République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Système d'information.

Sujet :

Le clustering (regroupement) de documents hiérarchique sur le Web

Présenté par : ADJENEG Nadia
HIMEUR Moufida

Promoteur : Mr FERFERA
Encadreur : Mr HOCINI Hatem

Organisme d'accueil : Centre de Développement des Technologies Avancées (CDTA).

Soutenue le: 27/09/05, devant le jury composé de :

Mr	Boukhlef	**Président**
Melle	Oussat	**Examinateur**
Mme	Boumahdi	**Examinateur**

- promotion 2004/2005 -

REMERCIEMENTS

Nous remercions avant tout Dieu d'avoir éclairé notre route et nous avoir donné la force, le courage et la volonté qui ont permis de réaliser ce travail.

Nous tenons à exprimer notre profonde gratitude et nos respectueux remerciements au chef de département et à tous les professeurs du département d'informatique de l'Université de Blida.

Nos vifs remerciements vont à Monsieur le Président et aux membres du jury d'avoir accepté d'évaluer notre travail.

Nous tenons à remercier notre promoteur Mr FERFERA pour son suivie et ses orientations.

Nous tenons à exprimer nos vifs remerciements à tous ceux qui, par leurs travaux, leurs idées, leurs présentations, leurs collaborations ou leurs relectures, ont participé de près ou de loin à la réalisation de ce mémoire, en particulier :

Mme Ghislaine pour le soutien dans les moments les plus difficiles, sa gentillesse, son rôle durant l'élaboration de cette thèse. Son sérieux restera exemplaire pour nous.

Tout le personnel de CDTA (Centre de recherche des technologies Avancées) spécialement le département de AS (Architecture des Systèmes), notre encadreur Mr HOCINI Hatem et les membres de l'équipe : BOUMARAF, BAIK, Lyes sans oublier GOUMAL Mehdi et TOUATI Tarek pour leurs aides et compréhensions.
Melle SOUABIR Fatiha pour son aide et son encouragement.

Mr Benjamin FUNG qui nous a aidées avec ses explications et ses propositions malgré la distance qui nous sépare.

Enfin nous tenons à remercier Mr SIAD Lamri (ingénieur) de son aide précieuse

Une mention particulière à Mr BENELKAID Ismail qui nous a aidées et encouragées pendant la réalisation de ce travail.

Dédicace

Je dédie ce modeste travail

A mes très chers parents, j'espère que vous serez toujours fiers de moi.

A mes grands parents qui m'ont encouragée et aidée avec leurs prière.

A mes sœurs Zohra et Samia.

A mes frères Adel, Omar, Walid et Aymen.

A ma voisine Fatiha qui m'a toujours aidée et encouragée.

A toute la famille, mes tantes, mes oncles, mes cousines et mes cousins.

A toutes mes amies Fatima, Imane, Hadjira, Malika, Lamya, Nabila, Habiba et Hayet.

A tous mes camarades d'étude et surtout Ismail, Nesrine, Dalila, Chahra, Merieme Ratiba et Fatima.

A mes meilleurs amis depuis ma première années d'étude.

A Moufida pour les bons moments qu'on a passé ensemble et à sa famille.

A tous ceux que j'aime et qui m'aiment.

A. NADIA

Dédicace

Je dédie ce modeste ouvrage

A mes très chers parents que j'aime très fort,

mon père qui s'est privé des caprices de
la vie au profit de mon bonheur,
ma mère qui s'est sacrifiée au bien
être de sa famille

A mes frères et sœurs Amina, Habiba, Khaled

et défunt Mohamed

A mes oncles et tantes, mes cousins et cousines et,

Pour n'oublier personne, à ma famille entière.

A tous mes amis et en particulier karima, Dalila,

Chahrazed, Mériame, Ratiba, Nessrine,
Hayet, Fatiha et bien sûr Ismail qui m'a tant
encouragée dans mes études pour l'obtention
de ce diplôme.

A mon binôme Nadia avec laquelle j'ai passé de très

beaux moments de ma vie et à toute
sa famille

A tous ceux que j'aime et qui m'aiment.

H. MOUFIDA

Sommaire

Chapitre II Règles associatives

Chapitre III Le clustering

Liste des figures

Liste des Tableaux

Résumé

Le développement explosif de l'information sur le web rend critique et indispensable l'élaboration de techniques et de modèles permettant de distinguer l'information importante et utile de celle qui est inutile. Pour répondre à ces besoins de découverte, un ensemble d'architectures, de démarches et d'outils, certains nouveaux, d'autres existant depuis longtemps, ont été regroupés sous le terme « **Data Mining** ». L'une des tâches les plus étudiées du Data Mining est sans doute la classification.

En effet le clustering (regroupement des documents) pièce maîtresse de data mining .Il vise à mettre les documents similaires ensemble dans des groupes (clusters). Le problème du clustering se résume en un problème de classification non supervisée. Ces clusters peuvent être classés dans une hiérarchie ou non : chaque fois un nouveau document arrive, il faut le mettre dans le ou les bons clusters.

L'objet de présent ouvrage est de faire une étude comparative de différents types et importantes méthodes du clustering , afin d'implémenter une méthode de classement appropriée. Cette méthode est utilisée dans le clustering hiérarchique des documents textuelles, elle est basée sur les «frequents itemsets » d'où son nom « Frequent Itemset Hierarchical Clustering »

Mots clés: Web mining, Data mining, Clustering, documents hiérarchiques, classification, Text mining, Les règles associatives, Frequent Itemset Hierarchical Clustering (FIHC).

Introduction Générale

Introduction Générale

L'événement marquant de ces dernières années dans le domaine de communication est caractérisé par l'écoulement énorme et la surabondance d'informations aux quelles nous faisons face chaque jour, ceci étant dû principalement à l'expansion de l'Internet, l'intranet, ainsi que de la messagerie électronique. Un grand nombre de données est disponible dans un temps minime et dans divers domaines de notre vie contemporaine : télécommunications, environnement, recherche, statistiques. La nature de ces informations varie en termes de volume, de disponibilité et d'importance.

L'assimilation de tout ce flux d'informations pose un grand problème quant au choix de son utilisation et de son efficacité au moment opportun et au bon endroit. Il a été constaté que la manipulation de l'information diffère selon le type d'utilisation à savoir :

- les éditeurs de l'information différencient leurs informations pour augmenter leurs valeurs ainsi que leurs revenus ;
- les directeurs d'information abaissent le coût des tâches qui exigent l'analyse des documents, et la fourniture d'un meilleur service à leurs clients ;
- les utilisateurs d'information ont l'accès direct à l'information appropriée, et découvrent de nouvelles idées et des relations y afférentes.

Ces besoins cruciaux ont amené les concernés à produire des logiciels de gestion des documents non structurés avec une grande nécessité de contrôler et de commander les grandes quantités d'informations textuelles et autres. Ces logiciels conduisent à de nouvelles informations exploitables par une gamme des systèmes de gestion des documents, des produits de balayage des textes, et des moteurs de recherche.

Les grandes entreprises utilisent une main d'œuvre nombreuse pour mettre ces documents dans une structure logique à usage postérieur.

L'une des solutions à cette problématique est d'automatiser et de regrouper les documents, d'où le terme scientifique, « document clustering » ce qui consiste à rassembler les documents semblables dans des groupes créés automatiquement permettant de donner à l'utilisateur une vision claire et facile sur les documents recherchés. L'accès à l'information devient alors plus précis et rapide.

Le travail présenté dans ce mémoire rentre dans un projet de recherche lancé par le Centre de Développement des Technologies Avancées (CDTA).

Le travail proposé entre dans le cadre du web mining et représente une contribution première à l'élaboration d'un système d'exploration de données sur le web.

Dans cette phase nous nous intéressons à l'application d'une méthode de regroupement (clustering) efficace basée sur l 'ensemble d'items les plus fréquents dans le document.

Notre étude permet la recherche de solution aux problèmes qui restent à résoudre en matière de clustering, malgré l'évolution des différentes méthodes existantes.

Ces problèmes se résument ainsi :

- la forme arbitraire des regroupements, génère souvent une limitation de l'exactitude des résultats ;
- la sensibilité de l'ordre des données d'entrée dans les différents algorithmes donne des solutions imprévisibles ;
- le traitement des données aberrantes dans les bases de donnée minimise l'exactitude des groupes ;
- la nécessité d'une connaissance antérieure du domaine afin de connaître la valeur des paramètres d'entrée dans les différents algorithmes.

Dans ce contexte notre contribution est la mise au point et l'implémentation d'une méthode permettant un clustering performant, cette méthode est fondée sur FIHC qui est efficace et *scalable*[1] et qui permet :

- de réduire considérablement le nombre des groupes par rapport aux documents ;
- de rendre pertinent l'ensemble des documents se rapportant à un groupe lorsque l'un des documents de ce groupe est pertinent dans une requête ;
- de visualiser un résultat sous forme d'une hiérarchie de groupe avec une vision globale des résultats.

Afin d'atteindre le but de ce travail, nous avons organisé notre mémoire comme suit.

Le premier chapitre sera consacré à l'étude des différents outils d'extraction de l'information : KDD(" Knowledge Discovery in Data base ") et ses différentes branches ("data mining" , "texte mining " , " web mining "). Nous présenterons les concepts fondamentaux, ainsi que les différents types d'applications.

Le deuxième chapitre, fait une présentation des règles d'associations pour la recherche des itemsets, avec les concepts fondamentaux ainsi que les algorithmes les plus utilisés, particulièrement l'algorithme « A priori ».

Dans le troisième chapitre nous aborderons le clustering en passant en revue les meilleures méthodes utilisées et en concluant avec une étude comparative.

Le quatrième chapitre sera consacré au développement de la méthode adoptée FIHC (Frequent Itemset based Hierachical clustering), en définissant en détail ses étapes.

Après avoir posé quelques repères théoriques sur ces nouvelles technologies, nous aborderons dans le cinquième chapitre la démarche de développement de la méthode " FIHC" suivant le cycle de vie en cascade en utilisant le langage de manipulation UML. Une implémentation de la solution retenue au niveau de la conception est établit, et enfin nous y présenterons quelques tests et résultats.

Dans la conclusion générale, nous faisons apparaître l'importance de la méthode FIHC.

[1] : Capacité de mise à l'échelle

Chapitre I

Data Mining

Chapitre I

Data Mining

1. Introduction

Aujourd'hui, nos capacités de collecte et de stockage de grandes masses de données de divers types, qui ont pris depuis ces dernières années une extension considérable et continuent d'évoluer rapidement, ont dépassé nos aptitudes à analyser, résumer et extraire la connaissance cachée à partir des données, car les méthodes classiques ne pouvaient plus manipuler les ensembles volumineux de données.

C'est à partir de ce problème, qu'est née une nouvelle génération d'outils, méthodes et techniques intelligents utilisés pour la fouille de données et la découverte de connaissances : ''Data Mining '' et ''Knowledge Discovery in Database (KDD)''.

La classification de cette grande masse de données est une phase très importante dans l'ensemble de tâches de Data mining qui est aussi une des plus importantes phases de processus de KDD. Le clustering est en effet une classification spéciale qui est considérée comme un des problèmes résolus par le data mining.

Dans ce qui suit, nous définirons ce qu'est le KDD enchaînée par une présentation de son processus pour obtenir une définition de Data Mining et sa relation avec le KDD, le Data Warehouse. Nous présentons ensuite les tâches de Data Mining, ses techniques, ses principales branches, quelques logiciels proposés dans le marché, et enfin quelques domaines d'application.

2. Définition du KDD

''Knowledge Discovery in Database'' (KDD) ou extraction de connaissances à partir de données (ECD), est un domaine aujourd'hui très en vogue, pour ne pas dire à la mode. Pour cela, on trouve autant de définitions du KDD qu'il y a de personnes à s'y intéresser. Cependant on peut le définir comme « un processus non trivial d'identification de structures inconnues, valides et potentiellement exploitables dans les bases de données » [FAY 96]. Cette définition est une des premières qui traite explicitement de KDD ; par la suite plusieurs tentatives de redéfinition sont apparues pour mieux préciser le domaine.

On peut aussi le définir comme une recherche de connaissances à partir d'une base de données. A partir d'un ''stock '' de données, on doit parvenir à un résultat lisible de ce qui compose la base de données, et ainsi prendre des décisions rapides. Alors le processus de KDD n'est pas une fin en soit, mais un outil à intégrer dans une politique [FAY 97].

Il consiste en l'extraction d'informations implicites, non connues et potentiellement utiles stockées dans des bases volumineuses [JEM 04].

Certains définissent le KDD comme un processus d'extraction de connaissances intensives, constituées d'interactions complexes, différées sur le temps, entre un être humain et une large base de données, soutenu par une suite hétérogène d'outils.

A cet effet, on utilisera des techniques de découverte de relations au sein de données, des techniques de reconnaissance de formes, d'analyse de données, le tout encapsulé dans

une technologie de type système expert, dans le but d'automatiser la découverte de l'information enfouie dans des bases de données typiquement très grandes [MAH 04].

Le KDD est un domaine de recherche en plein essor. Il se situe à la croisée de l'intelligence artificielle, des statistiques et des bases de données [HEB 03].

C'est un processus interactif et itératif qui peut analyser et interpréter un grand ensemble de données brutes afin d'en extraire des connaissances exploitables par l'utilisateur analyste qui y joue un rôle central [KOD 01]. Il cherche à équilibrer harmonieusement la qualité des données et la qualité de l'information (ou plus exactement la qualité des connaissances) mises à la disposition des décideurs [KOD 97].

Nous avons, d'après les différentes définitions suggérées, opté pour la suivante : le KDD peut être défini comme le processus qui génère automatiquement, à partir de base de données (relationnelles ou objets), les connaissances (informations) implicites contenues dans ces données.

3. Processus du KDD [LEF 98]

Le processus de KDD englobe un certain nombre de phases que nous expliquerons brièvement (**Figure I.1**), vu qu'il y a une seule phase qui sera détaillée dans les sections à venir et c'est la phase de modélisation (la phase de Data Mining).

3.1 Définition et position du problème

Cette première phase est celle où l'on expose le problème et où l'on définit les objectifs, les résultats attendus ainsi que les moyens de mesurer le succès de l'étape de Data Mining.

Dans cette phase introductive, il est intéressant de recueillir les intuitions et les connaissances des experts afin d'orienter le processus de découverte ou tout simplement pour identifier les variables les plus pertinentes susceptibles d'expliquer les phénomènes analysés. Cette phase se compose de trois étapes essentielles comme suit :

- formulation du problème : consiste à définir le problème et à le formuler sous une forme qui peut être traitée par les techniques et les outils de modélisation ;
- typologie du problème : savoir s'il s'agit d'un problème de structuration (lorsqu'on cherche à classifier des objets en sous-ensembles homogènes) ou alors d'un problème d'affectation (lorsqu'on connaît l'appartenance d'un élément à une ou plusieurs classes) ;
- les résultats attendus : avant de se lancer dans un processus de Data Mining, il faut savoir ce que l'on attend et ce que l'on compte faire de la connaissance extraite avec la prise en et considération l'identification des utilisateurs des résultats du processus de l'extraction de connaissances.

3.2 Collecte et sélection des données

Il s'agit dans cette phase de créer un ensemble de données cibles. Celles-ci sont collectées à partir de différentes documents papier, supports électroniques, fichiers internes ou externes, bases de données de type data warehouse ou data marts[2]. La recherche d'une sélection optimale de données est le point central d'un processus de KDD. Cette sélection nécessite souvent l'aide d'experts du domaine pour indiquer les variables qui ont une influence sur le problème à résoudre.

[2] : Data mart est un sous ensemble logique et physique d'un data warehouse.

La volonté d'intégrer toutes les variables avec un niveau très fin de définition entraîne un surdimensionnement du problème, qui nuit à la capacité de généralisation. Cette capacité de généralisation permet à un modèle de conserver un niveau de performance comparable entre la base d'apprentissage et la base de test. Si le nombre de variables est trop grand par rapport au nombre d'exemples, il devient presque impossible pour deux exemples de se trouver dans des parties proches. A ce niveau, l'élaboration de taxinomies[3], à partir des variables, permet d'en réduire le nombre (par exemple la transformation des départements en régions, des revenus en tranches de revenus ou de dates en intervalles).

Après la collection des données, l'utilisation de l'échantillonnage est nécessaire pour analyser les données de l'échantillon, afin de sélectionner les données pertinentes. En obtenant ces résultats, on peut les généraliser sur tout l'ensemble de données.

3.3 Nettoyage et préparation des données

Il s'agit de rassembler les données fournies par la phase précédente afin de les rendre plus fiables.

Une mauvaise qualité des données (erreurs de saisie, champs nuls, valeurs aberrantes), en fin de ce processus, peut produire des résultats qui sont peu pertinents, ce qui impose généralement cette phase de nettoyage de données. Celle-ci a pour objectif de corriger ou de contourner les inexactitudes ou les erreurs qui se sont glissées dans les données. Ces erreurs peuvent être de plusieurs natures :

- les données ayant des valeurs aberrantes, anormales par rapport leur type ;
- les données ayant des valeurs manquantes lors de la saisie, en obtenant des tables ayant des cases vides ;
- les données ayant des valeurs nulles.

L'existence des valeurs aberrantes ou nulles, ou alors l'absence de certaines valeurs ne convient pas toujours aux outils de ''Data Mining'' ; il faut donc gérer de manière spécifique ces valeurs selon plusieurs méthodes, parmi lesquelles nous citons :

1. exclure les enregistrements incomplets ; mais l'utilisation de cette méthode risque de réduire la base d'apprentissage.
2. remplacer les données par leur moyenne.
3. remplacer les données manquantes.
4. construire un premier *score*[4] , puis, grâce à des indicateurs statistiques, examiner tous les exemples qui contribuent trop fortement à la constitution de ce score. Un niveau de contribution anormal est souvent révélateur d'une donnée aberrante ou d'un exemple appartenant à une classe spécifique (par exemple, les encours d'une multinationale dans un échantillon de clients particuliers d'une banque).

3.4 Réduction et projection des données

Les données fournies par la phase précédente sont à ce niveau pertinentes et fiables. Il faut les transformer pour faciliter leur exploitation par les outils de modélisation. Ces transformations peuvent être de deux types, selon qu'elles modifient une ou plusieurs variables.

[3] : Une taxinomie est une méthode de classification des données

[4] : Un score est une sorte de note calculée à partir d'une équation. La détermination de l'équation se fait au moyen de techniques statistiques dites de *scoring*.

3.4.1 Transformation monovariable

a) La normalisation sert à aboutir à des ordres de grandeur comparables pour chaque variable. Il existe plusieurs techniques de normalisation, nous citons parmi elles les deux suivantes :

- la première consiste à soustraire à chaque valeur la valeur moyenne sur l'échantillon et à diviser cette différence par l'écart type calculé sur l'échantillon ;
- la seconde consiste à effectuer une transformation logarithmique de la variable afin de limiter l'impact de certaines valeurs exceptionnelles.

b) La transformation de dates en durée.

c) La transformation des données géographiques en coordonnées : cette approche consiste à adjoindre les coordonnées de longitude et de latitude (méthode de *géocodage*[5]), de façon à intégrer les contraintes de proximité dans le raisonnement.

3.4.2 Transformation multivariable

Lorsque les données brutes sont insuffisantes pour apporter un pouvoir prédictif, on procède par la combinaison de plusieurs variables élémentaires. Parmi les types de transformation multivariable on trouve : les ratios, les fréquences, les tendances, les combinaisons linéaires et non linéaires.

3.5 La modélisation (Data Mining)

La phase de modélisation qu'on appellera aussi phase de recherche du modèle, consiste à extraire la connaissance utile d'un ensemble de données bruitées et à la présenter sous une forme synthétique et compréhensible.

Il s'agit de la phase le plus souvent décrite sous le terme de ''Data Mining'' (détaillé dans la prochaine section) et qui repose, pour partie sur une recherche exploratoire de données.

C'est aussi l'étape de l'explicitation de l'objectif et de la stratégie d'analyse : exploration, classification, discrimination, segmentation, modélisation, prévision, etc....

Le choix des méthodes et des algorithmes se fait en privilégiant interprétabilité (modèles descriptifs) ou prédictibilité (modèle prédictifs) en mettant en œuvre des outils informatiques appropriées pour aboutir à une modélisation.

3.6 Interprétation et évaluation du modèle

L'évaluation du résultat permet d'estimer la qualité du modèle, afin de déterminer sa capacité ainsi que sa pertinence. Cette évaluation prend généralement une forme qualitative lors de la visualisation de la connaissance extraite sous forme graphique ou textuelle, qui contribue énormément à améliorer la compréhension et l'appréciation des résultats et facilite le partage de la connaissance ; et quantitative lors de la construction d'une base servant pour les tests pouvant ainsi vérifier si le modèle est capable de classer des données qu'il n'a jamais rencontrées auparavant.

[5] : Le géocodage est une technique de géomarketing qui transforme des adresses ou des éléments d'adresses en coordonnées géographiques.

3.7 Intégration de la connaissance

La connaissance ne sert à rien tant qu'elle n'est pas convertie en décision puis en action. Cette phase d'intégration de la connaissance consiste à implanter le modèle ou ses résultats dans les systèmes informatiques ou alors dans les processus de l'entreprise.

Elle est donc essentielle, puisqu'il s'agit de la transition du domaine des études au domaine opérationnel.

