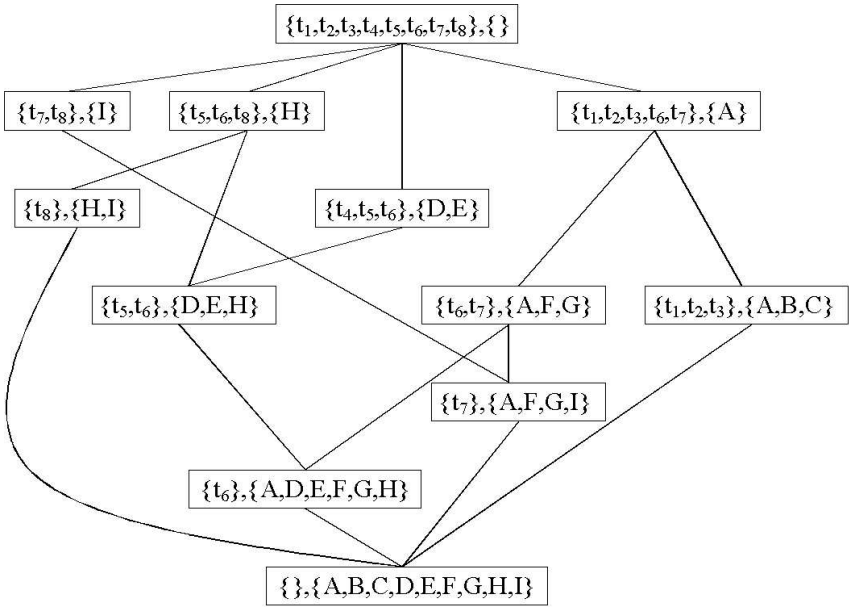

FIG. 3.4 – Treillis de concepts obtenu sur l'exemple

GALOIS [CR93], proposée par Carpineto et Romano, est une méthode incrémentale de construction du treillis de concepts. Lors de l'ajout d'une nouvelle transaction, l'algorithme parcourt le treillis et compare l'intension du concept courant avec les items de la transaction (description de la transaction). Si l'intension du concept courant est plus générale que la description de la transaction, alors cette dernière est ajoutée à l'extension. Si l'intention n'est pas comparable avec la description de la transaction, alors le concept reste inchangé. De nouveaux concepts peuvent être créés pendant ce parcours, notamment lorsque l'intersection entre le concept courant et la description de la transaction n'est pas déjà l'intension d'un concept existant. Ce principe est semblable à d'autres méthodes de construction incrémentale de treillis de concepts, comme celle de Godin et al. [GMA95]. Remarquons que cette méthode ne nécessite pas de fixer le nombre de concepts. *GALOIS* effectue une modification globale de la hiérarchie lors de l'ajout d'une nouvelle transaction, par conséquent, ne dépend pas de l'ordre d'insertion des transactions.

Nous pouvons aussi citer d'autres algorithmes construisant le treillis de concepts comme par exemple ceux proposés par Bordat [Bor86] et Ganter [Gan84]. Selon des expérimentations réalisées par Fu et al. [FMN04], celui de Ganter est le plus rapide des deux.

Les concepts du treillis sont formés de façon à ce que toutes les transactions d'un concept partagent un ensemble maximal d'items. Ce dernier est en fait un motif particulier appelé *motif fermé*. Nous présenterons et utiliserons ces motifs dans le chapitre 4, afin de découvrir des clusters.

3.6 Conclusion

Le tableau récapitulatif en annexe B, présente les principales méthodes de clustering. L'une des originalités de ce tableau est de présenter les méthodes en isolant celles basées sur les motifs et la notion de fréquence.

Parmi toutes les méthodes que nous avons abordées dans ce chapitre, il y a peu de méthodes qui autorisent le recouvrement d'éléments entre clusters. Or, cela permettrait de capter différents points de vue. *Fuzzy c-Medoids* permet de déterminer la probabilité d'appartenance d'un élément à un cluster. *PoBOC* recherche aussi des clusters non-disjoints. Mais ces méthodes sont basées sur une mesure de similarité entre les données, et elles ne donnent pas de descriptions claires des clusters obtenus.

Pour exploiter ensuite les clusters, il est important d'obtenir les éléments qui les composent et les descriptions expliquant le regroupement (par exemple sous forme de conjonctions d'items). Ainsi, un cluster peut être formalisé par un *bi-set* (cf. section 3.4) composé d'un ensemble d'items et d'un ensemble de transactions. *Bi-Clust* produit des bi-sets, mais ces derniers forment une partition des transactions et des items. Des approches basées sur la notion de fréquence permettent de produire des "moitiés" de bi-sets. Les méthodes comme celle de Wang et al. ne forment que les ensembles de transactions, et n'autorisent aucun chevauchement. Les méthodes de regroupement d'items, comme la méthode de Ronkainen et *ARHP*, peuvent servir à créer des clusters se chevauchant. Mais pour cela, il faut ranger les transactions dans les clusters dont les items sont vérifiés, car on n'obtient que les descriptions. Nous avons vu que cette tâche est délicate, et induit des problèmes. De plus, les descriptions des clusters peuvent ne pas être maximales (cf. section 3.3.2, paragraphe "Rangement des transactions") ce qui signifie une perte d'information. Nous avons remarqué que les motifs fréquents sont utilisés par plusieurs algorithmes, notamment *ARHP*. Cette méthode est une des rares à utiliser directement des motifs fréquents pour produire des clusters. Bien que cela ne forme que des clusters d'items, l'utilisation de ces motifs représente une voie intéressante, car il n'y a pas à définir de mesure de similarité. De plus, il est possible de compléter les bi-sets avec les transactions supportant chaque motif ou s'en approchant le plus.

Nous avons aussi abordé les méthodes de classification conceptuelle qui permettent d'obtenir en même temps la description des clusters et les éléments qui les composent. Les treillis de concepts présentent des idées originales telles que la maximalité de la description des clusters. Chaque concept est décrit par un ensemble de transactions, et les items partagés par ces dernières. En cela, un concept est un bi-set particulier. De plus, les concepts se recouvrent au niveau des éléments et aussi au niveau des items. Le treillis de concepts a une structure formellement définie qui ne dépend pas de divers paramètres, de l'ordonnancement des transactions ou de particularités algorithmiques. La hiérarchie obtenue n'est pas réduite à un arbre, et met en évidence de façon exhaustive les regroupements potentiellement intéressants des transactions par rapport à leurs items. Les concepts décrits par des ensembles d'items, sont facilement interprétables. De plus, la description d'un concept est maximale. La connexion de Galois fait que la description correspond à l'ensemble maximal des items partagés par les transactions. Par contre, le nombre de concepts n'étant pas limité, et souvent très important pour des données réelles, les résultats sont difficilement exploitables.

Les méthodes que nous avons présentées ne satisfont pas tous nos besoins tels que le chevauchement de transactions, l'obtention des descriptions des clusters, le nombre non fixé de clusters, et la gestion de quantités importantes de données. Remarquons que ce dernier point incite certains algorithmes à échantillonner les données. Nous avons néanmoins remarqué des points intéressants que nous pourrons utiliser pour définir notre propre méthode de clustering. Le premier point est l'utilisation des motifs fréquents qui permet de traiter des données qualitatives et qui ne nécessite pas de définir une mesure de similarité. Nous avons aussi noté que la notion de fermeture utilisée pour former les concepts du treillis de Galois, permet de capter un maximum de similarité pour un ensemble de transactions, et donc de produire des descriptions de taille maximale.

Dans le chapitre 4, nous présentons une nouvelle méthode de découverte de clusters, appelée ECCLAT, qui prend en considération ces notions, et qui se concrétise par un système permettant de produire des clusters répondant à nos besoins.

Chapitre 4

ECCLAT : une nouvelle méthode de découverte de clusters pour les données qualitatives

Sommaire

Dans ce chapitre, nous présentons une nouvelle méthode de découverte de clusters significatifs à partir d'importantes quantités de données qualitatives. Notre méthode est appelée ECCLAT (Extraction of Clusters from Concepts LATtice) [DC02a, DC02b]. Nous verrons dans la section 4.1 que les concepts définis par le treillis de Galois sont des motifs fermés, et forment un réservoir de clusters potentiels [DCHA01]. L'une des originalités d'ECCLAT est de tirer profit de résultats en fouille de données sur les représentations condensées de motifs, à savoir les motifs fermés fréquents [PBTL99]. ECCLAT extrait un sous-ensemble de concepts du treillis en utilisant une mesure d'évaluation (cf. section 4.2). Ce critère d'évaluation est basé sur la définition classique d'un "bon" cluster. Notons que notre critère est indépendant du domaine d'application.

ECCLAT permet de construire un clustering approché et de découvrir un ensemble de clusters significatifs avec un léger chevauchement d'éléments. Le comportement de la méthode est paramétré par l'utilisateur, et il n'est pas nécessaire de fixer le nombre de clusters.

Comme nous l'avons vu (cf. chapitre 2), nos besoins en clustering sont entre autres motivés par l'élaboration d'un système de recommandation de documents. Dans ce but, nous utilisons ECCLAT sur des données provenant de fichiers de traces de serveurs mandataires de France Télécom R&D (cf. section 4.3.2). Nous illustrons aussi notre approche sur des bases de données classiques (cf. section 4.3.1), et sur des données médicales des Discovery Challenges PKDD 2002 [CD02] (cf. section 4.3.2) et 2004 [DCS04] (cf. section 4.4).

4.1 Les motifs fermés fréquents : réservoir de clusters potentiels

Nous avons conclu à la fin du chapitre 3 que les motifs fréquents et les concepts du treillis de Galois (cf. section 3.5.2) présentent des propriétés intéressantes dans un but de clustering sur des données qualitatives. En effet, ils n'utilisent pas de mesure de similarité et correspondent à des bi-sets ce qui facilite l'interprétation des résultats. De plus, les concepts permettent d'obtenir des descriptions maximales. Le principal inconvénient est le nombre très élevé de concepts pour des données réelles, ce qui rend difficile l'exploitation des résultats.

Des travaux ont été réalisés pour réduire ce nombre en sélectionnant les concepts dits fréquents, c'est-à-dire dont l'extension contient un nombre minimum de transactions. Cela forme ainsi le treillis des concepts fréquents [Wai99], et représente un pas vers une sélection de concepts intéressants. Cependant, le taille du treillis reste importante.

Les récents travaux en découverte de connaissances dans les bases de données ont porté sur des algorithmes efficaces d'extraction de motifs, et plus précisément de représentations condensées de motifs. On dispose entre autres d'algorithmes permettant de rechercher les motifs fermés fréquents [PBTL99, BB00, STB$^+$02, ZH02] sur de grandes quantités de données. Ces motifs fermés fréquents correspondent aux concepts fréquents que nous avons précédemment évoqué. Notons aussi que l'algorithme *D-Miner* [BRB04] permet de rechercher des motifs fermés sous contraintes. La contrainte la plus simple étant celle définissant le seuil minimum de fréquence. Les travaux portant sur la recherche de motifs fermés fréquents intéressants de façon à diminuer leur nombre, sont rares. Pourtant de tels travaux seraient un premier pas vers une découverte de clusters.

Nous proposons donc d'utiliser les motifs fermés fréquents comme base pour former un ensemble de clusters, et ainsi de bénéficier d'algorithmes efficaces pour la découverte des concepts fréquents. Ces clusters candidats seront ensuite évalués puis sélectionnés.

Revenons à présent sur les notions de motif fermé et de fréquence, afin de présenter en détail leurs propriétés.

Un motif fermé est un ensemble maximal d'items (en respectant l'inclusion ensembliste) partagé par un ensemble de transactions. La définition 6 présente cette notion d'une façon plus formelle.

Définition 6 (motif fermé) *Soit X un motif, X est un motif fermé si et seulement si $h(X) = X$, où h est la connexion de Galois.*

Le bi-set composé d'un motif fermé X et des transactions supportant X, est comme nous l'avons dit précédemment, un concept du treillis de Galois. $g(X)$ désigne l'ensemble des transactions supportant le motif X. La fréquence de X ($\mathcal{F}(X)$) peut aussi se noter $|g(X)|$. Si $\mathcal{F}(X)$ est supérieure ou égale à $minfr$, alors X est un motif fermé fréquent.

Dans le reste de ce chapitre, X désignera un motif fermé fréquent, et $\mathcal{FC}$ (Frequent Closed) l'ensemble des motifs fermés fréquents.

Remarquons que les algorithmes de découverte de motifs fermés fréquents se basent sur une autre formulation de la connexion de Galois, donnée dans la définition 7.

Définition 7 (fermeture d'un motif) *Le motif fermé correspondant au motif Y (noté $h(Y)$), est l'intersection de toutes les transactions supportant Y.*

$$h(Y) = \bigcap_{t \in \mathcal{T}} \{f(t) \mid \forall i \in Y \ \ (t,i) \in \mathcal{R}\}$$

Les algorithmes comme *CLOSE* [PBTL99] ou *TITANIC* [STB+02], utilisent cette définition sur des motifs particuliers appelés générateurs (cf. annexe C). Pour découvrir les motifs fermés fréquents, nous pouvons aussi citer *CLOSET* [PHM00] et *CHARM* [ZH02].

D'autres algorithmes tels que *ACMINER* [BB00] et *MVMINER* (cf. annexe F), découvrant les motifs libres [BTP+02, BBR03, RC03a] qui est une autre représentation condensée de motifs, permettent aussi d'obtenir les motifs fermés. Nous utiliserons *MVMINER*, développé à l'université de Caen par François Rioult, pour nos expérimentations (cf. section 4.3).

Revenons sur notre exemple (cf. tableau 3.1) pour illustrer la notion de motif fermé fréquent. ABC est un motif fermé car il correspond à l'ensemble maximal d'items partagés par les transactions supportant au moins ABC. AB n'est pas fermé car ce n'est pas un ensemble maximal d'items : toutes les transactions ayant les items AB ont aussi l'item C. Autrement dit, nous pouvons ajouter l'item C à AB sans diminuer sa fréquence. Il est clair que dans un but de clustering, il est préférable de considérer ABC plutôt que AB.

$\mathcal{FC}$	Fréquence	Transactions
I	2	t_7 t_8
H	3	t_5 t_6 t_8
AFG	2	t_6 t_7
DE	3	t_4 t_5 t_6
ABC	3	t_1 t_2 t_3
A	5	t_1 t_2 t_3 t_6 t_7
DEH	2	t_5 t_6

TAB. 4.1 – Motifs fermés fréquents découverts sur notre exemple, $minfr$=2

Le tableau 4.1 récapitule les motifs fermés fréquents découverts sur l'exemple avec *minfr* égal à 2. La figure 4.1 représente le treillis de concepts fréquents correspondant.

Rappelons que le treillis contenant tous les concepts est présenté dans la figure 3.4 (cf. section 3.5.2).

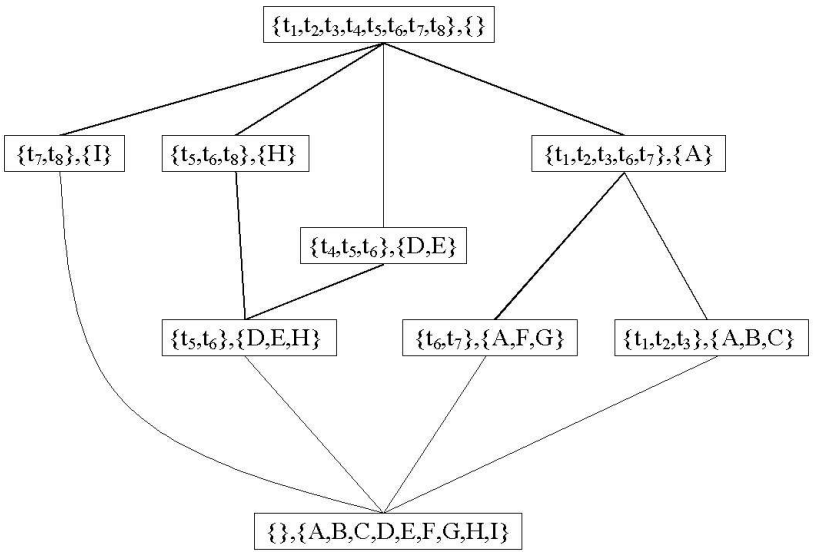

FIG. 4.1 – Treillis de concepts fréquents obtenu sur l'exemple, $minfr$=2

La fréquence est fondamentale pour extraire des clusters fiables. Cela permet de prendre en compte l'importance d'un cluster candidat et d'écarter les clusters qui risquent de présenter des artefacts.

La capture d'un maximum de similarité grâce aux motifs fermés pour décrire les clusters, et la notion de fréquence sont les bases de la sélection de clusters que nous présentons dans la section suivante.

Remarquons que dans le reste du chapitre, par abus de langage, nous appelons un cluster le motif fermé fréquent correspondant. En effet, étant donné un motif fermé fréquent, nous associons implicitement les transactions qui le supportent.

4.2 Evaluation et sélection de clusters candidats

Nous avons vu que les bi-sets composés de motifs fermés fréquents et des transactions les supportant représentent des clusters potentiels. Nous proposons maintenant une mesure d'évaluation de ces clusters candidats qui permet d'en sélectionner et de réduire leur nombre pour former une pseudo-partition des transactions.

4.2.1 Mesure d'évaluation d'un cluster candidat

Pour définir une mesure d'évaluation d'un cluster, il faut se poser la question : qu'est-ce qu'un bon cluster ? C'est une question difficile, car cela peut dépendre du domaine d'application et de l'interprétation que l'on va en faire.

Néanmoins, nous avons vu dans le chapitre 3 que la quasi-totalité des méthodes sont basées sur une maximisation de la similarité intra-cluster. Un "bon" cluster doit donc être très homogène. Nous avons traduit cela par une mesure d'*homogénéité*.

Les méthodes de clustering sont aussi basées sur la maximisation de la dissimilarité inter-cluster utilisée pour produire une partition des éléments. Dans notre contexte, nous ne voulons pas produire une partition stricte, mais autoriser le chevauchement d'éléments entre les clusters. Mais il est intéressant de noter que dans la pratique on ne souhaite pas avoir des clusters relativement semblables. Nous avons alors défini une mesure de *concentration*. Nous verrons que le calcul de la *concentration* ne nécessite pas de calculs directs d'un cluster par rapport à l'autre comme c'est le cas en général dans les algorithmes de clustering. Cela permet de limiter la mémoire utilisée et d'obtenir une complexité moindre.

Nous avons alors défini une mesure d'évaluation d'un cluster candidat, appelée *intérêt*, composée de l'*homogénéité* et de la *concentration*.

Homogénéité

L'*homogénéité* favorise les motifs fermés fréquents qui minimisent les "erreurs" induites sur ces motifs par leurs transactions. Nous comptabilisons pour chaque candidat le nombre d'items de leurs transactions, qui ne vérifient pas la définition du cluster. En cela, notre mesure favorise les motifs fermés fréquents ayant une forte similarité intra-cluster.

Nous présentons maintenant notre mesure. L'*homogénéité* d'un cluster X est calculée à partir de sa taille, sa fréquence $\mathcal{F}(X)$ (e.g. son nombre de transactions) et de la mesure de *divergence*.

Définition 8 (homogénéité d'un cluster candidat) *Soit X un cluster candidat, son homogénéité correspond à :*

$$homogénéité(X) = \frac{\mathcal{F}(X) \times |X|}{divergence(X) + (\mathcal{F}(X) \times |X|)}$$

La *divergence* d'un cluster permet de mesurer l'erreur de ses transactions par rapport à X, e.g. le nombre d'items non dans X, pour chaque transaction de $g(X)$. Pour une transaction, cela correspond au nombre d'items qui divergent par rapport aux items du motif fermé.

Définition 9 (divergence d'un cluster candidat) *Soit X un cluster candidat, sa divergence est définie de la façon suivante :*

$$divergence(X) = \sum_{t \in g(X)} |f(t) - X|.$$

Examinons les cas extrêmes. Dans le meilleur des cas, le cluster est "pur", $|t| = |X|$, alors la *divergence* est nulle, et l'*homogénéité* est égale à 1.

Dans le pire des cas, le motif fermé ne contient qu'un item, $|X| = 1$, et chacune de ses transactions contient tous les autres items, $|f(t) - X| = |\mathcal{I}| - 1$ où $t \in g(X)$. Nous avons alors $divergence(X) = \mathcal{F}(X) \times (|\mathcal{I}| - 1)$.

D'où

$$homogénéité(X) = \frac{\mathcal{F}(X)}{\mathcal{F}(X) \times (|\mathcal{I}| - 1) + \mathcal{F}(\mathcal{X})} = \frac{1}{|\mathcal{I}|}.$$

Nous avons donc :

$$\frac{1}{|\mathcal{I}|} < homogénéité(X) \leq 1.$$

Plus les transactions d'un cluster ont des items n'apparaîssant pas dans X, plus l'*homogénéité* du cluster tend vers $\frac{1}{|\mathcal{I}|}$.

Id.	Items
t_1	A B C
t_2	A B C
t_3	A B C
t_4	D E
t_5	D E **H**
t_6	**A** D E **F** **G** **H**
t_7	A F G I
t_8	H I

TAB. 4.2 – Illustration de l'*homogénéité* sur notre exemple

Reprenons notre exemple (cf. tableau 4.2), concernant le motif fermé fréquent ABC, tous les items de chacune de ses transactions appartiennent à ABC. La divergence est donc nulle. Nous avons alors :

$$homogénéité(ABC) = \frac{3 \times 3}{(0 + 0 + 0) + (3 \times 3)} = 1.$$

Pour le motif fermé fréquent DE, la transaction t_5 a un item qui diverge et quatre items divergent pour la transaction t_6. La divergence est donc égale à 5. Nous obtenons donc :

$$homogénéité(DE) = \frac{3 \times 2}{(0 + 1 + 4) + (3 \times 2)} = 0,545.$$

Discussion sur l'*homogénéité*

L'*homogénéité* joue le rôle de similarité intra-cluster pour ECCLAT. Remarquons qu'elle ne se réduit pas à comptabiliser les items fréquents et non-fréquents comme c'est le cas dans la méthode de Wang et al. (cf. section 3.3).

L'*homogénéité* et la fréquence minimum $minfr$ permettent d'obtenir un compromis entre beaucoup d'items et un nombre important de transactions supportant ces items. Si un cluster a beaucoup de transactions, beaucoup d'items doivent alors être partagés pour que la *divergence* soit faible, et l'homogénéité élevée. Si nous prenons le cas d'un cluster avec un nombre élevé d'items, il a de ce fait plus de chance que la divergence soit

faible. Une conséquence de cela, est que le nombre de transactions des clusters ayant les meilleures valeurs d'*homogénéité*, peut être relativement proche de $minfr$.

Nous remarquons que les motifs fermés fréquents maximaux ont tendance à être privilégiés par l'*homogénéité*, car le caractère maximal réduit la *divergence*. Mais, comme nous le montre l'exemple suivant (cf. tableau 4.3), l'*homogénéité* ne choisit pas forcément les maximaux. Si $minfr$ est égal à 2, ABC est un motif fermé fréquent, et $ABCD$ est le motif fermé fréquent maximal correspondant. L'*homogénéité* de ABC est égal à $(4\times3)/(7+4\times3) = 0,631$, alors que l'*homogénéité* de $ABCD$ est $(2\times4)/(5+2\times4) = 0,615$. Un motif fermé peut être plus homogène, selon notre mesure, que le motif fermé maximal correspondant.

Id.	Items
t_1	A B C D
t_2	A B C D E F G H I
t_3	A B C
t_4	A B C

TAB. 4.3 – Autre exemple de base de données transactionnelle

Remarquons que l'*homogénéité* d'un motif fermé dépend seulement de lui-même et peut ainsi être calculé simultanément au motif fermé. En effet, la *divergence* s'exprime aussi de la façon suivante :

$$divergence(X) = \sum_{t\in g(X)} |f(t) - X|$$

$$divergence(X) = \sum_{t\in g(X)} |f(t)| - \sum_{t\in g(X)} |X|$$

$$divergence(X) = \sum_{t\in g(X)} |f(t)| - \mathcal{F}(X) \times |X|.$$

Lors de la découverte d'un motif fermé, il suffit donc de stocker le nombre total d'items de toutes ses transactions. Une fois le motif fermé complétement formé, il ne reste plus qu'à retrancher le produit de sa fréquence avec son nombre d'items, et nous obtenons sa divergence. L'*homogénéité* est ensuite déduite. Le coût de calcul est donc minime. Nous avons intégré ce calcul dans le processus de découverte des motifs fermés fréquents tels que CLOSE (cf. annexe C) et MVMINER (cf. annexe F).

Concentration

Bien que nous ne cherchons pas une partition des transactions, il s'avère utile de définir une mesure s'apparentant à la dissimilarité inter-clusters que nous trouvons dans les algorithmes de clustering. Nous appelons *concentration* cette mesure. Elle a pour rôle de privilégier les motifs fermés fréquents qui concentrent des transactions. Elle permet notamment de limiter la duplication des transactions dans les clusters qui seront sélectionnés

par la suite. Remarquons que cela permet d'identifier des concepts du treillis de façon à trouver, si elles existent, des zones que nous pouvons qualifiées de "denses" par rapport aux transactions, et qui induisent un faible recouvrement entre les clusters.

Autrement dit, nous voulons favoriser des clusters ayant des transactions apparaîssant le moins possible dans l'ensemble des clusters. La *concentration* d'un cluster est donc définie en prenant en compte le nombre de clusters où chaque transaction apparaît. Nous définissons alors la fréquence absolue d'une transaction t, notée $\mathcal{F}'(t)$, comme le nombre de motifs fermés fréquents qui supportent t. $\mathcal{F}'(t)$ est la fréquence de t dans $\mathcal{FC}$.

Définition 10 (fréquence absolue d'une transaction) *Soient $\mathcal{FC}$ l'ensemble des motifs fermés fréquents étant donné un seuil $minfr$, et t une transaction. La fréquence de t est définie par :*

$$\mathcal{F}'(t) = |\{X \mid X \in \mathcal{FC} \quad et \quad t \in g(X)\}|.$$

Définition 11 (concentration d'un cluster candidat) *Soit X un cluster candidat, la concentration de X est définie par :*

$$concentration(X) = \frac{1}{|g(X)|} \times \sum_{t \in g(X)} \frac{1}{\mathcal{F}'(t)}$$

où $\mathcal{F}'(t)$ est le nombre de clusters candidats où t apparaît.

Examinons les cas extrêmes. Dans le meilleur des cas, toutes les transactions de $g(X)$ apparaîssent seulement dans X, $\mathcal{F}'(t) = 1$, alors la *concentration* est égale à 1. Dans le pire des cas, chaque transaction de $g(X)$ apparaît dans chaque motif fermé fréquent, $\mathcal{F}'(t) = |\mathcal{FC}|$, alors la *concentration* est égale à $\frac{1}{|\mathcal{FC}|}$.

Nous avons donc :

$$\frac{1}{|\mathcal{FC}|} < concentration(X) \leq 1.$$

Plus les transactions sont présentes dans l'ensemble des clusters, plus la *concentration* tend vers $\frac{1}{|\mathcal{FC}|}$.

Id.	Items
t_1	A B C
t_2	A B C
t_3	A B C
t_4	**D E**
t_5	**D E** H
t_6	A **D E** F G H
t_7	A F G I
t_8	H I

$\mathcal{FC}$	Fréquence	Transactions
ABC	3	t_1 t_2 t_3
DE	3	**t_4 t_5 t_6**
DEH	2	**t_5 t_6**
AFG	2	**t_6** t_7
I	2	t_7 t_8
A	5	t_1 t_2 t_3 **t_6** t_7
H	3	**t_5 t_6** t_8

TAB. 4.4 – Illustration de la *concentration* sur notre exemple

Dans notre exemple (cf. tableau 4.4), chaque transaction supportant ABC appartient à deux motifs fermés fréquents. Nous avons alors :

$$concentration(ABC) = \frac{1}{3} \times (\frac{1}{2} + \frac{1}{2} + \frac{1}{2}) = 0,5.$$

Pour DE, la transaction t_4 supporte seulement le motif fermé fréquent DE, la transaction t_5 supporte trois motifs fermés fréquents, et la transaction t_6 supporte cinq motifs fermés fréquents. Nous obtenons donc :

$$concentration(DE) = \frac{1}{3} \times (\frac{1}{1} + \frac{1}{3} + \frac{1}{5}) = 0,511.$$

Discussion sur la *concentration*

La *concentration* permet de mettre en évidence des concepts contenant des transactions "rares" dans $\mathcal{FC}$, s'ils existent. Si la *concentration* d'un motif fermé fréquent est égale à 1, alors cela peut signifier que soit tous ses items ne sont pas présents dans d'autres transactions, soit ils ne le sont pas assez pour former des motifs fermés fréquents.

Remarquons que par rapport à la dissimilarité inter-clusters des algorithmes classiques, la *concentration* ne nécessite aucun calcul pour chaque paire de clusters. Le calcul est donc plus efficace.

La *concentration* ne peut se calculer qu'une fois avoir obtenu tous les motifs fermés fréquents. Elle ne peut donc pas s'intégrer aux algorithmes de découverte de motifs. Néanmoins, il suffit seulement d'effectuer deux parcours de $\mathcal{FC}$ pour calculer toutes les concentrations (cf. algorithme 1). La complexité est donc linéaire suivant $|\mathcal{FC}|$.

Algorithme 1 Calcul de la *concentration*

Début
créer un tableau $freqp$ *à* $|\mathcal{T}|$ *éléments*; {stocke les $\mathcal{F}'(t)$}
initialiser les éléments de $freqp$ *à* 0;
pour tout $X \in \mathcal{FC}$ **faire**
  pour tout $t \in g(X)$ **faire**
    $freqp[t]$++;
  fin pour
fin pour
pour tout $X \in \mathcal{FC}$ **faire**

$$calculer \quad concentration(X) = \frac{1}{|g(X)|} \times \sum_{t \in g(X)} \frac{1}{freqp[t]};$$

fin pour
Fin

Intérêt

Finalement, nous définissons l'*intérêt* d'un cluster comme la moyenne de son homogénéité et de sa concentration. Nous avons $0 < intérêt(X) \leq 1$.

Définition 12 (intérêt d'un cluster candidat) *Soit X un cluster candidat, l'intérêt de X est définie par :*

$$intérêt(X) = \frac{homogénéité(X) + concentration(X)}{2}$$

En reprenant notre exemple, nous obtenons :

$$intérêt(ABC) = \frac{1 + 0,5}{2} = 0,75$$

et

$$intérêt(DE) = \frac{0,545 + 0,511}{2} = 0,528.$$

Le tableau 4.5 récapitule tous les motifs fermés fréquents avec leurs transactions, et les mesures d'*homonégéité*, de *concentration* et d'*intérêt*, pour $minfr$ égal à 2. Nous remarquons que ABC est le cluster candidat ayant le plus d'*intérêt* selon nos mesures.

$\mathcal{FC}$	Transactions	Homogénéité	Concentration	Intérêt
ABC	t_1 t_2 t_3	1	0,5	0,75
DE	t_4 t_5 t_6	0,545	0,511	0,528
DEH	t_5 t_6	0,666	0,266	0,466
AFG	t_6 t_7	0,6	0,266	0,433
I	t_7 t_8	0,333	0,416	0,375
A	t_1 t_2 t_3 t_6 t_7	0,263	0,406	0,334
H	t_5 t_6 t_8	0,272	0,344	0,308

TAB. 4.5 – Intérêt des motifs fermés fréquents, $minfr$=2

Discussion sur la mesure d'*intérêt*

Le coût pour calculer l'intérêt est faible car l'*homogénéité* est calculée lors du processus de découverte des motifs fermés fréquents, et le calcul de la *concentration* ne représente que deux parcours de $\mathcal{FC}$.

Notons que le poids de la *concentration* face à l'*homogénéité* est faible car la *concentration* n'est pas normalisée. Ceci est préférable vu que nous recherchons à obtenir du chevauchement. Si deux candidats ont une valeur proche ou égale d'*homogénéité*, alors la *concentration* va les départager.

Nous pouvons formuler quelques remarques, notamment sur les données rectangulaires où les transactions ont toutes le même nombre d'items. Dans ce cas, les motifs fermés fréquents ayant le même nombre d'items ont la même *divergence*. L'*homogénéité* est alors

proportionnelle à la fréquence. Le résultat est bien celui que nous voulons, car les clusters ayant le plus de transactions parmi ceux qui ont la plus grande description, sont privilégiés. Ceux sont donc bien les plus homogènes. Mais la *divergence* n'opère pas comme définie initialement. Si deux motifs fermés fréquents ont le même nombre d'items et de de transactions, alors seule la concentration les différencie.

Notre mesure pourrait aussi être adaptée et utilisée sur les données transposées de façon à éliminer le caractère rectangulaire [RC03b]. Remarquons qu'il ne serait pas nécessaire de réaliser en réalité cette transposition, les mesures "transposées" pouvant être calculées sur les données originales. Nous reviendrons sur ce point dans la conclusion de ce chapitre.

Après avoir déterminé l'*intérêt* de chaque cluster candidat, nous en sélectionnons parmi ceux qui ont les valeurs les plus élevées. Nous présentons maintenant l'algorithme utilisé pour cette étape.

4.2.2 Algorithme de sélection des clusters

La mesure d'*intérêt* calculée pour chaque cluster candidat n'est pas suffisante pour trouver des clusters. En effet, il ne suffit pas de sélectionner ceux qui ont les plus grandes valeurs, car deux clusters ayant une valeur d'*intérêt* proche peuvent avoir des ensembles d'items et de transactions très proches. Il est donc préférable de n'en prendre qu'un seul. Nous proposons donc un algorithme de sélection.

Nous utilisons la mesure d'*intérêt* définie dans la section précédente pour guider la sélection des clusters à partir du treillis des concepts fréquents. Un point innovant d'ECCLAT est sa capacité de produire un clustering avec un faible chevauchement entre les clusters (que nous appellerons "*clustering approché*") ou un ensemble de clusters avec chevauchement. Cette fonctionnalité dépend de la valeur d'un paramètre appelé M. M est un entier correspondant au nombre de transactions non encore classées, au cours de l'algorithme, que la sélection d'un cluster doit classer. Bien entendu, M est inférieur ou égal à $minfr$.

Si $minfr$ est assez bas, toutes les transactions apparaîtront dans au moins un motif fermé fréquent. Et si M est égal à 1, il est certain que toutes les transactions seront classées dans les clusters sélectionnés. Néanmoins, un chevauchement entre les clusters peut apparaître. Plus la valeur de M augmente, plus le chevauchement entre les clusters diminue mais plus la possibilité d'obtenir des transactions non classées augmente. De telles transactions sont alors regroupées dans un cluster appelé "reste".

L'algorithme de sélection est glouton. Préalablement, l'*intérêt* de chaque cluster candidat appartenant à $\mathcal{FC}$ est calculé. Le candidat ayant la plus grande valeur d'intérêt est sélectionné. Puis, tant qu'il y a des transactions à classer et qu'il reste des clusters candidats, nous sélectionnons celui ayant la plus grande valeur d'*intérêt* et contenant au moins M transactions non encore classées. La sélection des clusters est présentée dans l'algorithme 2.

Les résultats de la méthode sont liées à la valeur de M. Illustrons ce comportement avec notre exemple.