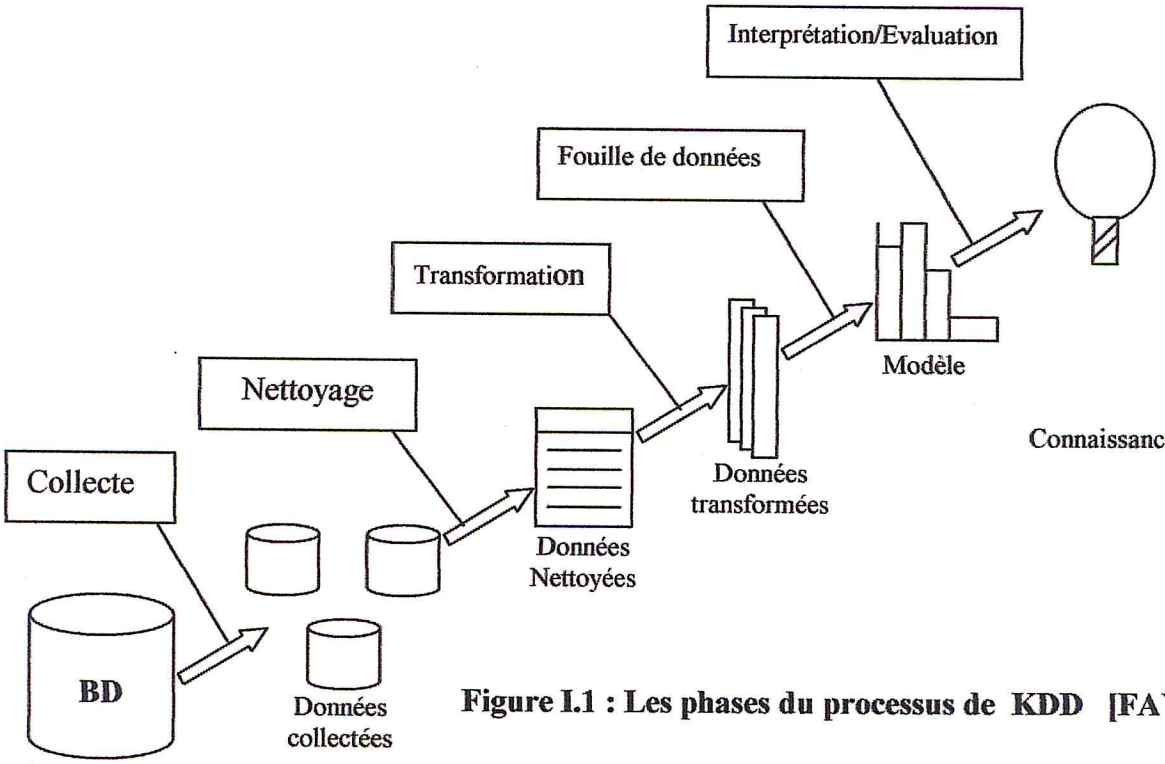


Figure I.1 : Les phases du processus de KDD [FAY 96].

Remarque : Dans le processus d'extraction de la connaissance (**figure I.1**) l'existence d'un "Data Warehouse" diminue le temps de réalisation d'un projet Data Mining et diminue (parfois élimine) quelques tâches pénibles dans le processus d'extraction de la connaissance (par exemple le nettoyage des données). A cause de cet intérêt nous allons parler brièvement de "data Warehouse" et sa relation avec le Data Mining dans les prochaines sections.

4. Définition du « Data Mining »

Le "Data Mining" ou " fouille des données" est un sujet d'actualité, il dépasse aujourd'hui le cercle restreint de la communauté scientifique pour susciter un vif intérêt dans le monde des affaires. Il existe plusieurs définitions du Data Mining ; nous en avons sélectionnés quelques unes :

- « La découverte de nouvelles corrélations, tendances et modèles par le tamisage d'un large volume de données » [DUV XX].
- « Un processus d'aide à la décision où les utilisateurs cherchent des modèles d'interprétation dans les données » (Karman Parsaye) [DUV XX].
- « Le Data Mining est un processus d'acquisition progressif de connaissances basé sur la combinaison et l'enchaînement de plusieurs techniques de modélisation, mais c'est l'intelligence de l'exploitant de données qui valorise l'information obtenue » [LEF 98].

- «L'extraction d'informations originales, auparavant inconnues et potentiellement utile, à partir des données » (Frawlay et Piateski-Shapiro) [DUV XX].
- «L'exploration et l'analyse, par des moyens automatiques ou semi-automatiques d'un large volume de données afin de découvrir des tendances ou des règles » (Michael J.A.Berry) [LEF 98].

Certains, considèrent le Data Mining comme le descendant et, selon certains, le successeur des statistiques telles qu'elles sont utilisées actuellement. Statistiques et Data Mining ont le même but, qui est de réaliser des modèles compacts et compréhensibles rendant compte des relations liant la description d'une situation à un résultat (ou un jugement) concernant cette description [BRE 97].

Parmi toutes les définitions suggérées, nous avons opté pour la suivante : le "Data Mining", traduit littéralement par "la fouille de données", est un processus non élémentaire pour l'exploration et l'analyse de grandes ensembles de données, généralement consignées dans des bases de données (relationnelles ou non) ou dans des entrepôts de données ("Data warehouses"), afin d'y découvrir de l'information implicite.

Cette information peut être de différentes natures : relations, corrélations, dépendances, associations, modèles, structures, tendances, classes, facteurs. Pour faire cette analyse et exploration, il faut utiliser des méthodes mathématiques, statistiques ou algorithmiques.

5. « Data Mining » et « KDD »

Le Data Mining et le KDD sont les noms souvent utilisés pour se rapporter à un champ de recherche très récent, qui est constitué de différentes techniques et méthodes provenant de plusieurs secteurs (statistique, apprentissage automatique, intelligence artificielles,...) employées pour extraire la connaissance à partir des ensembles de données [FRE 02].

Le terme "Data Mining", a été habituellement utilisé par les statisticiens, les analystes de données, et la communauté de système de gestion de l'information. Tandis que le KDD a été principalement utilisé par les chercheurs en intelligence artificielle et en apprentissage automatique [POL 00].

Il y a une autre distinction introduite et popularisée par [FAY 96]. Le point de vue adopté est que le KDD désigne l'ensemble du processus interactif et itératif d'extraction de connaissances utiles à partir des données. Tandis que le Data Mining est une principale étape dans ce processus, qui nécessite l'application des différents algorithmes pour assurer l'extraction des formes à partir des données, sans considérer les étapes où il question d'incorporer la connaissance du domaine, et l'interprétation des résultats.

6. « Data Mining » et « Data Warehouse »

Le processus de KDD présenté précédemment consiste en un rassemblement de données de plusieurs sources ; puis les organiser, et enfin transformer les données "brutes" en information utile (**Figure I.2**) [SCH XX].

Il est donc intéressant pour effectuer ce type de traitement de consolider les données dans une base de données spécifiquement conçue à cet effet, d'où le " Data Warehouse".

Un Data Warehouse, ou entrepôt de données, est défini comme une collection de données orientées sujet[6], intégrées[7], non volatiles[8] et historisées[9], organisées pour le support du processus d'aide à la décision. Il représente une vision idéale de maintenir un dépôt central (une grande base de données) de toutes les données. La centralisation des données est nécessaire pour maximiser l'accès et l'analyse d'utilisateur. [AND XX, LEF 98, BAS XX, GAR 00]

L'entrepôt de données est un prolongement de l'infocentre[10] dans un contexte reparti, où il devient possible de gérer des copies sur des machines différentes. Il s'agit d'intégrer et de matérialiser des vues de multiples sources de données, de les maintenir sur une machine séparée et de les exploiter pour le décisionnel, tout d'abord en OLAP (voir l'annexe A) [GAR 00].

Le Data Warahouse consiste à stocker et gérer des données, et le Data Mining va permettre d'analyser systématiquement ces données, pour en élaborer des données plus synthétiques et compréhensibles pour les décideurs [LEF 98].

Il y'a un autre grand avantage de stocker les données à utiliser dans un Data Warehouse, car le nettoyage[11] de données pour un Data Warehouse et pour le Data Mining est très semblable. Si les données ont été déjà nettoyées pour un Data Warehouse, alors elles n'auront très probablement pas besoin d'être nettoyées pour le Data Mining [CRO 99].

Un Data Warehouse précède donc le Data Mining (**Figure I.2**) Dans le processus d'extraction des connaissances à partir des données, mais n'est pas indispensable pour le Data Mining. Installant un Data Warehouse qui consolide des données de sources multiples, résout les problèmes d'intégrité des données [CRO 99].

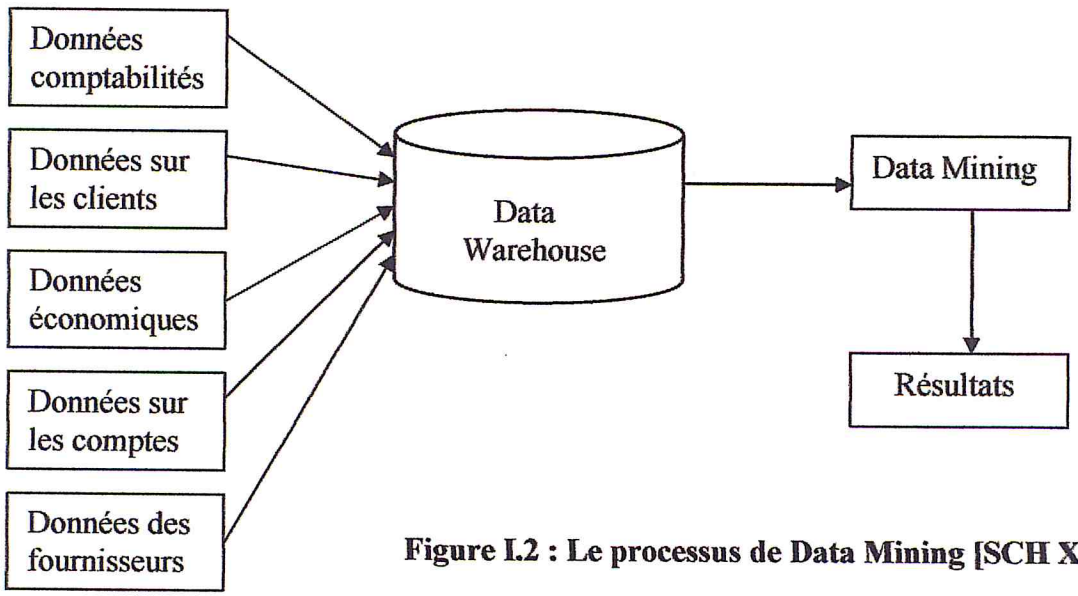


Figure I.2 : Le processus de Data Mining [SCH XX].

[6] : Les données sont organisées selon leurs sujets.

[7] : Toute donnée utile doit être intégrée dans le Data Warehouse. Cette données doit être mise en forme, unifier et doit avoir une description et un codage unique avant d'être intégrée, afin d'avoir un état cohérent.

[8] : Les données dans un data Warehouse ne peuvent en aucun cas être modifiées ou mises à jour.

[9] : Les données qui sont relatives à un passé de cinq à dix ans ou plus, sont utilisées pour des comparaisons, des prévisions et des prises de décisions.

[10] : Une technologie consistant à recopier les bases de données ou des parties de bases pour assurer les traitements décisionnels.

[11] : Une phase du processus KDD, il s'agit de corriger les erreurs et les inexactitudes de données.

7. Les tâches du Data Mining [COM 01] [DEL 02]

Les principales tâches ou natures de problèmes qui sont résolues par le Data Mining et ses techniques sont exposées ci-après :

7.1 La Classification

La classification est une opération de découpage d'un ensemble de données en un certain nombre de classes, et l'explication des caractéristiques de chaque classe. Elle consiste à examiner les caractéristiques d'un objet (donnée) nouvellement présenté afin de l'affecter à une classe d'un ensemble prédéfini.

Les objets à classer sont généralement représentés par des enregistrements et la classification elle même consiste à mettre à jour un champ de cet enregistrement par un code de classe spécifique. La classification se réfère à des événements discrets : oui, non, rougeole,...

Des exemples de tâche de classification sont :

- attribuer ou non un prêt à un client ;
- établir un diagnostic ;
- accepter ou refuser un retrait dans un distributeur.

7.2 L'estimation

L'estimation consiste à estimer la valeur d'un champ à partir des caractéristiques d'un objet. Le champ à estimer est un champ à valeurs continus : revenu, taille,...
Dans la réalité, l'estimation est utilisée dans un but de classification, il suffit de diviser le champ à estimer en intervalles et de considérer chaque intervalle comme une classe.

Parmi les exemples d'estimation on peut citer :

- estimer les revenus d'un client ;
- estimer la durée de vie (abonnement) d'un client.

7.3 La prédiction

La prédiction consiste à estimer une valeur future à partir de valeurs connues qui sont historisées. Nous présentons quelques exemples de tâches de prédiction à savoir :

- prédire les valeurs d'action ;
- prédire au vu de leurs actions passées les départs des clients.

7.4 Les règles d'associations

Elles consistent à déterminer les objets qui vont naturellement ensemble. L'exemple type est la détermination des articles qui se retrouvent ensemble dans un même ticket de supermarché d'où le terme « Analyse du panier de la ménagère ». Les règles d'associations sont utilisées par exemple pour ranger les articles qui vont généralement ensemble dans un même rayon ou dans un catalogue.

C'est une tâche qui nécessite de très grands jeux de données pour être effectives, pour cela, elle est considérée en plus comme une technique principales de Data Mining (nous allons étudier en détail cette technique dans le prochain chapitre) [LIN 97]. Si par exemple, les articles : boite pour chat et litière apparaissent souvent ensemble dans un même ticket de supermarché, on peut générer deux règles d'associations :

- les gents qui achètent des boites pour chat achèterons aussi de la litière avec une probabilité P1
- les gents qui achètent de la litière achèterons aussi des boites pour chat avec une probabilité P2.

7.5 La segmentation

La segmentation consiste à former des groupes (clusters) homogènes à l'intérieur d'un ensemble d'objets (données à segmenter). Pour cette tâche, il n'y a pas de classe à expliquer ou de valeur à prédire définie a priori, il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitué.

8. Les techniques de Data Mining

Avant de présenter les différentes techniques de Data Mining, il est important de bien préciser leur positionnement par rapport aux techniques statistiques. Les techniques de Data Mining utilisent les mêmes fondements théoriques que les techniques statistiques traditionnelles. Elles s'appuient sur des principes relativement similaires en introduisant un zeste d'intelligence artificielle et d'apprentissage automatique ; et chaque technique se base sur un ou plusieurs algorithmes connus.

Pour mieux illustrer ce qui vient d'être dit, nous présentons la figure **(Figure I.3)** qui montre une classification des méthodes et les techniques utilisées pour résoudre une tâche du Data Mining

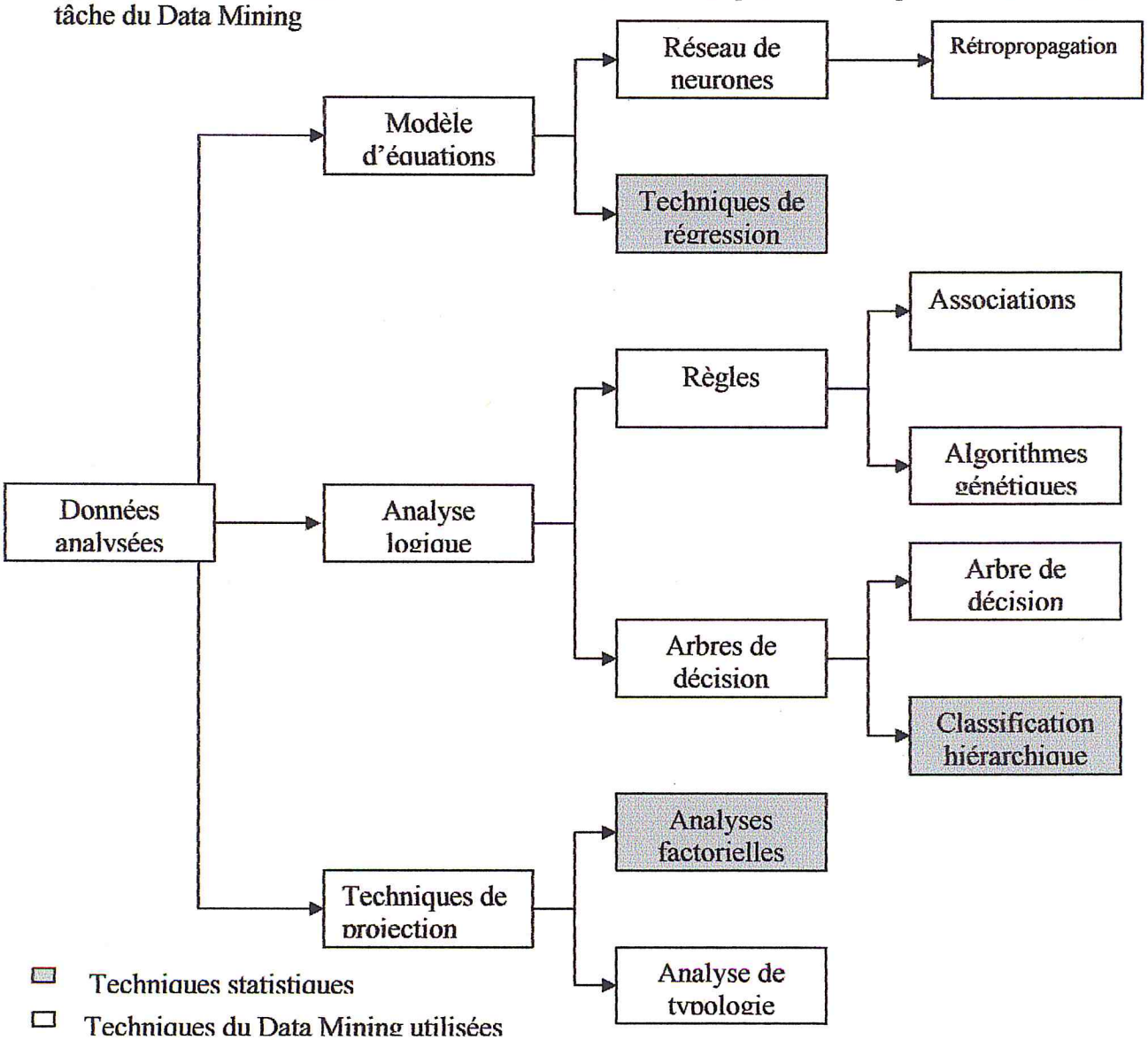


Figure I.3 : Classification des méthodes de Data Mining [LEF 98]

Dans ce chapitre, nous essayerons de résumer les techniques de Data Mining les plus utilisées, et dans le prochain chapitre nous allons détailler la technique qui nous intéresse.

8.1 Le raisonnement à base des cas (RBC) [BIC 99]

Les systèmes de **RBC** (raisonnement à base de cas), en anglais **CBR** (Case Based Reasoning), résolvent des problèmes par la comparaison d'exemples proches puisés dans un ensemble de cas préalablement stockés. Avec cette méthode de résolution, si une expérience passée et une nouvelle situation sont suffisamment " similaires ", toutes les conclusions appliquées à l'expérience passée restent valides et peuvent être appliquées à la nouvelle situation.

Les RBC suivent une procédure de recherche pour comparer les descriptifs du cas à traiter avec ceux des cas existants dans leur base interne. A ce titre, la capacité de résolution augmente au fil de l'arrivée de nouveaux exemples dans la base de référence. Donc, plus le nombre d'exemples est important, plus le RBC a de chances de retrouver un exemple proche, voire similaire.

Les principales étapes de construction d'un RBC sont les suivantes :

1. la collecte des données ;
2. la recherche des facteurs pertinents ;
3. l'indexation des données ;
4. les tests et l'amélioration de la performance.

Les applications des systèmes de RBC sont multiples. Cette technique rencontre un gros succès dans les domaines du service après-vente ou du diagnostic de panne, notamment dans les centres d'appels et les applications dites embarquées[12].

8.2 Les agents intelligents ou les knowbots [LEF 98]

Le terme ***knowbot*** est un condensé de *knowledge* et de *robot* ; il désigne ce que nous appelons en français les agents intelligents. Un agent est une entité physique ou abstraite, qui est capable d'agir sur elle- même et sur son environnement. Il dispose d'une représentation partielle de cet environnement et peut communiquer avec d'autres agents. Il poursuit un objectif individuel et son comportement est la conséquence de ses observations, de ses connaissances, de ses compétences et des interactions qu'il peut avoir avec d'autres agents et avec l'environnement.

La technologie des agents est récente. Elle est en évolution constante et les travaux de recherche sont montrés qu'un agent présente des caractéristiques précises. Il est :

- gérable : il prend ses instructions d'un homme ou d'un autre agent ;
- autonome : il préserve ses intérêts propres ;
- persistant : il sait ne rien faire sur de longues périodes ;
- fiable : il répond aux besoins de l'utilisateur ;
- prévoyant : il sait anticiper les besoins ;
- actif : il peut prendre des initiatives ;
- communicant : il interagit pour résoudre les différents conflits ;
- adaptatif : il sait changer d'environnement.

Les fonctions les plus innovantes d'un agent sont sa capacité à présenter ses intérêts et sa faculté à résoudre des conflits. A cause de ces fonctions, les agents sont utilisés dans

[12] : Un système embarqué est un système informatique qui doit se distinguer par sa robustesse de son matériel et le degré élevé d'autonomie de son logiciel. Un cas limite est celui des systèmes destinés à l'espace cosmique : le matériel informatique affronte alors un rayonnement très intense et destructif, tandis que le logiciel ne peut être réparé que difficilement par téléchargement.

plusieurs domaines d'applications. Ils sont utilisés à l'Internet comme des agents de navigation, en commerce électronique comme des conseillers électroniques, au suivi des tableaux de bord et dans autres applications.

8.3 Les algorithmes génétiques [LAU XX] [CRO 99]

Les algorithmes génétiques sont des procédures de recherche d'optimisation s'appuyant sur des techniques fondées sur la dynamique de la génétique biologique et de la sélection naturelle, c'est-à-dire, essentiellement la sélection, la reproduction et la mutation.

Les algorithmes génétiques décrivent l'évolution, au cours de générations successives, d'une population d'individus en réponse à son environnement. Ils sélectionnent les individus selon le principe de la survie du plus adapté. Comme leurs équivalents biologiques, les individus chromosomes sont constitués d'un ensemble de gènes qui ont chacun un rôle propre.

Dans une simulation génétique, les individus-chromosomes les mieux adaptés ont une probabilité plus élevée d'être sélectionnés et reproduits, donc d'être présents à la génération suivante. L'opération de mutation d'un gène permet de maintenir une certaine diversité dans la population. Cette diversité conduit à créer continuellement de nouvelles stratégies pour répondre aux changements aléatoires des gènes qui composent les chromosomes.

Quoique relativement récents, les algorithmes génétiques trouvent des applications dans de nombreux domaines. Ils sont utilisés dans l'industrie, en complément des techniques traditionnelles, pour résoudre des problèmes d'optimisation ou de contrôle de processus complexes (optimisation de la température d'un four ou de la pression d'un cylindre, par exemple), et dans le domaine des données spatiales et du géomarketing, pour optimiser des position dans l'espace (optimisation d'un plan d'affichage, choix des implantations d'automates bancaires, par exemple).

8.4 Les réseaux de neurones [LEF 98][COM 01]

Les réseaux de neurones sont connus depuis 1943, grâce aux travaux de MC Culloch et Pitts qui ont, les premiers, conçu un neurone logique. Le concept a connu un certain développement principalement dans le domaine de la cybernétique[13] qui a permis enfin l'apparition des réseaux de neurones d'aujourd'hui.

Les réseaux de neurones sont des algorithmes d'apprentissage, qui constituent un modèle de traitement informatique qui imite le fonctionnement de base du cerveau humain. Ils sont définis par des ensembles d'unités de traitement qui peuvent être des unités soit d'entrée, soit de sortie, soit cachées.

Les réseaux de neurones sont distingués en deux grandes catégories : les réseaux à apprentissage supervisé, dont on peut comparer le résultat avec une donnée en entrée, et les réseaux à apprentissage non supervisé, qui ne connaissent pas la réponse correcte, mais qui cherchent à découvrir la structure sous-jacente des données par une recherche des corrélations entre les entrées afin de pouvoir les organiser en catégories.

Souvent qualifiés de systèmes expérimentaux et immatures, les réseaux de neurones sont pourtant largement diffusés et utilisés de manière industrielle dans de nombreux secteurs d'activité.

[13] : La cybernétique est la science du contrôle des systèmes.

8.5 Les arbres de décisions [DEL 02] [LEF 98]

Un arbre de décisions est un enchaînement hiérarchique de règles logiques, qui comprennent une prémisse[14] (la première partie) et une conclusion [15] (la seconde partie), et qui sont construites automatiquement à partir d'une base d'exemples. Un exemple est constitué d'une liste d'attributs, dont la valeur détermine l'appartenance à une classe donnée.

La construction de l'arbre de décision consiste à utiliser des séquences de décisions pour subdiviser progressivement l'ensemble des exemples en sous-ensembles de plus en plus fins, ce qui constitue la forme arborescente de l'arbre.

L'ensemble d'origine, qui rassemble tous les exemples de la base, est appelé le nœud racine. Celui-ci est successivement découpé en sous-ensembles, appelés nœuds intermédiaires. Sur chaque nœud, une nouvelle évaluation est faite pour un découpage en sous-ensembles, pour avoir enfin des nœuds terminaux, qui sont appelés des feuilles.

Les applications des arbres de décisions sont de deux types : la construction d'un algorithme de segmentation d'une population dont les groupes d'affectation sont connus, et l'affectation d'une classe à un individu à partir de certains éléments descriptifs. Compte tenu de la simplicité du formalisme de restitution, les domaines d'application sont nombreux ; la liste ci-dessous reflète les principales applications :

- les études marketing, pour comprendre les critères prépondérants dans l'achat d'un produit ;
- le marketing direct, pour isoler les meilleurs critères explicatifs d'un comportement d'achat ;
- les ventes, pour analyser les performances par région, par enseignes ou par vendeur ;
- la gestion des stocks, pour analyser les ruptures, la qualité des fournisseurs ;
- l'analyse de risque, pour détecter les facteurs prédictifs d'un comportement de non-paiement.

8.6 Les réseaux bayésiens [LEF 98]

Les réseaux bayésiens sont une méthode classique d'apprentissage qui cherchent à construire un modèle de classification en utilisant la théorie des probabilités conditionnelles de Bayes. Cette méthode est utilisée pour associer une probabilité d'apparition d'un événement à la connaissance de certains autres événements.

Un réseau bayésien est un modèle graphique qui encode les probabilités entre les variables les plus pertinentes. Ce graphe orienté présente ces nœuds par des variables et les arcs par des dépendances entre les variables.

Parce que la méthode des réseaux bayésiens est relativement jeune, ses applications opérationnelles sont un peu moins nombreuses que celle des autres techniques exposées ici. Les premières applications sont la modélisation des processus d'alerte dans le domaine industriel et la prédiction du risque d'impayés dans le domaine des télécommunications.

---

[14] : La prémisse exprime une condition logique bâtie sur des tests portant sur des variables combinées par des opérateurs logiques (et, ou, non).

[15] : La conclusion est complétée par une fréquence d'appartenance (pour une variable qualitative) ou par une moyenne (pour une variable continue).

9. Quelques domaines d'application

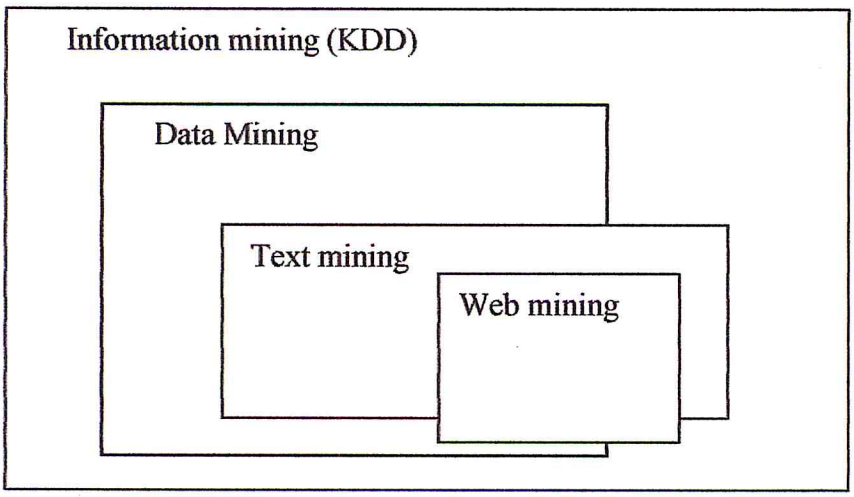
D'une manière générale, le Data Mining a une raison d'être partout où il existe de nombreuses informations et où les processus peuvent être améliorés, c'est-à-dire dans presque tous les secteurs d'activité.

- dans le domaine de l'analyse de risque , les compagnies d'assurance peuvent vouloir rechercher les caractéristiques des clients à haut risque et les compagnies bancaires déterminer si un crédit peut ou non être accordé à une certaine personne ;
- dans le domaine de marketing direct, le Data Mining offre un moyen de déterminer les caractéristiques (âge, profession, région,...) de la population à cibler pour un publipostage. Le courrier pourra ainsi être envoyé à la plus population offrant la plus haute probabilité de réponse ;
- dans tout secteur à forte concurrence, le Data Mining peut aider l'entreprise à identifier les clients susceptibles de quitter la société pour un concurrent ;
- dans la gestion des ressources humaines, pour l'étude du comportement des individus dans leur environnement de travail ;
- les applications sont également nombreuses dans le domaine médical, dans celui du contrôle des données financières en temps réel, dans celui de la détection du comportement frauduleux ainsi que de la segmentation de clientèle, etc....

10. Branches de Data Mining

Comme nous l'avons dit précédemment, le KDD est un processus d'extraction, à partir des bases de données, des connaissances, qui sont généralement utilisées pour aider à la prise de décision. Ce processus utilise des outils de Data Mining dans l'étape de la modélisation, et peut être appliqué soit sur des données d'où le "Data Mining"ou la fouille de données, soit sur des données textuelles d'où le" text mining", soit en appliquant les techniques de Data Mining sur le Web d'où le "Web mining".

Le schéma suivant présente ces différentes branches et présente aussi leur chevauchement, puis nous exposerons les deux dernières.



FigureI.4 : Génie des mines [PAS XX].

10.1 Le « Text mining »

Historiquement, le " Text mining" trouve ses origines dans trois domaines d'activité : dans l'amélioration des relations homme-machine surtout dans les premiers recherches en analyse linguistique et en traitement du signal , dans la traduction automatique ,et dans la caractérisation des textes pour faciliter le classement et la recherche des documents.

Le développement de ces travaux avec l'apparition du Data Mining a permis la naissance d'une nouvelle branche du Data Mining : le "text mining", qui est aussi considéré comme une des techniques de Data Mining [LEF 98]. Le text mining s'attache à construire une description objective et quantitative d'un texte en vue d'en dégager les traits marquants et de construire un résumé textuel ou graphique [LEF 98].

Dans cette section, nous allons définir le text mining, sa distinction avec le Data Mining, son processus, ses domaines d'application et finalement ses avantages et ses limites.

10.1.1 Définition du « text mining »

Le text mining est un nouveau domaine de recherche, qui essaye de résoudre le problème de la surabondance d'informations textuelles. A cause de son actualité nous trouvons plusieurs de définitions, parmi lesquelles nous allons citer quelques unes :

- « Le text mining regroupe l'ensemble des techniques issues du traitement automatique de la langue et de l'analyse de données permettant de retrouver des informations cachées dans de larges bases de données textuelles » (Bara et Nanceri, IBM) [LEF 98].
- « Le text mining est l'extraction d'informations à partir des formes ou patrons non manifestes (au sens de "hidden patterns") dans des grands corpus de texte. Autrement dit, l'objectif est le traitement de grandes quantités d'informations qui sont disponibles sous une forme textuelle et non structurée » [FEL 98].
- « Le text mining est la recherche des modèles dans des documents textuels, et peut être définie comme processus d'analyse du texte pour extraire de l'information utile pour des buts précis » [TEX XX].
- Certains le considèrent comme une prolongation normale du Data Mining [FAY 99].

La particularité du text mining réside dans le mélange de techniques linguistiques et statistiques provenant du Data Mining, de l'apprentissage automatique, du traitement automatique du langage naturel, de statistiques, du raisonnement à partir de cas et enfin de l'extraction de connaissances [POL 00].

Le text mining ressemble au Data Mining par sa volonté de découvrir de nouvelles informations à l'aide d'une analyse rapide et efficace des volumes d'informations sous formes textuelles. Il ne cherche pas à mettre au point des systèmes décrivant les langues dans leur ensemble ; mais il identifie certaines régularités de contenu et de forme pour effectuer une analyse simplifiée du texte [LEF 98].

Il repère les points clés ou marquants d'un texte en le découpant en unités élémentaires : mots, groupes de mots ou phrases. Puis, en utilisant ces algorithmes, il repère les plus fréquentes et calcule les associations entre ces unités [LEF 98]

Enfin, d'après les différentes définitions citées, nous avons adopté la suivante : le "text mining ", ou "Knowledge discovery in texts (KDT)", ou encore la fouille de textes (FT) peut être défini comme du Data Mining sur données textuelles. Il comprend

l'ensemble des techniques issues du traitement automatique du langage naturel et la fouille de données pour analyser et trouver les informations cachées dans des larges bases de données textuelles.

10.1.2 Distinction entre « Text Mining » et « Data Mining »

Le text mining est un nouveau domaine de recherche qui est considéré par certains comme une prolongation du Data Mining, c'est-à-dire comme une de ses techniques, mais il est considéré par d'autres comme une branche de Data Mining, autrement dit une fouille de données textuelles.

A cause de la diversité des points de vue, on trouve plusieurs distinctions entre le text mining et le Data Mining. Historiquement, le Data Mining est considéré comme la base du text mining au sens où celui-ci est l'extension du même but et du même processus vers des données textuelles. La distinction est donc fondée à son origine principalement sur la nature des données auxquelles s'adressent l'une et l'autre, d'une part des données numériques et factuelles, et d'autre part des données textuelles [POL XX].

Un autre élément de distinction est l'état de structuration des données. En général, le Data Mining travaille sur des données structurées et stockées dans des bases de données relationnelles. En revanche, le text mining travaille sur des données textuelles non structurées [FEL 98].

Le text mining se distingue du Data Mining également par les moyens techniques spécifiques qu'il faut employer pour traiter les données textuelles et non structurées [POL XX]. Pour bien éclairer cette distinction, nous allons présenter dans ce qui suit les différentes phases de processus du text mining, dans lesquelles il utilise ces différentes techniques.

10.1.3 Processus de Text Mining

Le processus de text mining qui ressemble au processus de KDD, mais sur des données textuelles et structurées, réunit un certain nombre de phases que nous allons expliquer brièvement :

A) Prétraitements de la collection de documents [POL XX]

Il s'agit dans cette phase de préparer les documents textuels à l'analyse en faisant la collecte à partir du Web sur des documents HTML, sur des bases de données soit bibliographiques soit textuelles.

Puis, en utilisant cette collection de documents, on met en place la gestion et le traitement des documents, qui sont en général sous la forme de données hétérogènes et sans structure fixe, dites données semi structurées (DSS), afin de leur appliquer un formalisme du type SGML[16] ou XML[17](voir l'annexe D) et de réaliser ainsi l'étiquetage de leurs attributs (par exemple, la date, le titre, les auteurs, la source, le corps du texte, et l'ensemble de termes caractérisant le document).

B) Traitement linguistique et indexation [LEF 98] [POL XX]

La phase du traitement linguistique et d'indexation, appelée souvent la fouille de textes, est destinée au traitement du langage écrit pour l'extraction de termes et l'indexation automatique des documents.

---

[16] : SGML ("Standard Generalized Mark-up Language") est une norme pour indiquer, par un système de balises spécifiques, la mise en page d'un document indépendamment du support matériel.

[17] : XML ("eXtensible Mark up Language") est un ensemble de spécifications désignant les règles à adopter afin de créer des documents structurés et universels mais libérés de toute contrainte.

Elle permet de faire la gestion des différents lexiques disponibles avec la suppression des termes existant en double dans les différents lexiques, l'indexation des notices textuelles par un traitement linguistique qui permet d'identifier les termes selon leur forme normale ou leurs variantes syntaxiques et de retenir une forme préférentielle en cas de synonymie. Mais cette phase de traitement suppose une validation humaine pour ne retenir que les meilleurs candidats de cette indexation automatique.

Nous pouvons citer les différentes étapes de cette phase.

1) La préparation des données

L'étape de préparation des données est une étape importante dans la fouille de textes. Elle permet de faire :

- ❖ l'élimination des mots vides, qui sont en général :
  - des articles (un, une, le, la),
  - des possessifs (ses, son),
  - des démonstratifs (ces, ce, cet, etc.…),
  - et des conjonctions (à) qui n'apportent pas de signification particulière au texte.

  Ces expressions sont trop présentes dans le texte, et sont spécifiques à la langue utilisée ;
- ❖ la lemmatisation[18], c'est-à-dire l'identification des différentes formes orthographiques d'un même mot :
  - racine de mots,
  - singulier ou pluriel,
  - pour les adjectifs, masculin ou féminin,
  - formes adverbiales, etc…

  Ce premier niveau de regroupement s'appuie sur un dictionnaire. L'exécution de cette analyse peut être automatique, surtout si le domaine est restreint. Il est néanmoins fréquent que le dictionnaire soit enrichi par l'utilisateur afin de tenir compte des formes d'écriture des différents interlocuteurs. Ce travail permet de ne définir comme mot-clé qu'une des formes du terme ;
- ❖ la recherche des mots signifiants, qui est faite à l'aide d'une analyse intermédiaire, qui est souvent nécessaire pour permettre à l'utilisateur de choisir les mots signifiants. Mais elle est tellement difficile à réaliser qu'aucun programme ne pourra détecter le sens de certains termes sans l'indication humaine sur le contexte. Un opérateur humain doit donc dresser une liste de mots ou d'expressions pour intégrer les spécificités de la langue et tenir compte du contexte de l'étude. Afin d'intégrer le degré de variabilité des contextes, les experts ont constitué des dictionnaires spécifiques à l'utilisateur, qui sont destinés à être utilisés automatiquement dans le cadre d'étude concernée.

2) La validation des indexations

Cette étape regroupe deux tâches distinctes:

1. un contrôle de la qualité de l'indexation automatique qui permet de supprimer les « bruits »[19] ;
2. un filtrage des termes qui risquent de perturber la phase ultérieure de classification.

---

[18] : La lemmatisation consiste en l'identification d'un mot par son lemme, c'est-à-dire par sa forme non fléchie : aimèrent → aimer ; aimeraient →aimer.

[19] : Le bruit est un terme non justifié selon un critère linguistique ou sémantique.

C) Analyse statistique [LEF 98]

1) La recherche des associations et la classification

Cette étape consiste à compter les termes ou les mots qui seront plus tard les mots clés de chaque texte ou document textuel, dans une séquence (qui peut se définir comme un nombre de mots (hors mots vides), un nombre de phrases ou de paragraphes). La séquence ne doit pas être trop courte, pour ne pas avoir beaucoup d'associations, ni trop longue, afin que les temps de calcul ne soient pas trop longs.

Ces mots clés sont utilisés pour établir les principales associations, en les regroupant dans des catégories homogènes telle que les mots clés souvent utilisés dans une même séquence seront classés dans une même catégorie.

Ce regroupement permet d'identifier le contexte d'utilisation, qui est défini par l'utilisation conjointe de plusieurs mots clés. Par exemple, lorsque les mots *problème*, *réception* et *inaudible* sont associés, on en déduit facilement un problème de difficulté de réception du message et donc une insuffisance de couverture de la zone. Le contexte d'appel du client sera donc identifié : couverture insuffisante.

2) La fixation du taux de support

Lors de la mesure des associations entre les mots clés, il faut être attentif à ne pas créer de « faux contextes ». Il est donc nécessaire de réaliser une phase de normalisation pour ajuster les poids des associations en fonction de la présence des mots dans le texte. Cette étape correspond à la définition du taux de confiance dans le calcul des associations, par exemple :

- si *problème de réception* est présent 50 fois ;
- et si *Angers* est présent 5 fois ;
- alors l'association *problème de réception+Angers* est présente 5 fois.

On peut voir qu'*Angers* est associé à 10% à *problème de réception* si on choisit le mot clé le plus présent (5/50), mais à 100% si on choisit le moins présent (5/5). Nous utiliserons donc majoritairement le support sur le terme le moins fréquent. Cette règle impose de définir un seuil minimal de présence du mot dans le texte, c'est le minimum support ; par exemple, les mots présents moins de 3 fois ne seront pas retenus.

Un même mot clé peut bien évidemment se retrouver dans plusieurs contextes. Afin de faciliter la compréhension globale du texte, les outils de text mining ont fait appel à des outils de représentation graphique dans la prochaine section.

D) Visualisation graphique

Après l'analyse des textes et la détermination des associations et des classes, une représentation graphique est nécessaire pour mieux visualiser les classes et les relations entre ces classes :

1) Le réseau sémantique [LEF 98]

Les associations entre les mots sont représentées sous la forme d'un réseau. Ce réseau ressemble fortement à un réseau bayésien par sa forme. Les nœuds représentent les mots et les liens mesurent la probabilité d'association entre les mots.

Afin de ne pas alourdir la représentation, seuls les mots et les associations les plus fréquents sont représentés. Ce type de représentation est dénommé réseau sémantique, qui permet de visualiser d'une manière quasi immédiate les mots et leur contexte.

Les concepts stockés dans le réseau sémantique sont reliés par des liens d'indexation. Grâce à ces liens, le réseau sémantique peut être utilisé comme un outil d'interrogation de la base de données textuelles traitée. En sélectionnant un mot (ou un groupe de mots), on peut visualiser tous les mots avec lesquels il est associé et accéder aux textes dans leur intégralité. Les liens construits entre le réseau sémantique et les données d'origine facilitent le travail de recherche et d'analyse.

2) La hiérarchisation des concepts [LEF 98]

Un second mode de représentation permet de réorganiser automatiquement le réseau sémantique sous une forme arborescente. Plutôt que de donner une représentation « à plat » du texte, il crée un arbre à partir des contextes les plus importants détectés dans le texte. La racine met en évidence le concept le plus important et les différentes branches correspondent aux contextes classés par ordre décroissant, il est possible d'identifier de manière descendante les concepts et les différentes déclinaisons de chacun d'eux.

La structure et l'utilisation de l'arbre sont différentes de leur équivalent dans le réseau sémantique. Ici, l'arbre n'est en effet pas récursif. Lorsqu'un concept est représenté ou sélectionné, seuls les concepts attachés à ce concept de manière décroissante sont représentés en éliminant tous les liens les plus faibles. Cet arbre correspond donc à une vision beaucoup plus épurée du texte ; sa structure hiérarchique et simplifiée permet à l'utilisateur d'appréhender plus rapidement les thèmes centraux du texte.

En règle générale, l'analyse peut introduire de la connaissance en modifiant l'organisation proposée par :

- l'incorporation ou l'exclusion de certains mots ou thèmes ;
- la modification du positionnement d'un mot ou d'un thème.

Remarque : cette phase d'analyse permet de combiner l'expertise humaine et la puissance informatique pour l'indexation des documents. Il est important de pouvoir sauvegarder cette organisation du contexte pour une utilisation future.

10.1.4 Les domaines d'application du Text Mining

Comme le Data Mining, le text mining sert à découvrir des renseignements, jusque-là inconnus, dans les bases de données textuelles, notamment à :

- découvrir les expressions les plus souvent employées dans un large volume de textes (par exemple, les comptes rendus d'appels d'un centre d'appel[20]) ;
- permettre les sélections des phrases ou des segments de textes en définissant un thème de recherche (indexation entre des contextes et des textes) ;
- ventiler les segments de textes par thème et sous-thème à l'aide d'un organiseur d'idées sous forme graphique (arbre hiérarchique) ;
- construire automatiquement des résumés, qui évitent à l'utilisateur de devoir lire intégralement le document (compression d'un texte de 90%) ;
- rechercher dans différents types de documents en utilisant le langage naturel : entretiens, lettres de réclamation, articles de presse, comptes rendus, etc. ;
- suivre l'évolution d'un marché (en étudiant les modifications des expressions à travers le temps) ;

[20] : Dans un centre d'appel, les téléopérateurs sont les personnes qui répondent aux appels. Pour limiter les formations nécessaires à ces téléopérateurs les systèmes de raisonnement à base de cas sont employées en tant que système d'aide à la décision pour diagnostiquer des pannes ou des problèmes d'utilisation et proposer des solutions de dépannage des utilisateurs.

- identifier de nouvelles associations de mots autour d'un sujet (recensement des mots les plus utilisés pour accéder à un site) ;
- croiser des informations textuelles, avec des informations stockées dans des bases de données (identification des profils de client les plus corrélées avec le contexte d'insatisfaction).