Algorithme 2 Sélection des clusters

Début
$TrClassées = \emptyset$; {Les transactions classées}
$Clusters = \emptyset$; {Les clusters résultats}
$Candidats = \mathcal{FC}$; {Les motifs fermés fréquents}
tant que ($|TrClassées| < |\mathcal{T}|$ *et* $Candidats \neq \emptyset$) **faire**
  Sélectionner $X \in Clusters$ / *intérêt*(X) *soit maximum et* $|g(X) - TrClassées| \geq M$;
  $Candidats = Candidats - X$;
  $Clusters = Clusters \cup X$;
  $TrClassées = TrClassées \cup g(X)$;
fin tant que
retourner $Clusters$;
Fin

Pour $M = 1$, le cluster candidat ayant la plus grande valeur d'*intérêt* est ABC (cf. tableau 4.5), il est donc sélectionné. Les transactions t_1, t_2 et t_3 sont classées. Puis, le cluster DE est choisi et les transactions t_4, t_5, et t_6 sont classées. Il reste alors à classer les transactions t_7 et t_8. Le cluster DEH est passé car il ne contient ni la transaction t_7 ni la transaction t_8. Le cluster AFG est sélectionné car il classe la transaction t_7. Il ne reste plus que la transaction t_8 à classer. Finalement, le cluster I est sélectionné. Toutes les transactions sont classées dans quatre clusters. Nous avons le chevauchement suivant : la transaction t_6 apparaît à la fois dans DE et AFG, et la transaction t_7 appartient à AFG et I. Intuitivement, lorsque nous observons les transactions t_6 et t_7 (cf. tableau 3.1), il n'y a aucune raison de les classer dans un cluster ou dans un autre. Notons que l'item A apparaît pour deux clusters, d'où aussi un chevauchement d'items.

Avec $M = 2$, nous obtenons les clusters ABC, DE and I. AFG n'est pas sélectionné car il ne permet de classer qu'une seule transaction restante. Il n'y a pas de chevauchement, nous obtenons une partition des transactions pour $M = minfr$ (cf. tableau 4.6).

Id.	Items
t_1	A B C
t_2	A B C
t_3	A B C
t_4	**D E**
t_5	**D E** H
t_6	A **D E** F G H
t_7	A F G I
t_8	H I

Clusters	Fréquence	Transactions
ABC	3	t_1 t_2 t_3
DE	3	**t_4 t_5 t_6**
I	2	t_7 t_8

TAB. 4.6 – Résultats d'ECCLAT sur notre exemple, $minfr$=2, M=2.

Avec $M = 3$, seulement deux clusters sont sélectionnés (ABC and DE) et un cluster "reste" est formé avec les transactions t_7 et t_8.

Discussion sur la procédure de sélection

D'un point de vue calculatoire, il est évident qu'une faible valeur de $minfr$ implique un nombre important de motifs fermés fréquents et un temps d'exécution plus élevé. En ce qui concerne M, plus la valeur de M augmente, plus le nombre de contraintes à vérifier augmentent et cela réduit l'efficacité de la procédure.

Les résultats expérimentaux (cf. section 4.3) montre que le choix d'une valeur de M proche de $minfr$ donne de bons résultats pour un clustering approché (à condition que $minfr$ soit relativement faible). Cela s'explique par le fait que les clusters candidats ayant une bonne valeur d'*intérêt* ont une fréquence proche de $minfr$ (cf. discussions sur les mesures).

Notons que le nombre maximum de clusters sélectionnés est limité, et dépend de $minfr$ et de M. En effet, nous avons $|\mathcal{T}|$ transactions à classer. Au pire, tous les clusters ne contiennent que $minfr$ transactions, et les clusters sélectionnés ne classent que M transactions. La sélection du premier cluster classe $minfr$ transactions, il reste alors $|\mathcal{T}| - minfr$ transactions à classer. Comme la sélection de chaque nouveau cluster permet de classer au moins M nouvelles transactions, le nombre maximum de clusters est $1 + \lfloor \frac{|\mathcal{T}|-minfr}{M} \rfloor$.

4.3 Expérimentations

Nous avons évalué ECCLAT sur des bases de données bien connues et utilisées dans de multiples travaux : `Mushroom`, `Votes` et `Titanic`. De telles bases de données sont appelées "benchmarks". Nous disposons de connaissances sur ces données, comme par exemple les classes auxquelles appartiennent les transactions. Cela nous permet d'évaluer la capacité d'ECCLAT à retrouver les classes.

Nous avons aussi utilisé notre méthode sur des données médicales lors du Discovery Challenge PKDD 2002, et sur des données provenant de fichiers de trace de serveurs mandataires de France Télécom R&D.

Pour simplifier la présentation des résultats, nous utilisons la notation relative (e.g. pourcentage) pour $minfr$. Pour M, nous utilisons la représentation absolue, mais nous indiquons aussi dans les tableaux la correspondance en notation relative.

4.3.1 Benchmarks

Protocole expérimental

Pour permettre une comparaison avec d'autres algorithmes de clustering, nous allons rechercher des clusterings approchés sur des benchmarks. Les algorithmes utilisés sont *K-Means* [Mac67], *EM* [DLR77], *Bi-Clust* [RR02], *COBWEB* [Fis87], *ARHP* [HKKM97] et Wang et al. [WCL99]. Rappelons qu'ils sont présentés dans le chapitre 3. Certains tableaux et résultats sont tirés d'articles originaux.

K-Means et *EM* nécessitant de fixer le nombre de clusters désirés, nous indiquons alors le nombre de classes présentes dans les données (k=2).

Rappelons que *ARHP* est une des rares méthodes utilisant directement les motifs fréquents pour former des clusters. Pour l'utiliser, nous avons téléchargé le programme *hMETIS*[10] auquel nous avons ajouté des programmes de pré-traitement (dont la recherche des motifs fréquents avec *APRIORI*) et de post-traitement (rangement au mieux des transactions, cf. section 3.3.2) que nous avons développés. Lors de nos expérimentations, le seuil $minfr$ pour la découverte de motifs fréquents est fixé à la même valeur utilisée pour ECCLAT. Il est aussi nécessaire de fixer le nombre de clusters. Comme *ARHP* effectue une partition des items, nous fixons ce nombre de façon à ce que chaque cluster obtenu soit décrit en moyenne par le même nombre moyen d'items que les clusters produits par ECCLAT. Notons qu'une fois terminée l'étape de rangement des transactions, plusieurs clusters peuvent être vides. Ces derniers sont alors supprimés des résultats.

ECCLAT ne produisant pas une partition stricte des transactions, nous ne pouvons pas utiliser les techniques classiques telle que la distance de Hausdorff entre deux partitions [KP77, LD03a] pour les comparaisons. De même, la *F-Measure* [SKK00] n'est pas adaptée car elle favorise les partitions où chaque classe correspond à un seul cluster. Pour comparer les résultats avec ceux d'autres méthodes, nous regardons alors la capacité à former des clusters purs, c'est-à-dire ne contenant que des transactions d'une même classe.

Une façon courante de mesurer l'impureté est d'utiliser une fonction d'entropie [Kul67]. Soit $P = (p_1, \ldots, p_j)$ la distribution de fréquences des classes d'un cluster, l'entropie de P noté $\varphi(P)$ est

$$\varphi(P) = -\sum_{i=1}^{j} p_i \times log\ p_i.$$

Plus $\varphi(P)$ est faible, plus le cluster est pur. $\varphi(P)$ est égal à 0 si et seulement si il existe i avec $p_i = 1$. Il n'y a alors qu'une seule classe pour les transactions du cluster.

L'entropie d'un ensemble $\mathcal{C}$ de clusters, est évaluée avec la formule suivante :

$$E_{\mathcal{C}} = \sum_{i=1}^{k} \frac{s_i \times \varphi(C_i)}{n}$$

où s_i est le nombre de transactions du cluster C_i, k le nombre de clusters, et n le nombre total de transactions. Plus $E_{\mathcal{C}}$ est faible, plus l'ensemble de clusters est pur.

Données utilisées

Pour les objectifs de l'expérience, les valeurs de classe dans les jeux de données sont ignorées, mais elles seront utilisées par la suite pour l'évaluation des résultats. Notons que nous n'effectuons aucun traitement particulier pour les valeurs manquantes. Si une valeur est manquante, il n'y aura pas d'item correspondant.

Les données utilisées étant discrètes, nous pouvons facilement les transformer en créant un item pour chaque attribut-valeur. Par exemple, pour un attribut `habitat` pouvant prendre les valeurs *bois*, *herbages*, *prairies*, *sentiers*, *agglomérations*, *déserts*, *feuillages*, nous formons 7 items pour cet attribut : *habitat=bois*, etc.

[10] *http://www-users.cs.umn.edu/~karypis/metis/hmetis/index.html*

La base de données bien connue `Mushroom`[11] contient 8124 transactions correspondant à des champignons (23 espèces des familles *Agaricus* et *Lepiota*). Les champignons sont identifiés comme `comestibles` ou `vénéneux`. Nous savons qu'il y a 4208 `comestibles` et 3916 `vénéneux`. Chaque transaction est décrite par 22 attributs donnant 116 items.

La base de données `Votes`[11] correspond aux résultats d'un questionnaire réalisé en 1984. 435 représentants du congrès américains ont répondu à ce questionnaire. Les données sont donc constituées de 435 transactions décrites par 16 attributs correspondant aux 16 questions. Les réponses étant binaires (oui/non), nous avons 32 items. Les représentants du congrès sont `démocrates` ou `républicains`, d'où deux classes. Nous en avons respectivement 267 et 168. Comme nous le verrons dans les résultats, cette base de données est "difficile" du fait qu'il y ait 288 valeurs manquantes, et de par sa nature. En effet, au sein d'un parti, il peut y avoir des divergences sur certains points comme par exemple l'éducation ou la politique militaire.

Les données de la base `Titanic`[12] ont été collectées lors de l'enquête effectuée par le British Board of Trade sur le naufrage de ce bâteau. 2201 personnes sont répertoriées dans ces données. Elles sont décrites par 3 attributs : classe (équipage ou $1^{ère}$ classe ou 2^{nde} classe ou $3^{ème}$ classe), âge (adulte ou enfant), et le sexe. Nous avons donc au total 2201 transactions et 8 items. Nous disposons aussi d'une valeur indiquant si la personne a survécu ou non. Au total, il y a 711 `survivants` et 1490 `naufragés`.

Résultats

Les tableaux 4.7 et 4.8 présentent, respectivement pour `Mushroom` et `Votes`, le nombre de clusters sélectionnés selon la valeur de $minfr$, avec M égal à 1. Les tableaux indiquent aussi le nombre maximum de clusters qui aurait pu être trouvé (pire des cas, cf. section 4.2.2). Le nombre de motifs fermés fréquents pour `Titanic` étant faible, le tableau correspondant n'est pas indiqué.

$minfr$ (%)	$\lvert\mathcal{FC}\rvert$	Nombre de clusters	ratio (%)	Nb. max. clusters
50	43	10	23,25	4063
40	112	13	11,6	4875
30	300	21	7	5687
20	887	30	3,38	6500
10	3580	61	1,7	7312
5	9737	144	1,47	7719

TAB. 4.7 – Nombre de clusters sélectionnés selon la valeur de $minfr$ (`Mushroom`, $M = 1$)

Nous constatons que le nombre de clusters est très inférieur au nombre de motifs fermés fréquents. Lorsque $minfr$ décroît, nous vérifions bien que le nombre de clusters sélectionnés n'augmente pas autant que le nombre de motifs fermés fréquents (autrement dit le ratio diminue). De plus, nous observons que ce nombre est très inférieur au nombre

[11] *http://www.ics.uci.edu/~mlearn/MLRepository.html*
[12] *http://www.amstat.org/publications/jse/jse_data_archive.html*

$minfr$ (%)	$\|\mathcal{FC}\|$	Nombre de clusters	ratio (%)	Nb. max. clusters
30	658	56 + 1 reste	8,5	305
20	7566	151 + 1 reste	2	349
15	15048	204 + 1 reste	1,5	370
10	37230	241 + 1 reste	0,6	392
5	83905	273 + 1 reste	0,3	414
2	156515	306 + 1 reste	0,1	427

TAB. 4.8 – Nombre de clusters sélectionnés selon la valeur de $minfr$ (`Votes`, $M = 1$)

maximum de clusters obtenu dans le pire des cas. Ce point est important, car un nombre de clusters trop important rendrait difficile leur exploitation.

Remarquons que pour `Votes`, nous obtenons un reste à chaque fois. Ce cluster "reste" ne contient qu'une seule transaction correspondant à un même `républicain`.

Les tableaux 4.9, 4.10 et 4.11 donnent, respectivement pour `Mushroom`, `Votes` et `Titanic`, le nombre de clusters sélectionnés, le nombre de transactions dans le cluster "reste", et le recouvrement moyen (i.e. nombre moyen de transactions communes entre deux clusters), en fonction de la valeur de M. Le nombre de clusters indiqué ne comptabilise pas l'éventuel reste.

Le seuil minimum de fréquence est fixé à 5% pour `Mushroom` et `Votes`. Par contre, étant donné que pour *Titanic* le nombre d'items est très faible, nous utilisons un seuil très bas. Avec $minfr$ égal à 1% (22 transactions), nous obtenons 33 motifs fermés fréquents.

M	Nombre de clusters	Taille du reste	Recouvrement moyen (nb. transactions)	Nb. moyen de transactions par cluster
1 (0,01%)	144	0	40	506,5
101 (1,25%)	42	60	29	494,6
203 (2,5%)	27	44	28	576,7
304 (3,74%)	20	60	15	563,8
406 (5%)	16	340	7	544,5
507 (6,2%)	12	442	16	749,5
609 (7,5%)	10	572	16	844,8

TAB. 4.9 – Résultats d'ECCLAT en fonction de la valeur de M (`Mushroom`, $minfr = 5\%$)

Le choix de $minfr$ correspond à un compromis entre le nombre de motifs fermés fréquents obtenus, le temps nécessaire à leur extraction, et la pertinence des motifs produits. En effet, nous cherchons à obtenir un réservoir relativement important de clusters candidats. Néanmoins, un seuil minimum de fréquence trop bas risque d'entraîner une impossibilité de calcul. De plus, les motifs fermés de très faible fréquence ne représentent pas des clusters pertinents (cf. section 4.1).

Notons que le temps d'exécution d'ECCLAT correspond environ au temps nécessaire à l'extraction des motifs fermés fréquents. Le calcul de la *concentration* et la sélection des

clusters sont très faibles comparés à la découverte des motifs. En utilisant *MVMINER* (cf. annexe F) modifié pour calculer l'*homogénéité*, l'extraction des motifs prend 42 secondes pour `Mushroom`, 8 secondes pour `Votes`, et moins d'une seconde pour `Titanic`, sur un Pentium III 1GHz avec 512Mo RAM.

M	Nombre de clusters	Taille du reste	Recouvrement moyen (nb. transactions)	Nb. moyen de transactions par cluster
1 (0,23%)	273	1	4,7	30,2
11 (2,5%)	33	10	6,7	50,7
22 (5%)	17	19	16,6	85,7
43 (10%)	8	57	21,3	108
65 (15%)	6	23	29,4	144,6

TAB. 4.10 – Résultats d'ECCLAT en fonction de la valeur de M (`Votes`, $minfr = 5\%$)

M	Nombre de clusters	Taille du reste	Recouvrement moyen (nb. transactions)	Nb. moyen de transactions par cluster
1 (0,04%)	13	0	1,32	177,2
5 (0,22%)	12	1	0,75	188,2
10 (0,45%)	11	6	0	199,5
20 (0,9%)	11	6	0	199,5
22 (1%)	11	6	0	199,5
25 (1,13%)	10	23	1,43	225,7

TAB. 4.11 – Résultats d'ECCLAT en fonction de la valeur de M (`Titanic`, $minfr = 1\%$)

Pour `Mushroom` et `Titanic`, avec M=1, toutes les transactions sont classées (car $minfr$ est assez faible), mais plusieurs chevauchements apparaissent. Nous obtenons un chevauchement minimum pour $M = minfr$. Cela paraît donc être un bon choix pour obtenir un clustering approché.

Comme nous l'avons dit précédemment, `Votes` est une base de données difficile. Il n'y a pas de réelle partition des transactions dans les données. De ce fait, ECCLAT n'est pas capable d'en former une artificiellement. ECCLAT ne peut que détecter une éventuelle partition. Avec M=1, toutes les transactions sauf le *républicain* évoqué précédemment, sont classées. Concernant le recouvrement moyen, il est logique de trouver une faible valeur pour M=1 car le nombre élevé de clusters fait baisser la moyenne. Plus M augmente, plus le nombre de clusters sélectionnés baisse, comme il y a toujours autant de chevauchement entre les clusters, la moyenne augmente.

Les tableaux 4.12, 4.13 et 4.14 présentent les résultats d'ECCLAT avec M égal à $minfr$. Pour chaque cluster, nous présentons le nombre d'items le décrivant, et la répartition de ses transactions selon les classes.

Avec `Mushroom`, un léger chevauchement est induit par les clusters 14 et 16. Les 14 autres clusters sont disjoints. Nous remarquons que nous avons un nombre relativement

cluster	#vénén.	#comest.	#items
1	0	432	16
2	0	432	16
3	0	432	16
4	0	432	16
5	648	0	15
6	648	0	15
7	432	0	14
8	432	0	14
9	432	0	14
10	432	0	14
11	0	768	14
12	0	512	14
13	352	96	12
14	288	896	10
15	0	416	9
16	72	560	5
reste	180	160	-

TAB. 4.12 – Clustering approché avec ECCLAT (`Mushroom`, $minfr$=5%, M=406 (5%))

cluster	#démocr.	#républi.	#items
1	3	24	14
2	5	29	14
3	20	2	13
4	43	4	12
5	41	2	11
6	7	44	10
7	25	6	10
8	74	8	8
9	19	8	8
10	107	17	7
11	19	70	7
12	27	3	5
13	124	20	5
14	34	103	5
15	173	25	3
16	58	106	3
17	165	42	1
reste	11	8	-

TAB. 4.13 – Clustering approché avec ECCLAT (`Votes`, $minfr$=5%, M=22 (5%))

important de clusters purs, aussi bien pour les `vénéneux` que les `comestibles`. En effet, 7 clusters purs correspondent à des champignons `comestibles`, et 6 autres à des champignons `vénéneux`. Notons que les motifs des clusters ne sont pas tous disjoints. Le chevauchement moyen est de 7 items. Ce recouvrement d'items apporte des explications supplémentaires sur les regroupements et peut être utilisé pour mieux caractériser les classes. Par exemple, prenons deux clusters de champignons disjoints au niveau de leurs transactions. Ces dernières sont toutes de la classe `comestibles`. Les motifs de ces clusters donnent deux explications sur le regroupement de champignons `comestibles`. De ce fait, nous constatons aussi un regroupement plus fin au sein de la classe `comestibles`. Si ces deux motifs ont des items communs, cela peut permettre de déduire des rôles plus ou moins importants de certains items. En effet, les items communs ont peut-être un rôle moindre que les autres.

Pour `Votes`, nous obtenons 17 clusters et un reste. Aucun cluster pur n'a été trouvé. Néanmoins, nous remarquons que nous pouvons étiqueter chaque cluster par un parti car à chaque fois il est possible d'identifier une forte majorité. Par exemple, le cluster 5 est composé de 41 `démocrates` et de 2 `républicains`, nous pouvons donc admettre que le motif décrivant ce cluster permet de caractériser les `démocrates`.

Nous pouvons formuler les mémes remarques concernant les clusters obtenus avec `Titanic`. Le cluster n°8 est composé à 91,6% de `naufragés`. Ce cluster peut donc servir à caractériser cette classe. Ainsi nous pouvons déduire que la majorité des hommes adultes en $2^{ème}$ classe n'ont pas survécu. Le cluster n°2 composé à 77,7% de `naufragés` indique

cluster	#survivants	#naufragés	#items
1	20	3	3
2	192	670	3
3	14	17	3
4	13	35	3
5	76	89	3
6	80	13	3
7	75	387	3
8	14	154	3
9	140	4	3
10	57	118	3
11	24	0	2
reste	6	0	-

TAB. 4.14 – Clustering approché avec ECCLAT (`Titanic`, $minfr$=1%, M=22 (1%))

	`Mushroom`	`Votes`	`Titanic`
ECCLAT	**0,137**	1,509	**0,476**
K-Means	0,471	0,366	0,618
EM	0,277	0,338	0,585
Bi-Clust	0,355	0,352	0,577
COBWEB	0,328	**0,334**	0,504
ARHP	0,533	0,335	0,550
Wang et al.	**0,120**	-	-

TAB. 4.15 – Comparaison des valeurs d'entropie obtenues par les algorithmes

que la majorité des hommes adultes de l'équipage ont péri. Le cluster n°7 indique que 83,7% des hommes adultes en $3^{ème}$ classe n'ont pas survécu. Le cluster n°9 composé à 97,2% de `survivants`, montre que la quasi-totalité des femmes adultes en $1^{ère}$ classe ont survécu. Le cluster n°1 montre que la quasi-totalité des femmes adultes de l'équipage ont survécu. Le cluster n°11, qui est le seul cluster pur, montre que tous les enfants en 2^{nde} classe ont survécu.

Comparaisons avec d'autres méthodes

Nous avons comparé nos résultats avec ceux produits sur nos trois jeux de données par les algorithmes *K-Means*, *EM*, *Bi-Clust*, *COBWEB* et *ARHP*. Ne disposant pas d'une implémentation de la méthode de Wang et al., nous ne présentons que les résultats sur `Mushroom` (disponibles dans l'article présentant l'algorithme).

Le tableau 4.15 récapitule les valeurs d'entropie calculées avec les ensembles de clusters produits par les différents algorithmes. Les tableaux de 4.16 à 4.31 présentent en détail la distribution des classes de chaque cluster pour chacun des algorithmes et jeux de données.

Concernant `Mushroom`, ECCLAT est en deuxième position derrière la méthode de Wang et al. Cette dernière produit 14 clusters (cf. tableau 4.21) lesquels sont obtenus hiérar-

cluster	#vénéneux	#comestibles
1	3100	648
2	816	3560

TAB. 4.16 – Exécution de *K-Means* sur `Mushroom`, k=2

cluster	#vénéneux	#comestibles
1	3086	0
2	830	4208

TAB. 4.17 – Exécution de *EM* sur `Mushroom`, k=2

cluster	#vénéneux	#comestibles
1	816	3774
2	1296	0
3	1768	242
4	36	0
5	0	192

TAB. 4.18 – Exécution de *Bi-Clust* sur `Mushroom`

cluster	#vénéneux	#comestibles
1	1566	0
2	1076	4208
3	1274	0

TAB. 4.19 – Exécution de *COBWEB* sur `Mushroom`

chiquement en scindant les clusters "reste" et en utilisant plusieurs valeurs de $minfr$. Nous obtenons ici un type de résultat qui se rapproche du notre. Les clusters sont assez nombreux, et dans certains cas, ils sont purs. Néanmoins, cette méthode a des difficultés à identifier différents cas de champignons `vénéneux`. Il y a seulement deux clusters qui s'y rapportent clairement (n°8 et n°9). Les résultats obtenus par notre approche sont plus discriminants (cf. tableau 4.12). De plus, nous n'utilisons pas plusieurs seuils pour $minfr$, et nous obtenons directement la description des clusters, ce qui n'est pas le cas avec cette méthode.

Les tableaux 4.18 et 4.19 représentent les résultats obtenus avec *Bi-Clust* et *COBWEB*. Nous obtenons ici plusieurs clusters purs pour les `vénéneux`. Mais nous remarquons que ces méthodes ont des difficultés à identifier clairement les champignons `comestibles`. Seul *Bi-Clust* trouve un petit cluster ne contenant que des champignons de cette classe. Un avantage de ces méthodes est que le nombre de clusters n'est pas fixé par l'utilisateur. Mais nous remarquons que le nombre de clusters est assez faible. Il aurait été intéressant d'obtenir des résultats plus "fins" montrant différents clusters au sein d'une même classe. Il est vrai qu'avec ECCLAT nous n'obtenons pas de partition des champignons, et de ce fait, le nombre de cluster est plus élevé. Nous avons tout de même 14 clusters disjoints. ECCLAT permet donc d'obtenir des résultats plus fins, proposant ainsi plusieurs approches pour interpréter et caractériser les classes de champignons.

Avec *ARHP*, nous obtenons le tableau 4.20. Le nombre de clusters est fixé à 9. Nous remarquons que le nombre réel de clusters résultats est 7. Nous constatons dans ces résultats, qu'il n'y pas de clusters purs avec un nombre important de transactions. Il y a tout de même quelques clusters qui ressortent comme le numéro 5 avec une majorité de `vénéneux` et le numéro 7 avec une "petite" majorité de `comestibles`.

Avec `Votes`, aucun des clusters obtenus avec les différentes méthodes ne sont purs. De plus, les clusters sensés caractériser les `républicains` sont assez mitigés. Par exemple, avec *K-Means* nous avons 45 `démocrates` et 154 `républicains` dans un même cluster (n°2). Les démocrates représentent 22,6% du cluster. Par contre, les clusters caractérisant

cluster	#vénéneux	#comestibles	#items
1	0	4	11
2	22	0	11
3	7	0	13
4	1	0	14
5	1556	28	14
6	2272	3824	14
7	58	342	14
Les 2 autres clusters sont vides			

TAB. 4.20 – Exécution de *ARHP* sur `Mushroom`, $minfr$=5%, k=9

cluster	#vénéneux	#comestibles
1	0	94
2	0	13
3	0	6
4	26	682
5	30	2631
6	37	121
7	61	69
8	287	0
9	3388	61
10	77	372
11	0	9
12	10	19
13	0	21
14	0	110

TAB. 4.21 – Exécution de la méthode de Wang et al. sur `Mushroom`

cluster	#démocrates	#républicains
1	222	14
2	45	154

TAB. 4.22 – Exécution de *K-Means* sur `Votes`, k=2

cluster	#démocrates	#républicains
1	221	9
2	46	159

TAB. 4.23 – Exécution de *EM* sur `Votes`, k=2

cluster	#démocrates	#républicains	#items
1	55	1	7
2	165	9	9
3	45	123	9
4	2	34	7
reste	0	1	-

TAB. 4.24 – Exécution de *ARHP* sur `Votes`, $minfr$=5%, k=4

cluster	#démocrates	#républicains
1	228	15
2	39	153

TAB. 4.25 – Exécution de *Bi-Clust* sur `Votes`

cluster	#démocrates	#républicains
1	220	8
2	47	160

TAB. 4.26 – Exécution de *COBWEB* sur `Votes`

les `démocrates` ressortent beaucoup plus. Par exemple, avec *Bi-Clust*, le cluster n°1 correspondant aux `démocrates` ne contient que 6% de `républicains`. Pour *COBWEB*, les `républicains` ne représentent que 3,5% dans le cluster (n°1) des `démocrates`. Remarquons que parmi les clusters produits par *ARHP*, le cluster n°4 ne contient que 5,5% de `démocrates`. Cependant, le nombre d'éléments de ce cluster est modeste (36 transactions).

Dans les clusters produits par ECCLAT (cf. tableau 4.13), les clusters caractérisant les `républicains` sont aussi plus mitigés que ceux représentant les `démocrates`. La proportion de `démocrates` dans les clusters représentant les `républicains` varie entre 11% pour le cluster 1, et 35,6% pour le cluster 16. En ce qui concerne la proportion des `républicains` dans les clusters `démocrates`, elle varie entre 4,6% pour le cluster 5 et 29,6% pour le cluster 9. Au final, bien qu'ECCLAT soit classé dernier selon la pureté globale, nous avons tout de même 5 clusters caractérisant clairement les républicains, et 10 caractérisant les démocrates. Notons que les clusters 9 et 16 ne sont pas considérés comme significatifs.

cluster	#survivants	#naufragés
1	499	817
2	212	673

TAB. 4.27 – Exécution de *K-Means* sur `Titanic`, k=2

cluster	#survivants	#naufragés
1	424	430
2	287	1060

TAB. 4.28 – Exécution de *EM* sur `Titanic`, k=2

cluster	#survivants	#naufragés
1	267	1060
2	444	430

TAB. 4.29 – Exécution de *Bi-Clust* sur `Titanic`

cluster	#survivants	#naufragés
1	374	176
2	42	0
3	295	1314

TAB. 4.30 – Exécution de *COBWEB* sur `Titanic`

cluster	#survivants	#naufragés	#items
1	118	30	3
2	390	1338	4
3	203	122	1

TAB. 4.31 – Exécution de *ARHP* sur `Titanic`, $minfr$=1%, k=3

Pour `Titanic`, nous remarquons que les partitions obtenues ne permettent pas de différentier clairement les `survivants` des `naufragés`. *COBWEB* (cf. tableau 4.30) trouve un cluster pur (cluster n°2) pour des `survivants`, mais ce cluster est assez petit. De même, le cluster n°11 produit par ECCLAT (cf. tableau 4.14) ne contient que des `survivants`. Notons qu'ici, ECCLAT obtient le meilleur score de pureté devant *COBWEB* et *ARHP*.

Si nous regardons les résultats produits par *ARHP* (cf. tableau 4.31). Le cluster n°1 correspond aux femmes et enfants en 2^{nde} classe, et ces derniers ont en majorité survécu. Le cluster n°2 correspond aux adultes, hommes de l'équipage ou en 2^{nde} classe. Ils ont

majoritairement péri lors du naufrage. Le cluster n°3 correspond aux personnes en $1^{ère}$ classe. Notons que ce cluster est assez mitigé et qu'il est difficile d'en tirer des conclusions. Nous retrouvons des conclusions évoquées précédemment avec les résultats d'ECCLAT. Néanmoins, il y a beaucoup moins d'informations utilisables. Cela est dû notamment au partitionnement des items effectué par *ARHP*.

Bilan

Ces expérimentations sur des benchmarks ont montré qu'ECCLAT est capable de trouver des clusters purs ou assez significatifs. En effet, ECCLAT arrive en deuxième position pour `Mushroom`, et obtient le meilleur score de pureté pour `Titanic`. En ce qui concerne `Votes` qui est une base difficile, le pureté obtenue est la plus faible, néanmoins ECCLAT fait ressortir localement des clusters significatifs sur chacune des classes permettant ainsi de proposer différentes caractérisations.

Nous avons clairement vu sur `Titanic` que les clusters fournis par ECCLAT sont facilement interprétables et apportent plus d'informations que les méthodes produisant une partition.

Nous pouvons aussi remarquer que s'il n'y a pas de réelle partition des transactions dans les données, alors ECCLAT n'est pas capable d'en former une artificiellement. Ce qui est le cas avec la base de données `Votes`. ECCLAT ne peut que détecter une éventuelle partition cachée dans les données, car elle est basée sur la recherche de motifs.

4.3.2 Données réelles

Discovery challenge PKDD 2002

Nous avons utilisé ECCLAT sur des données médicales lors du Discovery Challenge de la conférence PKDD (Principles and Practice of Knowledge Discovery in Databases) en 2002 [CD02]. Les bases de données ont été fournies[13] par l'hopital universitaire de Chiba (Japon). Elles contiennent des informations sur des patients atteints d'hépatites.

L'hépatite B et C sont des virus infectieux qui affectent le foie d'un patient. Ces infections sont importantes car elles sont un risque potentiel de développement de cirrhose du foie. Les indicateurs de telles maladies sont les fibroses hépatiques. Par exemple, la cirrhose du foie est caractérisée comme le stade terminal d'une fibrose du foie. Le mécanisme détaillé de la progression de la maladie est inconnu à ce jour.

Objectifs

Notre contribution au Discovery Challenge a été d'essayer de mieux estimer le niveau de la fibrose du foie à partir de résultats d'examens médicaux. Cette étape est actuellement déterminée par biopsie. L'idée est de substituer les biopsies par les examens de laboratoire, car une biopsie est invasif pour les patients. Nous voulions aussi montrer le potentiel de la découverte de clusters significatifs dans le domaine médical.

[13] *http://lisp.vse.cz/challenge/ecmlpkdd*2002/

Dans les expérimentations, nous formons des clusters regroupant des patients décrits par les résultats des examens effectués. Les caractéristiques de biopsie ne sont bien sûr pas utilisées puisqu'on cherche à remplacer cet examen. Dans le but de découvrir des combinaisons de résultats d'examens associés à un stade de la fibrose, nous estimons la pureté des clusters découverts en fonction du stade de la fibrose (5 différents stades).

Les données

Les données pour chaque patient sont complexes. Elles sont composées initialement de 6 tables. Nous avons par exemple la table `biopsy` contenant 960 enregistrements, la table `out-hospital_examinations` qui reporte les résultats d'examens hors hôpital et qui contient 31040 enregistrements, ou encore la table `in-hospital_examinations` qui synthétise les résultats d'examens réalisés à l'hôpital et qui contient plus de 1,5 millions d'enregistrements.

Nous avons pré-traité ces données de manière à obtenir deux fichiers correspondant à deux bases de données transactionnelles. Pour découvrir les relations entre le niveau de fibrose du foie et les examens médicaux, chaque transaction correspond à un patient. Une transaction regroupent des résultats d'examens.

Nous avons deux bases notées `bioexaout` et `bioexain` (cf. tableau 4.32), qui correspondent respectivement aux tables `out-hospital_examinations` et `in-hospital_examinations`. `bioexaout` contient 59 transactions et 17 items. `bioexain` est composée de 118 transactions et de 308 items. Pour chaque examen, autant d'items sont créés qu'il y a de types de résultats. Par exemple, l'item `HBC-AB+` signifie que le résultat de l'examen "Hepatitis B virus Code AntiBody" est positif. Les abréviations utilisées pour les examens sont décrites en annexe D.

	Nombre de patients	Nombre d'examens réalisés	Nombre d'examens différents	Nombre d'items
`bioexaout`	59	1122	11	17
`bioexain`	118	243653	172	308

TAB. 4.32 – Données construites à partir des tables `bioexaout` et `bioexain`

Résultats

Le tableau 4.33 présente les résultats obtenus avec ECCLAT sur `bioexaout`. Nous indiquons le nombre de motifs fermés fréquents, le nombre de clusters, la taille du cluster "reste" (nombre de transactions), et la taille des motifs fermés fréquents des clusters, en fonction de $minfr$ et de M. MIS signifie "MInimal Size of cluster descriptions". Cela correspond à la taille du plus petit motif des clusters. La taille d'un motif est bien entendu le nombre d'items. De la même façon, MAS signifie "MAximal Size of cluster descriptions", et correspond à la taille du plus important motif. AVS ("AVerage Size of cluster description") est la moyenne des tailles des motifs. Le tableau 4.34 présente les résultats obtenus sur `bioexain`.

$minfr$	$\|\mathcal{FC}\|$	M	Nombre de clusters	Taille du reste	MIS	MAS	AVS
3	93	1	23	0	1	6	3,52
		3	14	2	1	6	3,14
5	65	1	21	1	1	5	3,09
		5	9	2	1	5	2,66
7	44	1	15	1	1	5	2,6
		7	6	9	1	5	2,33

TAB. 4.33 – Résultats d'ECCLAT sur `bioexaout`

$minfr$	$\|\mathcal{FC}\|$	M	Nombre de clusters	Taille du reste	MIS	MAS	AVS
53	462041	1	34	0	1	20	9,55
		10	7	0	1	20	8,35
59	217952	1	28	0	11	19	16,92
		10	7	4	14	19	14,57
65	99968	1	23	0	10	17	15,74
		10	6	2	7	17	12
71	43396	1	19	0	9	16	14,52
		10	5	9	13	16	12,2

TAB. 4.34 – Résultats d'ECCLAT sur `bioexain`

Nous donnons à présent quelques clusters parmi ceux qui maximisent la pureté en fonction du stade de la fibrose. Dans les exemples donnés par la suite, chaque ligne correspond à un cluster. Nous donnons sa description sous forme d'une conjonction d'items (résultats d'examens), puis la fréquence (nombre de patients) ainsi que la distribution selon le stade de la fibrose. Les différents niveaux de fibrose sont notés `F0`, ..., `F4`. Ce dernier est le stade le plus sévère.