A cause de ces usages, le text mining est utilisé dans plusieurs domaines. Nous en citons quelques uns :

- la recherche des solutions à mettre en œuvre dans les documentations techniques. Le technicien n'a plus besoin de rechercher dans le sommaire. Il introduit les mots-clé relatifs au problème rencontré et accède directement aux différentes parties utiles du manuel. Une documentation technique indexée par un outil de text mining permet de gagner un temps considérable dans l'aide de diagnostic et peut être mise à disposition sur Internet .Le text mining est ici complémentaire des outils de raisonnement à base de cas.
- la recherche de précédents dans les archives juridiques, médicales.
- la veille concurrentielle avec des applications dans la supervision des brevets.

10.1.5 Avantages et limites du Text Mining [LEF 98]

Comme tout domaine et technique récent, le text mining offre des avantages mais aussi souffre de limites.

Les avantages

Grâce au text mining les entreprises peuvent tirer d'avantage des renseignements que leur apportent leurs clients via les centres d'appels, les messageries électroniques et les sites de discutions sur le net. Cette technique permet de transformer des stocks des documents électroniques en des sources d'alimentation pour comprendre les besoins des clients ; ses utilisateurs peuvent :

- voir le contenu « caché » des documents ;
- rapprocher des documents disséminés dans des divisions différentes et traitant le même thème ;
- effectuer des requêtes sur des documents textuels ;
- identifier et de résoudre des problèmes ;
- éliminer les goulets d'étranglement de recherche d'informations en identifiant les thèmes récurrents ;
- identifier des opportunités commerciales.

Les limites

Comme tout domaine de recherche récent le text mining ne répond pas à toutes les problématiques. De même que l'on n'utilise pas les techniques de Data Mining pour lancer une requête dans des bases de données, il ne serait pas raisonnable de mobiliser un outil de text mining dans la recherche des documents contenant un mot clé précisé. Les outils de text mining doivent donc être réservés aux tâches d'exploration et de recherche.

La sélection finale des textes les plus pertinents et la transformation des données en connaissance nécessitent un travail de synthèse de l'utilisateur. Les différentes technologies mises en œuvre sont un moyen d'accélérer le travail de découverte et de déchiffrage des informations ; elles apportent une aide précieuse dans le cadre de la recherche d'informations, mais une validation " humaine" de la pertinence du contenu est encore indispensable.

L'outil de text mining apporte une nouvelle puissance de traitement aux décideurs en identifiant des concepts et des relations au cœur des textes. Mais il appartiendra encore pendant de nombreuses années à l'homme de comprendre la signification et la pertinence de cette information par rapport à sa problématique.

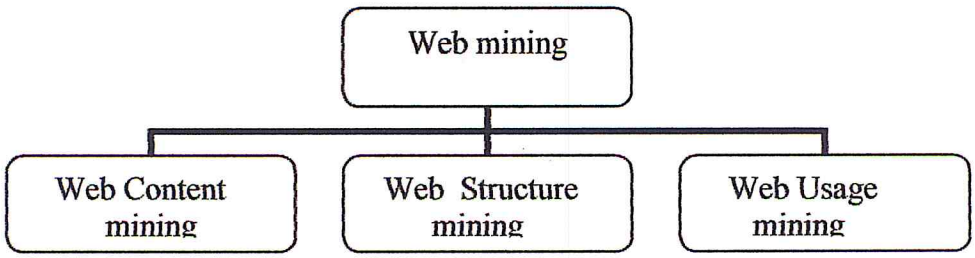
10.2 Le « Web Mining »

Le Data Mining s'applique dans différents contextes industriels pour l'évaluation de processus et la prise de décision. Lorsque les données analysées proviennent de l'Internet ou ont pour objectifs d'extraire des informations sur l'Internet on parle plutôt de Web mining.

Le Web mining est un domaine de recherche très récent, qui utilise les différentes techniques de Data Mining afin d'automatiser la recherche et d'extraire des connaissances utiles à partir des documents et services Web (O. Etzioni – 1996) [HAS 03].

Selon l'objectif visé, plusieurs types d'études peuvent être menées : les premières concernent l'analyse du contenu des pages Web, on parle de "Web Content mining" ; les secondes concernent l'étude des liens entre les site Web et l'extraction de l'information afin de déterminer les sites références d'une thématique *(" authorities")* et les sujets d'un site faisant de lui une référence , on parle alors de "Web Structure mining" ; enfin les dernières s'intéressent à l'usage, c'est-à-dire aux traces laissées sur les sites Web lors des connexions en utilisant des méthodes de traitement, de détection et d'analyse de comportement d'utilisateurs sur les sites Web. , on parle alors de " Web Usage mining "[HAS 03].

Ces trois types d'études sont présentées dans la figure 2 comme suit :



FigureI.5: le Web Mining [SCH XX].

11. Quelques solutions de Data Mining offertes sur le marché [COM 01]

De nombreux outils ont été construits à base des techniques du Data Mining ; citons quelques exemples de logiciels:

- ***ALICE d'ISoft*** : logiciel de Data Mining par arbres de décision, il permet de créer des modèles de segmentation.
- ***Intelligent Miner d'IBM*** **:** il couvre les tâches de segmentation, de prédiction, d'associations (temporelles ou non). Il utilise arbres de décision et les réseaux de neurones.
- ***Climentine d'ICL :*** il est construit à base d'arbres de décision, de réseaux de neurones, et d'autres techniques statistiques.

- ***SEM de SAS*** : il présente la particularité de permettre la construction de trois modèles d'apprentissage supervisé (réseau de neurones, régression et arbres de décision) en parallèle et de choisir, en final, le meilleur des trois.
- ***Saxon de PMSI*** : il est adapté à des problématiques d'apprentissage supervisé, telles que la classification, la prédiction, la prévision temporelle, mais aussi à des problématiques d'apprentissage non supervisé pour réaliser des segmentations automatiques.

12. Conclusion

Apres avoir fait le tour d'horizon de tout ce qui peut se rapporter au Data Mining, nous pouvons maintenant conclure que le Data Mining regroupe différents types d'outils, bien qu'ils n'aient pas les mêmes objets et n'utilisent pas les mêmes techniques ; néanmoins la réunion des objets que fournit chaque outil peut rapporter une meilleure extraction et représentation d'information.

Enfin, nous insistons sur le fait qu'une seule technique n'est pas suffisante pour résoudre un problème donné. Les règles d'association seront donc le sujet de prochain chapitre ; elles nécessitent comme les autres problèmes, la combinaison de plusieurs techniques en tenant compte du domaine où chacune est spécialisée.

Chapitre II

Règles Associatives

Chapitre II

Règles Associatives

1. Introduction

Comme nous l'avons présenté auparavant, le data mining est apparu comme une solution du problème de l'explosion des données. Il est constitué de l'ensemble des techniques et méthodes d'extraction des connaissances intéressantes, telles que règles, modèles, régularités, ou contraintes, à partir de données appartenant aux grandes bases de données.

La connaissance extraite devrait être non triviale, précédemment inconnue, implicite, et potentiellement utile parce qu'elle peut servir comme un important paramètre d'entrée pour prendre des décisions dans différents domaines industriels et commerciaux. Par exemple les commerçants veulent augmenter leur chiffre d'affaire en vendant aux clients un maximum de produits ; pour atteindre ce but il est impératif de savoir au préalable quels sont les articles que le client est susceptible d'acquérir à la fois.

Parmi les principales techniques de data mining la plus adéquate pour déterminer ce genre d'informations, est la technique d'extraction de règles associatives connue aussi sous le nom « des associations », qui est issue de la recherche en base de données [GAR 00] pour retrouver les corrélations entre produits dans le panier de la ménagère ou dans les caddies de supermarchés.

Dans ce chapitre nous décrirons l'extraction de règles associatives avec plus de détails que pour les autres techniques. Nous commencerons par la définition de la technique d'extraction des règles d'association, ses domaines d'application, le principe de la construction des règles d'association, et enfin les algorithmes les plus connus proposés pour l'extraction des règles associatives, et parmi eux l'algorithme Apriori qui va être utilisé dans l'algorithme " Frequent Itemset Hierarchical Clustering " (FIHC) pour la découverte des "frequents itemsets" ou les itemsets fréquents (dans ce mémoire nous allons utiliser souvent l'expression les itemsets fréquents au lieu "frequents itemsets").

2. Définition de la technique d'extraction des règles d'association (règles associatives)

Une règle associative ou règle d'association, qui est appelé aussi règle de décision est une règle conditionnelle qui se définit sous la forme d'une implication de la forme « si X (condition), alors Y (résultat) »ou plus formellement « $X \Rightarrow Y$ », où X et Y sont des ensembles d'objets (souvent des produits), traduisant le fait que si les objets X sont présents dans une transaction, alors les objets Y le sont avec une certaine probabilité [GAR 00].

Il est possible de mixer plusieurs conditions pour atteindre un résultat : « si X et Z, alors Y », comme il est possible aussi de combiner plusieurs opérateurs logiques insérés entre les conditions pour permettre l'extraction des règles conditionnelles dans des formats élaborés : « si X et non W, alors Y » [LEF 98].

La recherche des associations peut s'opérer sur l'ensemble des données (toutes les associations sont testées) ou sur une donnée cible (la conclusion étant fixée par l'utilisateur) [FAY 96].

Le principe général de la technique d'extraction des règles d'association est le suivant : on peut caractériser des sous-ensembles d'entités sur lesquelles ensuite on peut être conduit à prendre des décisions. Ces entités sont caractérisées par des valeurs ou des classes de valeurs, ou des attributs qui forment une chaîne logique de conditions. Ce qui ultérieurement sera traduit par « si les entités satisfont les conditions X_1, X_2,….., alors prendre la décision….. », décision en général liée à la variable Y dépendante [LEF 98].

Exemple : pour les personnes qui ont tel revenu, tel statut social, tel nombre de déménagements, et tel nombre d'emplois précédents, alors accepter une demande de crédit de tel type.

3. Quelques domaines d'application des règles associatives

Comme nous l'avons déjà vu pour chaque problème à résoudre par le data mining, il existe une technique qui lui est appropriée.

Parmi les domaines d'application très nombreux de la technique d'extraction des règles associatives, qui permettent de comprendre toutes les transactions commerciales nous citons les utilisations les plus fréquentes qui touchent :

- le domaine de l'analyse de risque, les compagnies d'assurance peuvent vouloir rechercher les caractéristiques de clients à haut risque et les compagnies bancaires déterminer si un crédit peut ou non être accordé;
- le domaine du marketing direct, cette technique sert à déterminer les caractéristiques (âge, région, profession…) de la population à cibler pour un publipostage ;
- le domaine de la grande distribution ; il est possible de déterminer des profits de consommateurs, l'effet des périodes de promotion, le contenu du panier de la ménagère, etc….

Plus généralement, l'analyse des associations s'applique avec succès à tous les problèmes dans lesquels l'apparition d'un événement est conditionnée par des événements passés : analyse des pannes dans l'industrie ou étude des décisions en sociologie [LEF 98].

4. Principe de construction des règles d'associations

Pour mieux illustrer cette démarche nous commencerons par donner les définitions les plus utiles qui concernent les règles d'association sous le titre de terminologie. Nous enchaînerons par l'explication de l'algorithme général d'extraction de ces règles, sans oublier l'application d'un exemple illustratif simple au fur et à mesure.

4.1 Terminologie des règles d'association [FAY 96] [MAH 04]

Etant donné une base de données BD (de transactions d'achats par exemple), en utilisant cette base nous allons citer quelques formalisations:

➔ soit $I= \{i_1, i_2, i_3, \ldots i_m\}$ un ensemble de *m* attributs distincts également appelés "***items***" (ou articles) ;

- soit D un ensemble de transactions où chaque transaction T est un ensemble d'items appelé "***itemset***" tel que $T \subseteq I$, c'est-à-dire que chaque transaction est de la forme <TID, i_1; i_2;…; i_k>, où TID est un identifiant qui identifie chaque transaction ;
- un "itemset" contenant k éléments est appelé "***k-itemset***", par exemple {A, B} est un 2-itemset ;
- soit X un ensemble d'items (itemset), T est dit contenant X si et seulement si $X \subseteq T$;
- une règle associative est une implication de la forme $X \Rightarrow Y$ où les itemsets X, $Y \subseteq I$ et $(X \cap Y=\varnothing)$;
- chaque règle associative appartenant à l'ensemble de transaction D est mesurée par deux indicateurs :
  - ***un support "s"*** d'un itemset lorsque un s% des transactions de D contiennent cet itemset ;
  - ***une confiance "c"*** d'une règle correspond à une probabilité c% qu'une transaction de D contienne Y sachant qu'elle contient X ;
- lorsqu'un itemset satisfait un paramètre entré par l'utilisateur qui est appelé le support minimum ("minimum support"), alors c'est un itemset fréquent ou "frequent itemset".

Après avoir vu les principales formalisations liées aux règles associatives, nous allons maintenant présenter ces définitions :

A) Support d'une règle associative (« Rule support ») [GAR 00]

Le support d'une règle est une mesure indiquant le pourcentage de transactions qui vérifient une règle associative ; en d'autres termes c'est le nombre de fois où un ensemble d'items coïncident.

En désignant par |X| le nombre de transactions comportant l'itemset X et par |BD|le nombre total de transactions dans la base, on a :

Support(X ⇒Y) = |X&Y|/ |BD| =s% de transactions vérifiant la règle.

Il faut noter que le support de X ⇒Y est le même que celui de Y ⇒X. On peut donc simplement le noter support (XY), car il s'agit bien finalement du support des itemsets XY (union des X et Y).

Exemple

Sur un ensemble de 500 000 transactions o na :
20 000 contiennent du café ;
30 000 contiennent du lait ;
10 000 contiennent du café et du lait.

Le support de chaque itemset est calculé comme suit :
support ({café}) = (20 000/500 000)*100%= 4% ;
support ({lait}) = (30 000/500 000)*100%= 6% ;
support ({café, lait}) = (10 000/500 000)*100%= 2%.

Dans cet exemple le lait et le café coïncident dans 10 000 transactions sur 500 000 existantes, alors le support est de 2%.

Le support n'est pas suffisant pour évaluer la force d'une règle. Il se peut en effet qu'une règle corresponde à des produits rarement achetés, mais soit très valide. Par exemple, lorsqu'on achète une brosse à dents, on achète très souvent du dentifrice. Cependant, on change rarement de brosse à dents. Donc, la règle (brosse à dents ⇒ dentifrice) a un faible support bien qu'elle soit très valide !

Pour mieux mesurer la validité d'une règle, la notion de **confiance** a été introduite, nous allons l'expliquer dans la prochaine section.

B) Confiance d'une règle associative (« Rule confidence ») [GAR 00]

La notion de confiance ou bien la confiance ("confidence") est une mesure indiquant le pourcentage de transactions qui vérifient la conclusion d'une règle d'association parmi celles qui vérifient la prémisse.

En d'autres termes elle mesure le degré de dépendance d'un ensemble d'items d'un autre ensemble d'items [LEF 98].

En utilisant les notations introduites dans la définition précédente, on obtient :

Confiance (X ⇒Y) =|XY|/ |X| = c% de transactions vérifiant l'implication

On peut aussi calculer la confiance à partir du support comme suit :

Confiance (X⇒Y) = support (XY) / support (X)

Exemple

Dans l'exemple précédent on constate que 20 000 transactions contiennent du café et parmi ces 20 000 transactions, 10 000 contiennent du lait, cela implique que quand les gens achètent du café , ils achètent aussi du lait dans 50% des cas. Alors la confiance pour la règle (café ⇒ lait) est de 50% (10 000 / 20 000).

Pour la règle inverse (lait ⇒ café) sa confiance est trouvée par 33% (10 000 / 30 000) c'est-à-dire que lorsque les gens achètent du lait, ils achètent également du café dans 33% des cas.

Remarque

L'expérience a montré que les règles intéressantes sont celles qui ont un support supérieur à un minimum (appelé minsup) et une confiance supérieure à un minimum plus élevé (minconf). Par exemple, on recherche toutes les règles ayant un support supérieur à 0.1 et une confiance supérieure à 0.7, pour que les règles trouvées soient intéressantes.

4.2 L'algorithme général d'extraction des règles d'association

Plusieurs algorithmes ont été proposés pour l'extraction des règles d'association, mais tous tournent autour de deux grandes et mêmes étapes qui sont [FAY 96] :

1) la découverte de ce qui est appelé les "frequent itemsets" ou les itemsets fréquents, où les itemsets fréquents sont un ensemble des items qui coïncident ensemble s fois (s étant le support déjà défini ci dessus) et s est supérieur au support minimum fixé par l'utilisateur, c'est-à-dire qu'une fraction minimum de transactions contient ces itemsets. Nous illustrons cette étape dans l'exemple prochain.

2) la génération des règles d'association à partir de l'ensemble des "frequent itemsets" trouvés lors de l'étape précédente ; l'idée générale de cette étape est de formuler toutes les implications possibles entre les différents attributs d'un même "frequent itemset" puis d'extraire, parmi ces implications, celles qui sont valides et qui satisfont le support minimum et la confiance minimum. Il est facile de calculer cette étape, mais elle est seulement utile pour l'extraction des règles d'association et n'est pas applicable dans FIHC (voir chapitre 4). Pour extraire les règles associatives valides il faut suivre la règle suivante :

si ABCD et AB sont deux fréquents itemsets (leurs supports sont supérieurs au support minimum fixé), on peut déterminer si AB⇒CD est une règle valide en calculant le ratio R : si R >= minconf (la confiance minimum fixée par l'utilisateur), alors la règle est valide, sinon elle est rejetée (voir l'exemple).

R = support (ABCD) / support (AB)

Exemple

Le tableau 1 présente une petite base de données de transactions T d'un supermarché contenant seulement quatre transactions avec les articles correspondants achetés. Supposons que le support minimum entré est de 50%, c'est-à-dire qu'un itemset i est fréquent si et seulement si au moins deux sur les quatre transactions contiennent l'itemset i.

Le tableau 2 présente tous les itemsets fréquents de T. Par exemple l'itemset {A} est fréquent parce qu'il apparaît dans trois transactions et son support est de 75%. De même, les itemsets {B} et {C} ont un support de 50%, ainsi sont considérés comme "frequent 1-itemset". {A, C} est "frequent 2-itemset" parce que tous les deux items A et C apparaissent ensemble en deux transactions avec un support de 50%. Cependant {A, B} n'est pas un "frequent 2-itemset" parce que les deux items A et B n'apparaissent ensemble que dans une seule transaction.

TID (l'ID de la transaction)	Items (les articles achetés)
100	A, B, C
200	A, C
300	A, D
400	B, E, F

Tableau II.1 : Base de données de transaction T.

Les itemset frequents	Le support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

Tableau II. 2 : Les fréquents itemsets de T.
(Minimum support =50%)

5. Les algorithmes les plus connus proposés pour l'extraction des règles d'association

Depuis la mise en place de la technique des règles associatives, un problème qui était posé et resté toujours poser pour les outils de data mining est le suivant : comment extraire les règles intéressantes pour l'analyse, c'est-à-dire des règles présentant des qualités de support et de confiance supérieures aux minima fixés ? Autrement dit, comment optimiser les calculs des indicateurs support et confiance sur de larges bases de données, et comment éliminer les règles inintéressantes ?

Les algorithmes actuels se basent sur la recherche des règles pertinentes pour le support, c'est-à-dire qu'ils extraient toutes les règles de support supérieur à une valeur donnée minsup et calculent les supports associés.

En plus de la pertinence des règles extraites, ces algorithmes essayent d'améliorer le temps d'exécution, et de réduire l'espace mémoire.

Remarque

Lorsqu'on dit algorithme d'extraction de règles associatives, on parle essentiellement d'algorithmes de découverte des itemsets fréquents, car la seconde étape (extraction des règles à partir les itemsets fréquents) est pratiquement commune à tous les algorithmes.

Parmi les algorithmes les plus performants proposés pour l'extraction de règles associatives nous citons :

- l'algorithme Apriori ;
- l'algorithme Apriori TID ;
- l'algorithme Apriori partitionné ;
- l'algorithme Bitmap ;
- l'algorithme Eclat ;
- l'algorithme " Count Distribution";
- l'algorithme " Data Distribution".

Nous allons maintenant étudier l'algorithme Apriori et nous verrons les autres algorithmes dans l'annexe B.

5.1 Présentation de l'algorithme Apriori

La recherche des règles intéressantes s'effectue donc par le calcul des supports de tous les ensembles d'objets et par rétention des ensembles de support supérieur au minimum demandé minsup pour trouver les ensembles fréquents.

Le premier algorithme proposé dans cette ligne est l'algorithme Apriori, qui est un algorithme fondamental proposé par R. AGRAWAL et R. SIKRANT en 1994 [AGR 94]. Il est la base de nombreux algorithmes de recherche de règles d'association.

Avant de présenter l'algorithme il faut tout d'abord admettre les notations et les propriétés des itemsets citées ci-dessous :

A) Les notations utilisées dans l'algorithme Apriori

L'algorithme Apriori et ses dérivés utilisent des notations qui sont regroupées dans le tableau ci-dessous :

k-itemset	Un itemset ayant k items
C_k	Un ensemble de k-itemsets candidats (potentiellement fréquents). Chaque membre de cet ensemble contient deux champs : • un pour l'itemset ; • l'autre pour le compteur du support.
L_k	L'ensemble des itemsets fréquents (ceux avec un support minimum). Chaque membre de cet ensemble contient également deux champs : • un pour l'itemset ; • l'autre pour le compteur du support.
C_{kchap}	L'ensemble des k-itemsets candidats lorsque les TIDs des transactions générées sont associés avec les candidats.

Tableau II.3 : Les notations utilisées dans l'algorithme Apriori [AGR 94].

B) Propriétés des itemsets [MAH 00]

Nous introduisons les principales propriétés des itemsets pour mieux éclairer les différentes étapes et concepts de cet algorithme :

Propriété 1 : Tout sous-ensemble d'un ensemble fréquent est fréquent.

Soit A un itemset fréquent, c'est-à-dire que le support (A) > minsup (minimum support). Supposons qu'il existe À ⊂ A tel que support (À) ≤ minsup. On a alors :
support (À) ≥ support (A) puisque À ⊂ A, ce qui implique
support (A) ≤ support (À) ≤ minsup et donc support (A) ≤ minsup et A est un itemset infréquent, ce qui contredit l'hypothèse.
Exemple : si {A, B, C} est un ensemble fréquent, {A}, {B}, {C}, {A, B}, {A, C}, {B, C} sont aussi fréquents puisque toute transaction qui contient {A, B, C} contient aussi n'importe lequel de ses sous-ensembles.

Propriété 2 : Tout sur-ensemble d'un ensemble infréquent est infréquent.

Soit A un itemset infréquent, c'est -à -dire que support (A) ≤ minsup.

Supposons qu'il existe À tel que A ⊂ À et support (À) > minsup. On a :

support (À) ≤ support(A) puisque A ⊂ À, ce qui implique

support (A) ≥ support (À)> minsup et donc support (A)> minsup et A est un itemset fréquent, ce qui contredit l'hypothèse.

C'est en partant des notations et des propriétés citées ci-dessus que l'algorithme Apriori a été fondé. Il fonctionne en trois phases :

- calcul des fréquences des ensembles d'un seul élément (les 1-itemsets) en éliminant les ensembles dont la fréquence n'atteint pas le support minimum ;
- on génère ensuite les ensembles candidats de deux éléments (les 2-itemsets) par jointure de l'ensemble des éléments fréquents obtenu à l'issue de la phase précédente ;
- on répète ce processus afin d'obtenir tous les ensembles fréquents.

Alors la génération des (k+1)-ensembles fréquents ((k+1)-itemsets fréquents) nécessite l'utilisation des k-ensembles fréquents, en prenant deux k-ensembles différents d'un élément et en les unissant pour composer un (k+1)-ensemble fréquent ; et ainsi de suite jusqu'à ce qu'il n'y ait plus d'itemsets fréquents de cardinalité supérieure ; ce qui produit k passes sur la base s'il existe des ensembles fréquents de taille k.

C) L'algorithme Apriori

Après l'explication du principe de l'algorithme Apriori, nous montrons dans la figure suivante son pseudo code :

```
L1= {1-itemset fréquent};
Pour (k=2; Lk-1≠∅ ; k++) faire
   Ck= Apriori- gen (Lk-1); // la génération de nouveaux candidats
   Pour tout transaction t ∈ D faire
      Ct= Subset (Ck , t) ; //analyse des candidats contenant t
      Pour tout c∈ Ck faire
         c.count++ ;
   Fin
   Lk={c∈ Ck | c.count ≥ minsup};
Fin
Answer = ∪k Lk  // ensemble de tout les itemsets fréquents
```

Figure II.1 : L'algorithme Apriori [AGR 94].

D) Explication de l'algorithme

Dès la première passe, l'algorithme commence par construire l'ensemble L_1 des itemsets de taille 1(ensemble des items) fréquents. Puis, afin d'obtenir l'ensemble complet des itemsets fréquents, il faut faire plusieurs passes sur les transactions.

Pour la K^{ieme} passe, il faut utiliser les itemsets fréquents L_{k-1} trouvés à la $(k-1)^{ieme}$ passe, pour servir à générer les itemsets C_k en utilisant la fonction **Apriori-gen** que nous décrirons ultérieurement.

En utilisant l'ensemble des nouvelles combinaisons générés pour calculer leurs supports à l'aide de la fonction **Subset (C_k , t)** (qui permet de déterminer parmi les candidats contenus dans C_k ceux qui sont aussi contenus dans une transaction t), et enfin éliminer ceux qui n'ont pas le support supérieur à minsup.

E) Génération des candidats dans Apriori

La fonction **Apriori-gen** permet de fournir les nouveaux candidats, et utilise pour construire les candidats de taille k comme paramètre les itemsets fréquents de taille k-1 ordonnés dans l'ordre lexicographique. Ainsi, on applique la propriété des itemsets qui veut que tout sous-ensemble d'un itemset fréquent est fréquent et on élimine des candidats tout sur-ensemble d'items infréquents. Le pseudo code de la cette fonction est représenté dans la **figure II.2**.

```
Ck ←∅ ;
Pour tout ensemble I1 d'items de Lk_1
   Pour tout ensemble I2 ≠ I1
      Si I2 et I1 ont un préfixe commun de taille k -2 et I1 [k -1] < I2 [k -1] alors
         Ck ←Ck ∪ (I1 ∪ I2 [k -1])
   Fin
Fin
Pour tout itemset c ∈ Ck
   Pour tout sous-ensemble de taille k -1 de c
      Si (s ∉ Lk_1) alors
         Supprimer c de Ck
   Fin
Fin
```

Figure II.2 : La fonction Apriori-gen (L_{k-1}) [MAH 00]

La génération des candidats par la fonction Apriori-gen est faite en deux étapes essentielles. La première étape est celle de jointure de tous les ensembles de L_{k-1} entre eux, pour donner comme résultat C^k_n itemsets possibles. La deuxième étape d'épuration permet de supprimer parmi les candidats générés ceux qui contiennent un sous ensemble de taille k-1 qui n'appartient pas à L_{k-1}.

Pour mieux comprendre cette fonction, nous employons cet exemple illustratif :

Exemple

Soit L_3= {{abc}, {abd}, {acd}, {ace}, {bcd}}

L'étape de jointure :

- {abcd} à partir de {abc} et {abd} ;
- {acde} à partir de {acd} et {ace}.

Alors on aura C_4= {{abcd}, {acde}}.

L'étape d'épuration :

- {acde} est supprimé car {ade} n'apparaît pas dans L_3.

Donc on n'aura à la fin que l'itemset {abcd} comme un itemset fréquent.

5.2 Exemple d'illustration

Pour mieux comprendre ces concepts qui peuvent paraître un peu flous, nous avons choisi de prendre l'exemple des tickets de caisse émis par un supermarché. La base d'analyse se compose de l'ensemble des transactions réalisées sur une période donnée.

Chaque transaction est représentée par un ticket de caisse, qui comprend un ensemble d'articles. Elle est considérée aussi comme un enregistrement à part entière de la base de données, avec le détail des articles ou des familles des articles [LEF 98].

Dans cet exemple, nous illustrons la démarche d'extraction des règles de décision afin de réaliser une campagne promotionnelle personnalisée avec l'édition de bons de réduction en fonction des achats : si on note la présence du café X dans la transaction, on édite un bon de réduction pour le sucre Z (à condition de trouver au préalable une règle X→Z avec un haut niveau de support et un haut niveau de confiance) [LEF 98].

Soit notre base de données citée dans le tableau II.4 (c'est une simple table pour des besoins de compréhension).

Ticket 1	Ticket2	Ticket3	Ticket4
farine	œuf	farine	œuf
sucre	sucre	œuf	chocolat
lait	chocolat	sucre	thé
		chocolat	

Tableau II.4 : La base de données de l'exemple.

Comme nous l'avons déjà expliqué, l'extraction des règles d'associations se fait en deux étapes distinctes. La première consiste à rechercher les itemsets fréquents en isolant les articles présentant un support supérieur au minimum support (minsup), puis combiner les articles (les items ou les 1-itemsets) les plus représentés pour générer les itemsets fréquents. La deuxième étape permet l'extraction des règles de décision à partir de cet ensemble des itemsets.

En appliquant ces deux étapes, nous obtenons les résultats présentés dans les tableaux suivants :

Étape 1 : la recherche des itemsets fréquents

Cette étape consiste à compter le nombre de fois où figure chaque article (item) (**Tableau II.6**), puis épurer l'ensemble trouvé afin de ne laisser que les itemsets fréquents en utilisant comme minimum support minsup=2 ,et comme minimum de confiance minconf=2 .

C_{1chap}

TID	Ensemble des itemsets (1-itemsets)
Ticket 1	{farine}, {sucre}, {lait}
Ticket 2	{œuf}, {sucre}, {chocolat}
Ticket 3	{farine}, {œuf}, {sucre}, {chocolat}
Ticket 4	{œuf}, {chocolat}, {thé}

→

Tableau II. 5: Les1- itemsets générés C_{1chap}.

C_1

1-itemset	support
{farine}	2
{sucre}	3
{lait}	1
{œuf}	3
{chocolat}	3
{thé}	1

→

L_1

1-itemset fréquent	support
{farine}	2
{sucre}	3
{œuf}	3
{chocolat}	3

Tableau II.6 : Les 1-itemsets générés et leurs supports.

Tableau II.7 : Les 1-itemsets fréquents et leurs supports.

Apres la première passe sur notre base de données pour produire les 1-itemsets fréquents (**Tableau II.7**), qui vont être utilisés dans la prochaine passe pour la génération d'autres itemsets.

C_{2chap}

TID	Ensemble des itemsets (2-itemset)
Ticket 1	{farine, sucre}
Ticket 2	{sucre, œuf}, {sucre, chocolat}, {œuf, chocolat}
Ticket 3	{farine, sucre}, {farine, œuf}, {farine, chocolat}, {sucre, œuf}, {sucre, chocolat}, {œuf, chocolat}
Ticket 4	{œuf, chocolat}

→

Tableau II. 8: Les 2-itemsets générés C_{2chap}.

C_2

2-itemset	support
{farine, sucre}	2
{farine, œuf }	1
{farine, chocolat }	1
{sucre, œuf }	2
{sucre, chocolat }	2
{œuf, chocolat }	3

→

L_2

2-itemset fréquent	Support
{farine, sucre}	2
{sucre, œuf }	2
{sucre, chocolat }	2
{œuf, chocolat }	3

Tableau II.9 : Les 2-itemsets générés et leurs supports C_2 .

Tableau II.10 : Les 2-itemsets fréquents et leurs supports L_2.

Dans la deuxième passe, toutes les combinaisons entre les différents 1-itemsets fréquents sont établies. Les 2-itemsets générés sont associés et les associer aux

transactions qui leur correspondent, d'où l'ensemble C_{2chap} (**Tableau II.8**). On compte alors leurs supports pour établir l'ensemble C_2 (**Tableau II.9**), et enfin épurer ce dernier pour obtenir l'ensemble des 2-itemsets fréquents L_2 (**Tableau II.10**), qui va servir dans la troisième passe.

C_{3chap}

TID	Ensemble des itemsets (3-itemset)
Ticket 2	{sucre, œuf, chocolat }
Ticket 3	{sucre, œuf, chocolat }

Tableau II. 11: Les 3-itemsets générés C_{3chap}.

C_3

3-itemset	support
{sucre, oeuf, chocolat}	2

Tableau II.12 : Les 3-itemsets générés et leurs supports C_3.

L_3

3-itemset	support
{sucre, oeuf, chocolat}	2

Tableau II.13 : Les 3-itemsets fréquents et leurs supports L_3.

Dans la troisième passe, nous n'obtenons qu'un seul itemset fréquent, ce qui indique l'arrêt de l'algorithme Apriori.

Remarque

L'utilisation de l'ensemble C_{kchap}, qui sert à calculer les supports se réduit considérablement au cours du processus , et permet de gagner beaucoup de temps de calcul dans l'algorithme *Apriori TID*, contrairement à l'algorithme *Apriori* qui utilise à chaque étape toute la base de données initiale pour calculer les nouveaux supports.

Étape 1 : Extraction des règles associatives

Une fois que les itemsets fréquents sont trouvés, la deuxième et dernière étape de l'algorithme d'extraction des règles d'association à partir des itemsets fréquents est appliquée comme suit :

L'itemset fréquent trouvé	Les implications possibles	Le support	La confiance	La validité
sucre, œuf, chocolat	sucre→ œuf sucre → chocolat œuf → chocolat sucre→œuf _chocolat ..... etc.	2 / 4 (50%) 2 / 4 (50%) 2 / 4 (50%) .... etc.	2 / 3 (66%) 2 / 3 (66%) 2 / 3 (66%) .... etc.	Oui Oui Oui .... etc.
farine, sucre	farine → sucre sucre→ farine	2 / 3 (66%) 2 / 3 (66%)	2 / 2 (100%) 2 / 3 (66 %)	Oui Oui

Tableau II.14 : Les règles obtenues et leurs support et confiance.

Remarques

❶ En générale les algorithmes d'extraction des règles d'association ne sont au fond que des algorithmes de comptages, avec détermination de probabilités en prenant des

ratios entre les différents nombres d'occurrences des différents items existant dans une base de données [FAY 96].

❷ L'algorithme Apriori et ses dérivés sont des algorithmes qui travaillent en largeur : ils comptent tous les 1-itemsets fréquents, puis les 2-itemsets fréquents, etc. [GAR 00].

❸ Lors de l'utilisation d'un algorithme d'extraction de règles associatives, il est très important de prendre en considération le nombre des items contenus dans la base de données et essayer de le réduire au maximum, car ceci peut influencer complètement la performance de l'algorithme utilisé (plus le nombre d'attributs est grand plus on a besoin d'espace mémoire pour contenir toutes les combinaisons) [FAY 96].

5.3 Le rapport d'Apriori avec les autres algorithmes [AGR 94]

Pour présenter le rapport d'Apriori avec les autres algorithmes (AIS et SETM), il faut avoir tout d'abord une vue globale sur ces deux algorithmes.

Les algorithmes AIS [AGR 93] et SETM [HOU 93] suivent la même procédure d'extraction de règles associatives que l'algorithme Apriori et ses dérivés, mais diffèrent essentiellement dans la manière de calcul des itemsets candidats.

Dans les algorithmes AIS et SETM, les itemsets candidats sont générés pendant le balayage de la base de données. Après la lecture de chaque transaction, on détermine quels sont parmi les itemsets fréquents trouvés dans la passe précédente, ceux qui sont présents dans la transaction, puis les nouveaux itemsets candidats sont générés en combinant ces itemsets fréquents avec les autres items de la transaction. Alors cette approche génère des itemsets trop petits et pas très intéressants.

Par ailleurs, l'algorithme Apriori et ses dérivés génèrent les itemsets candidats en utilisant uniquement les itemsets fréquents trouvés préalablement dans la passe précédente sans considérer les transactions de la base de données [FAY 96].

En plus, comme nous l'avons dit l'algorithme Apriori se base sur une propriété élémentaire concernant la fréquence d'un ensemble et de ses sous ensembles.

Avec ces deux principes, cette procédure est d'autant plus intéressantes parce qu'elle génère un nombre réduit d'itemsets candidats ce qui réduit donc considérablement l'espace mémoire requis.

Pour illustrer le choix d'utilisation de l'algorithme Apriori pour la découverte des itemsets fréquents dans l'algorithme FIHC, nous allons étudier l'évaluation de performance des différents algorithmes qui ont été déroulés sur des bases de données contenant différentes tailles et différents nombres de transactions qui sont affichés dans le tableau ci-dessous avec :

| T | : la taille moyenne des transactions.

| I | : la taille moyenne de l'itemsets.

| D | : la taille de base de données (ou le nombre de transactions).

Nom	| T |	| I |	| D |
T5.I2.D100K	5	2	100K
T10.I4.D100K	10	4	100K
T20.I4.D100K	20	4	100K
T20.I6.D100K	20	6	100K

Tableau II.15 : Les différents paramètres de bases de données utilisées [AGR 94].

D'après l'évaluation de performances de l'algorithme Apriori et les deux autres algorithmes présentés auparavant (AIS et SETM) faite par Agrawal, dont elle est représentée dans [AGR 94], nous avons présenté un tableau (tableau II.15) qui représente quelques bases de données utilisées dans cette évaluation. En choisissant la base T10.I4.D100K comme un échantillon de résultats obtenus dans cette évaluation pour tester deux algorithmes d'extraction des règles associatives (des algorithmes de découverte des itemsets frequents) : Apriori et SETM. Les résultats de tests sont affichés dans le tableau suivant :

Algorithme	Support minimum fixé (%)				
	2.0	1.5	1.0	0.75	0.5
SETM	41	91	659	929	1639
Apriori	3.8	4.8	11.2	17.4	19.3

Tableau II.16:Temps d'exécution en secondes de la base de données T10.I4.D100K [AGR 94].

En étudiant ce tableau , nous constatons qu'il y a une nette différence entre le temps d'exécution (élevé) nécessité par l'algorithme SETM et celui nécessité par l'algorithme Apriori.

Enfin, nous remarquons d'après tous les expériences et tous les résultats de tests représentés dans [AGR 94] que :

Quelle que soit la taille de la base de données analysée, quel que soit le nombre de variables qu'elle contient et le support minimum fixé, nous remarquons une nette différence entre le temps d'exécution des algorithmes Apriori et les autres algorithmes.

6. Conclusion

La recherche des règles d'association a donné naissance à de nombreux algorithmes, qui sont opérationnels pour des bases de millions de transactions dans plusieurs applications commerciales. Ces applications peuvent être analysées en utilisant un algorithme de recherche d'association. La grande distribution, l'assurance et le secteur bancaire sont des terrains potentiels d'application de ces algorithmes, mais ils sont utilisés aussi pour l'extraction des itemsets fréquents (qui est une étape préliminaire dans la procédure d'extraction des règles associatives) utilisé dans les différentes méthodes de clustering (regroupement) des documents et parmi eux l'algorithme FIHC.

La nette performance de l'algorithme Apriori illustrée dans la section précédente nous a permit de juger son utilisation comme une étape essentielle pour l'extraction des itemsets fréquents dans l'algorithme FIHC pour regrouper les documents (clustering), que nous allons détailler ces différentes méthodes et algorithmes dans le chapitre III, puis étudier les différentes phases de FIHC dans le chapitre IV.

Chapitre III

Le Clustering

Chapitre III

Le Clustering

1. Introduction

Les évolutions technologiques ainsi que le développement explosif de divers types de données et d'informations (des documents, des rapports, des page HTML ...etc) ont un impact certain sur l'entreprise et ont amenés celle-ci à des positions concurrentielles de moins en moins stables. Pour faire face à cet environnement hyper concurrentiel, les entreprises ont besoin d'organiser l'ensemble de documents dans une structure logique afin de minimiser le coût et le temps, d'où la naissance de « clustering » qui est considérer comme un outil de data mining ou un de ses problèmes [KIN 01].

Dans ce qui suit, nous définirons ce qu'est le " clustering " avec ses différents domaines d'application, par la suite nous présentons les méthodes les plus utilisées avec une étude comparative.

2. Le clustering

2.1 Définition de « clustering » [FUN 03]

Le clustering ou le regroupement est un processus qui rassemble un ensemble d'objets (données, informations, document ...) sous un ensemble de classes ou groupes appelés "clusters". Un cluster est une collections d'objets qui ont une similitude élevée (pour plus de détail voir l'annexe C) par rapport aux autres qui n'appartiennent pas au même ensemble ; ces objets sont basés sur leur valeurs d'attribut et sont traités collectivement sous un groupe . En d'autres termes, le regroupement est basé sur les principes de maximiser la similitude d'intra-clusters et de minimiser la similitude d'inter-clusters.

Le regroupement des informations et connaissances est une activité humaine ancienne importante ; donnons l'exemple d'un enfant qui apprend à distinguer les animaux, (les oiseaux, les poissons ...etc) et les plantes ; il les regroupe plusieurs fois, en s'améliorant à chacune en utilisant ses connaissances.

Le clustering est une classification automatique spéciale, autrement dit une classification non supervisée, c'est-à-dire il n'y a pas de classes prédéfinies. Pour pouvoir classer un document, il est nécessaire de comparer le contenu de ce document avec la description de chaque classe, et de déterminer la ou les classes les plus semblables ; la première tâche dans la classification est donc de créer une description pour chaque classe, cependant, dans beaucoup de cas en pratique une telle description n'est pas toujours disponible.

Le clustering est applicable dans des domaines varier, nous citons parmi eux :

- le commerce : il offre une véritable aide à découvrir les multiples types de clients et de les caractériser selon le modèle d'achat, l'importance, les régions ...etc ;
- la médecine : on trouve le clustering des maladies, de symptômes et des traitements qui mènent souvent aux taxinomies utiles ;

3. Les différents types et méthodes de clustering [CLE 04]

Comme le clustering est le regroupement des documents similaires, plusieurs méthodes et approches sont utilisées afin d'améliorer l'exactitude et la qualité des groupes, car une bonne méthode de regroupement doit garantir le mieux :

- une grande similarité intra-cluster,
- une faible similarité inter-cluster.

On peut catégoriser le clustering selon le type de résultat :

- le " hard clustering " qui est une construction disjointe des groupes où chaque document appartient a un et un seul cluster ;
- le " fuzzy clustering " dont la caractéristique principale est qu'un document participe dans chaque cluster avec un certain seuil (support) d'appartenance ;
- le " soft clustering " appelé aussi " overlapping ", dans lequel un document appartient à un ou plusieurs (pas tous les clusters) avec certaine valeur (degré) d'appartenance.

Le clustering est catégorisé aussi selon les méthodes :

- les méthodes de partitionnement dans lesquelles les algorithmes utilisés sont basés sur la construction de plusieurs partitions, puis leurs évaluations selon certains critères ;
- les méthodes hiérarchiques dans les quelles les algorithmes permettent de créer une décomposition hiérarchique des clusters selon certains critères ; les documents les plus similaires sont regroupés dans des clusters aux plus bas niveaux , tandis que les documents moins similaires sont regroupés dans des clusters aux plus haut niveaux. Parallèlement on y trouve des méthodes non hiérarchiques dont la structure des clusters n'est pas sous forme d'un arbre ;
- les méthode basées sur la densité dans lesquelles les algorithmes sont fondés sur des notions de connectivité et de densité, car les clusters sont représentés par des régions denses dans un espace de données séparées par des régions de densité inférieure ; l'idée générale est d'attribuer pour chaque donnée un point dans un cluster, par conséquent, le voisinage d'un rayon doit contenir au moins un nombre minimum de points de données. Ces méthodes sont bonnes pour le filtrage et parmi les algorithmes les plus connus, nous citerons " DBCAN ", " système optique " ... [FUN 03]
- les méthodes de grille sont basées sur une structure multi- niveaux de granularité. Elles permettent de quantifier l'espace par un nombre fini de cellules qui forme une structure de grille dans la quelle toutes les opérations de clustering s'effectuent ; la complexité [21] de ces méthodes est au moins linéairement proportionnelle au nombre d'objets, mais ce qui les caractérise est leur complexité indépendante du nombre d'objets de données, car elles ne dépend que du nombre de cellules dans chaque dimension dans l'espace quantifier [FUN 03].
- les méthodes basées sur fréquent " itemset " dans lesquelles les regroupements sont construits à la base d'un ensemble de mot clés fréquents dans l'ensemble de

[21] : La complexité c'est la mesure de temps de réponse de l'algorithme, o (n) le temps d'exécution de l'algorithme, n étant un nombre très lié a la taille de données.

documents textuels .Les méthodes hiérarchiques et de partitionnement sont celles qui ne peuvent pas s'appliquer vraiment au problème de la haute dimensionnalité dans les groupes de documents. Les méthodes de clustering basées sur les itemsets fréquents sont montrées comme des approches prometteuses pour la dimensionnalité de clustering élevée en littérature récente [FUN 03].Ces méthodes réduisent la dimension d'un espace de vecteur en employant seulement les "itemsets "fréquents pour grouper, car l'extraction fréquente d'itemset est une étape préliminaire dans leurs processus .Parmi les plus connus nous citerons " FIHC ", " HFTC ".