Avec la base `bioexaout` et $minfr$ égal à 3, nous pouvons produire des clusters purs mais avec peu de patients. Par exemple, le cluster suivant contient 4 patients, tous du stade `F3`.

```
HBC-AB+,HBS-AG+,HBE-AG-,HBE-AB+ : 4 : F0=0.0 F1=0.0 F2=0.0 F3=100.0 F4=0.0
```

Plusieurs clusters semblent être associés avec des stades sévères de fibrose (`F3` et `F4`). Par exemple, nous avons :

```
HBS-AB+,HBE-AG-,HBE-AB+ : 5 : F0=0.0 F1=0.0 F2=0.0 F3=25.0 F4=75.0
HBC-AB+,HBE-AG-,HBE-AB+ : 8 : F0=0.0 F1=14.29 F2=0.0 F3=57.15 F4=28.57
HBE-AB+ : 13 : F0=0.0 F1=16.67 F2=16.67 F3=41.67 F4=25.0
```

Les clusters présentés au-dessus semblent indiquer que `HBE-AB+` joue un rôle important en ce qui concerne les stades sévères de la maladie. Remarquons que ces clusters sont décrits seulement par 5 items : `HBC-AB+`, `HBE-AB+`, `HBE-AG-`, `HBS-AB+` et `HBS-AG+`.

D'autres clusters présentent des caractéristiques sur les premiers stades de fibrose (F1 et F2) :

HBE-AB-,HCV5'NCRRT-PCR+,HBE-AG- : 7 : F0=0.0 F1=66.67 F2=33.33 F3=0.0 F4=0.0
HBE-AB-,HBE-AG+ : 13 : F0=0.0 F1=58.33 F2=41.67 F3=0.0 F4=0.0
HBE-AB-,HCV5'NCRRT-PCR+,HBE-AG-,HBS-AG- : 6 : F0=0.0 F1=80.0 F2=20.0 F3=0.0 F4=0.0
HBE-AG+ : 16 : F0=0.0 F1=53.33 F2=40.0 F3=6.67 F4=0.0

Il serait très intéressant de réaliser des analyses statistiques de façon à confirmer (ou non) ces observations. Remarquons que nous n'avons pas énuméré tous les clusters obtenus. Les caractéristiques principales des clusters obtenus sont indiqués dans les tableaux 4.33 et 4.34.

Etant donné que la base bioexain contient un nombre beaucoup plus important d'items et que les données sont plus corrélées, le calcul des clusters nécessite une valeur plus élevée pour $minfr$. Cela tend à produire des clusters qui représentent en général plusieurs stades de fibrose. Les groupes ne sont pas assez "fins". Les clusters sont décrits avec un nombre plus important d'items que pour bioexaout. Nous présentons quelques exemples :

ALB=,CL=,K=,oudan+,G-GTP= : 54 : F0=0.0 F1=56.60 F2=30.19 F3=7.55 F4=5.66

ALB=,CL=,LAP=,nyuubi=,oudan+,U-BIL=,U-GLU=,U-KET=,U-PH+,U-PRO=,U-RBC=,U-SG+,G-GTP= : 53 : F0=0.0 F1=55.77 F2=30.77 F3=9.61 F4=3.85

ALB=,CRP=,K=,oudan+,U-BIL=,U-GLU=,U-PH+,U-PRO=,U-RBC=,U-SG+,F-A2.GL= : 53 : F0=1.92 F1=55.77 F2=26.92 F3=15.38 F4=0.0

CHE=,CL=,D-BIL=,HCV-AB=,I-BIL=,K=,LAP=,oudan+,T-BIL=,UA=,UN=,NA= : 56 : F0=1.79 F1=41.07 F2=35.71 F3=14.29 F4=7.14

D-BIL=,nyuubi=,oudan+,T-BIL=,U-BIL=,U-GLU=,U-KET=,U-PH+,U-PRO=,U-RBC=, U-SG+,G-GTP= : 56 : F0=0.0 F1=45.45 F2=40.0 F3=12.73 F4=1.82

Notons que beaucoup des items concernent des valeurs normales d'examens. Dans un cluster, il y a toujours quelques examens avec des valeurs anormales. Sur l'expérimentation avec $minfr$ égal à 53 et M égal à 1, nous obtenons 34 clusters, et seulement 3 valeurs anormales : oudan+, U-PH+ et U-SG+. Nous ne savons pas si ces examens ont un rôle spécial ou s'ils sont inclus dans un cluster parce qu'ils sont communs. Notons qu'il serait utile de réaliser plus expériences sur bioexain avec des valeurs plus faibles de $minfr$.

Bilan

En résumé, sur les données provenant des résultats d'examens hors hôpital, notre travail suggère un rôle particulier de plusieurs examens : HBC-AB+, HBE-AB+ et HBE-AG-

qui semblent être associés aux stades sévères de la maladie. Sur les données collectées à l'hôpital, à l'exception de trois valeurs anormales d'examens (`oudan+`, `U-PH+` and `U-SG+`), les autres examens sélectionnés ont une valeur normale, ce qui rend difficile l'interprétation médicale. A cause de valeurs manquantes sur cet attribut, nous avons dû retirer beaucoup d'informations dans la table `biopsy`. Il serait très intéressant et utile d'avoir plus de biopsies avec une valeur connue de stade de fibrose. Notons que plus de données de biopsie n'entraineraient pas un temps de calcul supplémentaire, car le coût algorithmique de notre méthode est principalement dû au nombre d'items.

Fichiers de trace de serveurs mandataires

Dans cette expérimentation, nous utilisons plusieurs fichiers de trace d'un serveur mandataire de France Télécom R&D pour former des communautés d'intérêt.

Objectifs

Nous recherchons des clusters d'utilisateurs en fonction des intérêts communs par l'intermédiaire des termes caractérisant les documents qu'ils ont consultés. Le but de cette expérience est d'évaluer le nombre de clusters et les recouvrements produits sur de telles données. Ces recouvrements sont très utiles et traduisent les divers centres d'intérêt des utilisateurs.

Les données

Les données contiennent 136 transactions et 17270 items. Les items sont des termes des pages HTML consultées par 136 utilisateurs du serveur mandataire, sur une période d'un mois. 18162 pages ont été consultées. Pour chaque page, nous avons extrait au plus 10 termes avec un extracteur basé sur la fréquence des mots significatifs, développé à France Télécom R&D et utilisé pour différentes études comme la variabilité des profils thématiques des utilisateurs [LFL00]. La collecte et le pré-traitement des données sont détaillées en annexe E.

Remarquons le nombre relativement faible d'items pour l'ensemble des documents, ce qui traduit que de nombreux documents portent sur des sujets proches.

Les données manipulées étant confidentielles car relatives aux salariés de France Télécom R&D, nous ne divulguons pas d'information sur leur contenu (URL, termes, ...).

Résultats

Le tableau 4.35 représente le nombre de clusters sélectionnés avec $M = 1$ en fonction de $minfr$. Le nombre maximum de clusters dans le pire des cas, est aussi indiqué. Nous constatons que le nombre de clusters est bien inférieur au nombre de motifs fermés fréquents et au nombre limite de clusters. Par exemple, pour $minfr$ égal à 40%, 16 clusters ont été retenus parmi 21507 motifs fermés fréquents. Dans le pire des cas, nous aurions obtenu 82 clusters. Ceci montre que les motifs fermés fréquents sélectionnés donnent une bonne représentation des utilisateurs, en terme de quantité.

Le tableau 4.36 indique le nombre de clusters sélectionnés, le nombre de transactions

$minfr$ (%)	$\|\mathcal{FC}\|$	Nombre de clusters	Nb. max. clusters
60	84	8	55
55	295	8	62
50	981	10	69
45	4482	13	75
40	21507	16	82
35	109237	25	89

TAB. 4.35 – Nombre de clusters sélectionnés en fonction de $minfr$ (trace d'un serveur mandataire, $M = 1$)

M	Nombre de clusters	Taille du reste	Recouvrement moyen (nb. transactions)	Nb. moyen de transactions par cluster
1 (0,7%)	16	0	50,5	71,3
3 (2,2%)	8	3	40,7	77
7 (5,1%)	4	14	31,8	79,7
10 (7,3%)	3	21	30,8	84,6
13 (9,5%)	3	12	45,3	104
27 (20%)	2	14	28,3	103,5
40 (30%)	1	45	0	91
54 (40%)	1	45	0	91

TAB. 4.36 – Résultats d'ECCLAT en fonction de M (trace d'un serveur mandataire, $minfr = 40\%$)

dans le cluster "reste", le recouvrement moyen, et le nombre moyen de transactions par cluster, en fonction de la valeur de M ($minfr = 40\%$). Toutes les transactions sont classées dans un cluster avec $M = 1$. Ce point est important pour utiliser les clusters dans de futures applications de Web Mining. De cette façon, aucun utilisateur ne sera mis à l'écart. Nous avons donc peu d'intérêts à utiliser une valeur élevée pour M. Remarquons que pour M égal à 40 ou 54, nous obtenons un seul cluster et un reste, cela s'explique par le fait que $minfr$ est trop élevé pour obtenir des résultats assez fins.

Concernant le recouvrement moyen, avec M égal à 1, nous avons en moyenne 50 utilisateurs communs entre deux clusters. Etant donné que la taille moyenne des clusters est de 71, cela traduit donc un fort recouvrement, tout comme les items parmi des documents consultés. Les clusters sont tout de même différenciés par 21 transactions en moyenne, ce qui est assez suffisant pour être significatif et exploitable.

Ce chevauchement est souhaitable dans plusieurs applications de Web Mining. Par exemple, un utilisateur peut être inclus dans un cluster correspondant à la *pêche* et à la *chasse* et aussi dans un cluster caractérisé par le *vélo* et la *montagne*. Le chevauchement permet alors de retrouver un tel utilisateur à partir de différentes requêtes. Notons qu'un algorithme de "pur" clustering, produisant une partition de transactions, assignerait un utilisateur à un seul cluster, ce qui signifie un seul point de vue.

4.4 ECCLAT vs. *PoBOC* : Discovery challenge PKDD 2004

Nous avons confronté ECCLAT et *PoBOC* (cf. Section 3.2.1) lors du Discovery Challenge PKDD 2004 [DCS04] sur des données relatives à l'athérosclérose[14]. Rappelons que tout comme ECCLAT, *PoBOC* a l'originalité de produire des clusters chevauchants. Les données sont constituées de patients décrits par des caractéristiques sociaux et des résultats d'examens. L'objectif est de détecter des facteurs de risque de la maladie.

Pour les deux algorithmes, nous avons utilisé les mêmes jeux de données pré-traitées, et donc les mêmes items, ce qui facilite l'interprétation et la comparaison. Le tableau 4.37 récapitule les deux jeux de données utilisés (`AthDeath` et `AthRisk`). Le premier jeu de données contient des patients morts de l'athérosclérose et des patients éventuellement morts mais d'autres maladies. Le second contient des patients normaux, des patients à risque, et des patients dont on ne connait pas la valeur de classe.

	Nombre de patients	Distribution selon les classes	Nombre d'items
`AthDeath`	748	124 (mort), 624 (autre)	117
`AthRisk`	1303	859 (risque), 276 (normal), 168 (inconnu)	108

TAB. 4.37 – Caractéristiques des données relatives à l'athérosclérose

D'un point de vue technique, ECCLAT présente l'avantage de produire directement des bi-sets. *PoBOC* produit des clusters chevauchants mais ne fournit pas de description des clusters. De façon à évaluer les résultats, un post-traitement des clusters a été mis en place pour compléter les bi-sets en calculant des motifs fermés à partir de ces clusters. Rappelons que la fermeture est très importante pour l'interprétation des résultats (cf. Section 4.1 et 4.3.1). Le principe est le suivant : pour un cluster donné, on détermine le motif représentant l'intersection de toutes les transactions du cluster. Ensuite, le cluster est étendu avec les transactions qui vérifient ce motif et n'appartiennent pas au cluster.

Le tableau 4.38 résume les résultats obtenus avec ECCLAT. Pour chacun des jeux de données et une valeur $minfr$, le nombre de motifs fermés fréquents est indiqué. On trouve aussi le nombre de clusters, le nombre de transactions dans le reste, le recouvrement moyen, et le nombre moyen de transactions dans les clusters, selon différentes valeurs de M. $minfr$ a été fixé à 5%. Pour `AthDeath`, cela représente un nombre minimum de 38 patients par cluster. Nous obtenons le plus petit nombre de clusters qui classent toutes les transactions, avec M=3. Pour `AthRisk`, $minfr$ représente 66 patients. Avec M=5, nous réduisons de manière significative le nombre de clusters tout en minimisant le cluster reste (un seul patient de classe inconnue n'est pas classé).

Les résultats obtenus par *PoBOC* sont présentés dans le tableau 4.39. Pour `AthDeath`, 155 clusters sont produits. Avant l'étape de description, la taille moyenne des clusters est

[14]Projet STULONG, *http://euromise.vse.cz/stulong−en/*

	$minfr$ (%)	$\|\mathcal{FC}\|$	M	Nombre de clusters	Taille du reste	Recouvrement moyen	Taille moy. des clusters
`AthDeath`	5	1412883	1	418	0	7	41,1
			3	181	0	5	41,5
`AthRisk`	5	1263715	1	626	0	13	73,3
			5	179	1	10	77,7

TAB. 4.38 – Résultats avec ECCLAT (Athéroclérose)

de 24,5 et le recouvrement moyen est de 1,4 transactions. Le calcul de la description élargit les clusters (367,7) et augmente le recouvrement (204). Avec `AthRisk`, nous obtenons 264 clusters ayant une taille moyenne de 32 transactions, et un recouvrement moyen de 2. Après description, un cluster a une taille moyenne de 670,2 et le recouvrement moyen correspond à 492 transactions. Remarquons que deux clusters ne sont pas décrits (l'intersection des transactions est vide).

	Nombre de clusters	Avant description		Après description	
		Recouvrement moyen	Taille moy. des clusters	Recouvrement moyen	Taille moy. des clusters
`AthDeath`	155	1,4	24,5	204	367,7
`AthRisk`	264	2	32	492	670,2

TAB. 4.39 – Résulats avec *PoBOC* (Athéroclérose)

Nous constatons que le calcul des descriptions a de grandes conséquences sur les clusters. En effet, ces derniers voient leur taille augmenter ce qui influe sur la pureté des clusters. Par exemple sur `AthRisk`, le nombre de clusters purs après l'étape de description est nul. Avec ECCLAT, nous en obtenons deux. Les descriptions produites sont moins précises que celles des clusters obtenus par ECCLAT. Les motifs découverts par ECCLAT contiennent en moyenne 10 items, ce qui correspond à plus du double de la taille moyenne de ceux obtenus avec *PoBOC*. De plus, certains clusters peuvent ne pas avoir de description.

Nous avons observé qu'ECCLAT et *PoBOC* n'ont pas produit les mêmes résultats, mais les interprétations tendent vers les mêmes conclusions. Le rôle important de la tension artérielle a été mis en évidence par les deux algorithmes. ECCLAT a trouvé les valeurs les plus élevées pour les patients morts des suites d'athérosclérose. *PoBOC* a découvert de faibles valeurs pour les patients de la classe `autre`. Nous pouvons dire ici que les résultats vont dans le même sens et se complètent. ECCLAT and *PoBOC* ont aussi montré que des valeurs élevées de la tension artérielle caractérisent les patients à risque.

Les méthodes ont fait apparaître que certains attributs ne semblent pas être significatifs pour définir des profils pronostics. Avec ECCLAT, nous avons remarqué les items correspondant à la situation de famille et aux examens d'urine. Pour *PoBOC*, l'utilité des examens concernant les triglycérides est soumis à discussion.

Parmi les patients de classe inconnue, *PoBOC* et ECCLAT sont d'accord pour en classer 22 dans le groupe à risque. Au total, nous avons trouvé que 32 patients de classe inconnue présentent des risques relativement à l'athérosclérose.

4.5 Bilan

ECCLAT permet de construire un clustering approché avec peu de chevauchement ou de découvrir un ensemble de clusters avec chevauchement. L'algorithme peut trouver une partition des transactions, s'il y en existe une. Mais il ne n'en créera pas artificiellement, par opposition à d'autres méthodes, étant donné qu'il se base sur la recherche de motifs cachés dans les données.

Nous avons vu qu'ECCLAT apporte des résultats intéressants au niveau de l'interprétation et de l'exploitation des clusters. Les chevauchements permettent d'obtenir plusieurs points de vue et de ce fait une meilleure caractérisation par rapport aux autres méthodes. Nous avons illustré cela sur les benchmarks `Mushroom`, `Votes` et `Titanic` où nous recherchions un clustering approché. Nous avons aussi utilisé des données médicales lors du Discovery Challenge PKDD 2002. Nous avons découvert des clusters de patients correspondant aux différents stades de fibrose du foie (niveau de gravité d'une hépatite). De tels clusters décrits par les examens sur les patients, peuvent mettre en évidence des facteurs pour estimer la gravité de la fibrose du foie. Nous avons fait apparaître quelques items liés aux stades sévères de la maladie.

Pour des applications en Web Mining, le chevauchement est aussi très important. Par exemple, le chevauchement est souhaitable pour regrouper des pages web ou des utilisateurs, et la navigation dans les hyper-documents [CGMZ00]. Cela correspond à différents points de vue pour classer la page ou l'utilisateur. Dans le cas de clusters d'utilisateurs, autrement dit des communautés d'intérêt, un utilisateur peut très bien appartenir à plusieurs communautés car il peut avoir plusieurs centres d'intérêt. Remarquons que plusieurs communautés distinctes peuvent aussi avoir un recouvrement dans leurs intérêts (chevauchement d'items) ce qui est possible avec ECCLAT. Nous avons effectué une première expérimentation de découverte de communautés sur des données issues de fichiers de trace d'un seveur mandataire de France Télécom R&D. Les clusters peuvent être ensuite utilisés pour diverses applications de personnalisation, comme par exemple la recommandation de documents aux utilisateurs. Nous abordons ce point et évaluons l'apport d'ECCLAT dans le chapitre 5.

Lors du Discovery Challenge PKDD 2004, nous avons comparé ECCLAT et *PoBOC* sur des données relatives à l'athérosclérose. Nous avons observé que les clusters produits sont différents, mais l'interprétation des résultats tend vers les mêmes conclusions. Les deux algorithmes ont mis en évidence le rôle important de la tension artérielle pour définir des profils pronostics. Cette comparaison a aussi montré l'avantage d'ECCLAT à produire directement les descriptions des clusters. En effet, une étape de post-traitement est nécessaire pour calculer les descriptions des clusters produits par *PoBOC*. Cette étape est délicate et peut conduire à ne trouver aucune description pour un cluster. De plus, les descriptions s'avèrent moins précises que celles produites par ECCLAT.

4.6 Conclusion

Nous avons proposé une nouvelle méthode, nommée ECCLAT, pour découvrir des clusters à partir de données qualitatives, utilisant de façon originale les motifs fermés fréquents. ECCLAT sélectionne des concepts à partir du treillis des motifs fermés fréquents. Nous utilisons les algorithmes récents de KDD pour découvrir les motifs fermés fréquents, ce qui permet une recherche efficace sur de grandes quantités de données. Contrairement aux méthodes existantes, notre approche n'utilise pas de mesure globale de similarité entre les transactions, mais est basée sur un critère d'évaluation d'un motif fermé fréquent appelé *intérêt* et qui est composé de l'*homogénéité* et de la *concentration*. Notons que le calcul de ces mesures et la sélection des clusters requièrent un faible coût de calcul.

ECCLAT permet de construire un clustering approché avec peu de chevauchement ou de découvrir un ensemble de clusters avec chevauchement. ECCLAT utilise uniquement deux paramètres ($minfr$ et M), intuitifs et simples à comprendre. $minfr$ traduit le nombre minimum d'éléments dans les clusters résultats, et donc la granularité des résultats. M traduit le nombre minimum de transactions différentes entre chaque paire de clusters obtenus. Le nombre de clusters résultats n'est pas fixé par l'utilisateur, mais est borné implicitement par la méthode de sélection. Pour rechercher un clustering approché, M doit avoir une valeur proche de $minfr$. Notons qu'un cluster "reste" contenant des transactions non classées, peut apparaître. Si nous désirons classer toutes les transactions, M doit être égal à 1, et $minfr$ doit être relativement faible pour que chaque transaction apparaisse au moins dans un motif fermé fréquent.

Nous obtenons une description maximale de chaque cluster sous forme d'une conjonction d'items, facilement exploitable comme nous l'avons vu sur des benchmarks et des données médicales. Nous avons montré l'intérêt de découvrir directement ces descriptions lors de notre comparaison avec l'algorithme $PoBOC$ qui produit des clusters chevauchants mais nécessite une étape de post-traitement pour calculer les descriptions. Remarquons que nous n'effectuons aucun contrôle sur le chevauchement éventuel d'items entre les clusters. Ce chevauchement peut aussi apporter des informations intéressantes sur la caractérisation de classe ou montrer des rôles plus ou moins importants de certains items. Nous avons aussi évalué de façon quantitative le recouvrement de transactions, sur des données issues de serveurs mandataires de France Télécom R&D, dans l'optique de développer des applications de Web Mining basées sur des clusters d'utilisateurs.

La dualité présente dans la notion de concept, nous laisse envisager la possibilité de travailler sur les données transposées. Cela serait un pas important vers la mise au point d'une méthode de bi-clustering pour former une bi-partition approchée (à la fois sur les transactions et sur les items). Nous pourrions en effet adapter nos mesures de façon "dual". Il ne serait alors pas nécessaire de réaliser en réalité la transposition [RC03b], les mesures "transposées" pouvant être déterminées dans l'algorithme calculant nos mesures sur les données originales.

Dans le chapitre 5, nous proposons un système de recommandation de documents basé sur des communautés d'intérêt. Nous utilisons des clusters chevauchants produits avec ECCLAT sur des données issues de serveurs mandataires pour simuler une utilisation dans CYRANO. Nous comparons les résultats avec ceux obtenus en utilisant une partition d'utilisateurs afin d'évaluer l'apport de notre méthode.

Chapitre 5

Recommandations de documents basées sur des clusters d'utilisateurs sous forme de bi-sets

Sommaire

En conclusion du chapitre 2, nous avons évoqué la nécessité d'avoir un système de recommandation de documents qui soit autonome, hybride, et basé sur des clusters pouvant capter les différents centres d'intérêts des utilisateurs. Les clusters chevauchants produits par ECCLAT (cf. chapitre 4) permettent de réaliser ce dernier point. Rappelons que nous nous plaçons dans le contexte de CYRANO (cf. chapitre 1) où les utilisateurs peuvent consulter des documents caratérisés par des ensembles de termes. Chaque cluster est un bi-set (cf. section 3.4) composé d'un ensemble de termes qui correspond à la description du cluster, et d'un ensemble d'utilisateurs.

Nous proposons dans ce chapitre un système de recommandation de documents basé sur des clusters d'utilisateurs [DLC03] sous forme de bi-sets. Notre système est autonome et hybride de type collaboration via la contenu. En effet, notre méthode s'apparente au filtrage basé sur le contenu mais utilise aussi des techniques de filtrage collaboratif (cf. chapitre 2).

Nous évaluons notre système en simulant des consultations de documents grâce à des fichiers de trace de serveurs mandataires qui s'apparentent à une future utilisation dans CYRANO. Le mécanisme de recommandation est générique à partir du moment où pour chaque cluster, sont indiqués les utilisateurs et la description du cluster sous forme d'une conjonction de termes. Nous comparons les résultats de recommandations obtenus avec des clusters chevauchants produits par ECCLAT, avec les résultats obtenus en utilisant des partitions d'utilisateurs formées par *ARHP* (cf. section 3.3.2).

5.1 Utilisation de bi-sets

Comme nous l'avons vu en conclusion du chapitre 2, il est nécessaire dans notre contexte de se baser sur des clusters pour effectuer des recommandations. Chacun de ces clusters doit être décrit par un ensemble de termes reflétant des centres d'intérêt, et par un ensemble d'utilisateurs. Ces deux ensembles forment un bi-set (cf. section 3.4). Cela écarte donc toutes les méthodes de clustering qui ne présentent pas de descriptions des clusters obtenus (cf. chapitre 3 et annexe B). ECCLAT satisfait pleinement ces contraintes.

Ces clusters ainsi définis permettent de mélanger des principes utilisés dans les systèmes de recommandations collaboratifs et basés sur le contenu. En effet, Les clusters sont formés en se basant sur les contenus des documents. Dans le cas d'ECCLAT, chaque cluster est décrit par les termes partagés par les utilisateurs de ce cluster. Il est donc possible d'effectuer les recommandations de façon collaborative : pour un utilisateur d'un cluster, recommander les documents qu'il n'a pas consultés et qu'ont consultés les autres utilisateurs du cluster. Nous pouvons aussi avoir une approche orientée sur le contenu : pour un document nouvellement arrivé dans CYRANO, nous pouvons utiliser l'ensemble des clusters comme filtre. Notons que *Amalthaea* utilise la même approche mais sur des clusters représentant les différents centres d'intérêt d'un seul utilisateur (cf. section 2.1). En utilisant nos clusters, nous pouvons recommander un document aux utilisateurs des clusters qui s'apparient le plus par rapport aux termes. Nous allons nous concentrons sur ce type de recommandations.

Cluster	Description	Utilisateurs
1	***sécurité*** *internet cinéma film*	2 4 5
2	*java* ***sécurité***	**1** 2 3
3	*internet protocole ip tcp*	**1** 5 6 7
4	***sécurité*** *maison*	7 8
5	*pêche rivière*	**1** 6 8

TAB. 5.1 – Un exemple de clusters d'utilisateurs produits avec ECCLAT

Nous voyons dans cette application tout l'intérêt du chevauchement éventuel des utilisateurs entre les clusters, et même le possible chevauchement des termes. Le tableau 5.1 montre des clusters produits par ECCLAT, avec leurs descriptions et les identifiants des utilisateurs contenus dans ces clusters. Le chevauchement d'utilisateurs permet de capter leurs divers centres d'intérêt. Sur cet exemple, trois centres d'intérêt ont été détectés pour l'utilisateur 1 : JAVA et sécurité, les protocoles TCP et IP, et la pêche. Le chevauchement de termes permet de capter différents contextes. En effet, le terme `sécurité` associé au terme `java` ne correspond pas au même contexte que s'il est associé avec le terme `maison`.

Remarquons aussi que nous pouvons obtenir des clusters où la description mélange différents sujets. Par exemple, le cluster n°1 est décrit par `sécurité`, `internet`, `cinéma` et `film`. Il est difficile de dégager du sens à partir de l'association de tous ces termes. Un tel cluster peut être produit car nous recherchons des motifs fermés (cf. section 4.1). Autrement dit, les utilisateurs 2, 4 et 5 ont consulté des documents ayant les termes `sécurité` et `internet`, mais ils ont aussi tous consulté des documents caractérisés par `cinéma` et `film`.

Notons que si un utilisateur est très intéressé par le `C++` et qu'il est le seul, nous n'allons pas le détecter avec ECCLAT. Nous prenons en compte les intérêts communs partagés par un groupe d'utilisateurs. Le nombre minimal d'utilisateurs par cluster est néanmoins modifiable, et correspond au paramètre $minfr$.

5.2 Calcul des recommandations

Dans cette section, nous présentons les bases de notre système de recommandation [DLC03]. Tout comme le système de Mobasher et al. (cf. section 2.1), nous avons un processus off-line et un processus on-line réalisant des recommandations. Lors du processus off-line, les clusters d'utilisateurs sont formés (clusters d'URL pour Mobasher et al.). Le principe de notre processus on-line est de calculer un taux de recouvrement entre un document et chacun des clusters afin de déterminer les utilisateurs potentiellement intéressés par le document.

Pour définir le taux de recouvrement, il est possible d'utiliser des mesures classiques comme les coefficients de Jaccard, Dice, ou Cosine. Elles sont notamment utilisées en clustering et en recherche d'information [FB92]. Mais elles prennent en compte le nombre de termes du cluster, ce qui pose un problème car le chevauchement de termes entre les clusters et le mélange de différents sujets dans la définition d'un cluster sont possibles.

Reprenons notre exemple (cf. tableau 5.1), il est tout à fait possible d'obtenir un cluster comme C_1 mélangeant différents sujets ({`sécurité internet`} et {`cinéma film`}). Notons aussi que C_1 et C_2 ont en commun le terme `sécurité`.

Soient $\mathcal{D}$ un document ayant comme description $K_{\mathcal{D}}$={`java http sécurité applet internet package`}, les scores obtenus par C_1 et C_2 sont les suivants :

$$Jaccard(\mathcal{D}, C_i) = \frac{|K_{\mathcal{D}} \cap K_{C_i}|}{|K_{\mathcal{D}} \cup K_{C_i}|} \qquad C_1 \rightarrow \frac{2}{8} \rightarrow 25\% \qquad C_2 \rightarrow \frac{2}{6} \rightarrow 33\%$$

$$Cosine(\mathcal{D}, C_i) = \frac{|K_{\mathcal{D}} \cap K_{C_i}|}{\sqrt{|K_{\mathcal{D}}| \times |K_{C_i}|}} \quad C_1 \rightarrow \frac{2}{\sqrt{24}} \rightarrow 40,8\% \quad C_2 \rightarrow \frac{2}{\sqrt{12}} \rightarrow 57,7\%$$

$$Dice(\mathcal{D}, C_i) = \frac{2|K_{\mathcal{D}} \cap K_{C_i}|}{|K_{\mathcal{D}}| + |K_{C_i}|} \qquad C_1 \rightarrow \frac{2\times 2}{10} \rightarrow 40\% \qquad C_2 \rightarrow \frac{2\times 2}{8} \rightarrow 50\%$$

Les clusters C_1 et C_2 ont le même nombre de termes communs avec le document $\mathcal{D}$. Mais comme C_1 mélange différents sujets, sa description est plus grande. Il est désavantagé par rapport à C_2, et obtient des scores plus faibles. Ce cas de figure ne doit pas se produire dans notre système de recommandation.

Etant donné que ces mesures ne sont donc pas adaptées à notre problème, nous en proposons une qui ne prend pas en compte la taille de la description du cluster. Cette mesure, notée CR, correspond au pourcentage de termes du document qui sont présents dans la description du cluster (cf. définition 13).

Définition 13 (Taux de recouvrement entre un document et un cluster) *Soient $\mathcal{D}$ un document, $K_{\mathcal{D}}$ l'ensemble des termes qui le caractérisent. Soit C_i un cluster composé d'un ensemble de termes K_{C_i} et d'un ensemble d'utilisateurs U_{C_i}. Nous calculons le taux de recouvrement de la façon suivante :*

$$CR(\mathcal{D}, C_i) = \frac{|K_{\mathcal{D}} \cap K_{C_i}|}{|K_{\mathcal{D}}|} * 100.$$

Rappelons que si C est un cluster produit par ECCLAT, alors C=(X,$g(X)$). Sa description K_C correspond au motif fermé fréquent X, et ses utilisateurs U_C correspondent à l'ensemble $g(X)$.

Pour un document et un cluster, si le taux de recouvrement est plus grand qu'un certain seuil, alors le document est recommandé aux utilisateurs du cluster.

Nous fixons un seuil minimum de recouvrement noté *mincr*. Si $CR(\mathcal{D}, C_i) \geq mincr$, alors nous recommandons le document $\mathcal{D}$ aux utilisateurs U_{C_i} (cf. algorithme 3).

La valeur de *mincr* influence bien entendu le nombre de documents recommandés. Plus la valeur de *mincr* augmente, plus le nombre de documents recommandés diminue. Il faut donc un compromis entre un nombre élevé de recommandations qui rendrait le système peu agréable et utilisable pour les utilisateurs, et pas assez de recommandations qui apporterait peu d'utilité.

Algorithme 3 Calcul des recommandations

Paramètres : $\mathcal{D}$ un document, $\mathcal{C}$ un ensemble de clusters, et *mincr* le seuil minimum de recommandation.
Retour : $reco(\mathcal{D})$ l'ensemble des utilisateurs à qui nous allons recommander le document $\mathcal{D}$.
Début
pour tout $C \in \mathcal{C}$ **faire**
  si $CR(\mathcal{D}, C) \geq mincr$ **alors**
    $reco(\mathcal{D}) = reco(\mathcal{D}) \cup U_C$;
  fin si
fin pour
Retourner $reco(\mathcal{D})$;
Fin

Sur notre exemple, on obtient les résultats suivants :

- K_{C_1}=`{sécurité internet cinema film}`, CR=33%.
- K_{C_2}=`{java sécurité}`, CR=33%.
- K_{C_3}=`{internet protocole ip tcp}`, CR=16,7%.
- K_{C_4}=`{sécurité maison}`, CR=16,7%.
- K_{C_5}=`{pêche rivière}`, CR=0%.

L'ordre des clusters est le suivant : C_1 à égalité avec C_2, puis C_3 à égalité avec C_4, et enfin C_5. Avec *mincr* fixé à 30%, le document $\mathcal{D}$ est recommandé aux utilisateurs des clusters C_1 et C_2. Nous remarquons bien que C_1 et C_2 obtiennent le même score.

5.3 Evaluations

Dans le but d'évaluer notre système de recommandations, nous avons utilisé des fichiers de trace provenant de serveurs mandataires de France Télécom R&D.

Dans ces expérimentations, nous comparons la réalité des consultations des utilisateurs avec des recommandations simulées par notre système. Il n'y a pas d'évaluation des recommandations par les utilisateurs. Notre système est virtuel. Nous évaluons l'apport, dans notre application, de l'utilisation des clusters chevauchants produits par ECCLAT par rapport à des partitions d'utilisateurs formées par *ARHP* (cf. section 3.3.2). Rappelons que *ARHP* utilise les motifs fréquents, et est notamment utilisé dans un système de recommandation proposé par Mobasher et al. (cf. section 2.1). Il fournit une partition des termes et des utilisateurs ce qui signifie que peu de centres d'intérêt sont captés par utilisateur.

Les données

Les données contiennent 147 utilisateurs (les transactions) et 8727 items. Les items sont des termes extraits des pages HTML consultées par les utilisateurs pendant une période d'un mois où 24278 pages ont été consultées. Cela correspond en moyenne à 5,5 pages consultées par utilisateur et par jour. Chaque page est décrite par un maximum de 10 termes obtenus avec un extracteur basé sur la fréquence des mots significatifs, développé à France Télécom R&D. Pour plus d'informations sur la collecte et la préparation des données, se reporter à l'annexe E.

Protocole

La simulation se déroule de la manière suivante. Préalablement, des clusters sous forme de bi-sets (cf. section 5.1) sont découverts en utilisant ECCLAT ou *ARHP*, à partir de l'ensemble des données issues des fichiers de trace. Ensuite, pour chaque document $\mathcal{D}$ présent dans ces fichiers de trace, le système détermine en utilisant les clusters, les utilisateurs supposés être intéressés par $\mathcal{D}$ (notés $UsersR(\mathcal{D})$). Cet ensemble correspond aux utilisateurs à qui le système aurait recommandé le document dans un contexte réel d'utilisation. Ensuite, le système vérifie à l'aide des fichiers de trace, si les utilisateurs qui avaient réellement consulté le document (notés $Users(\mathcal{D})$) auraient été avisés par le système (i.e. sont présents dans $UsersR(\mathcal{D})$).

Remarquons que si nous avions utilisé des fichiers de trace de serveurs web où les ensembles de documents et donc de termes sont connus et relativement stable dans le temps, nous aurions pu appliquer des techniques classiques de validation comme couper les données en deux, découvrir les clusters sur la première partie, déterminer des recommandations à partir des documents de la seconde partie, et vérifier sur les consultations toujours sur cette seconde partie. Néanmoins, pour un serveur mandataire, les ensembles de documents et plus spécialement de termes peuvent être très différents entre deux périodes. Cela rend difficile une première simulation et évaluation des résultats sans retour humain. Nous utilisons donc la même période pour découvrir les clusters et calculer les recommandations.

Nous utilisons une petite valeur pour *mincr* de façon à recommander un nombre important de documents. *mincr* est fixé à 20%.

Les résultats sont évalués grâce aux mesures suivantes :

$$échec(\mathcal{D}) = \frac{|Users(\mathcal{D}) - UsersR(\mathcal{D})|}{|Users(\mathcal{D})|}$$

$$rhit(\mathcal{D}) = \frac{|UsersR(\mathcal{D}) \cap Users(\mathcal{D})|}{|UsersR(\mathcal{D})|}$$

Le taux d'*échec* évalue le pourcentage d'utilisateurs qui ont consulté dans la réalité un document qui ne leur a pas été recommandé lors de la simulation. Le taux *rhit* (recommendation hit) mesure la quantité d'utilisateurs qui ont réellement consulté le document, parmi ceux qui ont reçus une recommandation sur ce document. De bons résultats correspondent donc à un taux moyen d'*échec* bas et un taux moyen *rhit* élevé.

Nous verrons dans les expérimentations que les taux *rhit* sont bas, ce qui signifie qu'un nombre important d'utilisateurs ont reçu des recommandations portant sur des documents qu'ils n'ont pas consulté en réalité. Peut-être qu'ils auraient été intéressés par ces documents, mais nous ne pouvons pas le vérifier. Pour cela, il serait nécessaire d'effectuer des expérimentations dans des conditions réelles d'utilisation, avec un retour des utilisateurs.

Néanmoins, l'association de ces deux taux permet de comparer différents résultats d'expérimentation. Si une expérience conduit à un taux d'*échec* plus bas qu'une autre, alors on peut conclure que l'expérience est meilleure à condition que le taux *rhit* soit supérieur.

5.3.1 Résultats avec des clusters produits par ECCLAT

De façon à prendre en compte tous les utilisateurs, nous avons fixé pour ECCLAT, $minfr$ à 10% et M à 1 (cf. tableau 5.2). Cela correspond à un nombre minimal de 14 utilisateurs par cluster. Avec ce seuil de fréquence, nous obtenons 454043 motifs fermés fréquents. 45 clusters ont été extraits. Le nombre moyen d'utilisateurs par cluster est de 21. Le chevauchement moyen des utilisateurs est de 7 utilisateurs. La taille moyenne des descriptions des clusters est de 9 items. En moyenne, 1,7 items sont communs entre les descriptions des clusters.

$minfr$ (%)	M	Nombre de clusters	Taille du reste
17	1	17	4
17	$minfr$	2	43
10	1	45	0
10	$minfr$	3	42

TAB. 5.2 – Informations sur les clusters d'utilisateurs produits avec ECCLAT

Le système a recommandé 11948 documents (cf. tableau 5.3) ce qui correspond en moyenne à 2,7 documents recommandés par jour à un utilisateur. Nous remarquons que 80% des documents recommandés ont été consultés. Le taux moyen d'*échec* est de 17,7%. Le taux moyen *rhit* est de 3,3%.

	Nb. documents recommandés	Taux moyen d'*échec* (%)	Taux moyen *rhit* (%)	% documents recommandés qui ont été consultés
ECCLAT	11948	17,7	3,3	80
ARHP, k=948	4212	87,6	2,1	11
ARHP, k=4	23937	24,8	0,8	75

TAB. 5.3 – Résultats des recommandations avec $mincr$=20%

5.3.2 Comparaison avec des clusters obtenus avec *ARHP*

Nous avons effectué la même expérience en utilisant *ARHP* pour produire une partition des utilisateurs. Cet algorithme utilise deux paramètres : le seuil minimum de fréquence $minfr$ pour la découverte des motifs fréquents, et le nombre de clusters k utilisé pour le partionnement. Nous avons fixé $minfr$ à 17% pour des raisons algorithmiques. En effet, avec ce seuil, nous obtenons 666857 motifs fréquents. Si nous baissons cette valeur par exemple à 15%, nous avons plus de 3,8 millions de motifs fréquents ce qui rend très difficile l'exécution de l'algorithme de partionnement.

Le choix du nombre de clusters est délicat. Ayant quelques connaissances sur les données grâce aux résultats produits par ECCLAT, nous avons fixé k à 948, de manière à obtenir en moyenne 9 items par cluster (comme les clusters d'ECCLAT). Nous avons en réalité obtenu 45 clusters. Nous remarquons qu'ici, les résultats sont très insuffisants (cf. tableau 5.3). Le taux moyen d'*échec* est de 87,6%, et le taux moyen *rhit* de 2,1%. De plus, le nombre de documents recommandés est très bas. En effet, seulement 4212 documents sont recommandés. Cela représente en moyenne moins d'un document récommandé (0,95 exactement) par jour et par utilisateur. Certe, trop de recommandations peu dérouter l'utilisateur, mais ici c'est très peu pour un système opérationnel.

Si nous fixons k à 45, de façon à obtenir "logiquement" le même nombre de clusters qu'avec ECCLAT, *ARHP* ne produit pas de clusters significatifs dans le sens où un cluster (parmi les 19 obtenus) regroupe à lui seul 125 utilisateurs.

Nous avons ensuite fixé k à 4 clusters. 23937 documents ont été recommandés (cf. tableau 5.3). Le taux moyen d'*échec* obtenu est 24,8%. 75% des documents recommandés ont été consultés. Ce taux est légèrement plus faible que les résultats obtenus avec les clusters produits avec ECCLAT. Par contre, le taux moyen *rhit* est de 0,8% ce qui est bien inférieur à ECCLAT.

En utilisant les clusters chevauchants produits par ECCLAT, nous avons donc obtenu de meilleurs résultats sur nos données. Nous avons aussi constaté qu'il est délicat d'utiliser un algorithme nécessitant de fixer le nombre de clusters, étant donné que l'on dispose *a apriori* d'aucune information sur les utilisateurs.

5.4 Conclusion

Nous avons proposé un système de recommandation de documents basé sur des clusters d'utilisateurs sous la forme de bi-sets, comme ceux produits par ECCLAT. Ces clusters sont décrits par des ensembles de termes qui sont utilisés pour calculer un taux de recouvrement avec la description d'un document qui sera éventuellement recommandé. Notre système est autonome et hybride. En effet, nous effectuons une découverte de clusters d'utilisateurs en se basant sur le contenu des documents, et non sur les préférences, ou des notes données par des utilisateurs comme *Tapestry*, *Grouplens* et *RecTree* (cf. chapitre 2), ou les documents eux-mêmes. L'aspect collaboratif est obtenu grâce à la détection des intérêts partagés entre les différents utilisateurs. Pour une utilisation dans CYRANO, nous nous sommes concentré sur les fonctions de filtrage et de distribution de documents. Néanmoins, nous pourrions tout à fait mettre en place un système collaboratif en proposant à un utilisateur d'un cluster, des documents qu'il n'a pas consultés et par contre que les autres utilisateurs du cluster ont consulté.

Nous avons évalué notre méthode en utilisant des fichiers de trace de serveurs mandataires. Nous avons effectué des expérimentations avec des clusters obtenus avec ECCLAT et aussi avec des partitions produites par *ARHP*. Sur nos données, les résultats obtenus sont meilleurs avec ECCLAT. Ces expérimentations correspondent à une simulation de fonctionnement de notre système de recommandation. Notons que ce contexte est difficile, et nous avons constaté un nombre important de recommandations non suivies d'effet. En perspective, il est donc logique d'effectuer des tests supplémentaires dans des conditions réelles avec un retour des utilisateurs sur les recommandations. Il serait aussi intéressant de modifier des systèmes de recommandation existants en y intégrant ECCLAT, et de comparer les résultats avec les systèmes originaux.

Remarquons que notre système pourrait remplacer le mécanisme de sélection des contenus à enregistrer dans l'espace de stockage de la CyraNode (cf. section 1.2.2). Par exemple, pour un document, le système pourrait calculer les taux de recouvrement par rapport à chaque cluster. La priorité du document serait alors le taux de recouvrement maximum trouvé (ou la moyenne, ...). Ainsi, nous pourrions obtenir des contenus selon les différents centres d'intérêt des utilisateurs, et non plus selon les centres d'intérêt majoritaires.

L'utilisation des clusters pour la recommandation de documents ou pour la sélection des contenus de la CyraNode, nécessite que ces clusters soient fréquemment mis à jour. Le développement d'une version incrémentale d'ECCLAT permettrait alors de proposer un système pseudo temps réel. Néanmoins, nous pouvons supposer que les centres d'intérêt des utilisateurs ne changent pas brusquement, et dans ce cas utiliser un processus qui recherche périodiquement les clusters d'utilisateurs (par exemple tous les soirs).

Dans la partie III, nous abordons le pré-chargement d'information dans les caches dans l'optique de disposer d'un processus actif de recherche et de collecte de documents pour les charger dans une CyraNode. Rappelons qu'une CyraNode ne dispose actuellement que de documents provenant de sources d'information telles que des serveurs mandataires-caches ou des chaînes satellites.

Troisième partie

Optimisation de la caractérisation des utilisateurs et application au pré-chargement d'information

Chapitre 6

Le pré-chargement d'information dans les caches

Sommaire

Dans cette dernière partie, nous abordons les systèmes de pré-chargement de documents dans les caches, dans l'objectif de doter une CyraNode d'un processus d'acquisition automatique de nouveaux documents (cf. section 1.3). Dans ce chapitre, nous faisons un état de l'art des méthodes de pré-chargement d'information dans le cache d'un navigateur ou d'un serveur mandataire (cf. chapitre 1). L'objectif de ce procédé, aussi appelé *prefetching*, est d'améliorer la qualité de service (QoS), ce qui se traduit par une accélération du chargement et une diminution du temps de latence. Cette dernière correspond à la durée entre le moment où l'utilisateur par l'intermédiaire de son navigateur demande un document et le moment où son navigateur obtient une réponse. Pour cela, les méthodes anticipent les accès des utilisateurs et chargent dans le cache les informations qui ont une forte probabilité d'être consultées. Un des problèmes est de trouver le bon équilibre entre le gain de QoS, et l'utilisation supplémentaire de bande passante induite par les pré-chargements.

Le pré-chargement peut s'effectuer de trois manières : entre les navigateurs clients et les serveurs web, entre les navigateurs clients et les serveurs mandataires, et entre les serveurs mandataires et les serveurs web. Nous portons une attention particulière au dernier cas qui se rapproche le plus de CYRANO.

6.1 Préliminaires

Les méthodes de pré-chargement utilisées dans les différents contextes cités précédemment, peuvent être classées selon deux familles. La première correspond aux méthodes analysant les pages HTML en cours de consultation de façon à télécharger à l'avance les images, les objets inclus, et les liens. La deuxième famille de méthodes est basée sur des études statistiques des successions d'URL et la recherche de motifs d'accès en utilisant les historiques de connexion. Parmi les différentes techniques utilisées, nous trouvons *PPM* qui représente l'historique sous forme d'une forêt où chaque arbre correspond à des séquences d'URL accédées. Nous présentons *PPM* dans le paragraphe suivant. Nous verrons que d'autres méthodes comme la découverte de règles d'association, sont aussi utilisées.

Les méthodes peuvent aussi être subdivisées selon qu'elles soient séquentielles (off-line) ou on-line. Dans ce dernier cas, lorsqu'un utilisateur demande un document, le serveur web, le serveur mandataire ou le client prédit la requête suivante et pré-charge immédiatement le document prédit au client. Dans le cas séquentiel, les documents détectés comme potentiellement intéressants pour l'utilisateur, sont pré-chargés pendant les périodes creuses.

6.1.1 Evaluation des méthodes de pré-chargement

Les mesures utilisées pour évaluer les performances de ces méthodes, sont basées sur des notions que nous avons évoquées dans le chapitre 1. Rappelons que lorsqu'un document présent dans un cache est redemandé, alors cela correspond à un *hit*. Si un document est demandé une première fois, et donc n'est pas présent dans le cache, cela provoque un *miss*. Les performances d'un cache sont notamment évaluées avec le taux de hit (cf. sections 1.1.2 et 1.1.4) calculé à partir des fichiers de trace qui enregistrent toute l'activité du cache.

Les différentes mesures pouvant être utilisées sont les suivantes :

- Le *rendement* correspond au taux de hit des documents pré-chargés (nombre de documents pré-chargés consultés par la suite par les utilisateurs, divisé par le nombre total de documents pré-chargés),
- Le taux de *requêtes sauvées* correspond au nombre de documents pré-chargés et consultés par la suite par les utilisateurs, divisé par le nombre total de requêtes,
- Le taux *bande passante sauvée* représente le total de la taille des documents pré-chargés et consultés par la suite, divisé par le total de la taille des documents demandés,
- La *bande passante gaspillée* est calculée grâce au total de la taille des documents pré-chargés qui ne sont pas consultés par la suite par les utilisateurs.

Un système de pré-chargement est efficace, si le rendement, les requêtes sauvées et la bande passante sauvée sont élevées. La bande passante gaspillée doit être bien entendue la plus faible possible. Le taux de hit global du cache est aussi amélioré grâce au nombre de requêtes sauvées. Notons que les notions et les mesures présentées, nous seront utiles dans le chapitre 8.

6.1.2 Une méthode clé : *PPM*

PPM (Prediction-by-Partial-Matching) [CW84] est un outil très utilisé dans les méthodes présentées dans ce chapitre. Elle permet en effet de représenter l'historique des documents consultés et d'effectuer des prédictions. A l'origine, *PPM* est un algorithme de compression de données basé sur les modèles statistiques de Markov. Cela correspond à réaliser une prédiction sur le caractère en cours d'examen du mot à compresser. Cette prédiction est effectuée grâce aux mots précédemment examinés.

Dans le contexte du Web, l'algorithme fonctionne de la même manière, sauf que les prédictions ne portent pas sur des caractères, mais sur des URL. Remarquons que pour la compression, le nombre de caractères est fini et connu. Pour un serveur web, l'ensemble des documents est connu et relativement stable. Pour un serveur mandataire, la méthode s'applique sur l'ensemble des URL déjà rencontrées. Mais le nombre de ces URL étant élevé et en constante évolution, cela pose des problèmes d'efficacité.

L'historique des accès est représenté par une forêt de profondeur Z=R+S, où R est le nombre des accès passés, et S le nombre d'étapes à prédire. Un nœud correspond à une URL, et au nombre de fois où elle a été consultée. Il y a une racine pour chaque URL candidate. A chaque requête sur une URL, une mise à jour est faite sur cette forêt ainsi que sur la liste des K dernières URL visitées. La figure 6.1 présente un exemple de cette structure avec R=1 et S=2. Sur cette figure, l'URL *A* a été consultée 13 fois. *C* a été visitée 2 fois après une consultation de *A*.

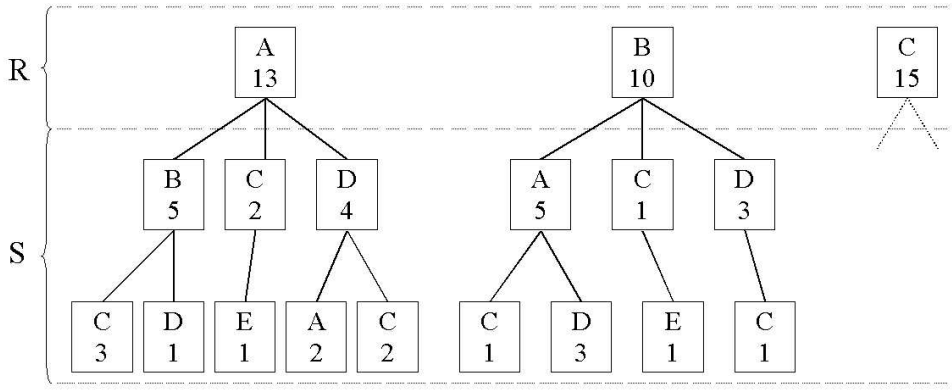

FIG. 6.1 – Exemple de structure utilisée par *PPM* pour la prédiction d'URL

L'algorithme utilise un autre paramètre en plus de R et S, qui correspond au seuil minimum pour les URL candidates. Seuls les documents dont la probabilité de sélection est supérieure à ce seuil, sont candidats.

Nous verrons avec les différents systèmes présentés par la suite, que cette approche est relativement efficace mais présente des inconvénients (cf. méthode de Nanopoulos et al.). De plus, cette structure est lourde à mettre en place, et demande des ressources machines et un maintien à jour important. Divers mécanismes sont alors mis en œuvre pour limiter la taille de la structure.

6.2 Les systèmes de pré-chargement de documents

Les systèmes de pré-chargement de documents sont classiquement organisés selon leur contexte d'utilisation. Dans un premier temps, nous présentons les systèmes qui peuvent être mis en place lorsque les acteurs sont des navigateurs clients et des serveurs web. Ensuite, nous abordons le cas où ce sont les serveurs mandataires qui envoient des documents aux navigateurs clients. Le dernier contexte est celui où des documents de serveurs web sont pré-chargés dans des caches de serveurs mandataires.

6.2.1 Serveurs Web vers les navigateurs clients

Les méthodes les plus basiques consistent à pré-charger les documents liés au document en cours de consultation par l'utilisateur grâce à son navigateur web. Les utilitaires comme *NetSonic Pro*[15] et *WebEarly 3*[16] fonctionnent sur ce principe en complément d'un navigateur sur la machine de l'utilisateur. Ils analysent la page HTML en cours de consultation, détectent les balises correspondant aux liens, et pré-chargent ces-derniers.

De nombreux travaux de recherche se sont concentrés sur les arrangements de pré-chargement entre le navigateur client et les serveurs web. Bestavros et Cunha [BC96] ont proposé un modèle pour la dissémination spéculative de documents du Web. Le moteur de prédiction est situé sur le serveur web, et est basé sur la probabilité qu'un utilisateur consultant un document i consulte un document j. Ce travail montre que les accès observés sur un serveur web peuvent être employés comme une efficace source d'information pour piloter le pré-chargement. En effet, les rendements obtenus varient entre 80% et 90%.

L'algorithme mis au point par Padmanabhan et Mogul [PM96] permet de borner la quantité de bande passante supplémentaire utilisée. Cet algorithme est basé sur les graphes de dépendance [GA94], équivalent à *PPM* avec R égal à 1. Les tests ont été effectués en simulant et rejouant des fichiers de trace de serveurs. Pour un consommation accrue de 40% de la bande passante, le temps de latence diminue de 36%. L'étude montre aussi que le préchargement de serveurs web aux clients individuels peut réduire le temps de latence de 45% mais double le trafic du réseau.

D'autres études se basent sur les règles d'association (cf. section 3.1) pour prédire les documents consultés sur un serveur. Nanopoulos et al. [NKM01] ont proposé un algorithme de recherche d'associations ordonnées en se basant sur l'algorithme APRIORI [AMS+96]. Remarquons que l'approche est différente de celle d'Agrawal et Srikant [AS95, AS96] qui recherchent des motifs séquentiels. L'algorithme permet d'extraire des motifs dans les fichiers de traces de serveurs web. Ces motifs sont ensuite utilisés par un autre algorithme réalisant les prédictions. Remarquons que les techniques basées sur la recherche de règles d'association, permettent de dériver des données l'ordre maximal (taille maximum des candidats) pour un seuil de fréquence donnée. Alors que *PPM* considère seulement des sous-séquences d'accès dans les transactions (la valeur maximale est S). Les comparaisons avec *PPM* et les graphes des dépendances ont montré que cet algorithme est plus efficace.

[15] *http://www.web3000.com/products/NetSonicpro/*
[16] *http://www.goto.fr/fr/produits/*

Nous pouvons formuler quelques critiques concernant les méthodes que nous venons de présenter. Premièrement, la forte sollicitation de la machine du client peut entrainer des problèmes d'efficacité et une surcharge réseau qui peuvent ralentir de façon importante la machine cliente, surtout si celle-ci prend la décision de pré-charger. Concernant les méthodes où les serveurs web prennent l'initiative, l'amélioration de la QoS de façon globale, implique que tous les serveurs web soient capables de prédire les documents consultés. Or cela est peu réalisable en pratique.

Nous ne nous attardons pas sur ce contexte car ces méthodes se basent exclusivement sur des documents déjà consultés. Dans le contexte d'une CyraNode, le pré-chargement de tels documents ne présente aucun intérêt. Nous voulons ajouter de nouveaux documents que les utilisateurs de CYRANO pourront alors consulter par la suite.

6.2.2 Serveurs mandataires vers les navigateurs clients

Les méthodes de préchargement entre les navigateurs clients et les serveurs mandataires, ne sont pas très répandues. Fan et al. [FCLJ99] ont proposé une approche qui compte sur le serveur mandataire pour prévoir que le document mis en cache pourrait faire référence à un futur accès en se basant sur le compresseur de données *PPM*. Le serveur mandataire profite de temps mort entre les requêtes de l'utilisateur pour envoyer des documents au cache du navigateur client. Les résultats de simulations montrent que le pré-chargement combiné avec un grand cache de navigateur et la delta-compression (compression des données à télécharger), peut réduire la latence jusqu'à 23,4%. Cette réduction de temps de latence est au détriment de la bande passante (jusqu'à 15% d'utilisation supplémentaire). Remarquons qu'il faut que les documents soient dans le cache du serveur mandataire pour être envoyés au client. Or cela n'est pas évident avec la politique de remplacement qui induit une certaine durée de vie d'un document dans le cache. On peut donc penser qu'à terme ce sont surtout les documents populaires qui feront l'objet de pré-chargement.

6.2.3 Serveurs web vers les serveurs mandataires

Nous abordons maintenant les techniques de pré-chargement de documents de serveurs web, dans un cache de serveur mandataire.

Les méthodes où les serveurs web ont l'initiative sont rares. Markatos et Chronaki [MC98] ont proposé que les serveurs web envoient régulièrement les 10 documents les plus populaires aux serveurs mandataires. Ils ont évalué la performance de cette stratégie en utilisant plusieurs fichiers de traces de serveurs web, et ont constaté qu'il est possible de prévoir plus de 40% des requêtes d'un client. Gwertzman et Seltzer [GS94] ont proposé une approche prenant en compte la situation géographique des serveurs mandataires. Un serveur web enverra des documents sélectionnés aux serveurs mandataires qui sont proches des clients de ce serveur web. Remarquons que ces techniques exigent la coopération des serveurs web, et que cette étude n'évalue pas la réduction du temps de latence.

Parmi les approches où les serveurs mandataires prennent la décision de pré-charger, nous trouvons des produits commerciaux, comme un serveur mandataire hardware proposé

par *CacheFlow*[17], effectuant de l'*embedded prefetching*. Cela correspond à pré-charger les images et objets contenus dans les pages HTML. Il n'y a pas de bande passante supplémentaire consommée car cela correspond juste à une anticipation des requêtes du client. Le temps de latence est réduit sur des documents ayant un grand nombre d'images. Ce serveur mandataire utilise aussi l'*adaptive refresh* qui rafraichit les documents les plus populaires du cache. Le logiciel *Wcol*[18] [CY97] prélève aussi à titre préventif les liens et les images inclus dans des documents.

Kim et al. [KKH00] ont proposé une méthode de pré-chargement séquentiel statistique. Ils préconisent une analyse périodique des accès des utilisateurs pour en extraire des motifs et générer des listes d'objets à pré-charger. Le nombre d'objets à pré-charger est limité, et la taille du disque alloué pour le pré-chargement peut être ajustée dynamiquement. La période utilisée pour la collecte de données est de 24 heures. Ils ont montré que l'augmentation possible du taux de hit des objets pré-chargés est entre 22% et 73%. Néanmoins, remarquons que les expériences ont été réalisées dans un environnement "académique", et qu'il n'y a pas d'informations sur les techniques utilisées pour la recherche des motifs, ni sur la consommation supplémentaire de bande passante.

Yang et al. [YZL01] ont utilisé les règles d'associations avec une prise en compte de l'ordre, formant ainsi des séquences d'accès, pour réaliser du pré-chargement. Une étape d'apprentissage est effectuée sur des fichiers de traces pour constituer un ensemble de règles ayant chacune une confiance minimum. Ces règles sont ensuite utilisées pour la prédiction. L'algorithme de prédiction fournit un ensemble de documents prédits à partir de la séquence courante d'accès d'un utilisateur, et un seuil minimum de confiance. Un point innovant de leur système est de proposer aussi une technique de cache prédictif en modifiant la politique de remplacement des objets dans le cache par l'ajout de la prise en compte de la fréquence d'accès des futurs objets.

Loon et Bharghavan [LB97] ont proposé un système de pré-chargement où les prédictions sont réalisées en tenant compte des profils des utilisateurs du serveur mandataire. Le système est une architecture hiérarchique de serveurs mandataires, où chaque cache connaît le profil du cache de niveau inférieur. Le profil d'un cache est un graphe valué, où un sommet est une URL avec son nombre de consultation, un arc $u \rightarrow v$ représente le fait que v est visité après u. A chaque requête d'un utilisateur, le moteur de prédiction génère une liste d'URL susceptibles d'être chargées par l'utilisateur en fonction de la page sur laquelle il se trouve. La liste est transmise à un cache appelé "manager" qui lance alors une procédure de mise à jour de son graphe et propage la mise à jour aux autres caches concernés. Remarquons que les objets inclus dans d'autres ne sont pas pris en compte dans le graphe, et que l'utilisation de la bande passante augmente de façon quadratique sur le temps d'utilisation de leur expérimentation. Le taux de hit obtenu est intéressant : 87,8%, mais cela concerne seulement 320 requêtes, ce qui est peu significatif.

Cunha et Jaccoud [CJ97] ont utilisé une méthode similaire pour définir les profils des utilisateurs à partir de fichiers de traces du département d'informatique de l'université de Boston. Les taux de hit obtenus sont du même ordre que ceux de Loon et Bharghavan.

[17] *http://www.cacheflow.com*
[18] *http://shika.aist-nara.ac.jp/products/wcol/wcol.html*

Notons qu'à cause de la nature des données utilisées pour les expérimentations, les études de Loon et Bharghavan, de Cunha et Jaccoud, et de Kim et al. ne sont pas représentatives de réelles conditions de fonctionnement.

Davison [Dav02] a proposé une approche très différente de celles que nous avons présentées. Les prédictions sont basées sur le contenu des pages visitées et la similarité entre les documents. Les pages récemment consultées sont utilisées comme un modèle pour ordonner les liens de la page en cours de consultation de façon à déterminer quels documents pré-charger. L'analyse du contenu des pages correspond à identifier les mots utilisés dans la définition des liens vers d'autres documents grâce aux balises HTML. Le système étudie la similarité entre les liens d'une précédente page avec les pages liés pour construire le modèle. Les évaluations montrent qu'il y a un potentiel significatif pour la prédiction basée sur le contenu des pages web. Ce type d'approche n'est pas réalisable dans CYRANO, car les documents présents dans une CyraNode ne sont pas liés entre eux.

6.3 Conclusion

Les méthodes que nous avons présentées, et spécialement celles utilisant des serveurs mandataires, sont basées uniquement sur les URL et les historiques de consultation. Toutes ces méthodes ne peuvent prédire que des documents qui sont déjà passés par le cache. Cette stratégie interdit de trouver et de pré-charger des documents nouveaux dans CYRANO. Néanmoins, nous avons remarqué des techniques et des points intéressants, comme l'utilisation de motifs, de régles d'association, l'ajout de paramètres comme la taille allouée au pré-chargement et la durée de la période d'apprentissage des habitudes et goûts des utilisateurs.

Ces études travaillent donc de manière fermée sur les URL correspondant aux documents déjà rencontrés. Cela implique alors que si les utilisateurs consultent souvent des nouveaux documents, alors les méthodes de prévision basées sur les historiques amélioreront peu les performances du cache.

Les approches basées sur le contenu des documents représentent alors une alternative idéale pour réaliser des prédictions sur des documents jamais consultés par les utilisateurs. Néanmoins, celle de Davison se base sur les liens entre les documents HTML. Or dans le contexte d'une CyraNode, les documents ne sont pas liés physiquement entre eux.

Au final, il paraît donc intéressant de réaliser du pré-chargement en utilisant des informations de plus haut niveau que les URL, comme les mots-clés ou des termes extraits de documents. De cette façon, nous pouvons alors ouvrir le système de pré-chargement vers de nouveaux documents ayant un lien thématique avec ceux déjà consultés. Nous développerons ces points dans les prochains chapitres.

Dans le chapitre 7, nous évaluons l'apport de l'utilisation de termes caractérisant les documents pour la recherche de régularités dans les accès Internet des utilisateurs. Nous proposons une nouvelle caractérisation des utilisateurs se basant sur la détection de mots émergents (plus adaptée dans un but prédictif). Dans le chapitre 8, nous présentons

un nouveau système de pré-chargement de documents utilisant cette caractérisation. Ce système a pour originalité de pouvoir pré-charger des documents non consultés par le passé.

Chapitre 7

Optimisation de la caractérisation des utilisateurs basée sur la détection de mots émergents

Sommaire

Afin de définir un système de pré-chargement de documents non consultés dans le passé, nous avons conclu à la fin du chapitre 6 qu'il est nécessaire d'utiliser les termes caractérisant les documents précédemment consultés par les utilisateurs, au lieu de se baser directement sur les URL. Un système de pré-chargement basé sur ce principe serait alors tout à fait adapté à CYRANO pour télécharger à l'avance de nouveaux documents pouvant potentiellement intéresser les utilisateurs.

Pour justifier l'emploi de termes qui demande des traitements supplémentaires et donc un certain coût, nous mesurons l'impact de l'utilisation d'URL ou de termes extraits de ces URL, sur la détection de régularités dans les accès des utilisateurs en utilisant des fichiers de trace de serveurs mandataires. Nous proposons pour cela un procédé d'évaluation de régularités dans un contexte prédictif, se basant sur une mesure que nous avons appelée *CQTS* [DL02]. Les résultats obtenus avec cette mesure permettent de spécifier des conditions d'utilisation d'un système de pré-chargement de documents.

Les expérimentations montrent la supériorité des termes face aux URL. Nous nous concentrons alors sur les termes des documents consultés pour définir les goûts des utilisateurs. Nous proposons une méthode de détection de certains termes que nous appelons "mots émergents" traduisant les tendances des utilisateurs [LD01]. Elles sont déterminées entre deux périodes. Le principe est d'identifier les termes référencés par des documents dont le nombre de consultations est en hausse significative entre les deux périodes. Les termes détectés comme émergents pour un utilisateur sont alors utilisés pour le caractériser sur la période analysée. Ces nouveaux profils utilisateurs seront utilisés par la suite pour définir un nouveau système de pré-chargement de documents.

7.1 Analyse de la régularité des accès des utilisateurs

La connaissance du comportement des utilisateurs lors de l'accès à l'information est intéressant de plusieurs points de vue. En particulier, il permet d'avoir une vision des modes d'accès à la connaissance et représente sur le plan commercial un enjeu important. Les travaux portant sur la modélisation du comportement à partir des traces de navigation utilisent essentiellement les distributions sous-exponentielles. Les plus connues sont les lois self-similaires, utilisées dans de nombreux travaux de modélisation du trafic web [LTWW93, Pax95, BCF+98, Lan00a]. Parmi les questions qui restent à l'étude, citons le choix du type de traces brutes à exploiter pour obtenir le meilleur potentiel de caractérisation (notre but dans cette section).

Nous étudions pour cela les possibilités de caractérisation à partir des traces d'accès des utilisateurs à des niveaux d'abstraction différents (URL et termes) en étudiant la régularité et leur potentiel prédictif. Cette étude permet de détecter si les utilisateurs retournent souvent sur les mêmes URL ou sur des documents portant sur les mêmes sujets. Autrement dit, cela permet de savoir si les accès utilisateurs ont une relative consistance dans le temps, ce qui est un point capital dans une application de pré-chargement de documents.

Nous effectuons une analyse des accès d'utilisateurs en utilisant des fichiers de trace de serveurs mandataires, en se basant sur les URL [DL02] et les termes extraits des documents consultés. Nous comparons les résultats obtenus avec ces deux approches [LD03b].

Dans le reste du chapitre, nous utiliserons le mot *item* comme terme générique pour désigner les URL ou les termes.

7.1.1 Description du procédé d'analyse

Pour réaliser notre étude, nous devons segmenter nos données de façon à suivre l'évolution et comparer les consultations des utilisateurs selon différentes périodes de temps, et cela pour les URL et aussi les termes. Pour chaque utilisateur, nous construisons deux séries symboliques notées S_u et S_t (cf. figure 7.1). La première représente l'ensemble des requêtes effectuées sur des URL. Ces informations sont ordonnées chronologiquement selon l'ordre de consultation. Remarquons qu'une URL peut être consultée à plusieurs reprises et donc apparaître à differents instants dans S_u. La seconde série correspond aux mêmes requêtes mais les URL sont remplacées par les termes w_j les caractérisant ($1 \leq j \leq 10$).

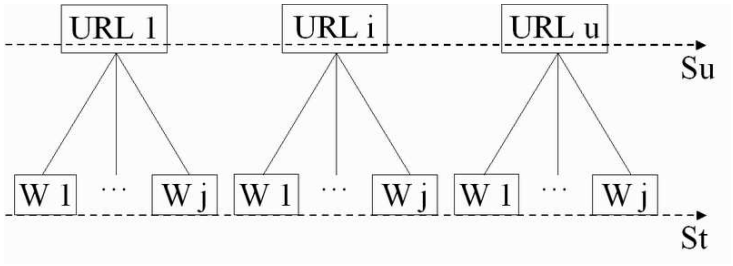

FIG. 7.1 – Les deux séries symboliques étudiées

Ces séries sont ensuite découpées en séquences de taille égale. La taille, notée T, correspond à un pourcentage du nombre total de requêtes. Un découpage selon le nombre de requêtes est préféré à un découpage directement fonction du temps car un utilisateur peut très bien avoir effectué peu de consultations voire aucune pendant une période relativement longue (demi-journée, journée, ...). Cela permet donc un meilleur suivi des consultations.

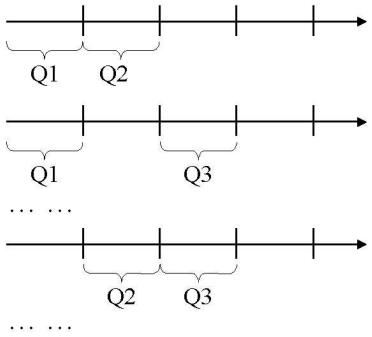

FIG. 7.2 – Procédé d'analyse

La figure 7.2 illustre le procédé utilisé. Pour la première séquence notée Q_1, nous calculons la régularité avec la séquence suivante notée Q_2. On réitère ce processus jusqu'à ce que tout l'espace des séquences soit couvert. La première séquence utilisée dans les comparaisons varient de Q_i à $Q_{i+n/2}$ (avec n le nombre total de séquences).

L'écartement entre deux séquences, noté Δ, est définie comme le nombre de séquences les séparant. Sur notre exemple, l'écartement entre Q_1 et Q_2 est égal à 0.

7.1.2 Calcul de la régularité

Une manière classique d'évaluer la régularité des accès utilisateurs est de calculer la *redondance* des requêtes. Cette redondance peut être interprétée comme illustrant le comportement de l'utilisateur. Par exemple, un comportement monolithique (accès concentrés sur peu d'items) aura une redondance élevée alors qu'un comportement plus dispersé (accès répartis sur un grand nombre d'items) aura une faible redondance.

Néanmoins, la redondance ne prend pas en compte une période par rapport à une autre. De ce fait, elle n'est pas adaptée à notre contexte. De plus, il est nécessaire d'utiliser une

mesure évaluant le potentiel prédictif, permattent ainsi de se placer dans un contexte similaire à la prévison et au pré-chargement de documents.

Nous introduisons alors une nouvelle mesure appelée *CQTS* (Common Queries in Temporal Sequences) [DL02]. Elle évalue le nombre de requêtes communes entre deux séquences traduisant la régularité des accès et la quantité d'information d'une séquence pour prédire les accès sur une autre (cf. définition 14).

Définition 14 (CQTS) *Soient Q_i et Q_j deux séquences d'analyse ($i < j$), la mesure CQTS sur ces deux séquences est déterminée par :*

$$CQTS(Q_i, Q_j) = \frac{|\{x \in Q_i \mid x \in Q_j\}|}{|Q_i|}.$$

$CQTS(Q_i, Q_j)$ calcule la proportion de requêtes de Q_i portant sur des items de Q_j. Plus les accès sont similaires entre les deux séquences, traduisant alors une meilleure régularité, plus cette proportion est élevée.

Prenons un exemple, soient deux séquences $Q_1 = \prec C, B, A, B, D \succ$ et $Q_2 = \prec A, B, B, B, C \succ$, on obtient $CQTS(Q_1, Q_2)$=80%. Cela signifie que 80% des informations contenues dans Q_1 correspondent à des accès effectués dans Q_2. Si $Q_1 = \prec C, B, A, B, B \succ$, on obtient alors 100%.