Les méthodes de partitionnement, méthodes hiérarchiques et méthodes basées sur fréquent " itemset " sont les plus utilisées dans le monde de clustering, elles sont considérées comme des méthodes de base pour d'autre nouvelles méthodes. Ces méthodes de base sont améliorées et reformulées selon les besoins, afin de donner des méthodes plus efficaces qui donnent des résultats beaucoup exacts. Nous citons les plus connus : la méthode " K-means " qui est une méthode de partition non hiérarchique (ne construit pas une hiérarchie de clusters), la méthode FIHC est une méthode basée sur fréquent "itemset " hiérarchique ... etc.

4. Comparaison des différentes méthodes selon les types de résultats (« hard, « soft » », « fuzzy »)

Selon Cardit et Wagstaff le "hard clustering" est basé sur l'idée que chaque document appartient à un seul cluster, c'est-à-dire que chaque cluster a ses propres membres .Les algorithmes de ce type sont simples et rapides, mais ils ont également quelques points de faiblesse :

- les algorithmes dépendent de la valeur des seuils (c'est des paramètres d'entrée) dans la construction des clusters ;
- les algorithmes dépendent de l'ordre de comparaison entre les différents clusters (calcul de similitude), particulièrement quand le seuil (support) prend une valeur très élevée dans le fusionnement ;
- les algorithmes ne sont pas efficaces lorsqu'un document comporte plusieurs thèmes, puisque il est associé a un seul cluster. La qualité de clustering baisse.

Prenant en compte les faiblesses cités ci-dessus, le "soft clustering" est apparu afin d'améliorer la qualité de clustering, car il permet d'assigner chaque document aux clusters multiples d'après les thèmes trouvés ; ceci est habituellement réalisé en employant les fonctions d'adhésion ("membership function") pour chaque document D_j et cluster C_i, cette fonction calcule M_{ij} qui indique le poids du document D_j dans le cluster C_i.

Le "soft clustering " est basé sur les approches itératives appliquant le modèle probabiliste ; il permet d'assumer au commencement l'attribution aléatoire des documents aux clusters, et ensuite, de prendre un premier modèle probabiliste en estimant les paramètres de ce dernier et de réitérer suivant deux étapes (EM) jusqu'à la convergence [CLE 04].

Ces deux étapes sont :

- l'espérance (e-étape) : calculer $P(C_i \mid D_j)$ pour chaque document donné dans le modèle courant, et re-marquer la probabilité des documents basés sur ces évaluations postérieurement ;

- la maximisation (m-étape) : ré estimer les paramètres de modèle (modèles «θ»), pour les probabilités remarquées.

On arrête ces opérations quand le clustering est bon.

Un algorithme " SImilarity-based Soft Clustering " (SISC) [KIN 01] est un efficace soft clustering algorithme basé sur la mesure de similarité.

Comme le " hard clustering ", le " soft " clustering présente des faiblesses :

- il n'est pas efficace quand on a une grande collection de données ;
- il recalcule les paramètres à chaque fois qui est une perte de temps.

Le dernier type est le fuzzy clustering .Les documents peuvent y être attribués simultanément à tous les clusters. Ainsi des rapports entre différents domaines peuvent être découverts, ce qui n'existe pas dans le " hard clustering ". Il est basé aussi sur les fonctions d'adhésion (" membership function "), c'est le " fyzzy " poids, qui est employé pour trouver la pertinence des documents et les relations entre eux. Les vecteurs de " fuzzy membership " sont trouvés par un procédé itératif de sorte qu'un vecteur d'un document puisse être rapproché linéairement des vecteurs de ses voisins.

Le fuzzy clustering n'est pas largement utilisé à cause de sa difficulté d'implémentation, l'algorithme le plus connu dans ce type est " Fuzzy Map Clustering " (FMC) [PAD 04].

5. Etude des différentes méthodes

5.1 Les méthodes hiérarchiques et non hiérarchiques [UTE XX] [ARG XX]

Les méthodes hiérarchiques comme leur nom l'indique consistent à construire des hiérarchies, afin d'obtenir une meilleure représentation des documents, à la différence des méthodes non hiérarchiques basées sur une structure plate où tous les clusters se situent dans le même niveau.

Nous allons montrer les aspects important et points remarquables de chaque catégorie. Commençons par les méthodes non hiérarchiques.

Les méthodes non hiérarchiques sont largement utilisées a cause de leur efficacité, car elles sont préférables si l'efficacité est importante, ou si on a un bon nombre de documents à traiter. La caractéristique principale des algorithmes appliqués est que le nombre de clusters désirés *k* comme un paramètre d'entrée doit être fourni.

Le processus de fonctionnement est illustrer comme suit :

- choisir aléatoirement les documents *de k clusters* comme *un centre* de regroupement un par cluster ;
- former les clusters initiaux basés sur ces centres ;
- réitérer, en modifiant à plusieurs reprises la réapparition des documents aux différents clusters pour améliorer le clustering global ;
- arrêter quand le regroupement converge ou après un nombre fixe d'itérations.

La méthode la plus connue dans cette catégorie est " **k-means** " ; c'est une méthode très utilisée à cause de sa souplesse et efficacité. Nous l'exposons plus loin.

Les méthodes hiérarchiques sont souvent retenues comme de meilleures approches de qualité, mais elles sont limitées en raison de leur équation quadratique de la complexité en temps (au moins O(n 2)) [ORL 04]. Ces méthodes sont préférables pour une analyse des données détaillée (bien structurée), et ont comme avantage qu'on a plus d'informations par rapport aux méthodes non hiérarchiques. De plus dans cette catégorie aucun algorithme n'est clairement préféré ; ces méthodes sont fondées sur l'idée qu'un niveau intermédiaire soit être regardé en tant que combinaison de deux clusters du prochain niveau plus bas soit peut dupliquer un cluster du prochain niveau plus élevé . Le résultat d'un algorithme hiérarchique peut être graphiquement montré comme arbre appelé " the dendogram ", cet arbre binaire montre graphiquement le fusionnement du processus et les clusters intermédiaires.

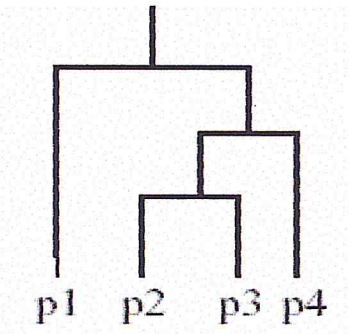


Figure III .2 : Le " dendogram " [STE 00].

Ces méthodes sont caractérisées par :

- une fonction de similitude pour déterminer la similitude de deux documents ;
- le regroupement des deux clusters les plus semblables jusqu'à ce qu'il y ait seulement un cluster ;
- la grande contenance du premier cluster.

Il y a deux méthodes de base hiérarchiques :

a) la méthode **agglomérative :** cette méthode commence par un cluster à chaque document ; l'étape suivante est le fusionnement de la paire la plus semblable ou la plus étroite pour fournir un cluster de niveau plus haut. Là, il est exigé la définition d'une fonction de similitude ou de distance entre les clusters.

b) la méthode **divisive :** cette méthode commence par un seul cluster et à chaque étape, on décompose un cluster en deux clusters jusqu'à arrivée aux clusters feuilles.

Les critères de fusion-éclatement [ORL 04]

Pour les méthodes hiérarchique, lorsqu'on a deux clusters C_1 et C_2.
C_1, C_2 sont fusionnés (éclaté) si il existe :

- $O_1 \in C_1$ et $O_2 \in C_2$, dist(O_1,O_2) <= seuil , ou
- $\nexists$ $O_1 \in C_1$ et $O_2 \in C_2$ tels que dist (O_1, O_2) >= seuil, ou
- dist (C_1 et C_2) < = seuil, avec

$$dist\ (C_1, C_2) = \frac{1}{n1 * n2} \sum_{o1 \in C1, o2 \in C2} dist\ (o1, o2)$$

Avec n_1=|C_1| et n_2=|C_2|.

- la biologie : il est employé dans le regroupement des gènes par catégorie pour découvrir les fonctionnalités semblables, afin de dériver les taxonomies végétales et animales ;
- la recherche : le clustering est utilisé avec le data mining et l'intelligence artificielle dans différent domaines tels que l'analyse de données textuelles (ADT) pour la description et l'exploration des connaissances.
- les moteurs de recherche : plusieurs méthodes de clustering sont applicables dans les moteurs de recherche afin de faciliter la recherche et de minimiser le temps d'exécution.

2.2 Le clustering des documents

Le clustering des documents est un processus automatique et un outil important dans différentes applications, plus précisément les moteurs de recherche. Il est utilisé pour des grandes base de données afin d'organiser les résultats de la recherche Web dans des groupes significatifs. Ces groupes permettent d'organiser les document selon certain critère ou formules pour donner à l'utilisateur une bonne vue globale sur les informations contenues dans ces documents de cluster.

Les techniques de clustering de document sont utilisés dans le champ de la recherche documentaire (" information retrieval ") et « text mining ».

Dans la recherche documentaire (RD), il est employé pour l'évaluation des similarités entre documents, car les documents similaires sont regroupés ensemble. Dans le cas ou une requête identifie un document pertinent par rapport à des besoins d'information exprimés par les utilisateurs, alors tous l'ensemble de documents est pertinents .Dans le « text mining » (TM) il est exploité pour produire des représentations synthétiques de vastes collections de documents, dans le cadre de procédures d'extraction d'information à partir de données textuelles (voir le processus de text mining dans le chapitre I) .Plus récemment le clustering a été proposé pour l'usage en recherche d'une collection de documents où en organisant le résultat retourné par un système "search engine" en réponse à une question d'utilisateur [STE 00].

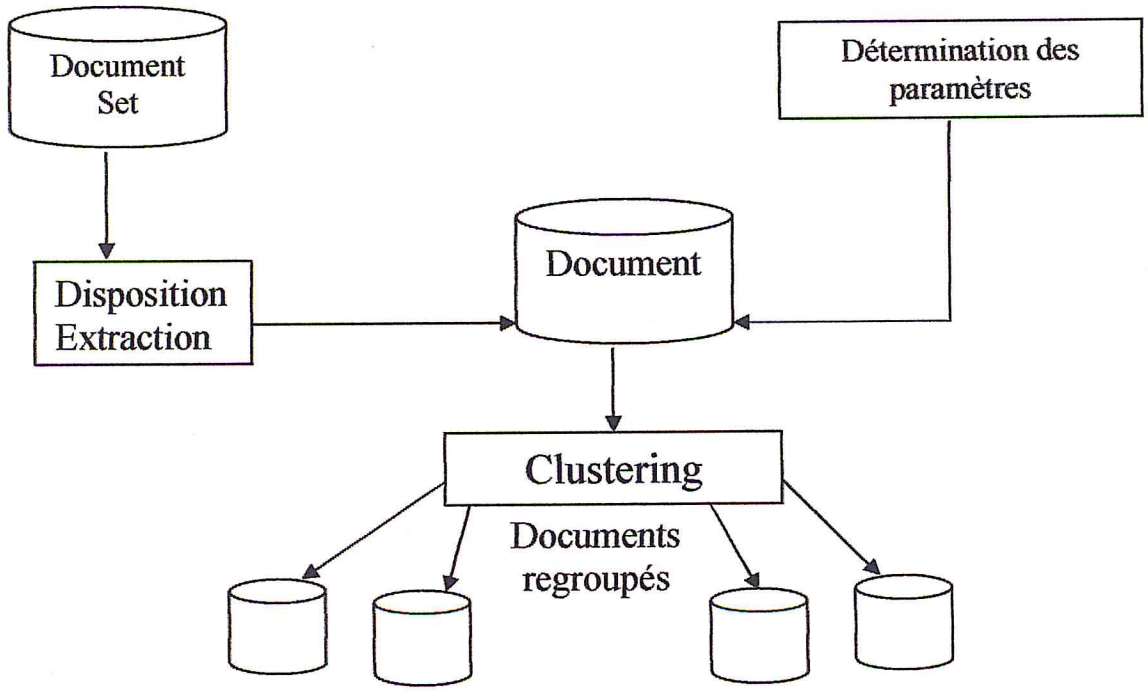


Figure III.1: Le clustering des documents [SUN 00].

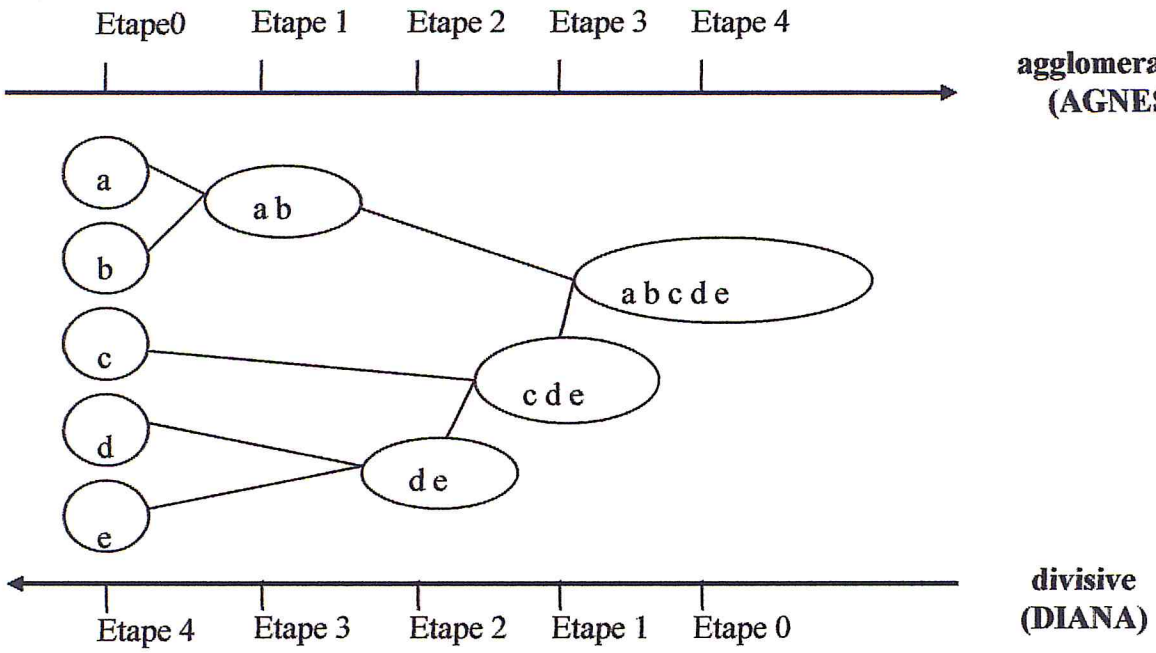


Figure III.3 : Les méthodes hiérarchiques [ORL 04].

5.2 La méthode « agglomérative » et la méthode « k-means »

Nous avons mentionné précédemment que les méthodes de partition, non hiérarchique sont très utilisées, spécialement k- means (MacQueen'67'). La méthode agglomerative est une méthode hiérarchique de type hard clustering ; de même k-means mais cette dernière est améliorée afin d'être utilisés aussi dans le soft clustering [ORL 04] .La méthode k- means est modifiée à chaque fois de nouveaux besoins sont apparus. En résultat, plusieurs dérivés sont apparus tels que " *k-medoids* " ou " PAM " (Partition around medoids), " bisecting K-means algorithm ", " basic K-means algorithm "...

Le résultat d'une étude indique que la méthode agglomérative est souvent meilleur que K-means, bien que plus lente [STE 00] ; d'autres études soumettent à la contrainte que ces résultats soit obtenus avec des données non-document ,mais ce qui est en accord d'après les différentes expériences est que K-means est employée en raison de sa efficacité, tandis que la méthode hiérarchique agglomérative est employée en raison de sa qualité [STE 00].

Pour mieux comprendre nous représentons d'une manière brève les deux méthodes.

5.2.1 L'algorithme de la méthode agglomerative [CSE 99]

L'algorithme agglomeratif hiérarchique (AHC) a l'avantage de fournir une hiérarchie, car on a une bonne organisation des documents, les étapes de la méthode sont les suivantes :

a) appliquer une méthode de distance entre les clusters, car si on a N points, on a besoin de plusieurs mesures pour créer la table de distance (après la construction des clusters initiaux) ;

b) créer la matrice de similitude ;

c) trouver la plus petite valeur dans la matrice de similitude, ensuite regrouper les clusters les plus similaires dans un seul cluster, puis recalculer la distance entre les nouveaux clusters avec les autres ;

d) la distance entre les clusters est obtenue par le calcul successif de la matrice de similitude.

Dans la méthode agglomerative on a trois variantes de calcule du distance entre deux clusters :

- lien simple (" Single Link "): la distance entre deux clusters est indiquée par la distance entre leurs membres les plus étroits. Cette variante produit un cluster avec la propriété que chaque membre de ce dernier est lié moins étroitement à son voisin qu'à n'importe quel point en dehors de son cluster ;
- lien complet (" Complete Link "): la distance entre deux clusters est indiquée par la distance entre leurs membres les plus éloignés ;
- moyenne de groupe (" Group Average ") : la distance entre deux clusters est mesurée entre les centres de surface de chacun.

La figure ci- dessous montre les deux premières variantes.

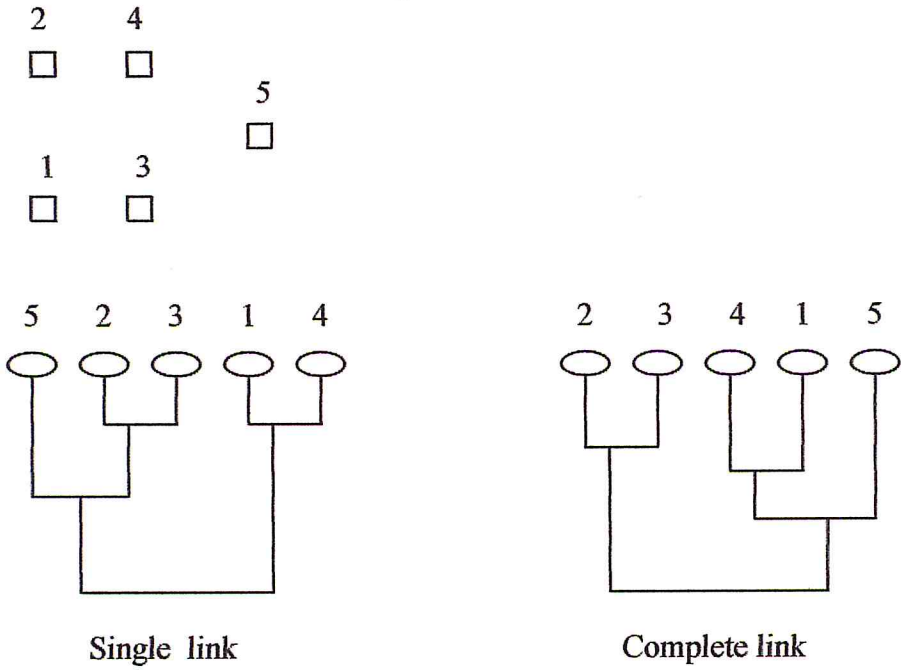


Figure III. 4 : Les distances dans l'agglomerative [CSE 99].

La méthode agglomerative (AHC) est hiérarchique ; son algorithme est caractérisé par la complexité quadratique, car dans la première itération, toutes les méthodes de (AHC) doivent calculer la distance de toutes les paires de ***n*** documents ; la complexité est d'O(n^2), dans chacune des itérations de fusionnement *de n* - 2 suivants ; elle doit calculer la distance entre les clusters les plus récemment créés et tous autres clusters existants , afin de maintenir une exécution globale d'O(n^2) ; le calcul de la distance entre eux doit être faite un dans un temps constant.

5.2.2 L'algorithme de la méthode k-means [ORL 04] [DEL 02]

La méthode k-means est une méthode de partition non hiérarchique ; elle est relativement efficace à cause de sa complexité $O(t\ k\ n)$: n est un ensemble d'objets (document), k est un ensemble de clusters, et t est l'ensemble d 'itérations, on a normalement $k, t << n$.

On suppose que les documents sont des vecteurs à valeurs réelles ; les clusters sont basés sur les centres de surface ou le centre de gravité ; l'attribution des documents aux clusters est basée sur la distance aux centres de surface courants de chaque cluster.

L'algorithme " *k-means* " est en quatre étapes :

a) choisir k objets formant ainsi k clusters ;

b) (ré) affecter chaque objet O au cluster C_i de centre M_i tel que dist (O, M_i) est minimal ;

c) recalculer M_i de chaque cluster (le barycentre) ;

d) aller à l'étape' b' si on vient de faire une affectation.

Un exemple représentatif

- A= {1, 2, 3, 6, 7, 8, 13, 15, 17}. Créer 3 clusters à partir de A
- On prend 3 objets au hasard. Supposons que c'est 1, 2 et 3. Cela donne C_1= {1} avec M_1=1, C_2= {2} avec M_2=2 et C_3= {3} avec M_3=3.
- Chaque objet O est affecté au cluster au milieu duquel O est le plus proche, alors 6 est affecté à C_3 car dist(M_3,6) < dist(M_2,6) et dist(M_3,6) < dist(M_1,6). On a C_1= {1} avec M_1=1, C_2= {2} avec M_2=2 et C_3= {3, 6, 7, 8, 13, 15, 17} avec M_3=69/7=9.86.
- dist (3, M_2) < dist (3, M_3), 3 passe dans C_2. Tous les autres objets ne bougent pas. C_1= {1}, M_1=1, C_2= {2,3}, M_2=2.5, C_3= {6, 7, 8, 13, 15, 17} et M_3= 66/6=11.
- dist (6, M_2) <dist (6, M_3), 6 passe dans C_2. Tous les autres objets ne bougent pas. C_1= {1}, M1=1, C_2= {2, 3, 6}, M_2=11/3=3.67, C3= {7, 8, 13, 15, 17}, M_3= 12.
- dist (2, M_1) <dist (2, M_2), 2 passe en C1. dist (7, M_2) <dist (7, M_3) 7 passe en C_2. Les autres ne bougent pas. C_1= {1,2}, M_1=1.5, C_2= {3, 6, 7}, M_2=5.34, C_3= {8, 13, 15, 17}, M_3=13.25. Donc C_1= {1, 2, 3}, M_1=2, C_2= {6, 7, 8}, M_2=7, C_3= {13, 15, 17}, M_3=15.
- dist (3, M_1) <dist (3, M_2), 3 passe en 1. dist (8, M_2) <dist (8, M_3) 8 passe en 2
- Plus rien ne bouge

Cet exemple illustre l'application de l'algorithme k-means où on choisit aléatoirement les clusters initiaux.