7.1.3 Expérimentations

Protocole

CQTS est calculée sur chaque paire de séquences selon notre procédé d'analyse (cf. figure 7.2), par utilisateur et en moyenne sur tous les utilisateurs.

Nous évaluons dans un premier temps la valeur de *CQTS* en moyenne sur tous les utilisateurs, en fonction de l'écartement entre deux séquences (cf. section 7.1.1), sur chacune des deux séries (URL et termes).

Ensuite, nous présentons la distribution des utilisateurs selon la valeur maximale de *CQTS* obtenue par chacun avec les séquences basées sur les URL et sur les termes. Dans le reste du chapitre, la valeur maximale de *CQTS* est notée *cqmax*.

Dans chacune de ces expérimentations, nous confrontons les résultats de l'approche basée sur les URL avec ceux obtenus en utilisant les termes.

Données utilisées pour l'étude

Nous utilisons la totalité des données à notre disposition. Ces données sont issues de serveurs mandataires-caches opérationnels de France Télécom R&D Caen sur une durée de 17 mois. Elles concernent 331 utilisateurs et représentent plus de 1,5 millions de requêtes filtrées sur des documents de type texte, ce qui correspond en moyenne à 4563 requêtes par utilisateur. Chaque document est décrit par 10 termes. Pour plus d'informations se reporter à l'annexe E.

Résultats

La figure 7.3 présente la moyenne des valeurs de $CQTS$ obtenues sur l'ensemble des utilisateurs, en fonction de l'écartement entre deux séquences analysées. Différentes tailles de séquences sont utilisées (T=1, 2, 3, 5, 10, et 20%). Par exemple, le premier point d'une courbe indique la moyenne des valeurs de $CQTS$ obtenues avec des séquences consécutives (i.e. Δ=0).

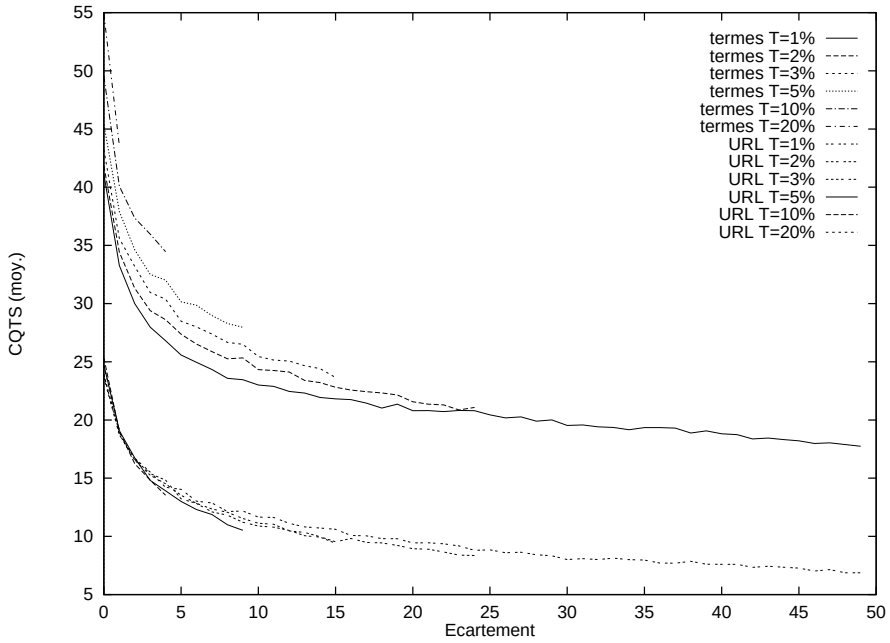

FIG. 7.3 – Valeur moyenne de $CQTS$ en fonction de l'écartement des séquences

Avec les termes, les courbes obtenues ont toutes des valeurs de $CQTS$ supérieures à celles obtenues avec les URL. Quelque soit le type d'item, la valeur moyenne maximale de $CQTS$ est réalisée lorsque l'écartement est le plus petit ($\Delta = 0$). Pour les URL, cette valeur moyenne maximale n'est pas très différente selon la taille des séquences étudiées (entre 23,5% et 25%). Cela signifie qu'en moyenne une information sur quatre dans une séquence donnée, est valide pour prédire les accès de la séquence suivante. Les différences sont plus marquées avec les termes. $CQTS$ est compris entre 41% et 55% ce qui correspond à quasiment une information sur deux. Les termes extraits des documents consultés apportent donc nettement plus d'information par rapport aux URL, afin de détecter des régularités et prédire de futurs accès.

Nous remarquons qu'aussi bien pour les URL que les termes, la valeur moyenne de $CQTS$ décroît rapidement en fonction de l'écartement des séquences. La période de validité d'une prédiction est donc très courte. Un système de pré-chargement de documents doit donc calculer des prévisions d'accès sachant que celles-ci auront une faible portée. De plus, la durée de la période d'analyse doit être adaptée en fonction de celle où va se porter les prévisions. Plus cette dernière est lointaine, plus la période d'analyse doit être grande pour connaître suffisamment les habitudes des utilisateurs.

Nous observons que la valeur moyenne de $CQTS$ en fonction de l'espacement des séquences comparées, suit une loi sous-exponentielle. Après transformation logarithmique (cf. figures 7.4 et 7.5), la valeur de $CQTS$ en fonction de Δ pour une taille de séquences donnée, est représentée par une quasi-droite. Les droites obtenues selon les différentes valeurs de T, ont un coefficient de régression de 0,99 ce qui montre une excellente corrélation.

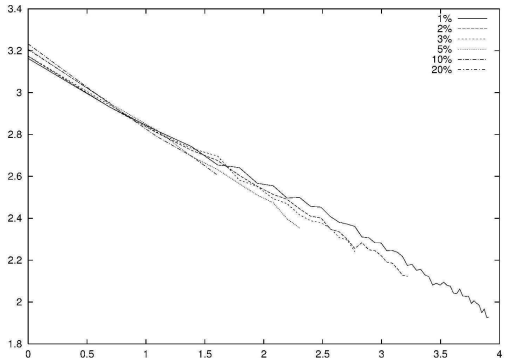

FIG. 7.4 – Application du logarithme aux courbes de la figure 7.3 (URL)

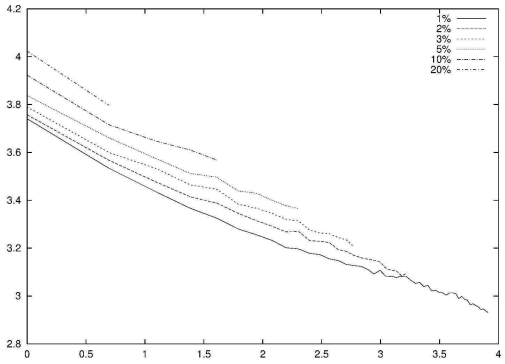

FIG. 7.5 – Application du logarithme aux courbes de la figure 7.3 (Termes)

La loi sous-exponentielle liant $CQTS$ et Δ s'exprime alors de la manière suivante. Soient $cqmax$ la valeur maximale de $CQTS$ et k une constante, nous avons donc : $log(CQTS) = -kTlog(\Delta) + log(cqmax)$, où $-kT$ est le coefficient directeur de la droite.

$$log(CQTS) = log(\Delta^{-kT}) + log(cqmax)$$
$$log(\frac{CQTS}{cqmax}) = log(\Delta^{-kT})$$
$$\frac{CQTS}{cqmax} = \Delta^{-kT}$$
$$\text{donc} \quad CQTS = cqmax.\Delta^{-kT}$$

Ces premières constatations suggèrent d'effectuer des études plus approfondies de la capacité de $CQTS$ à modéliser le comportement des utilisateurs. En effet, comme nous l'avons évoqué au début de cette section, les lois sous-exponentielles représentent une voie intéressante pour la modélisation du comportement des utilisateurs [Lan00a].

Les figures 7.6 et 7.7 présentent la distribution des utilisateurs selon la valeur maximale de $CQTS$, respectivement pour les URL et les termes. $cqmax$ est indiquée en abscisse. Le nombre d'utilisateurs correspond à l'ordonnée.

Avec les URL, nous constatons, qu'à l'exception des extrêmes, la répartition est relativement homogène. Pour $cqmax$ comprise entre 90 et 100%, on a 73 utilisateurs (22% du total).

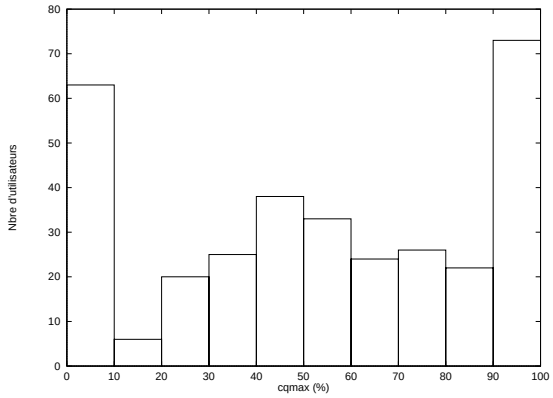

FIG. 7.6 – Distribution des utilisateurs selon *cqmax* (URL)

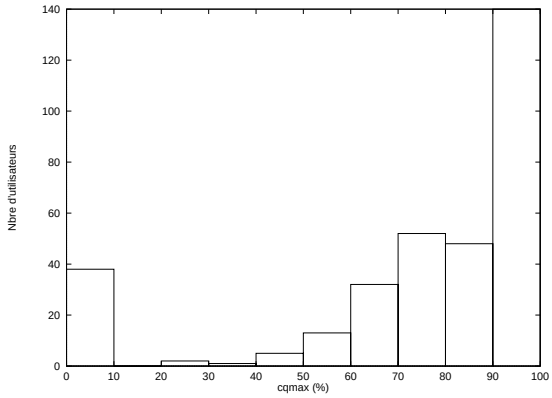

FIG. 7.7 – Distribution des utilisateurs selon *cqmax* (Termes)

La répartition est plus concentrée sur des valeurs élevées de *cqmax* pour les termes. 140 utilisateurs (42% du total) ont une *cqmax* supérieure à 90%. De plus, 38 utilisateurs ont une *cqmax* inférieure à 10% contre 63 utilisateurs pour le cas des URL. Cela montre donc que les termes produisent de meilleurs résultats.

7.1.4 Bilan

Nous avons montré en analysant des traces réelles de navigation que les termes décrivant les documents consultés permettent une description plus puissante que les URL. Nous avons observé une certaine cohérence des accès avec les termes grâce à notre méthode d'analyse.

L'explication de ces différences entre URL et termes tient à l'étendu possible du nombre d'items. Le nombre de termes croit vite puis est limité rapidement par le vocabulaire des langues. Le nombre d'URL est potentiellement très grand (combinaison des termes). Les deux ensembles ont des croissances très différentes suivant l'évolution des accès.

Les expérimentations ont aussi mis en évidence qu'il est nécessaire de calculer régulièrement des prévisions d'accès dans un système de pré-chargement. Les prévisions ont en effet une très courte portée. La durée de la période d'analyse permettant de calculer des prévisions, doit être adaptée en fonction de celle où va se porter les prévisions. Plus cette dernière est lointaine, plus la période d'analyse doit être grande pour apprendre suffisamment les habitudes des utilisateurs.

Nous avons aussi souligné la capacité intéressante de la self-similarité qui permet de modéliser le trafic web et le comportement des utilisateurs. Il paraît intéressant de poursuivre dans cette voie en se basant sur notre mesure de régularité $CQTS$ et d'effectuer des expérimentations complémentaires.

Les efforts et les temps de calcul supplémentaires pour extraire et gérer des termes de documents consultés, sont donc justifiés étant donné que les termes apportent des informations de meilleure qualité dans un but prédictif.

7.2 Détection de mots émergents

Pour une meilleure caractérisation des utilisateurs, il est recommandé d'utiliser les termes décrivant les documents qu'ils ont consultés. Dans cette section, nous proposons d'affiner cette caractérisation en détectant certains termes, appelés "mots émergents", traduisant les goûts et les tendances des utilisateurs [LD01]. De cette façon, nous obtenons des profils "prédictifs" plus adaptés à un contexte de pré-chargement.

7.2.1 Contexte

Nous définissons un système de variables attachées à un utilisateur. Ces variables correspondent à des termes de la langue française (ou autre) répartie en plusieurs catégories comme par exemple, un thème (végétal, animal, automobile, ...), un type (action, suspens, rire, classique, pop, ...), un support (film, documentaire, musique, ...) ou toute autre catégorie.

Chacun de ces termes est affecté d'une pondération reflétant le niveau d'importance du terme pour l'utilisateur. Ces pondérations sont calculées par itération au fur et à mesure que l'utilisateur accède à un document pouvant être décrit par les termes correspondant (cf. gestion de la personnalisation de la CyraNode, section 1.2.2).

L'ensemble de ces termes représente un espace algébrique multidimensionnel dans lequel évolue le profil global de l'utilisateur. Les différentes catégories ainsi que les relations entre les catégories sont stockées (par exemple dans un base de données, cf. annexe E) de manière à pouvoir reconstituer la chronologie d'évolution de chaque terme.

Prenons un exemple concernant des vidéos :

t_1 : L'utilisateur visionne un film d'action avec des automobiles.

Catégorie 1 : Thème
auto
1

Catégorie 2 : Type
action
1

Catégorie 3 : Support
film
1

t_2 : L'utilisateur regarde un documentaire sur les arbres

Catégorie 1 : Thème	
auto	arbre
1	1

Catégorie 2 : Type
action
1

Catégorie 3 : Support	
film	documentaire
1	1

t_3 : L'utilisateur visionne un documentaire sur les automobiles

Catégorie 1 : Thème	
auto	arbre
2	1

Catégorie 2 : Type
action
1

Catégorie 3 : Support	
film	documentaire
1	2

t_4 : L'utilisateur regarde une émission sur le sport automobile

Catégorie 1 : Thème	
auto	arbre
3	1

Catégorie 2 : Type
action
1

Catégorie 3 : Support		
film	documentaire	sport
1	2	1

7.2.2 Méthode de détection d'émergences

Compte tenu du contexte que nous avons fixé précédemment, chaque terme peut être représenté dans un repère chronologique. Les figures 7.8 et 7.9 illustrent l'évolution des termes `auto` et `arbre` de notre exemple. Nous remarquons que le terme *auto* est dominant par rapport au terme *arbre*. La pondération du terme `auto` sur la période $]t_2, t_3]$ est égale à 2. Elle augmente encore après cette période. Par contre, la pondération du terme `arbre` stagne à 1.

La pondération d'un terme w sur une période i est notée P_w^i. Par exemple, $P_{auto}^{]t_2,t_3]}$=2 et $P_{arbre}^{]t_2,t_3]}$=1.

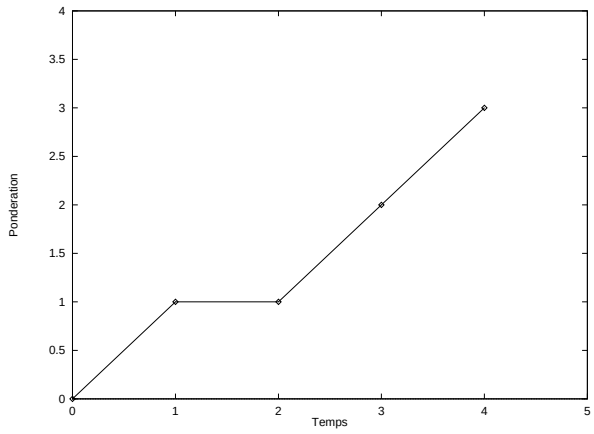

FIG. 7.8 – Evolution du terme *auto*

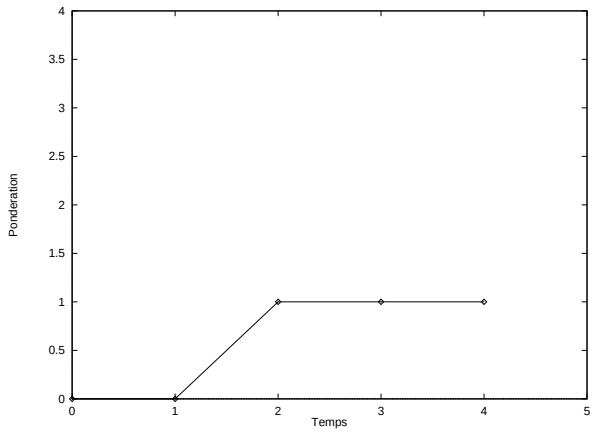

FIG. 7.9 – Evolution du terme *arbre*

Pour une période et un terme donnés, si la variation de la pondération du terme entre cette période et la période précédente est supérieure ou égale à un seuil S, alors le terme est un *mot émergent* (cf. définition 15).

Définition 15 (Mot émergent) *Soient Q_i et Q_j deux périodes telles que $i < j$, w un terme, et S un seuil. w est un mot émergent sur Q_j par rapport à Q_i si :*

$$\Delta P_w^{i,j} = P_w^j - P_w^i \geq S.$$

Le seuil correspond à la moyenne des variations du terme observées sur de précedentes périodes :

$$S = \frac{1}{nb} \sum_{l=1}^{nb} \Delta P_w^{l-1,l}$$

où nb est le nombre de périodes précédentes.

Si nb=1, seule la période précédente (i dans la définition) est prise en compte. Le terme est alors émergent si sa pondération a simplement augmenté entre les deux périodes.

La méthode de détection d'émergences est appliquée pour chaque terme. L'ensemble des mots émergents détectés sur une période pour un utilisateur est alors utilisé pour le caractériser.

7.2.3 Applications

La caractérisation des utilisateurs basée sur les mots émergents, peut être utilisée dans différents contextes d'accès à l'information, actifs (accès au Web) ou passifs (télé-diffusion).

Par exemple, connaissant les mots émergents d'un utilisateur (ou de plusieurs), un diffuseur peut anticiper et diffuser un certain type d'information et un contenu cohérent avec les évolutions des intérêts de cet utilisateur.

Elles présentent aussi un grand intérêt pour le pré-chargement (cf. chapitre 6) et la recommandation de documents (cf. chapitre 2). Par exemple, à partir des mots émergents, il est intéressant d'anticiper des recherches d'informations et de proposer les résultats aux utilisateurs. Le chapitre suivant propose l'utilisation des mots émergents pour le pré-chargement d'information. Les documents récupérés grâce aux mots émergents sont alors chargées dans un cache et disponibles pour d'autres utilisateurs.

7.3 Conclusion

Les expérimentations que nous avons menées sur des fichiers de trace de serveurs mandataires, ont montré la supériorité de l'utilisation des termes décrivant les documents consultés face aux URL de ces documents. Pour cela, une nouvelle mesure de régularités des accès a été introduite dans un but prédictif. Cette mesure permet en effet de déterminer la quantité d'information valide sur une séquence pour prédire des accès sur une séquence suivante. Les expérimentations ont aussi fait ressortir la nécessité de déterminer fréquemment les prévisions car elles ont une rapide péremption. D'où l'importance de disposer, par exemple dans un système de pré-chargement de documents, d'une gestion de la durée de la période utilisée pour réaliser des prédictions, et de paramètres pour la modifier. Nous avons aussi observé un comportement self-similaire de notre mesure en fonction du temps. La self-similitude est très utilisée dans la modélisation du trafic web. Il serait donc intéressant de poursuivre dans cette voie et d'effectuer des expérimentations supplémentaires pour aboutir à un éventuel modèle du comportement des utilisateurs.

La caractérisation des utilisateurs doit donc se baser sur les termes des documents qu'ils ont consultés. Nous avons proposé d'affiner cette caractérisation en ne sélectionnant que certains termes appelés mots émergents. Pour cela, nous avons défini une méthode de détection d'émergences. Elle calcule le nombre de fois qu'un terme fait référence à un document consulté sur une période, puis compare cette valeur avec celle obtenue sur une période précédente. Si la variation est supérieure à un certain seuil, alors le terme est un mot émergent. La caractérisation d'un utilisateur sur une période correspond à l'ensemble des mots émergents détectés sur cette période par rapport à la précédente. Ces profils utilisateurs présentent un intérêt important pour la diffusion ciblée d'information aux utilisateurs, la recommandation et le pré-chargement de documents.

Dans le chapitre 8, nous proposons un nouveau système de pré-chargement appelé PrefetchEm. Ce dernier utilise la caractérisation des utilisateurs basée sur les mots émergents pour pré-charger des documents dans un cache. Nous effectuons une première évaluation de ce système dans le contexte général des serveurs mandataires-caches.

Chapitre 8

Nouveau système de pré-chargement de documents basé sur la détection d'information émergente

Sommaire

Ce chapitre correspond à la réalisation pratique de nos travaux sur le pré-chargement de documents dans les caches. Nous décrivons un nouveau système de pré-chargement basé sur la détection d'information émergente [LD04]. Ce système a pour originalité de trouver de nouveaux documents susceptibles d'intéresser les utilisateurs. Ces documents n'ont a priori aucun liens directs (au sens hypertexte) avec d'autres précédemment consultés, contrairement aux méthodes classiques qui ne se basent que sur les historiques de consultations et les liens hypertextes (cf. chapitre 6). De ce fait, notre système est assez ambitieux. Les mots émergents (cf. chapitre 7) détectés sur une période sont injectés dans un moteur de recherche sur le Web, de façon à trouver des documents ayant un lien thématique. Certains de ces documents sont ensuite pré-chargés dans un cache, et seront alors disponibles pour la période suivante.

Ce système est développé sous la forme d'un module pour CYRANO (cf. chapitre 1), et aussi dans un logiciel autonome appelé PrefetchEm [DL03] intrégrant un serveur mandataire-cache Squid. Nous réalisons des expérimentations avec ce logiciel en rejouant des fichiers de trace de serveurs mandataires. Nous évaluons l'apport du système en comparant les résultats avec ceux obtenus par un serveur mandataire Squid sans pré-chargement.

8.1 Description du système

Notre système est séquentiel. Il effectue une analyse périodique des accès des utilisateurs et procède au téléchargement anticipé de documents.

Les mots émergents (cf. section 7.2) sont détectés à intervalles réguliers pour chacun des utilisateurs de la CyraNode ou du serveur mandataire. Les mots émergents liés à un utilisateur sont ensuite injectés dans un moteur de recherche sur le Web, pour trouver des documents en rapport avec ses intérêts. On bénéficie de cette façon de technologies semblables à celles mises en œuvre par les moteurs de recherche.

Au final, une sélection de documents les plus fréquemment recontrés dans les résultats des recherches, est pré-chargée dans l'espace disque de la CyraNode ou dans le cache du serveur mandataire. Si un des documents pré-chargés est demandé par un utilisateur sur la période suivante, il lui est immédiatement envoyé. Cela permet ainsi de réduire considérablement le temps de téléchargement et de latence (amélioration de la QoS).

Le système est constitué de deux étapes : la recherche de documents, et le téléchargement de certains de ces documents.

Etape 1 : recherche de documents pour chacun des utilisateurs
- détection des mots émergents sur une période i par rapport à la période $i-1$
- recherche de documents en intérogeant un moteur de recherche sur le Web, avec comme arguments les mots émergents.

Etant donné que les mots émergents liés à un utilisateur sont injectés dans un moteur de recherche, leur nombre doit être limité. Seuls les e mots les plus émergents sont sélectionnés. Un mot est émergent dès que sa pondération a augmenté (seuil S fixé à 1).

Nous autorisons aussi p nouveaux mots (i.e. n'apparaissant pas dans la période $i-1$). Cela permet d'éviter une stagnation, et de faire évoluer l'ensemble des mots.

Prenons comme exemple les tableaux de la figure 8.1. Si on recherche au maximum deux mots émergents sans autoriser de nouveaux mots, sur la période i par rapport à $i-1$, on obtient alors : `sécurité` et `réseau`. Ces deux mots représentent les plus fortes progressions. Par contre, si on recherche deux mots émergents en autorisant un nouveau, `sécurité` et `java` sont sélectionnés.

Période $i-1$	
Terme	Pondération
ip	9
sécurité	8
réseau	3
moto	2

Période i	
Terme	Pondération
ip	8
sécurité	**12**
réseau	*4*
JAVA	6

FIG. 8.1 – Exemple de détection de mots émergents

La recherche de document est effectuée grâce à un module séparé. Il est paramètré par le moteur choisi, le type de requête ("ou" / "et"), et les types de documents recherchés (HTML, images, vidéos, pdf, ps, ...). Les différents moteurs intérogeables par notre

système sont : Altavista[19], Voila[20], Lycos[21], Caloga[22] et Singingfish[23]. Par défaut, des documents HTML sont recherchés sur Altavista avec une disjonction des mots envoyés.

Etape 2 : chargement d'une sélection de documents
- les N documents les plus présents dans les résultats des recherches effectuées, sont téléchargés dans un espace disque dédié, jusqu'à une taille totale maximum fixée (notée $Tmax$).

Le fait de charger les documents qui apparaissent le plus dans les différentes recherches permet d'augmenter la probabilité que ces documents soient demandés par la suite. On bénéficie alors d'un effet de mutualisation, comme dans le fonctionnement d'un serveur mandataire-cache (cf. section 1.2.1).

Le système a donc 5 paramètres :
- d, la durée des périodes
- e, le nombre de mots émergents sélectionnés par utilisateur
- p, le nombre de nouveaux mots autorisés
- N, le nombre maximum de documents téléchargés
- $Tmax$, la taille de l'espace disque alloué pour le pré-chargement.

Rappelons que nous avons noté l'importance de pouvoir gérer et modifier les paramètres d, N et $Tmax$, lors de nos précédentes réflexions (cf. sections 6.3 et 7.3).

Le processus de pré-chargement est lancé tous les jours à un instant défini par l'administrateur du système. Cette périodicité est identique à celle utilisée dans le système séquentiel de pré-chargement de Kim et al. (cf. section 6.2.3).

8.2 Architecture générale

8.2.1 Module de la CyraNode

Pour CYRANO, le système est développé dans un module qui interagit avec la base de données de la CyraNode (cf. section 1.2.2), et aussi avec le module de gestion de l'espace de stockage de manière à ajouter les documents pré-chargés. La détection des mots émergents utilise la base de données de la CyraNode qui stocke tous les accès et les termes décrivant les documents consultés par les utilisateurs. Les paramètres du pré-chargement sont bien entendu modifiables par l'administrateur de la CyraNode, lors de l'activation du module.

[19] *http://www.altavista.com*
[20] *http://www.voila.fr*
[21] *http://www.lycos.fr*
[22] *http://web.caloga.com*
[23] *http://www.singingfish.com*

8.2.2 Logiciel PrefetchEm

Le logiciel PrefetchEm [DL03] est autonome. Il est constitué d'un serveur mandataire-cache Squid auquel nous avons ajouté une base de données identique à celle développée pour CYRANO. Les accès des utilisateurs ainsi que les termes des documents consultés sont stockés dans cette base par un processus qui analyse en permanence les fichiers de trace du Squid. Le format de ces fichiers est présenté dans la section E.1.1 de l'annexe E. A cela s'intégre le module de pré-chargement évoquée précédemment. La figure 8.2 présente l'architecture générale du logiciel.

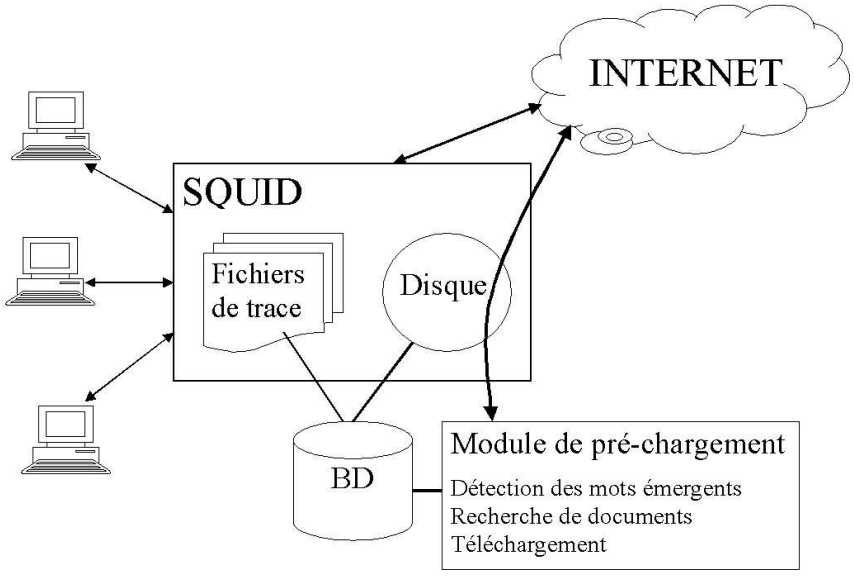

FIG. 8.2 – Architecture générale de PrefetchEm

Lorsqu'un utilisateur consulte une URL, une ou plusieurs lignes apparaissent dans le fichier de trace du Squid. Le processus d'alimentation de la base de données interprète ces lignes et insère dans la base les informations correspondante (date de consultation, identifiant de l'utilisateur, URL, ...). Des termes sont ensuite extraits à partir du contenu de l'URL (cf. annexe E) pour la décrire. Le contenu peut se trouver dans le cache du Squid, sinon il est récupéré à partir du site d'origine.

8.3 Expérimentations

8.3.1 Protocole

Nous évaluons le fonctionnement de PrefetchEm en simulant des accès contenus dans des fichiers de trace de serveurs mandataires [LD04]. Les performances sont calculées en analysant les fichiers de trace produits par le Squid intégré à PrefetchEm, avec les mesures présentées dans la section 6.1.1 du chapitre 6. Nous comparons les résultats obtenus avec ceux produits avec les mêmes accès sur un Squid sans pré-chargement (cf. figure 8.3). La même taille de cache est utilisée pour le Squid classique et PrefetchEm. Le rejeu des accès

est réalisé par un script utilisant la commande *wget*[24] permettant de récupérer le contenu d'URL sur Internet.

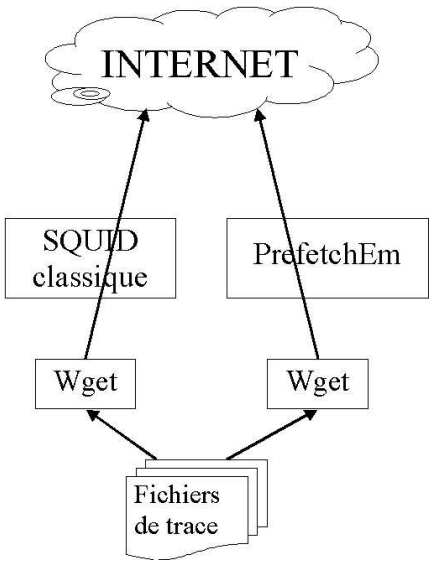

FIG. 8.3 – Simulation et évaluation de PrefetchEm par rapport à un Squid classique

Le temps indiqué pour chaque accès dans les traces est utilisé pour simuler le fonctionnement du logiciel dans des conditions réelles. Les accès sont rejoués jusqu'à ceux effectués à 19h dans la réalité. Le rejeu est alors suspendu pendant le temps nécessaire à pré-charger des documents. Lorsque les opérations de pré-chargement sont terminées, le rejeu reprend jusqu'aux accès effectués à 19h le jour suivant, etc.

d est fixé à 4 heures, e à 3, et p à 1 [LD04] (cf. section 8.1). Les mots émergents sont donc déterminés sur les quatre dernières heures (15h-19h) par rapport aux quatre heures précédentes (11h-15h). Pour chaque utilisateur, 3 mots émergents dont un nouveau, sont détectés et envoyés au moteur de recherche Altavista. Les requêtes sont du type "ou", et portent sur des documents textuels (HTML, txt, pdf, ...). Le choix de 3 mots pour constituer les requêtes est motivé par les résultats d'études statistiques[25]. Ces dernières montrent que les requêtes envoyées aux moteurs de recherche comportant 3 mots représentent 25,61% (au niveau mondial en décembre 2003 et janvier 2004), et arrivent en seconde position derrière celles composées de 2 mots.

Nous testons plusieurs politiques de sélection de documents parmi les résultats fournis par le moteur de recherche. La première est appelée $prefetchEm10$, et correspond au chargement des 10 premières URL obtenues pour chaque utilisateur. La seconde, $prefetchEm2+$, télécharge les URL présentes dans au moins 2 résultats de recherche, permettant ainsi de bénéficier d'un effet de mutualisation. Dans toutes nos expérimentations, $Tmax$ n'est pas limitée.

[24] *http://www.gnu.org/software/wget/wget.html*
[25] *http://www.journaldunet.com*

Une expérience supplémentaire ($prefetchEmAll20$) est effectuée en considérant l'ensemble des utilisateurs comme un seul utilisateur. Les mots émergents traduisent alors les tendances de l'ensemble des utilisateurs. Les premières URL obtenues par une unique recherche sont donc pré-chargées (20 URL maximum).

Les URL pré-chargées sont consignées dans un fichier de trace interne au module de pré-chargement. Il permet en complément du fichier de trace du Squid de calculer les améliorations induites par le système (nombre de requêtes sauvées, ...). Les calculs des taux de hit sont effectués en supposant une taille de cache infini, ce qui est courant dans les simulations de pré-chargement [Dav02].

Notons que cette simulation est coûteuse et longue à effectuer. Elle demande en effet une bonne fiabilité du réseau et des ressources importantes.

8.3.2 Données utilisées

Le rejeu des accès implique l'utilisation de fichiers de trace très récents pour avoir une forte probabilité que les URL soient encore valides. Les données utilisées lors de nos expérimentations dans les chapitres précédents sont relativement anciennes. Nous avons donc utilisé des accès récents d'un trafic réel de France Télécom R&D. Seules les requêtes correspondant aux documents textuels ont été gardées de façon à pouvoir extraire des termes. Les fichiers de trace obtenus couvrent 7 jours correspondant à 791069 requêtes effectuées sur 227325 URL par 2000 utilisateurs. Remarquons que le nombre de jours n'est pas très élevé car comme nous l'avons dit précédemment cette expérimentation est très coûteuse.

8.3.3 Résultats

Le tableau 8.1 présente les résultats obtenus en moyenne pour un jour. Le Squid de référence (sans pré-chargement) est appelé *classique*.

Nous indiquons pour chacune des expériences les valeurs moyennes des mesures d'évaluation sur les 7 jours : le taux de hit, le nombre de requêtes sauvées, le rendement, la bande passante sauvée, la bande passante gaspillée. La proportion de bande passante sauvée par rapport à la bande passante totale induite par le pré-chargement est aussi indiquée. Le gain de taux de hit par rapport au Squid classique est reporté en dernière ligne.

Rappelons que le nombre de requêtes sauvées correspond au nombre d'URL pré-chargées consultées une première fois. On ne comptabilise pas les autres fois car les URL peuvent être mises dans le cache lorsqu'elles sont consultées une première fois, et le fait de les avoir pré-chargées ne change rien sur les consultations suivantes de ces URL. Néanmoins, le nombre de requêtes effectuées sur des URL pré-chargées par le système est un bon indicateur sur la popularité de ces dernières et donc sur une certaine pertinence des prévisions réalisées. Cette information est donc reportée dans le tableau.

	classique	*prefetchEm*10	*prefetchEm*2+	*prefetchEmAll*20
Taux de hit (%)	71,55	71,3	**74,13**	71,65
Nb. URL pré-chargées	-	9221	1055	14
Nb. requêtes sauvées	-	115	20	1
Nb. total de requêtes sur URL pré-chargées	-	1595	709	38
Rendement (%)	-	1,25	1,77	7,70
Bande passante sauvées (en Ko)	-	9520,3	2203,1	42,8
Bande passante gaspillées (en Ko)	-	557018,3	84996,1	*871,01*
Bande passante sauvée / pré-chargée (%)	-	1,68	2,52	4,68
Gain taux de hit (%)	-	0	**+3,6**	+0,14

TAB. 8.1 – Résultats de la simulation de PrefetchEm (en moyenne pour un jour)

PrefetchEm10

Pour *prefetchEm*10, le taux de hit moyen sur les 7 jours est légèrement inférieur à celui du Squid *classique*. Cela est du à quelques erreurs de téléchargement d'URL causées par des problèmes réseaux et d'indisponibilité du serveur Web d'origine. Au lieu d'obtenir des *hit* dans les fichiers de trace, des codes d'erreur apparaissent.

Les résultats montrent qu'une approche purement individuelle n'est pas envisageable en pratique. Le nombre d'URL pré-chargées est élevée alors que très peu d'entre elles sont consultées (en moyenne 115 sur 9221). La bande passante gaspillée est donc très importante. La proportion de bande passante induite par le pré-chargement qui est réellement utile est de 1,68%. Cette solution est donc à écarter.

PrefetchEmAll20

Si on considère l'ensemble des utilisateurs du serveur mandataire comme un seul utilisateur (*prefetchEmAll*20) et que l'on pré-charge au maximum 20 URL, on contrôle parfaitement la bande passante gaspillée (seulement 871,01 Ko). Cependant, les URL pré-chargées sont très peu consultées (une en moyenne). Cela est du en partie au faible nombre d'URL pré-chargées, mais surtout aux possibles divergences de sujets obtenus lors de la détection des mots émergents. Ces derniers sont en effet moins précis qu'une détection par utilisateur. Cela a donc des répercussions importantes sur les résultats des recherches de documents.

PrefetchEm2+

L'expérience *prefetchEm*2+ correspond à l'amélioration la plus élevée du taux de hit (+3,6%). L'effet de mutualisation a une forte influence sur la sélection des documents téléchargés : 9 fois moins de documents comparé à *prefetchEm*10. Le nombre d'URL

pré-chargées consultées est plus faible, étant donné que l'on télécharge moins d'URL, mais le rapport entre la bande passante sauvée et la bande passante utilisée pour le pré-chargement est meilleur (2,52% face à 1,68% pour $prefetchEm10$).

Bilan

Pour chacune des expériences, le nombre d'URL pré-chargées et consultées par la suite est faible, ce qui se traduit par des rendements peu élevés. Cependant, nous remarquons que le nombre de requêtes sur ces URL est relativement important. Par exemple, 709 requêtes sur 20 URL pré-chargées pour $prefetchEm2+$. Cela montre que certaines URL trouvées sont très demandées et présentent donc un grand intérêt pour les utilisateurs.

Notre système permet donc de trouver des documents qui intéressent fortement les utilisateurs. Néanmoins, la proportion de ces documents par rapport à l'ensemble des documents pré-chargés n'est pas assez élevée pour que cela se traduise dans le calcul du rendement.

8.4 Conclusion

Nous avons proposé un nouveau système de pré-chargement de documents dont l'originalité est de trouver des documents axés sur les intérêts des utilisateurs mais n'ayant a priori aucun liens hypertextes avec les documents dèjà consultés.

Cette approche est adaptée à notre problème. Rappelons qu'une CyraNode telle que définie initialement dans le projet, ne dispose pas de processus actif pour récupérer des documents (cf. section 1.3). Dorénavant, elle est en mesure de rechercher et télécharger des documents susceptibles d'intéresser les utilisateurs. Nous avons en effet développé un module de pré-chargement s'intégrant dans une CyraNode. L'administrateur peut paramètrer ce module de telle sorte que des documents soient pré-chargés tous les jours en accord avec les mots émergents détectés.

Nous avons aussi présenté le logiciel PrefetchEm qui correspond à une implantation autonome du système dans le contexte général des serveurs mandataires-caches. Les simulations ont été effectuées en rejouant des accès extraits de fichiers de trace de serveurs mandataires-caches de France Télécom R&D. Le contexte des expérimentations est assez difficile. En effet, elles ont demandé des ressources et une fiabilité du réseau sur une longue période. Les résultats ont montré que le rendement est assez faible. Il est vrai que notre approche est assez ambitieuse car contrairement aux méthodes existantes de pré-chargement, nous n'utilisons pas les URL et les liens hypertextes des documents déjà consultés (cf. chapitre 6), mais les termes de ces documents qui représentent un niveau plus abstrait. Cependant, nous avons remarqué que les URL pré-chargées présentent un grand intérêt pour les utilisateurs du fait du nombre élevé de requêtes sur celles-ci. De plus, par rapport à un Squid sans pré-chargement, le taux de hit du cache est amélioré en moyenne de 3,6% dans le cas d'une détection individuelle des mots émergents et d'une sélection des documents mutualisant les résultats.

Les résultats indiquent que le système est capable de trouver des documents très demandés par les utilisateurs. Ces derniers seraient peut-être intéressés par les autres documents pré-chargés mais ne savent pas qu'ils sont disponibles. Il serait donc plus rentable de suggérer aux utilisateurs ces documents. Ainsi, nous pourrions coupler notre système de pré-chargement avec la recommandation de documents (cf. chapitres 2 et 5). Les utilisateurs de la CyraNode serait alors avisés de l'arrivée d'un nouveau document susceptible de les intéresser.

Nos expérimentations tendent à montrer que la mutualisation des résultats obtenus pour chaque utilisateur produit de meilleures prévisions que des traitements purement individuels. Néanmoins, un traitement de l'ensemble des utilisateurs vu comme un seul, n'est pas efficace. Il serait donc intéressant d'étudier l'utilisation de communautés d'intérêt où le regroupement des utilisateurs serait basé sur les mots émergents. Pour cela, nous pouvons utilisé ECCLAT (cf. chapitre 4) afin de bénéficier du chevauchement possible entre les communautés permettant de capter les différents centres d'intérêt des utilisateurs. De plus, ECCLAT produit les descriptions des communautés qui correspondraient dans notre cas à des associations de mots émergents. La recherche et le pré-chargement de documents pourrait donc s'effectuer avec la description de chaque communauté.

Conclusion générale

L'objectif de ce travail était de développer des systèmes pour améliorer l'accès à l'information à la fois en terme de rapidité et de sélection. Nos travaux se sont déroulés au sein du projet RNRT CYRANO qui a pour but de rapprocher vers les utilisateurs les contenus susceptibles de les intéresser, et pour cela met en œuvre une réplication sélective de l'information. Nous avons fortement participé aux divers développements du projet, et tout particulièrement le module de gestion de la personnalisation d'une CyraNode (élément composant le système CYRANO). Afin d'incorporer des services de personnalisation aux CyraNodes, nous avons proposé des systèmes adéquats de recommandation de documents aux utilisateurs pour faciliter la sélection de l'information, et de pré-chargement d'information pour accélérer les temps de téléchargement.

Notre première contribution est la proposition d'un système de recommandation de documents reposant sur une nouvelle méthode de découverte de clusters, appelée Ecclat, à partir de données qualitatives. Chaque cluster correspond à une communauté d'utilisateurs identifiée automatiquement selon des intérêts partagés. Ecclat a la particularité de produire des clusters *chevauchants* où une transaction (dans notre cas un utilisateur) peut appartenir à plusieurs clusters. De plus, Ecclat fournit une *description* de chaque cluster sous forme d'items vérifiés par toutes les transactions du cluster. Pour notre application, les items correspondent à des termes de documents consultés.

La notion de motif fermé fréquent est utilisée de façon originale par Ecclat pour former les descriptions des clusters. Les transactions vérifiant tous les items d'un motif fermé fréquent sont associées à ce motif pour former un bi-set. Les bi-sets sont alors considérés comme des clusters potentiels. L'atout majeur des motifs fermés fréquents est la maximalité. En effet, un motif fermé fréquent représente l'ensemble maximal d'items partagés par un ensemble de transactions. Il n'existe donc pas d'item supplémentaire vérifié par ces transactions. Ce point est capital dans un but de clustering.

Ecclat sélectionne certains bi-sets à partir du treillis des motifs fermés fréquents (aussi appelé treillis des concepts fréquents) afin d'identifier des communautés d'intérêt. Pour une recherche efficace sur de grandes quantités de données, nous utilisons les algorithmes récents de fouille de données pour découvrir les motifs fermés fréquents. Contrairement aux méthodes existantes, notre approche n'utilise pas de mesure globale de similarité entre les transactions, mais est basée sur un nouveau critère d'évaluation d'un motif appelé *intérêt*, et qui est composé de l'*homogénéité* et de la *concentration*. Le calcul de ces mesures et la sélection des clusters requièrent un faible coût de calcul. Notons que le nombre de clusters résultats (qu'il est difficile d'estimer a priori) n'est pas fixé par l'utilisateur.

Le chevauchement entre les clusters est très important dans notre contexte d'application web, montrant ainsi différents points de vue. Grâce au chevauchement, un utilisateur peut être présent dans plusieurs communautés correspondant à divers centres d'intérêt. Les algorithmes de clustering autorisant le chevauchement d'éléments sont rares. Les méthodes existantes, comme *Fuzzy c-Medoids* et *PoBOC*, ne construisent pas de descriptions des clusters obtenus. A notre connaissance, ECCLAT est la seule méthode à produire directement des clusters non-disjoints et leurs descriptions.

Nous avons effectué différentes évaluations d'ECCLAT, à la fois sur des benchmarks de façon à comparer les résultats avec ceux d'autres méthodes, et sur des données réelles issues du domaine médical et de France Télécom R&D. ECCLAT a montré de bonnes capacités à trouver des clusters purs par rapport à une valeur de classe, face à d'autres méthodes sur les benchmarks. De plus, nous avons montré que les descriptions des clusters donnent un avantage certain à ECCLAT pour l'interprétation et l'utilisation des clusters. Notons qu'il est tout à fait possible d'obtenir des chevauchements éventuels d'items entre les descriptions des clusters. Cela peut aussi apporter des informations intéressantes sur la caractérisation de classes ou montrer des rôles plus ou moins importants de certains items. Lors de nos expérimentations sur des données médicales pour le Discovery Challenge PKDD 2002, nous avons découvert des clusters de patients correspondant aux différents stades de fibrose du foie (niveau de gravité d'une hépatite). De tels clusters décrits par les examens sur les patients, ont fait apparaître quelques items liés aux stades sévères de la maladie. Lors du Discovery Challenge PKDD 2004, nous avons confronté ECCLAT avec *PoBOC* sur des données relatives à l'athérosclérose. Nous avons observé que les clusters produits sont différents, mais l'interprétation des résultats tend vers les mêmes conclusions. Les deux algorithmes ont mis en évidence le rôle important de la tension artérielle pour définir des profils pronostics. Cette comparaison a aussi confirmé l'avantage d'ECCLAT à produire directement les descriptions des clusters. En effet, l'étape de post-traitement nécessaire pour calculer les descriptions des clusters produits par *PoBOC*, est délicate et peut conduire à ne trouver aucune description. De plus, les descriptions s'avèrent moins précises que celles produites par ECCLAT.

En ce qui concerne la recommandation de documents, les descriptions des clusters sont utilisées pour calculer un taux de recouvrement avec la description d'un document qui sera éventuellement recommandé aux utilisateurs des clusters. Notre système est autonome et hybride, car nous effectuons une découverte automatique de clusters d'utilisateurs en se basant sur le contenu des documents, et non sur les préférences, ou des notes données par des utilisateurs, ou les documents eux-mêmes. Pour une utilisation dans CYRANO, nous nous sommes concentrés sur les fonctions de filtrage et de distribution de documents. Nous avons évalué notre méthode en utilisant des fichiers de trace de serveurs mandataires de France Télécom R&D. Ces expérimentations correspondent à une simulation de fonctionnement de notre système de recommandation. Les expérimentations ont été effectuées avec des clusters obtenus avec ECCLAT et aussi avec des partitions produites par *ARHP*. L'utilisation des clusters produits par ECCLAT ont permis d'obtenir de meilleurs résultats, ce qui montre l'avantage du chevauchement entre les clusters.

Notre deuxième contribution est la définition d'un nouveau système de pré-chargement de documents ayant la particularité de ne pas utiliser les liens hypertextes des documents

déjà consultés par les utilisateurs. Ce point est impératif dans le contexte CYRANO, car les documents présents dans une CyraNode n'ont pas a priori de liens directs au sens hypertexte. Or, les approches existantes se basent principalement sur les historiques de consultations, les séquences d'URL visitées, et les liens hypertextes. Nous avons donc étudié l'utilisation d'informations de plus haut niveau à savoir les termes. Pour cela, nous avons confronté les URL et les termes extraits de ces URL pour la détection de régularités. Une nouvelle mesure de régularités des accès dans un but prédictif a été introduite. Cette mesure permet de déterminer la quantité d'information valide sur une période d'accès pour prédire des accès sur une période suivante. Les expérimentations que nous avons menées sur des fichiers de trace de serveurs mandataires de France Télécom R&D, ont montré la supériorité de l'utilisation des termes décrivant les documents consultés face aux URL de ces documents, pour détecter des régularités dans les accès des utilisateurs.

La caractérisation des utilisateurs doit donc se baser sur les termes des documents qu'ils ont consultés. Nous avons proposé d'affiner cette caractérisation en ne sélectionnant que certains termes appelés *mots émergents*, pour traduire les goûts et les tendances des utilisateurs d'une façon plus dynamique dans un but prédictif. Nous avons défini une méthode de détection d'émergences. Elle calcule le nombre de fois qu'un terme fait référence à un document consulté sur une période, puis compare cette valeur avec celle obtenue sur une période précédente. Si la variation est supérieure à un certain seuil, alors le terme est un mot émergent. La caractérisation d'un utilisateur sur une période correspond à l'ensemble des mots émergents détectés sur cette période par rapport à la précédente. Ces profils utilisateurs présentent un intérêt important pour des applications comme la diffusion ciblée d'information aux utilisateurs et le pré-chargement de documents.

Notre système de pré-chargement utilise les mots émergents pour trouver des documents via un moteur de recherche sur le web. Nous avons développé un module de pré-chargement pour les CyraNodes, ce qui leur permet de disposer d'un processus actif pour récupérer des documents susceptibles d'intéresser les utilisateurs. L'administrateur peut paramètrer ce module de telle sorte que des documents soient pré-chargés tous les jours en accord avec les mots émergents détectés.

Nous avons aussi présenté le logiciel PrefetchEm qui correspond à une implantation autonome du système de pré-chargement dans le contexte général des serveurs mandataires-caches. Les simulations ont été effectuées en rejouant des accès extraits de fichiers de trace de serveurs mandataires-caches de France Télécom R&D. Les résultats ont montré que le rendement est assez faible. Il est vrai que notre approche est assez ambitieuse car contrairement aux méthodes existantes de pré-chargement, nous n'utilisons pas les URL et les liens hypertextes des documents déjà consultés, mais des termes qui sont plus abstraits. Cependant, nous avons remarqué que des URL pré-chargées présentent un grand intérêt pour les utilisateurs du fait du nombre élevé de requêtes sur celles-ci. De plus, par rapport à un serveur mandataire-cache Squid sans pré-chargement, le taux de hit du cache est amélioré en moyenne de 3,6% dans le cas d'une détection individuelle des mots émergents et un pré-chargement des documents apparaissant dans plusieurs résultats de recherche obtenus pour les utilisateurs. Nos expérimentations tendent à montrer que cette mutualisation des résultats produit de meilleures prévisions que des traitements purement individuels.

Discussion et perspectives

Le système de recommandation que nous avons proposé a vocation à fonctionner dans un contexte opérationnel. L'utilisation de clusters d'utilisateurs nécessite donc qu'ils soient fréquemment mis à jour. Nous pouvons supposer que les centres d'intérêt des utilisateurs ne changent pas brusquement, et dans ce cas utiliser un processus qui recherche périodiquement les clusters d'utilisateurs (par exemple tous les soirs). Mais, il serait plus adapté de développer une version incrémentale d'ECCLAT permettant alors de proposer un système pseudo temps réel. Les voies possibles pour réaliser cela, sont les techniques de construction de treillis de Galois [GMA95], et les algorithmes de découverte incrémentale de motifs fréquents [TC00], qu'il faudrait adapter.

Nos travaux sur le clustering (cf. chapitre 4) ont montré l'intérêt de calculer simultanément les clusters et leurs descriptions. Les algorithmes de bi-clustering, comme par exemple *Bi-Clust*, effectuent cela mais n'autorisent pas le chevauchement de transactions, ni d'items. Or nos travaux peuvent permettre de former des bi-partitions avec un léger chevauchement. La mesure d'*intérêt* d'un cluster candidat que nous avons proposé dans ECCLAT, repose sur des calculs portant sur les items du cluster et aussi sur ses transactions. Nous ne contrôlons que le chevauchement au niveau des transactions. Néanmoins, la dualité présente dans la notion de concept, nous laisse envisager la possibilité de travailler sur les données transposées et d'effectuer un contrôle du chevauchement d'items. Cela serait un pas important vers la mise au point d'une méthode de bi-clustering pour former une bi-partition approchée. Notons, qu'il ne serait alors pas nécessaire de réaliser en réalité la transposition [RC03b], les mesures "transposées" pouvant être déterminées dans l'algorithme calculant nos mesures sur les données originales.

Concernant le pré-chargement de documents, les résultats de nos expérimentations indiquent que PrefetchEm est capable de trouver des documents très demandés par les utilisateurs, mais un petit nombre d'entre eux sont réellement consultés par la suite. Les utilisateurs seraient peut-être intéressés par les autres documents pré-chargés mais ne savent pas qu'ils sont disponibles. Il serait donc pertinent de suggérer aux utilisateurs ces documents. Ainsi, nous pourrions combiner notre système de pré-chargement avec la recommandation de documents. Les utilisateurs d'une CyraNode seraient alors avisés de l'arrivée d'un nouveau document susceptible de les intéresser.

Nous avons aussi montré qu'une mutualisation des résultats obtenus avec le moteur de recherche pour chaque utilisateur, donne de meilleurs résultats qu'un traitement individuel ou global. L'utilisation de communautés d'intérêt comme celles découverte par ECCLAT, semble donc être une voie très intéressante pour améliorer la qualité des résultats du pré-chargement. De telles communautés seraient formées en fonction des mots émergents partagés par les utilisateurs.

Notre méthode de détection d'émergences ne permet pas de trouver des associations de mots émergents permettant de faire des liens entre plusieurs mots. Or, une notion récente appelée motif émergent [DL99, SCR04], assez proche de nos mots émergents, permettrait de trouver de telles associations. Un motif émergent est une association d'items

davantage présente dans une classe de transactions que dans l'ensemble des autres. Au lieu de considérer des classes de transactions, on pourrait couper les données en fonction des séquences de consultations des documents. Un utilisateur serait alors caractérisé par des motifs émergents. Chacun de ces motifs serait composé de mots se trouvant dans un ou plusieurs documents consultés. L'utilisation de motifs émergents apporterait une meilleure précision dans les requêtes au moteur de recherche afin de trouver des documents à pré-charger.

Nous terminons par quelques perspectives plus générales. ECCLAT pourrait être utilisé pour la mise en place d'autres applications web comme l'organisation automatique de documents. Cela permettrait à des portails ou des moteurs de recherche de déterminer automatiquement une arborescence thématique de documents, en bénéficiant du chevauchement et des descriptions produits par ECCLAT. Enfin, l'ensemble de nos travaux pourrait servir à définir un système automatique de diffusion ciblée d'information, dans un contexte marketing.

Bibliographie

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, June 1998.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Database. *ACM SIGMOD (Special Interest Group on Management of Data)*, 22(2) :207–216, may 1993. Washington DC, USA.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328, Menlo Park, CA, 1996. AAAI Press.

[AS94] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.

[AS95] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proceedings of the International Conference on Data Engineering (ICDE)*, Taipei, Taiwan, march 1995.

[AS96] R. Agrawal and R. Srikant. Mining Sequential Patterns : Generalizations and Performance Improvements. In *the Fifth International Conference on Extending Database Technology*, Avignon, France, march 1996.

[BA99] R. J. jr. Bayardo and R. Agrawal. Mining the Most Interesting Rules. In *Proceedings of the 5th ACM SIGKDD*, august 1999.

[Bal97] M. Balabanovic. An Adaptive Web Page Recommendation Service. In *the First International Conference on Autonomous Agents*, pages 378–385, Marina del Rey, CA, USA, February 1997.

[Bay98] R.J. Bayardo. Efficiently Mining Long Patterns From Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 85–93, Seattle, June 1998.

[BB00] J. F. Boulicaut and A. Bykowski. Frequent Closures as Concise Representation for Binary Data Mining. In Springer-Verlag, editor, *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, pages 18–20, Kyoto, Japan, april 2000.

[BBPP99] I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The Maximum Clique Problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 4, Boston, MA, 1999. Kluwer Academic Publishers.

[BBR03] J. F. Boulicaut, A. Bykowski, and R. Rigotti. Free-sets : a Condensed Representation of Boolean Data for the Approximation of Frequency Queries. *Data Mining and Knowledge Discovery*, 7(1) :5–22, 2003.

[BC96] A. Bestravos and C. Cunha. Server-initiated Document Dissemination for the WWW. *IEEE Data Engineering Bulletin*, 19(3) :3–11, September 1996.

[BCF+98] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. On the Implications of Zipf's law for Web Caching. In *Proceedings of the 3rd International WWW Caching Workshop*, June 1998.

[Bis92] G. Bisson. Conceptual Clustering in a First Order Logic Representation. In *Neumann*, pages 458–462, 1992.

[BJ01] J. F. Boulicaut and B. Jeudy. Mining Free Sets under Constraints. In *Proceedings of the International Database Engineering and Application Symposium*, Grenoble, July 2001.

[Bor86] J.-P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Math. Sci. Hum.*, 24(96) :31–47, 1986.

[BRB04] J. Besson, C. Robardet, and J.-F. Boulicaut. Constraint-Based Mining of Formal Concepts in Transactional Data. In *Proceedings of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04), LNAI 3056*, pages 615–624, Sydney, Australia, May 2004.

[BRBR04] J. Besson, C. Robardet, J.-F. Boulicaut, and S. Rome. Constraint-based Concept Mining and its Application to Microarray Data Analysis. *Intelligent Data Analysis journal, IOS Press*, April 2004.

[BTP+02] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Pascal : un algorithme d'extraction des motifs fréquents. *Technique et sciences informatiques*, 1 :65–95, 2002.

[CD02] B. Crémilleux and N. Durand. Search for Factors Estimating the Stage of Liver Fibrosis Based on the Discovery of Meaningful Clusters. In *Proceedings of the PKDD'02 Discovery Challenge on Hepatitis Data, co-located with the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*, Helsinki, Finland, August 2002.

[CGMZ00] B. Crémilleux, M. Gaio, J. Madelaine, and K. Zreik. Discovering Browsing Paths on the Web. In *Proceedings of the Conference on Human-System Learning*, pages 9–18, Paris, France, 2000. Europia.

[CHW01] S. H. S. Chee, J. Han, and K. Wang. RecTree : An Efficient Collaborative Filtering Method. In *Proceedings of the International Conference on Data Warehouse and Knowledge Discovery (DaWak'01)*, Munich, Germany, September 2001.

[CJ97] C. R. Cunha and C. F. Jaccoud. Determining WWW User's Next Access and its Application to Pre-fetching. In *Proceedings of the IEEE Symposium on computers and Communication (ISCC'97)*, Alexandria, Egypt, july 1997.

[CMV04] G. Cleuziou, L. Martin, and C. Vrain. PoBOC : un algorithme de soft-clustering. Applications à l'apprentissage de règles et au traitement de

données textuelles. In *Actes des 4ème Journées d'Extraction et Gestion des Connaissances (EGC'04)*, volume I, pages 217–228, Clermont-Ferrand, France, Janvier 2004.

[CR93] C. Carpineto and G. Romano. Galois : An Order-Theoretic Approach to Conceptual Clustering. In *Proceedings of the Machine Learning Conference*, pages 33–40, 1993.

[CS95] P. Cheeseman and J. Stutz. Bayesian Classification (AutoClass) : Theory and Results. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, Menlo Park, 1995. the AAAI Press.

[CW84] J. G. Cleary and I. H. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications*, 32(4) :396–402, 1984.

[CY97] K. Chinen and S. Yamaguchi. An Interactive Prefetching Proxy Server for Improvement of WWW Latency. In *Proceedings of the Seventh Annual Conference of the Internet Society (INET'97)*, June, 1997.

[Dav02] B. D. Davison. Predicting Web Actions from HTML Content. In *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia (HT'02)*, pages 159–168, College Park, Maryland, USA, June 2002.

[DC02a] N. Durand and B Crémilleux. ECCLAT : a New Approach of Clusters Discovery in Categorical Data. In M. Bramer, A. Preece, and F. Coenen, editors, *Proceedings of the 22nd SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES'02), BCS Conference Series*, pages 177–190, Cambridge, UK, December 2002.

[DC02b] N. Durand and B. Crémilleux. Extraction of a Subset of Concepts from the Frequent Closed Itemset Lattice : A New Approach of Meaningful Clusters Discovery. In *Proceedings of the International Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD'02), co-located with ECAI'02*, pages 24–25, Lyon, France, July 2002.

[DCHA01] N. Durand, B. Crémilleux, and M. Henry-Amar. Discovering Associations in Clinical Data : Application to Search for Prognostic Factors in Hodgkin's Disease. In S. Quaglini, P. Barahona, and S. Andreassen, editors, *Proceedings of the 8th Conference on Artificial Intelligence in Medecine in Europe (AIME'01), Lecture Notes in Artificial Intelligence*, volume 2101, pages 50–54, Cascais, Portugal, July 2001.

[DCS04] N. Durand, G. Cleuziou, and A. Soulet. Discovery of Overlapping Clusters to detect Atherosclerosis Risk Factors. In *Proceedings of the PKDD'04 Discovery Challenge on Atherosclerosis Data, co-located with the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, Pisa, Italy, September 2004.

[DE84] W. Day and H. Edelsbrunner. Efficient Algorithms for Agglomerative Hierarchical Clustering Methods. *Journal of Classification*, 1 :1–24, 1984.

[Did71] E. Diday. La méthode des nuées dynamiques. *Revue de Statistiques Appliquées*, 19(2) :19–34, 1971.

[DL99] G. Dong and J. Li. Efficient Mining of Emerging Patterns : Discovering Trends and Differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Knowledge Discovery and Data Mining (SIGKDD'99)*, pages 43–52, San Diego, USA, August 1999.

[DL02] N. Durand and L. Lancieri. Study of the Regularity of the Users' Internet Accesses. In H. Yin et al., editor, *Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'02), Lecture Notes in Computer Science*, volume 2412, pages 173–178, Manchester, UK, August 2002.

[DL03] N. Durand and L. Lancieri. PrefetchEm v1.0 : Pré-chargement d'information dans un proxy-cache basé sur la détection d'Emergences, 2003. Logiciel déposé à l'Agence de Protection des Programmes, le 19 septembre 2003 sous le numéro IDDN.FR.001.380034.000.S.P.2003.000.30810.

[DLC03] N. Durand, L. Lancieri, and C. Crémilleux. Recommendation System Based on the Discovery of Meaningful Categorical Clusters. In V. Palade et al., editor, *Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'03), Lecture Notes in Artificial Intelligence*, volume 2773, pages 857–863, Oxford, UK, September 2003.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39 :1–38, 1977.

[EKSX96] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd international Conference on Knowledge Discovery and Data Mining (KDD'96)*, Portland, Oregon, August 1996.

[FB92] W. B. Frakes and R. Baezayates. *Information Retreival Data Structures and Algorithms*. Prentice Hall, Englewood CLiffs, NJ, 1992.

[FCLJ99] L. Fan, P. Cao, W. Lin, and Q. Jacobson. Web Prefetching between Low-Bandwidth Clients and Proxies : Potential and Performance. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'99)*, Atlanta, may 1999.

[Fis87] D. H. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, 2 :139–172, 1987.

[FMN04] H. Fu and E. Mephu Nguifo. How Well go Lattice Algorithms on Currently used Machine Learning TestBeds ? In *Actes des 4ème Journées d'Extraction et Gestion des Connaissances (EGC'04)*, volume II, pages 373–384, Clermont-Ferrand, France, Janvier 2004.

[GA94] J. Griffioen and R. Appleton. Reducing File System Latency using a Predictive Approach. In *Proceedings of the USENIX Technical Conference*, Boston, 1994.

[Gan84] B. Ganter. Two Basic Algorithms in Concept Analysis. Technical Report Preprint 831, Technical University Darmstadt, 1984.

[GGM02] F. Giannotti, C. Gozzi, and G. Manco. Clustering Transactional Data. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*, pages 175–187, Helsinki, Finland, August 2002.

[GGR99] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS : Clustering Categorical Data using Summaries. In Surajit Chaudhuri and David Madigan, editors, *Proceedings of the 5th ACM SIGMOD international Conference on Knownledge Discovery and Data Mining*, pages 73–83, New York, August 1999.

[GLF89] J.H. Gennari, P. Langley, and D.H. Fisher. Models of Incremental Concept Formation. *Artificial Intelligence*, 40 :11–61, 1989.

[GMA95] R. Godin, R. Missaoui, and H. Alaoui. Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices. *Computational Intelligence*, 11(2) :246–267, 1995.

[GNOT92] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communication of the ACM*, 35(12) :61–70, 1992.

[GRS98] S. Guha, R. Rastogi, and K. Shim. CURE : An Efficient Clustering Algorithm for Large Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.

[GRS99] S. Guha, R. Rastogi, and K. Shim. ROCK : A Robust Clustering Algorithm for Categorical Attributes. In *Proceedings of the 15th International Conference on IEEE Data Engineering (ICDE'99)*, pages 512–521, 1999.

[GS94] J. Gwetzman and M. Seltzer. The Case for Geographical Pushing-caching. In *Proceedings of the HotOs Conference*, 1994.

[HB89] S.J. Hanson and M. Bauer. Conceptual Clustering, Categorization and Polymorphy. *Machine Learning*, 3 :343–372, 1989.

[HC02] H. Harb and L. Chen. Video Scene Description : An Audio Based Approach. In *Actes de la Journée francophone d'accès intelligent aux documents multimédias sur l'Internet (MediaNet'02)*, Sousse, Tunisie, Juin 2002. Hermes.

[HKKM97] E. H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering Based on Association Rule Hypergraphs. In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 9–13, Tucson, Arizona, 1997.

[HKKM98] E. H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph Based Clustering in High-Dimensional Data Sets : a Summary of Results. *Bulletin of the Technical Commitee on Data Engineering*, 21(1), march 1998.

[Hua99] Z. Huang. Extensions to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery*, 2(3) :283–304, 1999.

[JFM97] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher : A Tour Guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 770–775, Nagoya, Japan, August 1997.

[JM01] A. Jebali and M. Makpangou. Replica Divergence Control Protocol in Weakly Connected Environment. In *Proceedings of the IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, October 2001.

[KHK99] G. Karypis, E. H. Han, and V. Kumar. CHAMELEON : Hierarchical Clustering Using Dynamic Modeling. *IEEE Computer*, 32(8) :68–75, 1999.

[KK99] G. Karypis and V. Kumar. Multilevel k-way Hypergraph Partitioning. In *Proceedings of the 36th Design Automation Conference*, august 1999.

[KKH00] S. H. Kim, J. Y. Kim, and J. W. Hong. A Statistical, Batch, Proxy-Side Web Prefetching Scheme for Efficient Internet Bandwidth Usage. In *Proceedings of the 2000 Network+Interop Engineers Conference*, Las Vegas, May 2000.

[KMM+97] J. A Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L .R. Gordon, and J. Riedl. GroupLens : Applying Collaborative Filtering to Usenet News. *Communication of the ACM*, 40(3) :77–87, March 1997.

[Koh95] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, New-York, 1995.

[Kol83] J. L. Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7 :243–280, 1983.

[KP77] M. Karonski and Z. Palka. On Marczewski-Steinhaus Type Distance between Hypergraphs. In *Zastosowania Matematyki Applicationes Mathematicae*, volume 16, pages 47–57, 1977.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data : An Introduction to Cluster Analysis*. John Wiley and Sons - Interscience (Series in Applied Probability and Statistics), New York, 1990.

[Kri01] R. Krishnapuram. Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. *IEEE Transactions on Fuzzy Systems*, 9(4) :596–607, 2001.

[Kul67] S. Kullback. *Information theory and statistics*. Chapman and Hall, New York - Dover, 1967.

[Lan00a] L. Lancieri. *Mémoire et Oubli : Deux Mécanismes Complémentaires pour Caractériser les Différents Acteurs de l'Internet dans leurs Interactions*. PhD thesis, Université de Caen, décembre 2000.

[Lan00b] L. Lancieri. Procédé et dispositif de réutilisation d'information ayant fait l'objet d'une réception antérieure dans un réseau de télécommunication tel que le réseau Internet, 2000. Brevet déposé à l'INPI le 17 octobre 2000, numéro 2815435.

[Lan01] L. Lancieri. The Concept of Informational Ecology or Interest of the Information re-Use in the Company. In *Proceedings of the 3rd International Conference on Enterprise Information Systems (IEEE, AAAI, TCNA)*, Setubal, Portugal, 2001.

[Lan04] L. Lancieri. Reusing Implicit Cooperation : A novel approach to knowledge management. *TripleC (Cognition, Communication, Co-operation)*, 2(1) :28–46, 2004.

[LB97] T. S. Loon and V. Bharghavan. Alleviating the Latency and Bandwidth Problems in WWW Browsing. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'97)*, december 1997.

[LBBS01] L. Lancieri, N. Berthier-Bonnel, and L. Stumme. To Exploit the Collective Intelligence Thanks to the Co-operative Replication. In *Proceedings of the International Conferences on Info-tech & Info-net (ICII'01)*, Beijing, China, 2001.

[LCM02] L. Lancieri, L. Chen, and M. Makpangou. Projet RNRT Cyrano : Système de Réplication Actif Personnalisé de Flux Audiovisuel sur Internet. In *Actes de la Journée francophone d'accès intelligent aux documents multimédias sur l'Internet (MediaNet'02)*, pages 49–60, Sousse, Tunisie, Juin 2002. Hermes.

[LD01] L. Lancieri and N. Durand. Procédé et système de détection d'un phénomène d'émergence d'information à partir de données numériques échantillonnées, 2001. Brevet déposé à l'INPI le 22 juin 2001, numéro 0108267.

[LD03a] L. Lancieri and N. Durand. Evaluating the Impact of Users' Profiles Dimension on its Characterization Effectiveness : Method Based on the Evaluation of Users' Communities' Organization Quality. In *The International Symposium on Computational Intelligence for Measurement Systems and Applications (CIMSA 2003)*, pages 130–134, Lugano, Switzerland, July 2003.

[LD03b] L. Lancieri and N. Durand. Internet Users' Behavior : Compared Study of the Access Traces and Application to the Discovery of Communities. *Submitted to the IEEE Transactions on Systems, Man and Cybernetics (SMC)*, 2003.

[LD04] L. Lancieri and N. Durand. Activity Forecast of the Internet Users : Compared Performances of Active and Passive Forecast Strategies. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA'04)*, pages 770–775, Innsbruck, Austria, February 2004.

[Leb87] M. Lebowitz. Experiments with Incremental Concept Formation : Unimem. *Machine Learning*, 2 :103–138, 1987.

[LFL00] S. Legouix, J. P. Foucault, and L. Lancieri. A Method for Studying the Variability of Users' Thematic Profile. In *Proceedings of the WebNet'00, Association for the Advancement of Computing in Education (AACE)*, San Antonio, 2000.

[LFP03] F. Le Fessant and S. Patarin. MLdonkey, a Multi-Network Peer-to-Peer File-Sharing Program. In *Proceedings of the International Conference on Functional Programming (ICFP'03)*, Uppsala, Sweden, August 2003.

[Lie95] H. Lieberman. Letizia : An Agent that Assists Web Browsing. In *the Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95*, pages 924–929, Montréal, Québec, Canada,, August 1995.

[LLDS02] L. Lancieri, S. Legouix, N. Durand, and N. Saillard. CyraNode v1.4 - Système de Réplication Actif Personnalisé de Flux Audiovisuel sur Internet, 2002. Logiciel déposé à l'Agence de Protection des Programmes, le 4 avril 2002 sous le numéro IDDN.FR.001.140011.000.S.P.2002.000.30810.