L'algorithme k-means est très simple, mais il souffre de certaines faiblesses. Le problème le plus pertinent est que l'algorithme a comme paramètre d'entrée le nombre de clusters construits, donc on doit spécifier k (nombre de clusters) à chaque fois, car l'exécution multiple permet de donner des clusters différents relativement à chacune. D'un autre côté k-means n'est pas applicable en présence d'attributs qui ne sont pas du type intervalle, car on a un calcul de moyenne .Une autre faiblesse est que les clusters sont construits par rapport à des objets inexistants (les milieux) ; en plus, l'algorithme k-means ne peut pas découvrir les groupes *non convexes*. En résultat nous pouvons dire que le choix des premiers clusters peut avoir comme conséquence un taux de convergence faible, où la convergence de clustering soit optimale. Comme solution le choix des bons clusters peut être amélioré en utilisant une heuristique ou les résultats d'une autre méthode.

Une méthode de partition produit un clustering plat, mais l'application répétée de la même méthode peut également fournir un clustering hiérarchique. De même, les méthodes hiérarchiques peuvent être employées pour produire une représentation plate de clusters [ORL 04].

La méthode k-means est améliorée et redéfinie afin d'être utilisé dans les différents types de clustering .Une de ces dérivés est " bisecting k-means".

5.2.3 La méthode « bisecting k-means » [STE 00]

La méthode bisecting k-means est une méthode de partition où on utilise une hiérarchie. C'est une méthode hiérarchique divisive, L'expérience a montré qu'elle est meilleure que l'agglomirative en terme d'efficacité et d'exactitude.

L'algorithme comporte les étapes suivantes :

- choisir un cluster pour le décomposer ; il y' a plusieurs manières de sélectionner le cluster à dédoubler, mais l'expérience prouve qu'il n'y a aucune différence significative en termes d'exactitude .Habituellement, on choisit le plus grand regroupement ;
- utiliser l'algorithme " k-means " pour subdiviser le cluster choisi ;
- répéter l'étape 2 pour un nombre constant de fois ;
- répéter les trois étapes ci-dessus jusqu'à ce que le nombre désiré de clusters soit atteint.

La méthode bisecting k- means est une méthode dérivée de k-means. Elle maintient les mêmes étapes dans la construction des clusters mais elle tend à produire des clusters de taille relativement uniforme contrairement a k-means, dont la taille des clusters est largement différente ; on sait que si la taille d'un cluster est petite, ces clusters sont de haute qualité, mais cela ne contribue pas beaucoup à la mesure globale de qualité du "clustering ".

Plusieurs variantes de la méthode "k-means" sont utilisées et améliorées ou dans d'autres cas utilisées avec d'autres méthodes de partition, telles que la méthode "Principal Direction Divisive Partitioning " (PDDP), ou combinées avec d'autres méthodes comme pour l'algorithme "buckshot"(algorithme de chevrotine).

5.2.4 L'algorithme « buckshot » [CUT 92] [HAI 98]

L'algorithme est une combinaison entre "k-means" et "AHC", car il utilise les deux méthodes. La succession des étapes se présente ainsi :

- prendre d'abord aléatoirement un échantillon de documents de taille n, appliquer l'algorithme "Hiérarchical agglomerative Clustering " (HAC) sur cet échantillon qui prend seulement la complexité O (n) du temps ;
- employer les résultats de (HAC) en tant que regroupements initiaux pour l'algorithme "K-means";
- l'algorithme global a une complexité O (n) et permet d'éviter les problèmes de mauvais choix de clusters initiaux.

Cet algorithme résout un problème de "k-means" concernant le choix des regroupements initiaux. En conséquence, une amélioration constante de temps ; de plus l'algorithme devient de plus en plus déterministe.

5.3 La méthode « Hierarchical Frequent Term-based Clustering » (HFTC) [FUN 03]

Elle a été récemment développée et utilise la notion des itemsets fréquents.Cette méthode produit une hiérarchie résultante efficace pour être utilisée dans la recherche. Bien que les deux méthodes HFTC et FIHC soient des méthodes de clustering hiérarchiques basées sur les itemsets fréquents ; elles sont complètement différentes en termes de critère groupant , de leur stratégie pour l'identification des clusters et de leurs résultats hiérarchiques.

HFTC prend à chaque fois le prochain itemset fréquent (représentant le prochain cluster), pour réduire au minimum le chevauchement des documents qui contiennent l'itemset et quelques itemsets restants. Donc le résultat du clustering dépend beaucoup de l'ordre de sélection des itemsets.

5.4 La méthode «Frequent Itemset_based Hierarchical Clustering » (FIHC) [FUN 03]

FIHC est une méthode de regroupement des documents basée sur le principe de repérage des mots fréquemment utilisés dans un document, elle est à l'origine des règles d'association présentée précédemment (chapitre II) ; elle fait partie du " hard clustering"; dans cette méthode un tel ensemble de mots est appelé " frequent itemset ", ces mots fréquents sont utilisés comme des étiquettes pour des ensembles de groupes "clusters" (regroupement). La représentation de la méthode est dans le chapitre IV.

Les dispositifs principaux de cette méthode sont les suivants :

- ***la réduction de dimension*** : pour chaque document, on attribue un vecteur qui maintient la fréquence d'apparition de chaque mot fréquent dans le document, car c'est un facteur principal pour l'efficacité de FIHC.
- ***l'augmentation de l'exactitude*** : l'expérience a montré que FIHC surpasse plusieurs autres méthodes en termes d'exactitude, même lorsque la méthode est appliquée à une grande collection de documents.

- ***le nombre de "clusters" est facultatif*** : la plupart des méthodes utilisées dans le "clustering" exigent d'indiquer le nombre de " clusters " désiré comme paramètre d'entrée dans les algorithmes, tandis que FIHC n'oblige pas l'utilisateur à le donner.
- ***une meilleure stratégie de réduction de l'arbre*** : cette méthode a comme résultat un arbre où les " clusters " sont organisés. FIHC permet de tailler l'arbre dans le cas où on a trop de "clusters". Cette caractéristique augmente l'exactitude en fusionnant les " clusters " qui sont semblables.
- ***la facilité de la recherche*** : la forme hiérarchique des résultats obtenus permet de faciliter la recherche dans le cas où cette méthode est exploitée dans un moteur de recherche.
- ***L'efficacité et la scabilité*** : FIHC complète le processus de regroupement de centaines et même de milliers de documents dans un délai minime, tandis que certains algorithmes ne peuvent produire la même solution qu'après des heures d'exécution. Donc l'expérience a montré que FIHC est sensiblement plus efficace et scalable que ces concurrents.

FIHC est une méthode de clustering hiérarchique complètement différente des autres méthodes hiérarchiques. Elle nous permet de former d'abord tous les clusters en assignant les documents aux clusters et de construire une hiérarchie basée sur leurs similitudes d'inter-cluster. Les clusters dans la hiérarchie résultante sont non recouverts («non-overlapping"), le cluster parent contient seulement les documents généraux.

Dans FIHC nous ne suivons pas un ordre séquentiel pour choisir les clusters, mais plutôt nous assignons les documents aux meilleurs clusters parmi tous les clusters disponibles.

Nous allons présenter cette méthode en détail dans les chapitres suivant.

6. Conclusion

Après cette étude, on constate qu'il n'y a pas une méthode absolument meilleure ou une méthode absolument faible. Chaque méthode est approprié pour un contexte et un type bien défini de clustering , et ce contexte est enrichi par ensemble de critères reliés entre eux tels que la nature des données utilisées , les fonctions de mesure employées, les paramètres d'entrée ... etc.

La qualité d'un regroupement dépend de la mesure de similarité utilisée par la méthode et de son implémentation.

Chapitre IV

Présentation de La méthode FIHC

Chapitre IV

Présentation de La méthode FIHC

1. Introduction

La méthode FIHC est une méthode "Cluster- centred ". Elle mesure la cohésion entre les clusters utilisant les fréquents itemset (voir le chapitre II pour voir plus de détail sur les itemsets). La fraction de l'ensemble des documents donne un tel groupe de mots communs nommés " Global frequent itemset " qui forme la base des clusters initiaux dont chacun est attribué à une matière ("topic") ; la propriété essentielle de ces mots fréquents est le nombre d'occurrences dans tous les documents. Pour illustrer l'importance de cette propriété dans le clustering, nous citons cet exemple :

On considère deux " Global frequent itemset ", "*apple* " et "*window*" ; les documents qui contiennent le mot "*apple* " parlent sur les fruits et les produits fermiers et les documents qui contiennent le mot "*window*" parlent sur la rénovation, autrement dit si les deux mots se trouvent en même temps dans un nombre de documents, on peut donc identifier d'autres matières qui parlent sur les " *les système d'exploitations"* ou *" computers "*, la découverte de ces nouvelles matières (" topic") dans la première étape permettre d'améliorer la solution de clustering.

2. La détermination des « Global frequent itemset » [FUN 03]

Pour générer les "global fréquents itemset "ou les itemsets fréquents globaux nous utilisons l'algorithme « A priori » qui est présenté dans le chapitre II.

Dans cette partie les documents sont traités un par un pour repérer tous les mots en eliminant tous les mots vides et non significatifs (comme, le, la, de, nous, pour, en conséquence, car, etc...), qui sont rassemblés dans un fichier appelé « stop word ». Tous les mots repérés sont rassemblés dans ce qui est appelé *" 1- itemset "*, ils sont considérés comme premiers "global fréquents itemset ". Pour chaque document est sauvegardée la fréquence appropriée de chacun des mots trouvés ; cette opération permet la construction des vecteurs sous le nom de *" feature vector "* regroupés dans une structure appelée *"Document Set "*.

L 'ensemble de ces " global fréquents itemset " constituent la dimension de ces vecteurs ; donc ils ont une dimensionnalité minimum, ils sont utilisés à la place des vecteurs de documents de plus haute dimension ; en conséquence, ils améliorent de manière significative l'efficacité et la scalabilité de toutes les opérations de regroupement suivantes.

Dans le tableau IV.1 un exemple est donné. Un ensemble de douze documents est choisi, dont les noms indiquent leurs classes naturelles.

Après l'application de l'algorithme "a priori" aux vecteurs de document, nous calculons les "global fréquents itemset " qui sont : "écoulement", "forme", "couche",

Le support de 1-itemset « patient » est 42 % parce que la moitié des documents dans l'ensemble contiennent le mot ou l'item « patient ».

Dans cet exemple, l'itemset est fréquent seulement si son support est plus grand ou égal à 30 % (minimum support).

La construction des clusters est faite en deux étapes : construction des clusters initiaux et construction des clusters disjoints.

3. La construction des clusters [FUN 03]

3.1 La construction initiale des clusters

Un premier cluster est construit pour chaque "global frequent itemset". Tous les documents contenant cet item set sont inclus dans le même cluster. Puisqu'un document habituellement contient plus qu'un " frequent itemset ", alors le même document peut apparaître dans des clusters initiaux multiples, c'est -à –dire que les clusters doivent assurer la propriété suivante : tous les documents dans un cluster contiennent tous les mots dans le "global frequent itemset " qui le définit, car ces mots sont considérés obligatoires pour chaque document dans un cluster ; donc ces global frequent itemset sont employés comme des *clusters labels (étiquettes)* pour identifier les clusters.

L'étiquette de cluster a deux buts : d'abord, elle établit la structure hiérarchique dans l'étape de construction d'arbre. De plus elle est présentée à l'utilisateur pour faciliter la recherche.

On utilise l'exemple présenté dans le tableau IV.2 pour représenter la construction initiale des clusters. Pour chaque " global frequent itemset " dans le tableau un cluster initial est construit, là où son étiquette est constituée par les mots de global frequent itemset correspondant. Par exemple l'étiquette de cluster de *C (patient; traitement)* est *« patient; traitement »*. Pour illustrer la construction initiale des clusters nous employons le document *med.*6, qui apparaît dans les clusters *C (patient; traitement)*, *C (patient)*, *C (résultat)* et *C (traitement)* car il contient tous les" global frequent itemset " de ces clusters ; en d'autres termes, il contient tous ces " cluster label ". Les clusters initiaux sont montrés dans le tableau (**Tableau IV.3**) ; la troisième colonne est expliquée plus tard :

Cluster	Document dans cluster	Cluster frequent items et leur clusters support
C {écoulement}	cran.1, cran.2, cran.3, cran.4, cran.5	{écoulement CS=100%}, {couche CS=100%}.
C {forme}	cisi.1, cran.1, cran.3, med.2, med.5	{forme CS=100%}.
C {couche}	cran.1, cran.2, cran.3, cran.4, cran5	{couche CS=100%}, {écoulement CS=100%}.
C {patient}	med.1, med.2, med.4, med.5,med.6	{patient, CS=100%}, {traitement CS=83%}, {résultat CS=80%}.
C {résultat}	cran.3, med.1, med.2, med.5, med.6	{résultat CS=100%}, {patient CS=100%}, {traitement CS=75%}.
C{ traitement }	med.1, med.2, med.3, med.4, med.6	{traitement CS=100%}, {patient CS=80%}.
C {écoulement, couche}	cran.1, cran.2, cran.3, cran.4, cran.5	{écoulement CS=100%}, {couche CS=100%}.
C {patient, traitement}	med.1, med.2, med.4, med.5, med.6	{patient CS=100%}, {traitement CS=100%}.
C{patient , résultat	med.1, med.2, med.5, med.6	{résultat CS=100%}, {patient CS=100%}, {traitement CS=75%}.

Tableau IV.3 : Initial cluster.

(minimum cluster support =70 %)

3.2 La construction disjointe des clusters

Après la construction des clusters initiaux sachant que chaque document appartient à un ou plusieurs clusters initiaux, nous allons faire une construction disjointes de ces clusters. Cette étape garantit qu'un document appartient à un seul cluster, et également chaque document dans un cluster contient toujours les items de cluster label.

Intuitivement, un premier cluster *C i* est bon pour un document *doc* j. S'il y a beaucoup de" frequent itemset " dans *doc j* et qu'ils apparaissent dans plusieurs documents dans *C i,* alors ils sont considérés comme point de référence pour un cluster et sont employés sous le nom de « *cluster frequent items* » afin de grouper les documents semblables. Nous disons qu'un item *x* est *cluster frequent* dans un cluster *C i* si *x* est contenu dans une certaine fraction minimum des documents dans C_i, *cluster support x* est le pourcentage des documents dans *Ci* qui contiennent *x*.

La troisième colonne dans le tableau (**Tableau IV.3**) montre pour chaque cluster les " cluster frequent itemset " et leur support. Par exemple, les " cluster frequent itemset " de cluster *C* (résultat) sont "patient", "traitement", et "résultat". Les deux items « patient » et « résultat » ont un support 100% de cluster support, parce que tous les documents dans le cluster contiennent ces items (ce sont des " cluster label "), " traitement " a un support de 75 %, parce que trois parmi quatre documents contiennent cet item.

Pour rendre ces clusters non-recouverts[1], une fonction est appliquée pour assigner chaque document a un et un seul cluster ; c'est la fonction score. Cette fonction mesure la qualité d'un premier cluster *C i* pour un document *doc j* car un document *doc* j est attribué au cluster initial *C i* de plus haute valeur de *score i* ; après cette tâche, chaque document appartient exactement à un cluster .

Score (Ci ←docj) = [∑n(x) * cluster support(x)] - [∑ n(x0) * global support(x0)]

x : représente un " global frequent itemset " dans doc_j qui est également un "cluster fréquent items " dans C_i .

x 0 : représente un " global frequent itemset " dans doc_j qui n'est pas "cluster fréquent items " dans C_i .

n (x) : est la fréquence d'apparition de *x* dans le "feature vector" de doc_j .

n (x 0) : est la fréquence d'apparition de *x0* dans le "feature vector" de doc_j.

Le premier terme de la fonction score montre que si " global frequent itemset " x dans docj est un "cluster frequent " dans cluster C_i, on peut mesurer l'importance de la fréquence de l'item x dans les différents clusters, car pour un document doc j cette dernière est multipliée par le cluster support dans C_i.

Le second terme de la fonction score pénalise le cluster C_i si on a le " global frequent itemset " **x0** dans doc_j qui n'est pas un "cluster frequent" dans C_i. La fréquence de **x0** est multipliée par le global support qui est vu comme une importance de **x0** dans l'ensemble de documents ou comme poids de pénalité de cet item.

Une propriété unique de cette fonction de points « score» est « la fréquence locale », c'est-à-dire le nombre d'occurrences d'un item dans un document .Cette fréquence est pris en considération en tant qu'élément du critère groupant. Cela est différent avec les autres méthodes de clustering de documents ("frequent itemset-based method "), où seulement la présence ou l'absence d'un itemset dans un document est considérée, mais la fréquence locale qui est une information importante, elle n'est pas utilisée.

Pour comprendre pourquoi la fréquence locale est cruciale, considérons un " global frequent itemset " "tennis" qui apparaît vingt fois dans le document *doc k* et " global frequent itemset " "football" qui apparaît seulement une fois dans *doc k*. Supposons qu'il y a deux clusters : un sur le tennis et l'autre sur le football. Si la fréquence d'un item dans *doc k* est ignorée, alors les deux " global frequent itemset " sont considérés d'une importance égale ; néanmoins, il est plus plausible de classifier *doc k* dans le cluster du "tennis", plutôt que dans le cluster du "football" ; car cette perspicacité importante est encapsulée dans la fonction de points «**score** ».

Considérons le tableau (**Tableau IV.3**) pour trouver le cluster le plus approprié pour le document *med.*1. Nous employons pour expliquer le calcul :

- *Score (C (patient)* ← *med.*1),
- *Score(C (résultat)* ← *med.*1),
- *Score (C (traitement)* ← *med.*1),
- *Score (C (patient; traitement)* ← *med* 1).

Score (C (patient) ← *med.1)* = 8 * 1 + 1 * 0, 80 +2 * 0,83 = 10. 46.
Score (C (résultat) ← *med.1)* = 8 * 1 + 1 * 1 + 2 * 0.75 = 10.5.
Score (C (traitement) ← *med .1)* = 8 * 0.8 + 2* 1 – 1 * 0.42 = 7.95.
Score (C (patient, traitement) ← *med.1)* = 8 * 1 + 2 * 1 – 1 * 0.42 = 9.58.

Les" global frequent itemset " de *med.*1 sont "patient", "résultat", et "traitement".Leur valeur dans le feature vector sont : 8, 1, et 2 respectivement. "patient"

[1] : Un cluster qui possède que ses propres documents.

et "traitement" sont des "frequent cluster" dans le cluster *C* (*traitement*) ; par conséquent ces deux items apparaissent dans la première partie de la fonction « **Score** » et leur fréquences sont multipliées par leur cluster support correspondants au cluster *C* (*traitement*) : 0.80 et 1 respectivement. L'item "résultat" n'est pas un "cluster fréquent" dans le cluster *C* (traitement), donc, il apparaît dans la deuxième partie de pénalité et sa fréquence est multipliée par son global support qui est égal à 0.42. Après le calcul de la fonction score pour chaque cluster initial de med.1, le document *med.*1 est assigné au cluster qui a la plus grande valeur de la fonction score. Le document med.1 est affecté a C (résultat) qui a la plus grande valeur de score.

Dans le cas où il existe deux clusters qui ont la même plus haute valeur de score le document concerné est affecté intuitivement au premier cluster dont le document existe .

Après avoir assigné chaque document à un cluster, le tableau (**Tableau IV.4**) illustre les clusters disjoints.

Cluster	Document dans cluster	Cluster frequent items et leurs clusters support
C {écoulement}	cran.1, cran.2, cran.3, cran.4, cran.5	{écoulement CS=100%}, {couche CS=100%}
C {forme}	cisi.1	{forme CS=100}
C {couche}		vide
C {patient}		{patient, CS=100%}, {traitement CS=83%}, {résultat CS=80%}.
C {résultat}	med.1, med.2, med.5, med6	{résultat CS=100%}, {patient CS=100%}, {traitement CS=75 %}.
C {traitement}	med.3	{traitement CS=100%}.
C {écoulement, couche}		vide
C {patient, traitement}	med.4	{patient CS=100%}, {traitement CS=100%},
C {patient, résultat}		vide

Tableau IV.4 : Cluster disjoint.

Remarques

1. Il y a une différence importante entre l'étiquette de cluster " *cluster label* "et l'ensemble de "cluster frequents" liés à un cluster ; car chaque document dans un cluster doit contenir tous les items dans le " *cluster label* ", tandis que un "cluster frequents" item doit nécessairement apparaître dans certaines fractions de documents dans un cluster .En d' autres termes, l'étiquette est employée comme une identité de cluster et "cluster frequents" comme une description de la matière du cluster.
2. Après avoir assigné tous les documents à leurs meilleurs clusters initiaux, nous avons besoin de recalculer les "cluster frequents" pour chaque cluster afin de refléter une mise à jour de clustering.

❸ Dans la construction disjointe des clusters le recalcule de "cluster fréquents items" de cluster C_i est nécessaire pour les clusters ayant des labels qui sont des combinaisons de label de super cluster. Dans le tableau **Tableau IV.4** la troisième colonne reflète les clusters avec une mis à jour de "cluster frequents" items. Les descendant potentiel du cluster C (traitement) est *C (patient, treatment)*. Tout en recalculant le "cluster frequents" de *C* (traitement), tous les documents de tous les deux *C (patient, traitement)* et *C* (traitement) sont considérés.

4. La construction de l'arbre de clusters [FUN 03]

La construction de l'arbre final est basée sur la similitude entre les clusters. Car au cas où un arbre contiendrait trop de clusters, deux méthodes de taille sont appliquées pour le fusionnement des clusters semblables afin de raccourcir et rétrécir l'arbre de clusters.