[LMV02] N. Labroche, N. Monmarché, and G. Venturini. A New Clustering Algorithm based on the Chemical Recognition System of Ants. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02)*, pages 345–349, Lyon, France, July 2002. IOS Press.

[LTWW93] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic. In Deepinder P. Sidhu, editor, *ACM SIGCOMM'93*, pages 183–193, San Francisco, California, 1993.

[Mac67] J. B. MacQueen. Some Methods of Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[Mar96] E. P. Markatos. Main Memory Caching of Web Documents. *Computer Networks and ISDN Systems*, 28 :893–906, 1996.

[MC98] E. P. Markatos and C. E. Chronaki. A Top-10 Approach to Prefetching the Web. In *Proceedings of the Annual Conference of the Internet Society (INET)*, Geneva, July 1998.

[MCS99] B. Mobasher, R. Cooley, and J. Srivastava. Creating Adaptive Web Sites through Usage-Based Clustering of URLs. In *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, Chicago, november 1999.

[Mit82] T. M. Mitchell. Generalization as Search. *Artificial Intelligence*, 18(2) :203–226, 1982.

[Mou97] A. Moukas. Amalthaea : Information Discovery and Filtering Using a Multiagent Evolving Ecosystem. *International Journal of Applied Artificial Intelligence*, 11(5) :437–457, 1997.

[MS83] R. S. Michalski and R. E. Stepp. Learning from Observation : Conceptual Clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning, An Artificial Intelligence Approach*, volume 1, pages 331–363. Morgan Kauffmann, 1983.

[NH94] R. Ng and J. Han. Efficient and Effective Clustering Method for Spatial Data Mining. In *Proccedings of the International Conference on Very Large Databases (VLDB)*, pages 144–155, Santiago, Chile, 1994.

[NKM01] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. Effective Prediction of Web-user Accesses : A Data Mining Approach. In *Proceedings of the WebKDD Workshop (WebKDD'01)*, San Francisco, 2001.

[NW97] D. S. W. Ngu and X. Wu. SiteHelper : A Localized Agent that Helps Incremental Exploration of the World Wide Web. In *Proceedings of the 6th international World Wide Web Conference*, pages 691–700, Santa Clara, CA, 1997.

[OKN01] S. Oyanagi, K. Kubota, and A. Nakase. Application of Matrix Clustering to Web Log Analysis and Access Prediction. In *Proceedings of the WebKDD Workshop (WebKDD'01) co-lacated with the ACM SIGKDD International Conference on Knowledge Discovery in Databases (KDD'01)*, San Francisco, CA, August 2001.

[Pas00] N. Pasquier. *Data Mining : Algorithmes d'Extraction et de Réduction des Règles d'Associations dans les BD.* PhD thesis, Université de Clermont-Ferrand, 2000.

[Pax95] V. Paxon. Fast Approximation of Self-Similar Network Traffic. Technical Report LBL-36750, University of California, Berkeley, April 1995.

[Paz99] M. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5) :393–408, 1999.

[PBTL99] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Information Systems*, 24(1) :25–46, Elsevier, 1999.

[PHM00] J. Pei, J. Han, and R. Mao. CLOSET : An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[PM96] V. N. Padmanabhan and J. C. Mogul. Using Predictive Prefetching to Improve World Wide Web Latency. *Computer Communication Review*, 26(3) :22–36, july 1996.

[PMB96] M Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert : Identifying interesting web sites. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 54–61, Portland, Oregon, 1996.

[RC98] A. Ragel and B. Crémilleux. Treatment of Missing Values for Association Rules. In Korb K. B. Wu X., Kotagiri R., editor, *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 98*, volume 1394 of *Lecture notes in artificial intelligence*, pages 258–270, Melbourne, Australia, 1998.

[RC99] A. Ragel and B. Crémilleux. MVC - A Preprocessing Method to Deal with Missing Values. *Knowledge-Based Systems*, 12(5/6) :285–291, 1999.

[RC03a] F. Rioult and B. Crémilleux. Condensed Representations in Presence of Missing Values. In *Proceedings of the 5th International Conference on Intelligent Data Analysis (IDA'03), Lecture Notes in Computer Science*, volume 2810, pages 578–588, Berlin, Germany, August 2003.

[RC03b] F. Rioult and B. Crémilleux. Optimisation d'extraction de motifs : une nouvelle méthode fondée sur la transposition de données. In *Actes de la Conférence Apprentissage*, pages 299–313, Laval, France, Juillet 2003. Presses Universitaires de Grenoble.

[Rob02] C. Robardet. *Contribution à la classification non supervisée : proposition d'une méthode de bi-partitionnement.* PhD thesis, Université de Lyon 2, Juillet 2002.

[Ron98] P. Ronkainen. Attribute Similarity and Event Sequence Similarity in Data Mining. Technical Report C-1998-42, University of Helsinki, october 1998.

[RR02] C. Robardet and C. Rigotti. Etude de méthodes de recherche locale pour la construction de bi-partitions. *Extraction de Connaissances et Apprentissage (RSTI-RIA-ECA)*, 16 :705–728, 2002. Hermès France.

[Sai03] N. Saillard. *Optimisation des architectures de réplication dédiées à Internet, par la caractérisation du trafic.* PhD thesis, Université de Caen, Novembre 2003.

[SCR04] A. Soulet, B. Crémilleux, and F. Rioult. Condensed Representation of Emerging Patterns. In *Proceedings of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, pages 127–132, Sydney, Australia, May 2004.

[SCZ98] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster : A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)*, pages 428–439, New York City, NY, USA, August 1998.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. Technical Report #00-034, University of Minnesota, 2000.

[STB+02] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing Iceberg Concept Lattices with TITANIC. *Journal on Knowledge and Data Engineering*, 2002.

[TC00] S. Thomas and S. Chakravarthy. Incremental Mining of Constrained Associations. In *Proceedings of the 7th International Conference on High Performance Computing (HiPC'00)*, pages 547–558, Bangalore, India, December 2000.

[THH01] A. K. H. Tung, J. Hou, and J. Han. Spatial Clustering in the Presence of Obstacles. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 359–367, Heidelberg, Germany, April 2001.

[TK00] P. N. Tan and V. Kumar. Interestingness Measures for Association Patterns : a Perspective. Technical Report TR00-036, University of Minnesota, 2000.

[Wai99] K. Waiyamai. *Découverte et Gestion des Connaissances dans les Bases de Données (Data Mining) : Une Approche par Treillis de Concepts Fréquents.* PhD thesis, Université de Clermont-Ferrand, 1999.

[Wan99] J. Wang. A Survey of Web Caching Schemes for the Internet. *ACM Computer Communication Review*, 29(5) :36–46, October 1999.

[WCC01] Y. H. Wu, Y. C. Chen, and A. L. P. Chen. Enabling Personalized Recommendation on the Web Based on User Interests and Behaviors. In *Proceedings of the 11th International Workshop on Research Issues in Data Engineering (RIDE-DM'01)*, pages 17–24, Heidelberg, Germany, April 2001.

[WCL99] K. Wang, X. Chu, and B. Liu. Clustering Transactions Using Large Items. In *ACM Conference on Information and Knowledge Management (CIKM)*, USA, 1999.

[Wil82] R. Wille. Restructuring Lattice Theory : an Approach based on Hierarchies of Concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Reidel, Dordrecht-Boston, 1982.

[Wil89] R. Wille. Knowledge Acquisition by Methods of Formal Concept Analysis. *Data Analysis, Learning Symbolic and Numeric Knowledge*, pages 365–380, New-York : Nova Science, 1989.

[WSO02] D. Wilson, B. Smyth, and D. O'Sullivan. Improving Collaborative Personalized TV Services. In M. Bramer, A. Preece, and F. Coenen, editors, *Proceedings of the 22nd SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence (ES'02), BCS Conference Series*, pages 265–278, Cambridge, UK, December 2002.

[WYM97] W. Wang, J. Yang, and R. Muntz. STING : A Statistical Information Grid Approach to Spatial Data Mining. In *Proceedings of the Twenty-Third International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece, 1997.

[YZL01] Q. Yang, H. H. Zhang, and T. Li. Mining Web Logs for Prediction Models in WWW Caching and Prefetching. In *Proceedings of the seventh ACM SIGKDD International Conference on knowledge Discovery and Data Mining (KDD'01)*, San Francisco, California, USA, August 2001.

[ZH02] M. J. Zaki and C.-J. Hsiao. CHARM : An Efficient Algorithm for Closed Itemset Mining. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)*, Arlington, April 2002.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Linvy. BIRCH : An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montréal, Québec, 1996.

Annexe A

L'interface utilisateur d'une CyraNode

La figure A.1 présente la page d'accueil d'une CyraNode. Le cadre en haut à gauche de cette page permet d'entrer un pseudo et un mot de passe pour se connecter au système.

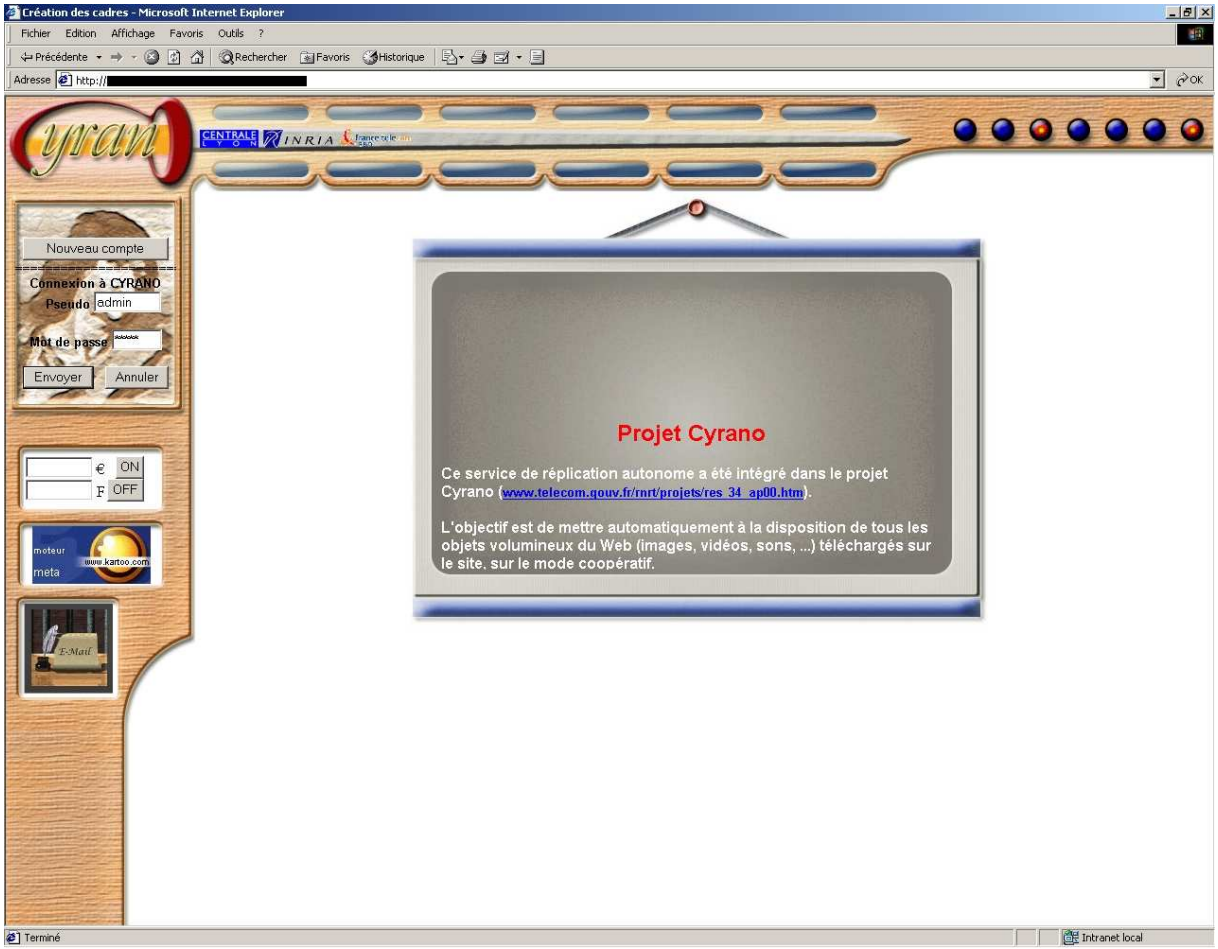

FIG. A.1 – Connexion à l'interface du système CYRANO

Un fois que l'utilisateur est authentifié, une page personnalisée apparaît (cf. figure A.2). Les derniers documents ajoutés dans la CyraNode sont présentés selon leur catégorie.

La page indique quelques statistiques comme le nombre de documents stockés dans la CyraNode, la répartition de ces documents selon les catégories. A droite de ces statistiques, l'utilisateur peut entrer des mots et sélectionner une catégorie pour effectuer une recherche de documents parmi ceux stockés dans la CyraNode. Le cadre en haut à gauche, donne accès à diverses fonctions comme l'affichage de l'historique des recherches, la configuration de la CyraNode (si l'utilisateur est un administrateur).

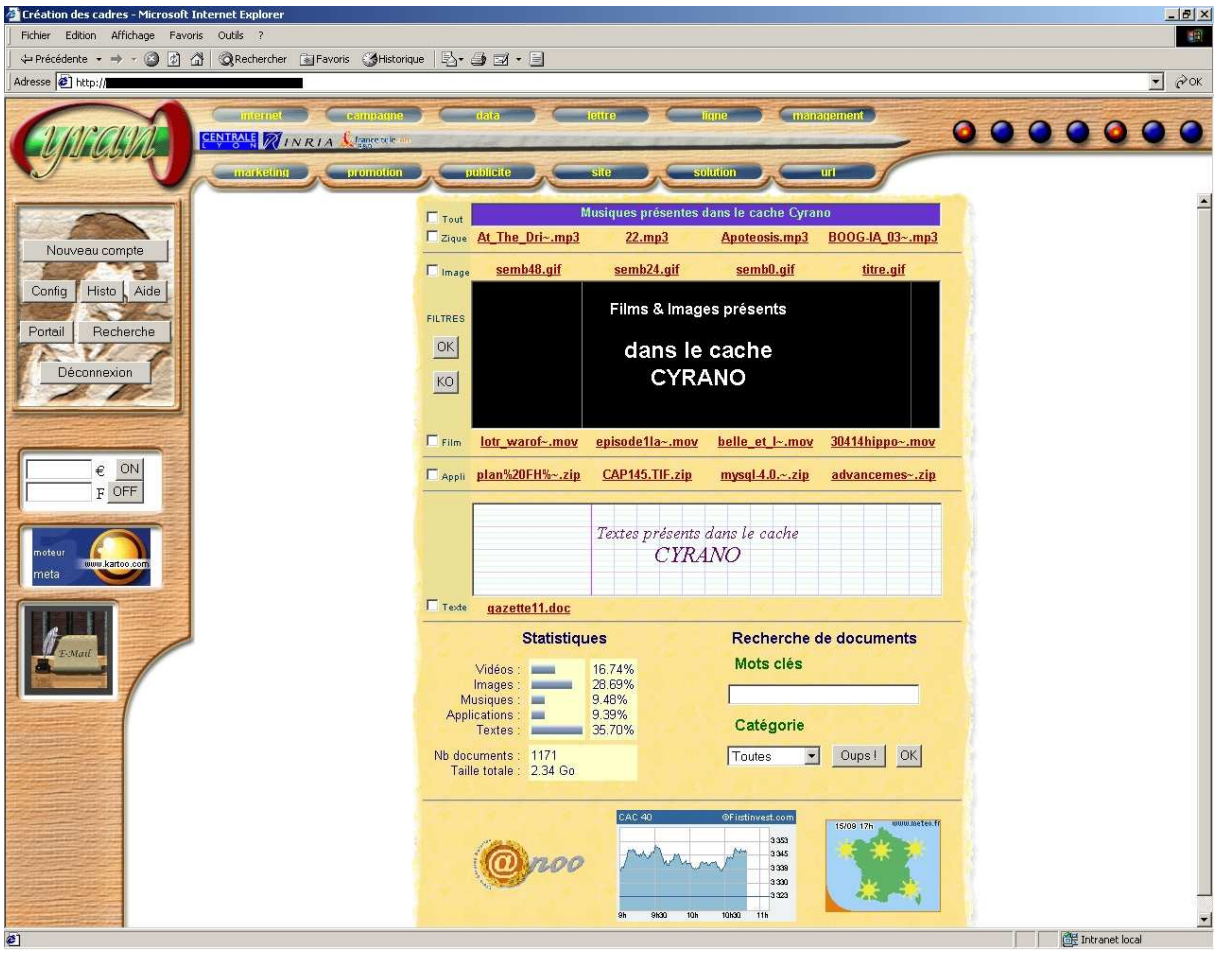

FIG. A.2 – Page d'accueil personnalisée de l'interface de CYRANO

FIG. A.3 – Exemple de boutons personnalisés de recherche rapide

La personnalisation s'effectue essentiellement sur les boutons affichés en haut de la page. Ces boutons permettent de lancer des recherches rapides sur les mots correspondants. La figure A.3 présente un exemple de boutons. Les mots apparaissant sur les boutons correspondent à une combinaison des mots les plus fréquemment entrés dans les

recherches de l'utilisateur, et des mots les plus présents dans les documents consultés par l'utilisateur.

La figure A.4 présente le résultat d'une recherche de documents sur toutes les catégories avec les mots-clés "réseau" et "IP". Pour chaque résultat apparaissent le nom du document, l'URL d'origine, les mots-clés, la taille, et la date d'arrivée dans la CyraNode. Si l'utilisateur est un administrateur, il a la possibilité d'effacer des documents ou de poser un filtre pour que des documents ne puissent pas être consultés par les autres utilisateurs.

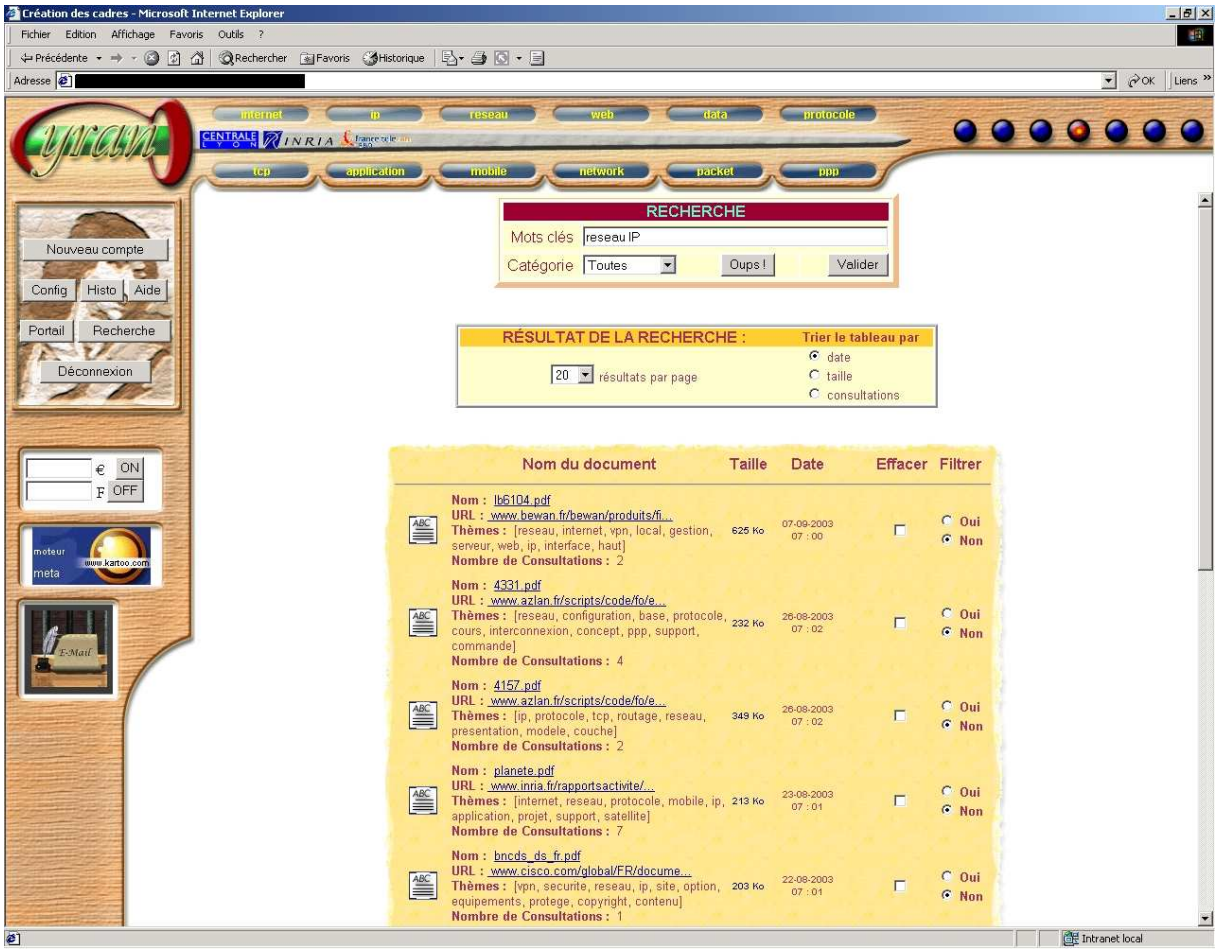

FIG. A.4 – Recherche de documents par l'interface de CYRANO

Lorsque l'utilisateur a trouvé un document qu'il l'intéresse, il clique sur le nom de celui-ci. Le document est alors ouvert par le navigateur (cf. figure A.5).

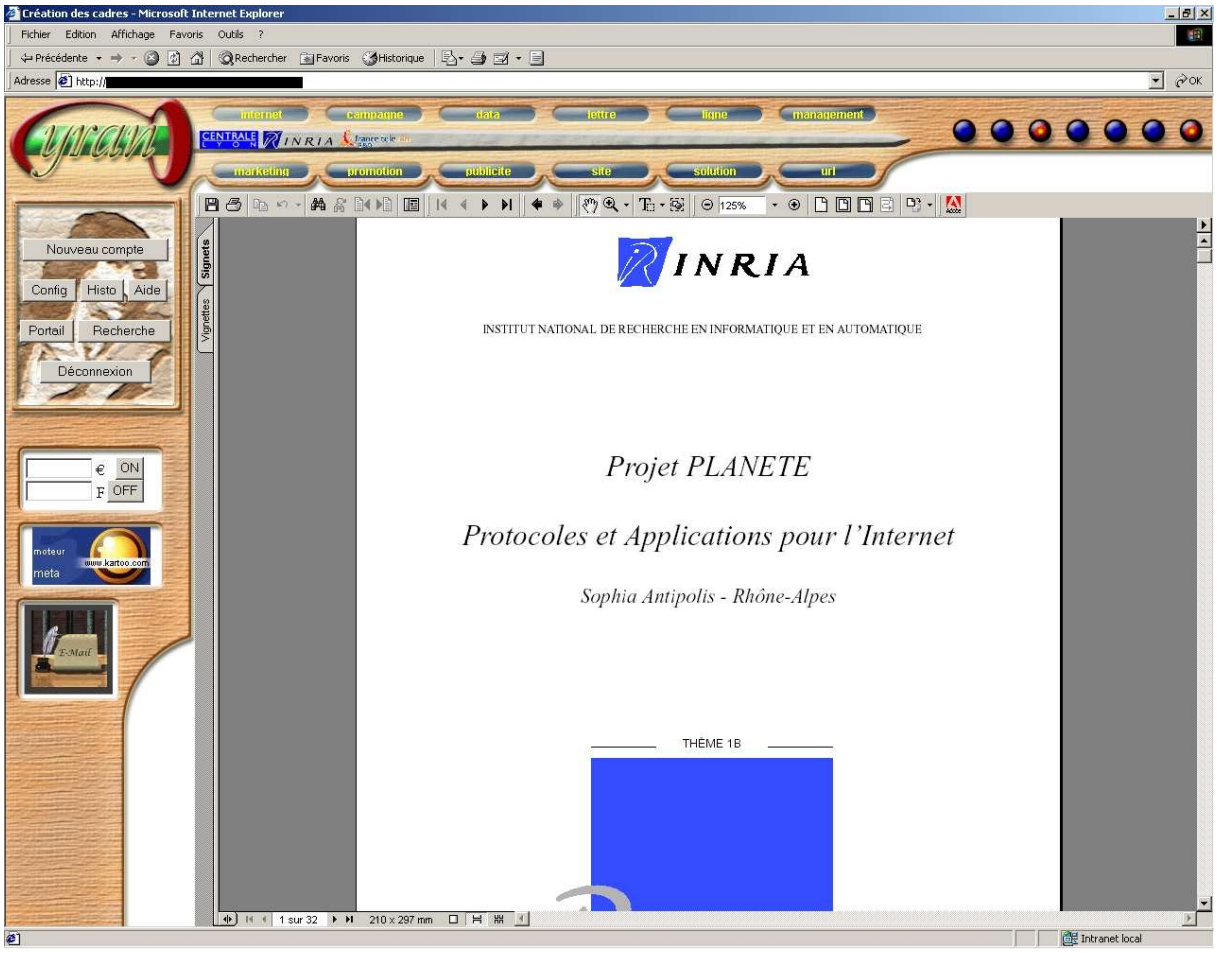

FIG. A.5 – Consultation d'un document par l'interface de CYRANO

Annexe B

Récapitulatif des méthodes de clustering

Le tableau B.1 présente une synthèse des méthodes de clustering présentées dans le chapitre 3, en fonction des critères suivant : le type de données utilisées, la nécessité de fixer le nombre de clusters, le possible chevauchement d'éléments entre les clusters, et la présence d'une description des clusters résultats expliquant pourquoi les éléments ont été regroupés. Dans le cas de regroupement d'items, la description correspond aux éléments des clusters. Dans certains cas, nous employons le terme "possible". Cela signifie que l'algorithme seul ne satisfait pas le critère, mais grâce à un calcul supplémentaire, il est possible de le satisfaire (sans une garantie absolue). Les catégories des méthodes sont représentées par une lettre, P : partitionnement, H : hiérarchique, F : basée sur la fréquence, C : classification conceptuelle.

Nom	Catégorie	Type de données	Nombre fixé de clusters	Chevauchement d'éléments	Description des clusters
K-Means	P	Quantitatives	Oui	Non	Non
K-Modes	P	Qualitatives	Oui	Non	Non
Nuées dynamiques	P	Quantitatives	Oui	Non	Non
PAM	P	Quantitatives	Oui	Non	Non
CLARA	P	Quantitatives	Oui	Non	Non
CLARANS	P	Quantitatives	Oui	Non	Non
Fuzzy c-Medoids	P	Qualitatives	Oui	Oui (probabilité)	Non
PoBOC	P	Quant. & Qual.	Non	Oui	Non
DBSCAN	P	Quantitatives	Oui	Non	Non
OPTICS	P	Spatiales	Non	Non	Non
STING	P	Spatiales	Non	Non	Non
WaveCluster	P	Spatiales	Non	Non	Non
EM	P	Quantitatives	Oui	Non	Non
AutoClass	P	Quant. & Qual.	Oui	Oui (probabilité)	Oui
SOM	P	Quantitatives	Oui	Non	Non
AntClust	P	Quant. & Qual.	Non	Non	Non
COD	P	Quantitatives	Oui	Non	Non
Bi-Clust	Bi-P	Qualitatives	Non	Non	Oui
AGNES	H	Quantitatives	Non	Non	Non
DIANA	H	Quantitatives	Non	Non	Non
CURE	H	Quantitatives	Oui	Non	Non
BIRCH	H	Quantitatives	Non	Non	Non
ROCK	H	Qualitatives	Oui	Non	Possible
CHAMELEON	H	Qualitatives	Oui	Non	Non
TrK-Means	F (P)	Qualitatives	Oui	Non	Oui
Wang et al.	F (P)	Qualitatives	Non	Non	Possible
CLIQUE	F (P)	Qualitatives	Non	Non	Oui
CACTUS	F (H)	Qualitatives	Non	Non	Oui
Ronkainen	F (H)	Qualitatives	Non	Possible	Possible
ARHP	F (P)	Qualitatives	Oui	Possible	Possible
Ping-Pong	F (C)	Qualitatives	Non	Oui	Oui
CLUSTER	C	Qualitatives	Non	Non	Oui
COBWEB	C	Qualitatives	Non	Non	Oui
GALOIS	C	Qualitatives	Non	Oui	Oui

TAB. B.1 – Récapitulatif des principaux algorithmes de clustering

Annexe C

Découverte de motifs fermés fréquents et calcul de l'*homogénéité* avec l'algorithme *CLOSE*

Sommaire

Le but de cette annexe est de présenter l'algorithme *CLOSE* recherchant les motifs fermés fréquents, et d'indiquer les modifications nécessaires pour le calcul de l'*homogénéité* de chacun de ces motifs. Nous présentons aussi un exemple complet d'exécution de l'algorithme.

C.1 Recherche des motifs fermés fréquents

L'algorithme *CLOSE* [PBTL99] détermine les motifs fermés fréquents à partir d'une base de données transactionnelle. Il permet de déduire de ces motifs tous les motifs fréquents. Les travaux montrent qu'il est plus rapide que l'algorithme *APRIORI* [AMS+96] sur des données denses et/ou corrélées. Ces deux algorithmes procédent par niveaux. Premièrement, les motifs de taille 1 sont examinés, puis ensuite ceux de taille 2 sont générés et examinés, etc. Bien entendu, l'algorithme s'arrête lorsqu'il n'y a plus de motifs à examiner. Pour rendre efficace cette recherche, les algorithmes se basent sur des critères d'élagage supprimant des motifs inutiles lors de la génération des motifs de taille supérieure. Nous verrons que *CLOSE* introduit un critère d'élagage supplémentaire par rapport à *APRIORI*.

Pour rechercher les motifs fermés fréquents, *CLOSE* (cf. algorithme 4) utilise des motifs dits "générateurs". Un générateur va permettre de trouver un motif fermé en effectuant l'intersection de toutes les transactions contenant ce générateur. De cette manière, *CLOSE* va calculer les motifs fermés de tous les générateurs de taille 1 en parcourant la base de données (calculant en même temps les fréquences), puis sélectionner les fréquents, et

déterminer les générateurs de taille 2 à partir de ceux de taille 1, et va recommencer jusqu'à épuisement des générateurs.

Algorithme 4 CLOSE

$FCC_1 = \{$motifs de taille 1$\}$;
pour $(i = 1; FCC_i.générateur \neq \oslash; i++)$ **faire**
  $FCC_i.fermeture = \oslash$;
  $FCC_i.fréquence = 0$;
  $FCC_i =$ Gen-Closure(FCC_i) ; {produit la fermeture de chaque générateur}
  pour tout $c \in FCC_i$ **faire**
    si $(c.fréquence \geq minfr)$ **alors**
      $FC_i = FC_i \cup \{c\}$;
    fin si
  fin pour
  $FCC_{i+1} =$ Gen-Generator(FC_i) ; {créer les générateurs pour l'itération i+1}
fin pour
retourner $FC = \bigcup_{j=1}^{i-1} \{FC_j.fermeture, FC_j.fréquence\}$;

Le parcours de la base de données qui permet le calcul des motifs fermés et des fréquences est effectué par la fonction *Gen-Closure* (cf. algorithme 5). La fonction qui génère les nouveaux générateurs, s'appelle *Gen-Generator* (cf. algorithme 6). Notons qu'elles utilisent une fonction *Subset* qui prend en paramètre un ensemble de générateurs et une transaction, et qui retourne les générateurs contenus dans cette transaction. *Gen-Generator* fusionne les motifs de taille i ayant leur (i-1) premiers items en commun [AMS+96], et effectue deux élagages. Le premier est issu d'*APRIORI* et est justifié par les propriétés 1 et 2. Le second élagage élimine les générateurs qui sont inclus dans les motifs fermés fréquents déjà calculés, car ces générateurs vont générer les mêmes motifs fermés (cf. théorème 1).

Propriété 1 *Tous les sous-ensembles d'un motif fermé fréquent sont fréquents.*

Propriété 2 *Tous les super-ensembles d'un motif fermé non-fréquent sont non-fréquents.*

Théorème 1 (justification du 2^{eme} élagage) *Soient I un ensemble de générateurs de taille i, et $S = \{s_1, s_2, \ldots, s_j\}$ un ensemble de (i-1)-sous-ensembles de I où $\cup s_k = I$. Si $\exists s_a \in S$ t.q. $I \subseteq h(s_a)$, alors $h(I) = h(s_a)$.*

Tout comme *APRIORI*, *CLOSE* utilise une structure d'arbre de hachage pour stocker les motifs [PBTL99]. Cette structure a été modifiée pour pouvoir stocker des informations supplémentaires comme la fermeture. Nous utilisons les notations suivantes.

FCC_i contient les candidats composés de trois champs :
- *générateur* : motif de taille i,
- *fermeture* : motif fermé candidat produit par la fermeture de *générateur*,
- *fréquence* : compteur stockant le fréquence du générateur (elle est égale à la fréquence du motif fermé).

FC_i a la même structure mais correspond aux motifs fermés fréquents.

Un exemple complet d'exécution est présenté dans la section C.3.

Algorithme 5 Gen-Closure

pour tout transaction t de $\mathcal{B}$ **faire**
  G_t = Subset($FCC_i.générateur$, t) ; {les générateurs contenus dans la transaction t}
  pour tout $g \in G_t$ **faire**
    si ($g.fermeture = \oslash$) **alors**
      $g.fermeture = t$; {initialise avec les items de la transaction t}
    sinon
      $g.fermeture = g.fermeture \cap t$;
    fin si
    $g.fréquence++$;
  fin pour
fin pour
retourner $\bigcup\{c \in FCC_i \mid c.fermeture \neq \oslash\}$;

Algorithme 6 Gen-Generator

{fusionner les motifs de FC_i qui partagent leur (i-1) premiers items}
insert into $FCC_{i+1}.générateur$
select $p.item_1, p.item_2, \ldots, p.item_i, q.item_i$
from $FC_i.générateur$ p, $FC_i.générateur$ q
where $p.item_1 = q.item_1, \ldots, p.item_{i-1} = q.item_{i-1}, p.item_i < q.item_i$;
{1^{ere} étape d'élagage : propriétés 1 et 2}
pour tout $p \in FCC_{i+1}.générateur$ **faire**
  pour tout i-sous-ensemble s de p **faire**
    si ($s \notin FC_i.générateur$) **alors**
      supprimer p de $FCC_{i+1}.générateur$;
    fin si
  fin pour
fin pour
{2^{eme} étape d'élagage, on élimine les nouveaux générateurs qui vont donner des fermés déjà calculés (voir théorème 1) : on supprime les nouveaux (i+1)-générateurs qui sont inclus dans un motif fermé fréquent de FC_i}
pour tout $p \in FCC_{i+1}.générateur$ **faire**
  S_p = Subset($FC_i.générateur$, p) ; {générateurs contenus dans p}
  pour tout $s \in S_p$ **faire**
    si ($p \subseteq s.fermeture$) **alors**
      supprimer p de $FCC_{i+1}.générateur$;
    fin si
  fin pour
fin pour
retourner $\bigcup\{c \in FCC_{i+1}\}$;

C.2 Calcul de l'*homogénéité*

Comme nous l'avons vu dans la section 4.2, le calcul de l'*homogénéité* d'un motif fermé est simple. Pour la *divergence*, il suffit de déterminer le nombre total d'items des toutes ses transactions. Puis, une fois le motif fermé complétement formé, il ne reste plus qu'à retrancher le produit de sa fréquence et de son nombre d'items, de la somme précédente.

Rappelons les formules :

$$divergence(X) = \sum_{t \in g(X)} |f(t)| - \mathcal{F}(X) \times |X|$$

$$homogénéité(X) = \frac{\mathcal{F}(X) \times |X|}{divergence(X) + (\mathcal{F}(X) \times |X|)}$$

D'où

$$homogénéité(X) = \frac{\mathcal{F}(X) \times |X|}{\sum_{t \in g(X)} |f(t)|}.$$

Pour mettre cela en œuvre, il faut ajouter un champ *homogénéité* à la structure correspondant à un motif fermé candidat (FCC) ou un motif fermé fréquent (FC).

Le calcul de la somme des items des transactions de X s'effectue dans *Gen-Closure*. Il faut alors ajouter une instruction après l'incrémentation de la fréquence, de façon à ajouter à l'*homogénéité* du générateur le nombre d'items de la transaction examinée (cf. algorithme 7).

Algorithme 7 Gen-Closure Modifié

pour tout transaction t de $\mathcal{B}$ **faire**
  G_t = Subset($FCC_i.générateur$, t) ; {les générateurs contenus dans la transaction t}
  pour tout $g \in G_t$ **faire**
    si $(g.fermeture = \oslash)$ **alors**
      $g.fermeture = t$; {initialise avec les items de la transaction t}
    sinon
      $g.fermeture = g.fermeture \cap t$;
    fin si
    $g.fréquence++$;
    g.homogénéité += **|t|**; {**pour le calcul de l'homogénéité**}
  fin pour
fin pour
retourner $\bigcup\{c \in FCC_i / c.fermeture \neq \oslash\}$;

Dans l'algorithme principal *CLOSE*, si un candidat est fréquent, alors avant de l'ajouter dans FC_i, il faut mettre à jour son *homogénéité*. Pour cela, nous divisons le produit de la fréquence et du nombre d'items de la fermeture par l'ancienne valeur d'*homogénéité* (cf. algorithme 8).

Algorithme 8 CLOSE Modifié

$FCC_1 = \{\text{motifs de taille 1}\}$;
pour $(i = 1; FCC_i.générateur \neq \oslash; i++)$ **faire**
  $FCC_i.fermeture = \oslash$;
  $FCC_i.fréquence = 0$;
  FCC_i = Gen-Closure(FCC_i) ; {produit la fermeture de chaque générateur}
  pour tout $c \in FCC_i$ **faire**
    si $(c.fréquence \geq minfr)$ **alors**
      c.homogénéité $= \frac{\textbf{c.fréquence} \times |\textbf{c.fermeture}|)}{\textbf{c.homogénéité}}$; {**calcul de l'homogénéité**}
      $FC_i = FC_i \cup \{c\}$;
    fin si
  fin pour
  FCC_{i+1} = Gen-Generator(FC_i) ; {créer les générateurs pour l'itération i+1}
fin pour
retourner $FC = \bigcup_{j=1}^{i-1} \{FC_j.fermeture, FC_j.fréquence, FC_j.homogénéité\}$;

C.3 Exemple d'exécution

Prenons un exemple simple, utilisé dans [PBTL99], contenant 5 transactions et 5 items. Nous fixons le seuil $minfr$ à 2 transactions.

Id.	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

Initialisation

FCC$_1$.*générateur* = {A, B ,C ,D ,E} ;

i=1

FCC$_1$.*fermeture*={} ;
FCC$_1$.*fréquence*=0 ;

FCC$_1$= *Gen-Closure*(FCC1) ;
  t=1 t1={A, C, D}
  G$_1$= *Subset*({A, B, C, D, E}, {A, C, D}) = {A, C, D}
  g=A A.*fermeture*={}, alors A.*fermeture*={A, C, D}, A.*fréquence*=1, A.*homogénéité*=3
  g=C C.*fermeture*={A, C, D}, C.*fréquence*=1, C.*homogénéité*=3
  g=D D.*fermeture*={A, C, D}, D.*fréquence*=1, D.*homogénéité*=3

t=2 t2={B, C, E}
G_2 = *Subset*({A, B, C, D, E}, {B, C, E}) = {B, C, E}
g=B B.*fermeture*={B, C, E}, B.*fréquence*=1, B.*homogénéité*=3
g=C C.*fermeture*={A, C, D}, ∩ {B, C, E}={C} C.*fréquence*=2, C.*homogénéité*=6
g=E E.*fermeture*={B, C, E}, E.*fréquence*=1, E.*homogénéité*=3

t=3 t3={A, B, C, E}
G_3 = *Subset*({A, B, C, D, E}, {A, B, C, E}) = {A, B, C, E}
g=A A.*fermeture*={A, C, D} ∩ {A, B, C, E}={A, C}, A.*fréquence*=2, A.*homogénéité*=7
g=B B.*fermeture*={B, C, E} ∩ {A, B, C, E}={B, C, E}, B.*fréquence*=2, B.*homogénéité*=7
g=C C.*fermeture*={C} ∩ {A, B, C, E}={C} C.*fréquence*=3, C.*homogénéité*=10
g=E E.*fermeture*={B, C, E} ∩ {A, B, C, E}={B, C, E}, E.*fréquence*=2, E.*homogénéité*=7

t=4 t4={B, E}
G_4 = *Subset*({A, B, C, D, E}, {B, E}) = {B, E}
g=B B.*fermeture*={B, C, E} ∩ {B, E}={B, E}, B.*fréquence*=3, B.*homogénéité*=9
g=E E.*fermeture*={B, E}, E.*fréquence*=3, E.*homogénéité*=9

t=5 t5={A, B, C, E}
G_5 = *Subset*({A, B, C, D, E}, {A, B, C, E}) = {A, B, C, E}
g=A A.*fermeture*={A, C} ∩ {A, B, C, E}={A, C}, A.*fréquence*=3, A.*homogénéité*=11
g=B B.*fermeture*={B, E} ∩ {A, B, C, E}={B, E}, B.*fréquence*=4, B.*homogénéité*=13
g=C C.*fermeture*={C} ∩ {A, B, C, E}={C}, C.*fréquence*=4, C.*homogénéité*=14
g=E E.*fermeture*={B, E} ∩ {A, B, C, E}={B, E}, E.*fréquence*=4, E.*homogénéité*=13

FCC_1 :

Générateur	Fermeture	Fréquence	Homogénéité
A	A, C	3	11
B	B, E	4	13
C	C	4	14
D	A, C, D	1	3
E	B, E	4	13

On ne garde que ceux qui ont une fréquence ≥ 2.

FC_1 :

Générateur	Fermeture	Fréquence	Homogénéité
A	A, C	3	0,545
B	B, E	4	0,615
C	C	4	0,285

(E, {B, E}, 3) est éliminé car {B, E} apparaît déjà avec le générateur B.

FCC_2 = *Gen-Generator*(FC_1) ;
On fusionne les motifs de FC1 qui partagent leurs i-1 premiers items
Ici i-1=0, alors on a {A, B}, {A, C}, et {B, C}.
Les trois ensembles passent la première étape d'élimination
Dans la deuxième étape, A, C est éliminé.
p={A, C}
S_p = *Subset*({A, B, C}, {A, C}) = {A, C} // générateurs contenus dans p
s=A, s.*fermeture*={A, C}, p⊆s.*fermeture*,
donc p={A, C} est supprimé de FCC_2.*générateur*

FCC_2.*générateur* = {{A, B}, {B, C}}

i=2

FCC_2.*fermeture*={} ;
FCC_2.*fréquence*=0 ;

FCC_2 = *Gen-Closure*(FCC2) ;
t1={A, C, D}
G_1= *Subset*({{A, B}, {B, C}}, {A, C, D}) = {}

t2={B, C, E}
G_2 = *Subset*({{A, B}, {B, C}}, {B, C, E}) = {{B, C}}
{B, C}.*fermeture*={B, C, E}, {B, C}.*fréquence*=1, {B, C}.*homogénéité*=3

t3={A, B, C, E}
G_3 = *Subset*({{A, B}, {B, C}}, {A, B, C, E}) = {{A, B}, {B, C}}
{A, B}.*fermeture*={A, B, C, E}, {A, B}.*fréquence*=1, {A, B}.*homogénéité*=4
{B, C}.*fermeture*={B, C, E}, {B, C}.*fréquence*=2, {B, C}.*homogénéité*=7

t4={B, E}
G_4 = *Subset*({{A, B}, {B, C}}, {B, E}) = {}

t5={A, B, C, E}
G_5 = *Subset*({{A, B}, {B, C}}, {A, B, C, E}) = {{A, B}, {B, C}}
{A, B}.*fermeture*={A, B, C, E}, {A, B}.*fréquence*=2, {A, B}.*homogénéité*=8
{B, C}.*fermeture*={B, C, E}, {B, C}.*fréquence*=3, {B, C}.*homogénéité*=11

FCC_2 :

Générateur	Fermeture	Fréquence	Homogénéité
A, B	A, B, C, E	2	8
B, C	B, C, E	3	11

FC_2 :

Générateur	Fermeture	Fréquence	Homogénéité
A, B	A, B, C, E	2	1
B, C	B, C, E	3	0,818

$FCC_3 = Gen\text{-}Generator(FC_2)$;
Il n'y a pas de générateurs de FC_2 qui ont leur i-1=1 premier item en commun.
Donc FCC_3.*générateur*={}

i=3

FCC_3 est vide, donc Fin.

Résultat

Motif fermé fréquent	Fréquence	Homogénéité
A, C	3	0,545
B, E	4	0,615
C	4	0,285
A, B, C, E	2	1
B, C, E	3	0,818

Annexe D

Signification des abréviations utilisées pour les examens concernant les hépatites (Discovery Challenge PKDD 2002)

Abréviation	Description
4-COL	collagen type IV
4-COL/E	collagen type IV
A/G	the ratio of albumin to globulin
A1.GL	alpha1-globulin
A2.GL	alpha2-globulin
A2PI	alpha2-plasmin inhibitor activity
ACAIG-G	ACA IgG (AntiCardiolipin Antibodies)
ACE	angiotensin converting enzyme
ACLIG-G	cardiolipin antibody, beta2-glycoprotein-1 complex
AFP/E	alpha-fetoprotein
ALB	albumin
ALP	alkaline phosphatase
ALP-1	alkaline phosphatase isoenzyme, 1
ALP-2	alkaline phosphatase isoenzyme, 2
ALP-3	alkaline phosphatase isoenzyme, 3
ALP-4	alkaline phosphatase isoenzyme, 4
ALP-5	alkaline phosphatase isoenzyme, 5
ALP-sonota	alkaline phosphatase isoenzyme, other
AML1MTG8	AML1/MTG8mRNA
AMY	alpha-amylase
ANISO	anisocyte
APO-A1	apolipoprotein A1
APO-A2	apolipoprotein A2
APO-B	apolipoprotein B
APO-C2	apolipoprotein C2

APO-C3	apolipoprotein C3
APO-E	apolipoprotein E
APTT	activated partial thromboplastin time
ASK	antistreptokinase
ASO	anti-streptolysin O
AT3	antithrombin III
ATY-LY	atypical lymphocyte
B-AMY	beta-amylase
B-CELL	B cells percentage
B.GL	beta-globulin
BA	basophilic leukocyte
BLAST	blast cell
C	3 the third component of complement
C	4 the fourth component of complement
C-BIL	bilirubin, conjugated
C3	complement 3
C4	complement 4
CA	calcium
CA-50/E	carbohydrate antigen 50
CA125	carbohydrate antigen 125
CA15-3/E	carbohydrate antigen 15-3
CA19-9/E	carbohydrate antigen 19-9
CCR	creatinin clearance
CEA/E	carcinoembryonic antigen
CH50	complement, total
CHE	cholinesterase
CL	chloride
CONC-1	Fishberg concentration test
CONC-2	Fishberg concentration test
CONC-3	Fishberg concentration test
CONC-4	Fishberg concentration test
CONC-5	Fishberg concentration test
CPK	creatine kinase
CRE	creatinine
CRP	C-reactive protein
CU	copper
CYFRA	cytokeratin 19 fragment
D-BIL	bilirubin, direct
D-D	komento D-dimer, comment
D-daima-	D-dimer
DILU-1	Fishberg dilution test
DILU-2	Fishberg dilution test
DILU-3	Fishberg dilution test
DILU-4	Fishberg dilution test
DILU-5	Fishberg dilution test
DILU-6	Fishberg dilution test
DILU-7	Fishberg dilution test
DILU-8	Fishberg dilution test

EO	eosinophilic leukocyte
ERYTHR.B	erythroblast
F-10	coagulation factor X activity assay
F-11	coagulation factor XI activity assay
F-12	coagulation factor XII activity assay
F-13	coagulation factor XIII activity
F-2	coagulation factor II activity assay
F-5	coagulation factor V activity assay
F-7	coagulation factor VII activity assay
F-8	coagulation factor VIII activity assay
F-9	coagulation factor IX activity assay
F-CHO	cholesterol, free
FDP	fibrinogen/fibrin degradation products, blood
FDP	komento FDP, comments
FE	iron
FER	ferritin
FSH/E	follicle-stimulating hormone
FT3	triiodothyronine, free
FT4	thyroxine, free
G-GTP	gamma-glutamyltranspeptidase
G.GL	gamma-globulin
GLU	glucose
GOT	glutamic-oxaloacetic transminase
GPT	glutamic-pyruvic transminase
GSM	gamma-seminoprotein
GTT	glucose tolerant test
GTT-120	glucose level after 120 minutes (GTT)
GTT-15	glucose level after 15 minutes (GTT)
GTT-180	glucose level after 180 minutes (GTT)
GTT-30	glucose level after 30 minutes (GTT)
GTT-60	glucose level after 60 minutes (GTT)
GTT-90	glucose level after 90 minutes (GTT)
HB-SAG	Antibody of Hepatitis B Surface
HBA1	hemoglobin A1
HBA1C	hemoglobin A1c
HBC-AB	hepatitis B virus core antibody
HBD	hydroxybutyrate dehydrogenase
HBE-AB	hepatitis B virus e antibody
HBE-AG	hepatitis B virus e antigen
HBS-AB	hepatitis B virus surface antibody
HBS-AG	hepatitis B virus surface antigen
HBV-DNA	hepatitis B virus DNA, quantitative
HB	kakunin HB confirmation
HCT	hematocrit
HCV-AB	hepatitis C virus antibody
HCV-RNA	hepatitis C virus RNA, quantitative
HCVRNA	umu hepatitis C virus RNA, qualitative
HDL-CHO	high-density lipoprotein cholesterol

HGB	hemoglobin
HIV-AB	human immunodeficiency virus, antibody
HPT	complexed factor H, HPT
HTLV-1	human T-cell lymphotropic virus I antibody
HVA	homovanillic acid
HYPERSEG	hypersegmentation
I-BIL	bilirubin, indirect
I-P	inorganic phosphate
ICG	indocyanine green
ICG-10	ICG level after 10 minutes
ICG-15	ICG level after 15 minutes
ICG-5	ICG level after 5 minutes
IG-A	immunoglobulin A
IG-D	immunoglobulin D
IG-E	immunoglobulin E
IG-G	immunoglobulin G
IG-M	immunoglobulin M
IRI/E	immunoreactive insulin
K	potassium
LA	lupus anticoagulant
LAP	leucine aminopeptidase
LDH	lactate dehydrogenase
LDH1	lactate dehydrogenase isoenzyme, 1
LDH2	lactate dehydrogenase isoenzyme, 2
LDH3	lactate dehydrogenase isoenzyme, 3
LDH4	lactate dehydrogenase isoenzyme, 4
LDH5	lactate dehydrogenase isoenzyme, 5
LDL-CHO	low-density lipoprotein choleterol
LH/E	luteinizing hormone
LY	lymphocyte
MCH	mean corpuscular hemoglobin
MCHC	mean corpuscular hemoglobin concentration
MCV	mean corpuscular volume ?
METAMYEL	metamyelocyte
MG	magnesium
MO	momocyte
MPV	mean platelet volume
MYEL	myelocyte
NA	sodium
NEFA	non-esterified fatty acid
NH3	ammonia
NSE	neuron specific enolase
PC	protein C
PCT	plateletcrit
PDW	platelet distribution width
PIC/E	alpha2-plasmin inhibitor-plasmin complex
PIVKA-2	protein induced by vitamin K absence-II
PL	phospholipid

PLG	plasminogen
PLT	platelet count
POIKILO	poikilocyte
POLYCHRO	polychromatic cell
PRL/E	prolactin
PRO-F	protein fractionation
PROMYEL	promyelocyte
PSP120	phenolsulfonphthalein test
PSP15	phenolsulfonphthalein test
PSP30	phenolsulfonphthalein test
PSP60	phenolsulfonphthalein test
PT	prothrombin time
Pro-GRP	pro-gastrin releasing peptide
RA	rheumatoid arthritis test
RBC	red blood cell count
RDW	red cell distribution width
RET	reticulocyte
RF	rheumatoid factor, quantitative
RPR	syphilis serology test by RPR, qalitative
RPR-T	syphilis serology test by RPR, quantitative
SA	sialic acid (neuraminic acid)
SEG	segmented cell
SQ-CELL	squamous cell
ST.	stab cell
T-BA	bile acids, total
T-BIL	bilirubin, total
T-C/S	T cytotoxic/suppressor cells ratio
T-CELL	T cells percentage
T-CHO	cholesterol, total
T-I/H	T helper/inducer cells ratio
T3	triiodothyronine
T4/E	thyroxine, total
TAT/E	thrombin anti-thrombin III complex
TG	triglyceride
TH/TS	T helper/suppressor cells ratio
TIBC	total iron-binding capacity
TM	thrombomodulin
TP	protein, total
TPAI-C	tissue plasminogen activator, plasminogen activator inhibitor-1 complex
TPHA	syphilis serology test by TPHA, qalitative
TPHA-T	syphilis serology test by TPHA, quantitative
TSH/E	thyroid-stimulating hormone
TST	testosterone
TT	complexed factor T, TT
TTT	thymol turbidity test
U-ALB	albumin, urine
U-BIL	bilirubin, urine

U-CA	calcium, urine
U-CL	chloride, urine
U-CRE	creatinine, urine
U-CU	copper, urine
U-FE	iron, urine
U-GLU	glucose, urine
U-GTT120	glucose level after 120 minutes (GTT), urine
U-GTT180	glucose level after 180 minutes (GTT), urine
U-GTT30	glucose level after 30 minutes (GTT), urine
U-GTT60	glucose level after 60 minutes (GTT), urine
U-GTT90	glucose gevel after 90 minutes (GTT), urine
U-IP	inorganic phosphate, urine
U-K	potassium, urine
U-MG	magnesium, urine
U-NA	sodium, urine
UA	uric acid
UIBC	unsaturated iron-binding capacity
UN	blood urea nitrogen
VMA	vanillylmandelic acid
W-OTHER	other white blood cell
WBC	white blood cell count
ZTT	zinc sulfate turbidity test
anmonia	NH3
oudan	Jaundice
kanrei	cold agglutinin titer
gyoushuu	coagulation
ku-musu-	D direct Coombs(direct anti-globulin test)
ku-musu-	I indirect Coombs(indirect antiglobulin test)
kecchin-30	a blood sedimentation test (30min)
kecchin-60	a blood sedimentation test (60min)
kou	RNP ribonucleoprotein antibodies
kou	SSA/RO Sjogren's syndrome A(Ro) antibodies
kou	SSB/LA Sjogren's syndrome B(La) antibodies
kousentoromea	centromere antibody
koutaika	dilution level of antibody
komento	A comment A
komento	B comment B
komento	C comment C
komento	D comment D
komento	E comment E
komento	F comment F
komento	G comment G
komento	H comment H
komento	I comment I
komento	J comment J
komento	K comment K
komento	L comment L
komento	N comment N

sairoido	thyroglobulin antibodies
senketsu	occult blood, urine
tokiso	toxoplasma antibody
nyuubi	chyle
ben	hitohemo hemoglobin, stool
maikurozo-mu	microsomal antibody
youketsu	hemolysis

Annexe E

Pré-traitement de données issues de serveurs mandataires-caches

Sommaire

Lors de nos expérimentations, nous utilisons des données extraites de fichiers de trace de serveurs mandataires. Cette annexe décrit les étapes de préparation de ces données (cf. figure E.1). Elles sont issues de fichiers de trace de deux serveurs mandataires-caches de France Télécom R&D Caen. Elles ont été collectées pendant 17 mois (du 1er janvier 1999 au 31 mai 2000) et représentent les accès effectués par 331 utilisateurs.

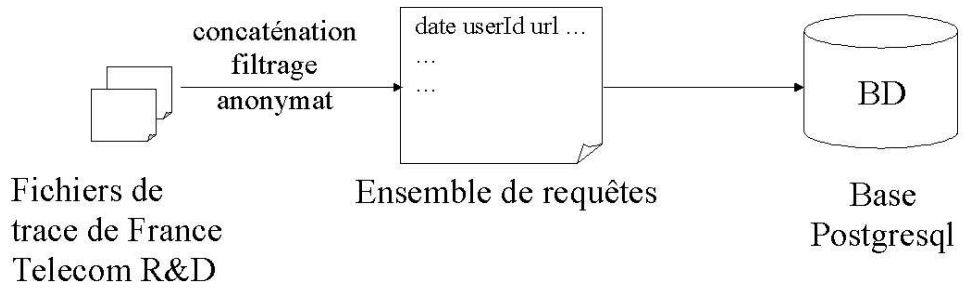

FIG. E.1 – Chaîne de pré-traitement des données

E.1 Traitement des données brutes

Les traces des serveurs mandataires correspondent à des fichiers au format texte où chaque ligne représente une requête : une demande ou l'envoi de données. Toutes les informations correspondant à la requête et à son traitement sont enregistrées.

E.1.1 Format des données

Voici le format classique d'une ligne de fichier de trace d'un serveur mandataire-cache Squid[26] :

time elapsed remotehost code/status bytes method URL rfc931 peerstatus/peerhost mimetype

- *time* est le temps Unix correspondant à la requête,
- *elapsed* est la durée en millisecondes du traitement effectué par le cache,
- *remotehost* correspond à l'adresse IP de la machine de l'utilisateur effectuant la requête,
- *code* est le résultat de la requête sur le cache (`TCP_HIT`, `TCP_MISS`, `TCP_DENIED`, ...),
- *status* indique le code retour HTTP (succès : 2xx, erreur coté client : 4xx, erreur coté serveur : 5xx, ...),
- *bytes* est la taille de l'objet *URL* en octets,
- *method* indique si c'est une demande d'un objet (`GET`), l'envoi de données (`POST`), ...,
- *mimetype* indique le type mime de l'objet correspondant à l'URL (`text/html`, `image/jpeg`, `video/mpeg`, ...),
- *rfc931* correspond au nom de l'utilisateur effectuant la requête (notons que cette information est désactivée par défaut),
- *peerstatus/peerhost* représentent les informations envoyées par l'éventuelle hiérarchie. Un serveur mandataire peut en effet avoir un cache père. Si il ne possède pas un document demandé, il intéroge alors son père.

Exemple de trace :

1062044424.526 0 10.xxx.xxx.xxx TCP_HIT/200 2125 GET http://www.toto.org/page.html - NONE/- text/html

Cette ligne signifie que le 28 août 2003 à 6 heures 20 minutes et 24 secondes, la machine 10.xxx.xxx.xxx a demandé l'objet *http://www.toto.org/page.html* correspondant à une page HTML de 2125 octets. La requête s'est traduite par un succès (*200*). De plus, le serveur mandataire disposait de l'objet dans son cache (*TCP_HIT*).

E.1.2 Préparation

Nous faisons l'hypothèse que chaque adresse IP correspond à un utilisateur. Pour des raisons de confidentialité, les adresses IP sont remplacées par des identifiants dans la totalité de nos données.

Les traces ont été épurées, seules les requêtes correspondant à un GET et n'ayant pas entrainées une erreur (4xx, 5xx) sont gardées. Toutes les requêtes portant sur les moteurs

[26] *http://www.squid-cache.org*

de recherche et les portails les plus connus (Google, Voila, Altavista, Yahoo, Wanadoo, Francetelecom, ...) sont supprimées.

Nous n'avons gardé que les requêtes correspondant à un objet de type mime *text/html*. Cette approche est justifiée d'un point de vue cognitif, car les pages textuelles sont fortement liées à une démarche explicite des utilisateurs. Les objets inclus sont plutôt, en terme de probabilité, une conséquence d'un accès primaire qui est le plus souvent textuel [Lan00a]. De plus, le texte est plus facilement caractérisable en terme sémantique que des images ou des vidéos par exemple. Cette restriction a aussi pour but de diminuer le cardinal de l'ensemble de toutes les requêtes et de faciliter l'exploitation des données.

Au final, le nombre total de requêtes sur les deux serveurs mandataires s'élève à 1510358 (392853 URL). Cet ensemble correspond à : 4563 requêtes par utilisateur, soit environ 8,6 requêtes par utilisateur et par jour.

E.2 Stockage dans une base de données

Après avoir été filtrées, les requêtes sont stockées dans une base de données PostgreSQL. Cela permet d'effectuer facilement toutes les opérations que nous désirons sur les données, comme par exemple la sélection des accès effectués sur une période.

Cette base est similaire à celle utilisée dans CYRANO (cf. chapitre 1). Elle est composée de différentes tables stockant toutes les informations sur les requêtes. Par exemple, pour les accès, les informations sont la date, l'identifiant de l'utilisateur, l'URL, la méthode, le code retour, etc. Pour les objets, on a l'URL, le type mime, ..., et des termes correspondant aux descriptions.

E.3 Calcul des descriptions des documents

Chaque document est décrit par un ensemble de termes extraits de son contenu. Les 10 termes les plus fréquents dans un document, qui n'appartiennent pas à une *stop list*, sont utilisés pour le décrire. Le français et l'anglais sont pris en compte. La stop list contient des mots qualifiés de "vides", comme par exemple les articles, les conjonctions et les prépositions.

Annexe F

Extraction de représentations condensées de motifs avec *MVMINER*

Sommaire

Dans cette annexe, nous décrivons brièvement le logiciel *MVMINER* développé à l'université de Caen par François Rioult[27]. Ce logiciel permet de découvrir les motifs δ-libres à partir d'une base de données transactionnelle, et est robuste en présence de valeurs manquantes [RC03a]. Nous avons utilisé ce logiciel pour extraire les motifs fermés fréquents (pouvant être dérivés des motifs libres), et calculer leur *homogénéité*. Les modifications pour ce calcul ont été réalisées en collaboration avec l'auteur, et sont similaires aux ajouts effectués dans l'algorithme *CLOSE* (cf. annexe C). Notons que nous n'utilisons pas la prise en compte de valeurs manquantes, ni la possibilité d'introduire des approximations sur le calcul des fréquences grâce au δ [BB00]. Ce dernier est fixé à 0 de manière à extraire les motifs exacts. Néanmoins, la prise en compte des valeurs manquantes et l'utilisation d'un δ peuvent ouvrir d'intéressantes perspectives pour Ecclat.

F.1 Motifs libres et motifs fermés

Un motif A est libre s'il n'existe pas de règle logique qui soit vérifiée entre ses sous-ensembles, i.e., il n'existe pas deux motifs distincs X et Y tels que $A = X \cup Y$, $Y \neq \emptyset$ et $X \rightarrow Y$ soit une règle vérifiée par toutes les transactions.

Une manière simple d'introduire cette notion est de représenter le treillis des motifs et de former des classes d'équivalence telles que les élements d'une même classe aient la même fréquence et soient reliés par l'inclusion ensembliste.

Les motifs libres sont alors les motifs minimaux (en nombre d'items) des classes d'équivalence. Les motifs fermés sont les maximaux. Remarquons que pour une classe, il peut y avoir plusieurs motifs libres, mais il y a un seul motif fermé.

[27] *http://users.info.unicaen.fr/~frioult/*

Formellement, un motif est libre s'il n'est pas inclus dans la fermeture de l'un de ses propres sous-ensembles stricts.

La figure F.1 présente un exemple de treillis de motifs dont la fréquence est supérieure ou égale à 2. AB et AE sont des motifs libres, et $ABCE$ est un motif fermé. Ils appartiennent à la même classe d'équivalence, et ont une fréquence égale à 2. AB est libre car il n'est pas inclus dans la fermeture de A ($=AC$), ni dans celle de B ($=BE$). Par contre, ABC n'est pas libre car la fermeture de AB est égale à $ABCE$.

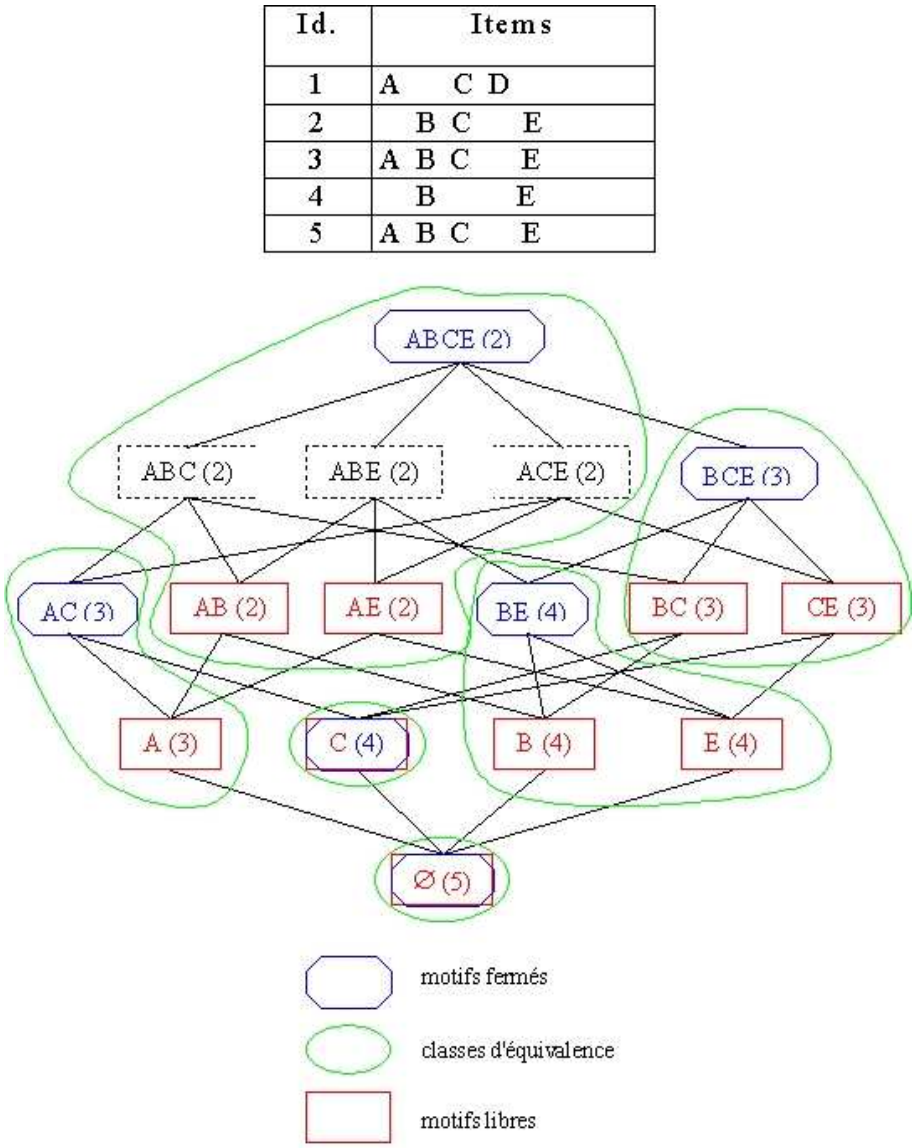

FIG. F.1 – Treillis de motifs fréquents et classes d'équivalence

Les motifs libres sont utilisés comme des générateurs de motifs fermés [BBR03]. La contrainte de liberté étant anti-monotone tout comme la contrainte de fréquence, les algorithmes par niveaux sont alors efficaces pour les découvrir.

Remarquons que les générateurs utilisés dans l'algorithme *CLOSE* [PBTL99] sont des motifs libres. Les étapes d'élagage utilisées dans cet algorithme permettent de garder un motif libre par classe d'équivalence de façon à trouver le motif fermé correspondant (cf. section C.1 en annexe C).

Nous présentons dans la section suivante le principe utilisé dans *MVMINER* pour découvrir les motifs libres et les motifs fermés fréquents.

F.2 Fonctionnement général

MVMINER procède de la même manière que les algorithmes par niveaux comme par exemple *APRIORI* [AS94] et *CLOSE* [PBTL99].

Il est donc composé de deux étapes :

- Génération : l'ensemble des candidats est dérivé de l'ensemble temporaire des motifs libres.
- Vérification : les candidats sont parcourus avec l'aide de la base de données pour déterminer s'ils sont libres fréquents.

Les motifs sont stockés dans un arbre préfixe. La génération des candidats de longueur l revient à fusionner les nœuds ayant le même parent et vérifiant que tous les sous-ensembles de longueur $l-1$ sont aussi libres (i.e. sont dans l'arbre).

MVMINER a été modifié pour extraire les concepts fréquents (motifs fermés fréquents et les transactions qui les supportent). Rappelons que la fermeture de A se note $h(A)$ (cf. section 3.5.2).

Les deux étapes ont alors été adaptées de la façon suivante :

- Pendant la génération : soit A un motif libre fréquent, deux motifs $A \cup \{a\}$ et $A \cup \{b\}$ sont fusionnés pour générer $A \cup \{a,b\}$ sauf si $\exists x \in A \cup \{a,b\}$ tq $x \in h(A \cup \{a,b\} \setminus \{x\})$.
- Pendant la vérification : les identifiants des transactions contenant un motif sont stockés pour construire l'extension $g(A)$. La fermeture de chaque motif est aussi calculée ([BBR03] donne une façon précise pour réaliser efficacement cette étape). Si le candidat est accepté, le concept correspondant $(h(A), g(A))$ est affiché.

Le calcul de l'*homogénéité* s'effectue d'une façon similaire à celle que nous avons détaillée dans la version modifiée de l'algorithme *CLOSE* (cf. section C.2 en annexe C).

Résumé

Le cadre de ces travaux correspond au projet RNRT CYRANO, dont le but est d'améliorer l'accès à l'information, à la fois en terme de sélection et de rapidité. Cette thèse propose un système de recommandation de documents aux utilisateurs. Ce système s'appuie sur une nouvelle méthode de découverte de clusters, appelée ECCLAT, pour regrouper les utilisateurs en communautés d'intérêt. Le principal avantage de cette méthode est de produire des clusters non-disjoints d'utilisateurs à partir de données qualitatives. ECCLAT a aussi été utilisée dans d'autres domaines comme la médecine. Un nouveau système de pré-chargement de documents, appelé PrefetchEm, est aussi proposé. Il repose sur une optimisation de la caractérisation des utilisateurs, basée sur la détection d'informations émergentes pour traduire leurs goûts et leurs tendances. Les expérimentations sont réalisées avec des accès réels effectués par des utilisateurs de serveurs mandataires-caches de France Télécom R&D.

Mots-clés : technologies de réplication, recommandation de documents, pré-chargement d'information, classification non-supervisée, clusters non-disjoints, treillis de concepts, motifs fermés fréquents, bi-sets, émergence d'information.

Title

Extraction of clusters from concept Lattice : Application to the discovery of interest communities in order to improve the access to information.

Abstract

This work is related to the CYRANO RNRT project, whose aim is to improve the access to information, in term of rapidity of download and information retrieval. This thesis proposes a new recommendation system of documents, which is based on a new method of clusters discovery, called ECCLAT, to group the users into interest communities. The main advantage of this method is to produce overlapping clusters of users from categorical data. ECCLAT was also used in other domains such as medicine. A new prefetching system, called PrefetchEm, is also presented. This system is founded on an optimisation of the user characterizations, based on the detection of emerging information in order to reflect their tastes and their tendencies. The experimentations are done with real accesses performed by users of proxies-caches of France Telecom R&D.

Keywords: replication technologies, recommendation of documents, prefetching, clustering, overlapping clusters, concept lattice, frequent closed itemsets, bi-sets, emergence of information.

Discipline

Informatique

France Télécom R&D — 42 rue des Coutures, 14066 Caen Cédex
Groupe de Recherche en Informatique, Image, Instrumentation de Caen — CNRS UMR 6072