4.1 Construction de l'arbre

Dans ce qui suit, nous expliquons comment construire un arbre hiérarchique non-recouvert de clusters. L'arbre résultant de clusters a deux objectifs :

- ❖ fournir une structure logique à la recherche.
- ❖ chaque cluster a exactement un parent ; la matière (topic) d'un cluster parent est plus générale que celle des deux clusters enfants, mais elle est semblable à un certain degré.

Nous nous rappelons que chaque cluster emploie un " global frequent itemset " en tant que son étiquette (" cluster label "). Dans l'arbre de clusters, le noeud de racine rassemble les documents non similaires de différents clusters et a comme cluster label "null ", constituant le niveau 0 de l'arbre. Le niveau 0 regroupe tous les clusters avec une étiquette de "global frequent 1-itemset". Les clusters de niveau 2 sont les clusters avec une étiquette de "global frequent 2-itemset " et ainsi de suite pour chaque niveau. La profondeur de l'arbre est égale à la taille du plus grand "global frequent k-itemset ".

La figure **Figure IV.1** illustre l'algorithme utilisé pour la construction de l'arbre de clusters.

4.2 L'algorithme utilisé

Trier tous les clusters par le nombre d'itemsets de leur cluster label dans un ordre décroissant ;
Pour chaque cluster C_i dans la liste {
//enlever le noeud feuille vide
Si C_i ne contient aucun document et il n'a aucun cluster enfant **alors** {
sauter ce cluster vide C_i, et essayer le cluster C_{i+1} } ;
//identifier tous les parents potentiels
k = le nombre d'itemsets dans C_i cluster label ;
Potentiel Parents = trouver tous les clusters contenant l'étiquette de cluster C_i avec *ki-1* itemset qui est un sous-ensemble de C_i cluster label
//choisir le parent le plus semblable
doc (C_i) = fusion de tous les documents dans le sous-arbre C_i dans un document combiné simple ;
Calculer les points (score) de *doc* (C_i) contre chaque *Potentiel Parents* ;
Placer le cluster parent qui a le plus hauts points dans le parent de C_i ;

Figure IV.1 : L'algorithme de la construction de l'arbre de cluster [FUN 03]

Dans ce qui suit, on note un cluster avec "cluster label"de global frequent k-itemset comme **k-cluster**.

Dans cet algorithme, les *k* - clusters apparaissent toujours à un niveau plus élevé que (*k* - 1)-clusters dans l'arbre. Un arbre ascendant est construit en commençant par le niveau le plus haut. Le cluster label est considéré comme un item obligatoire, donc la construction de l'arbre est basée sur les clusters label comme suite :

- pour chaque *k* – cluster C_i, nous identifions tous les potentiels parents qui sont (*k-1*) clusters et qui ont comme étiquette, un sous-ensemble de C_i cluster label ;
- la prochaine étape est de choisir le "meilleur" parent parmi ces parents potentiels. Le critère de choix de meilleur parent est semblable à choisir le meilleur cluster pour un document dans l'étape précédente (la construction disjointe de cluster) ;
- fusionnons d'abord tous les documents dans le (*k-1*)-clusters comme un document conceptuel simple *doc* (C_i), et calculant les points (score) de *doc* (C_i), contre chaque parent potentiel. Celui qui a le plus hauts points deviendrait le parent de C_i. Dans l'exécution réelle, l'opération de fusionnement des documents dans *doc* (C_i) est accompli en ajoutant les "feature vector" des documents de cluster C_i ;
- les noeuds vides des feuilles sont enlevés pendant la construction de l'arbre.

Il arrive souvent que des nœuds non feuille ont habituellement un ensemble de nœuds enfants qu'ils contiennent beaucoup de documents. Cette situation peut apparaître quand les documents sous une matière ("topic") sont bien catégorisés pour les sous matières (sub topic), car le nœud qui est vide sert comme un intermédiaire pour organiser les sub-topic sous la même catégorie, donc il facilite énormément la recherche.

Ces nœuds non feuille sont gardés pour la prochaine étape « greffer l'arbre ».

A partir des clusters de tableau **Tableau IV.4**, l'arbre est construit en commençant par les 2-clusters (c'est -à- dire, les clusters avec 2-itemsets comme étiquette de cluster). Les clusters feuille vides sont supprimés .puisque il existe un seul 2-cluster qui est *C* (*patient, traitement)*, donc son parent est cherché. Les parents potentiels sont *C* (*patient)* et *C* (*traitement)*. Le poids de doc(*C* (*patient; traitement*)) est calculé comme suit :

doc(*C* (*patient, traitement*))= la somme des "feature vector" de *C* (*patient, traitement)* = *doc (Ci)* = (0, 0, 0, 1, 0, 9)

Score (*C* (*patient)* ← *doc (Ci)*)= 9 * 0.83 + 1*1= 8.47

Score (*C* (*traitement)* ← *doc (Ci)*)=9 * 1 – 1*0.42= 8.58

Puisque le cluster *C* (*traitement)* a la plus haute valeur de score, donc il est choisi comme un parent de *C* (*patient, traitement*).L'arbre résultante est représenté dans la figure **Figure IV.2** en dessous.

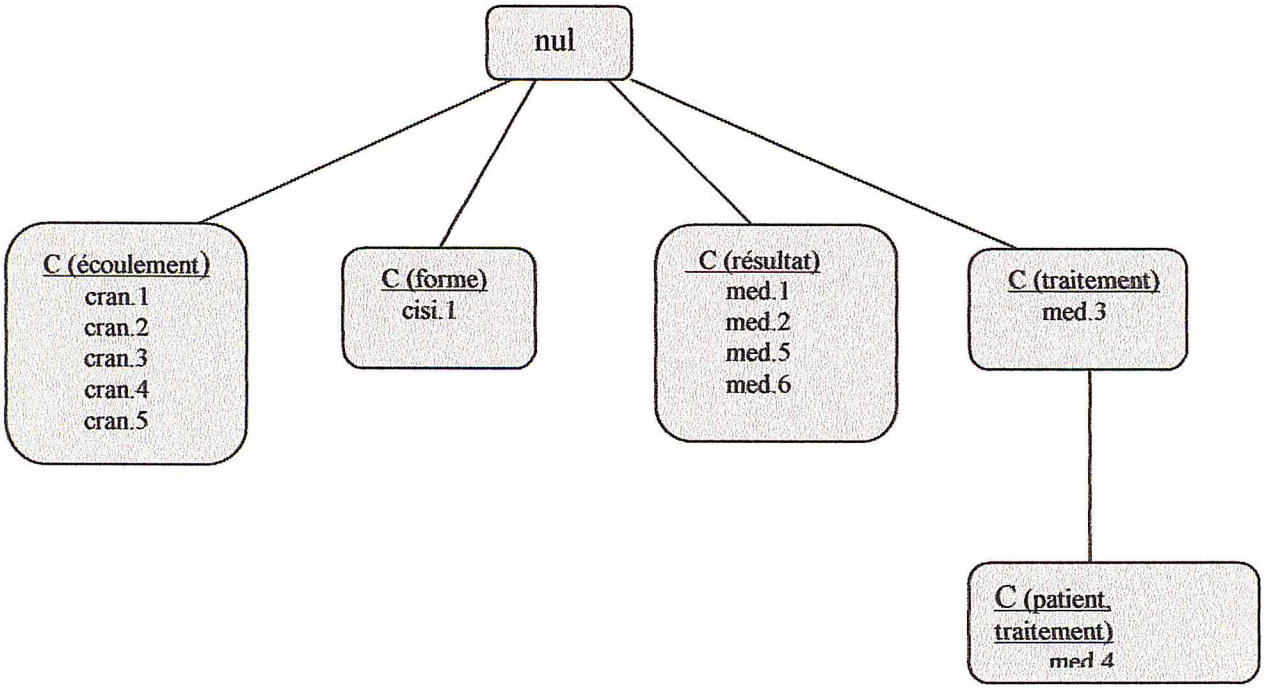


Figure IV.2 : L'arbre construit à partir de Tableau IV.4.

5. Greffer l'arbre (l'optimisation) [FUN 03]

Un arbre de clusters peut être large et profond, particulièrement quand un bas global minimum support est employé ; par conséquence, il est probable que des documents de la même matière seraient distribués sur plus d'un cluster, qui mèneraient à une exactitude de clustering faible. Le but de cette étape est de fusionner les clusters semblables afin de produire une hiérarchie de matière qui permet de faciliter la recherche et d'augmenter l'exactitude.

Avant d'introduire les techniques de taillement de l'arbre, définissons d'abord la formule d' *inter- cluster similarité*, car c'est la notion clé dans le fusionnement des clusters. Pour mesurer la similitude inter-clusters entre deux clusters C_a et C_b, une autre notion est définit c'est la similitude de C_b à C_a, et la similitude de C_a à C_b. L'idée est de traiter un cluster en tant qu'un document (en combinant tous les documents dans le cluster) et on mesure ses points contre un autre cluster en utilisant la fonction « score » définie précédemment. La seule différence est que la fonction score est normalisée pour voir l'effet de la variation de la taille des documents.

La similitude de Ci à C_j est définie comme suit :

$$\mathbf{Sim(Ci \leftarrow C\ j) = ((Score(Ci \leftarrow doc\ (C\ j)) / (\sum x\ n(x) + \sum x0\ n(x0))) + 1}$$

***doc* (*C j*)**: représente la combinaison de tous les documents dans le sous-arbre de C_j en un document simple ;

x : représente un " global frequent itemset " dans *doc j* qui est également un "cluster frequent items " dans C_i.

x 0 : représente un " global frequent itemset " dans *doc j* qui n'est pas "cluster fréquent items " dans *C i* .

***n* (*x*)** : est la fréquence de *x* dans le "feature vecteur" de *doc* (*C j*) ;

$n(x0)$: est la fréquence de $x\ 0$ dans le items dans le "feature vecteur" $doc(Cj)$.

Pour expliquer la normalisation de $\sum x\ n(x) + \sum x0\ n(x0)$; le global support et le cluster support de la fonction score sont entre [0,1], car la maximum valeur de score est $\sum x\ n(x)$, et la minimum valeur est de $-\sum x0\ n(x0)$, donc le score est normalisé en divisant sur la somme $\sum x\ n(x) + \sum x0\ n(x0)$. Les points (valeurs) sont dans la marge de [- 1.1]. Pour éviter des valeurs négatives de similitude, une valeur est ajoutée +1 à la limite. En conséquence, la gamme de la fonction *Sim* est [0. 2].

La similitude d'inter-cluster entre C_a et C_b est définie comme la moyenne géométrique normalisée des deux points de score normalisé $Sim(C_a \leftarrow C_b)$ et $Sim(C_b \leftarrow C_a)$:

$$Inter\ Sim(C_a \leftrightarrow C_b) = [\ Sim(C_a \leftarrow C_b) * Sim(C_b \leftarrow C_a)\ ]^{1/2}$$

$Sim(C_a \leftarrow C_b)$: est la similarité de C_b vers C_a.

$Sim(C_b \leftarrow C_a)$: est la similarité de C_a vers C_b.

L'avantage de la moyen géométrique est que deux clusters sont considérés comme semblables seulement si les deux valeurs de $Sim(C_a \leftarrow C_b)$ et $Sim(C_b \leftarrow C_a)$ sont hautes. La gamme de la fonction *Inter Sim* est également [0.2]. Des valeurs plus élevées impliquent une similitude plus élevée entre deux clusters.

Si la valeur *Inter Sim* ci-dessous 1 implique que la fréquence d'itemset différents a excédé l la fréquence d'itemset semblables. Par conséquent, la valeur de *Inter Sim* est 1 et indique que les deux clusters sont semblables.

Nous présentons maintenant les deux techniques de taillement de l'arbre.

5.1 Taille d'enfant « child pruning »

L'objectif de cette étape est de raccourcir l'arbre en remplaçant les clusters enfants par leurs parents. Le critère de taille est basé sur la similitude « inter- clusters » entre le parent et ses enfants, car un enfant est taillé si et seulement s'il est semblable à son parent ;donc cette technique ne dégrade pas la pureté du parent. Cette technique est basée sur la lois suivante : la sous matière d'un enfant (par exemple balle de tennis) est très semblable à la matière de son parent (par exemple tennis) ; dans le cas où cette similitude est trop spécifique, elle peut être enlevée et l'enfant est taillé.

Le procédé est présenté sur la figure **Figure IV.2** au-dessous.Un balayage de l'arbre en ascendant pour chaque noeud non feuille est établi, ensuite un calcul de *Inter Sim* pour chaque enfant ainsi que leurs descendants est effectué. Si *Inter Sim* est au-dessus de 1, alors le cluster est taillé, par conséquent ses enfants deviennent les enfants de leur grand-père. Cette étape est appliquée seulement au niveau 2 vers le bas au fond de l'arbre car c'est illogique de comparer les clusters de niveau 1 à la racine, car ils rassemblent les documents non semblables.

L'algorithme utilisé

```
Balayer l'arbre en ascendant ;
Pour chaque nœud non feuille C_i dans l'arbre
{Calculer Inter Sim de C_i avec chacun de ses enfants comprenant leurs descendants ;
Tailler le cluster enfant si Inter Sim est au-dessus de 1 ;
}
```

Pour déterminer si le cluster *C (patient, traitement)* est taillé, on calcul la similitude d'inter-clusters entre *C (traitement)* et *C (patient, traitement)*.

Sim (*C (traitement)* ← *C (patient, traitement)*) = [(9 * 1-1*0.42) / (9+1)] +1= 1.85

Sim (*C (patient, traitement)* ← *C (traitement)*)= [(2 * 1) / 2] + 1= 2

Inter Sim (*C (traitement)* ← *C (patient; traitement)*) = $[(1.85 * 2)]^{1/2}$ =1.92

Pour calculer *Sim* (*C (traitement)*← *C (patient; traitement)*, nous combinons tous les documents du cluster *C (patient, traitement)* avec l'addition de leurs "feature vectors". Le vecteur additionné est (0, 0, 0, 1, 0, 9).

Puisque la similitude inter-clusters est au-dessus de 1, cluster *C (patient, traitement)* est taillé. Le résultat est illustré dans la figure **Figure IV.3** suivante :

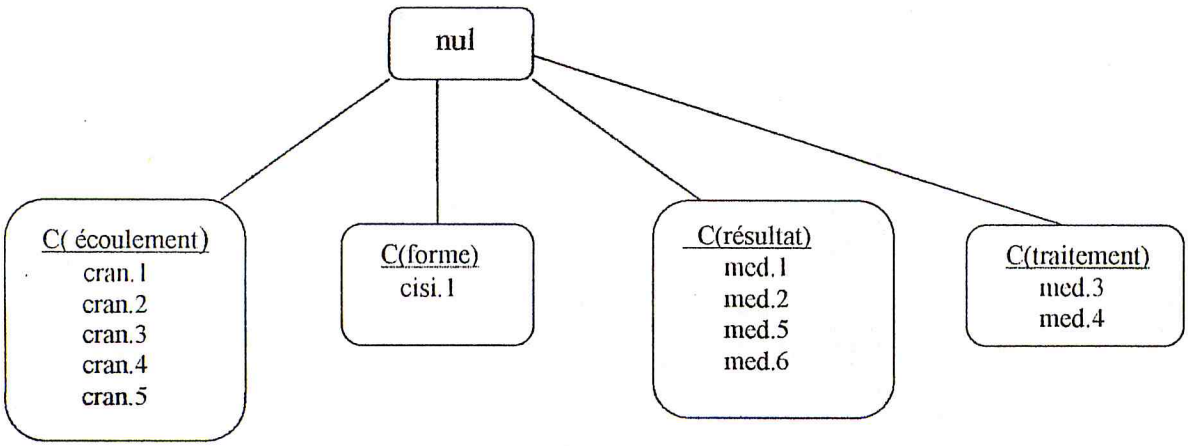


Figure IV.3 : L'arbre avec des enfants taillés.

5.2 Le fusionnement d'enfants de même parent

Le fusionnement d'enfant de mêmes parents rétrécit l'arbre en fusionnant les sous-arbres semblables au niveau 1.

L'algorithme représenté dans la figure **Figure IV.4** illustre le procédé :

- calculer *Inter Sim* pour chaque paire de clusters au niveau 1 d'un arbre. Le fusionnement continu entre les clusters qui ont la plus haute *Inter Sim* jusqu'à ce que le nombre personnalisé de clusters entrés par l'utilisateur soit atteint.
- on cas où un utilisateur n'indiquerait pas le nombre désiré de clusters, l'algorithme se terminerait quand toutes les paires de cluster ont *Inter Sim* inférieur ou égal à 1.
- cette étape est appliquée seulement au niveau 1, car l'application pour chaque niveau de l'arbre est trop chère informatiquement

Cependant, dans cet algorithme le nombre de clusters non vides de niveau 1 est toujours limité par le nombre de "globals frequents itemset ".

L'algorithme utilisé

Pour chaque paire de clusters au niveau 1 de l'arbre

{Calculer *Inter Sim* ainsi que leurs descendants ;

Stocker le résultat dans une matrice ;

}

Répéter

{Choisir la paire de cluster qui a le plus haut *Inter Sim* ;

Fusionner le cluster le plus petit dans le cluster le plus grand avec leurs descendants ;

Mettre à jour la matrice de similitude inter-clusters ;

} **Jusqu'à ce que** l'utilisateur ait indiqué le nombre de clusters

Figure IV.4 :l'algorithme de fusionnement d'enfants d'un même parent [FUN 03].

Application de l'algorithme

On considère l'arbre de la figure **Figure IV.3**. L'*Inter Sim* est calculé pour chaque paire de clusters au niveau 1.Si l'utilisateur n'a pas indiqué le nombre désiré de clusters, FIHC terminera et renverra l'arbre comme cela est représenté dans la figure **Figure IV.3.**

Supposons que l'utilisateur a indiqué le nombre de clusters à trois, l'algorithme taillerait un cluster au niveau 1 utilisant la matrice de similitude qui représente les valeurs de similarité et d'inter-similarité entre toutes les paires de clusters. Dans notre exemple, les clusters sont :

C (écoulement), *C (forme)*, C (résultat) et *C (traitement)*.

La matrice de similitude est représentée dans le **Tableau IV.5** ci- dessous.

Paire de clusters (Ci , Cj)	Sim (Cj ← Ci)	Sim (Ci ← Cj)	Inter-sim (Ci ↔ Cj)
C (écoulement) & C (forme)	0.71	0.58	0.64
C (écoulement) & C (résultat)	0.77	0.87	0.81
C (écoulement) &C (traitement)	0.58	0.58	0.58
C (forme) & C (résultat)	0.58	0.65	0.61
C (forme) & C (traitement)	0.58	0.58	0.58
C (résultat) & C (traitement)	1.2	0.79	0.97

Tableau IV.5 : Matrice de similitude

C (résultat) et C (traitement) est la paire avec la plus haute valeur d'Inter-similarité. Donc le cluster le plus petit C (traitement) est fusionné avec le cluster le plus grand C (résultat).

L'arbre final avec les trois clusters est représenté dans la figure suivante **Figure IV.5.**

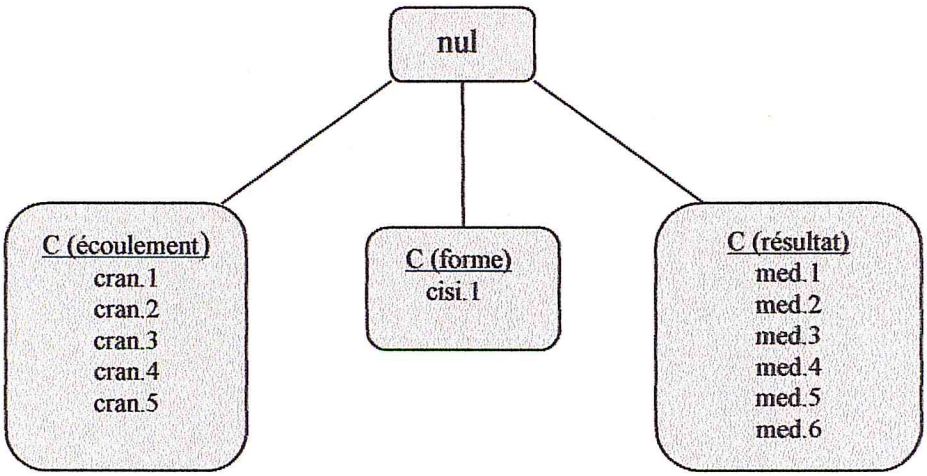


Figure IV.5 : L'arbre final.

6. Conclusion

Après qu'on a représenté les différentes étapes de méthode FIHC, on constate que cette dernière donne comme résultat un arbre de clusters non profond, car les deux techniques utilisée (le fusionnement d'enfants de même parent et la taille d'enfant), permettent de donner un arbre qui favorise la lecture rapide .En plus l'utilisation d'une fonction de poids (Score), qui permet d'assigner les documents aux meilleurs clusters, augmente l'exactitude de clustering.

La méthode FIHC utilise deux paramètres obligatoires, le " minimum support " et le cluster support pour la génération des "globals frequents itemset", tandis que le nombre de cluster est un paramètre facultatif, car l'arbre de clusters est construit même si ce dernier est inconnu. Ces caractéristiques rendre FIHC assez robuste.

En raison des apports appréciables de la méthode FIHC, nous avons choisi de l'utiliser pour la conception de notre implémentation. C'est pourquoi, dans le chapitre suivant, nous présentons la démarche de développement de cette méthode appliqué au clustering

Chapitre V

La démarche de Développement

1. Introduction

Le clustering de document est déjà utilisé dans un grand nombre de domaines de l'informatique .Nous allons nous intéresser à la méthode **FIHC** présentée précédemment, à la quelle nous allons introduire des modifications dans l'implémentation dans le but d'optimiser le résultat élaboré. Le résultat est un arbre de clusters, présenté dans un fichier XML. Chaque cluster est représenté par plusieurs données souvent difficiles à interpréter. C'est pourquoi, nous allons présenter le résultat sous forme de diagrammes d'un langage formel « **UML** » plus facile à interpréter.

Nous commençons ce chapitre par le choix de la méthode de développement suivie, et pour réaliser notre conception, nous présentons toutes les étapes de ce développement :

- la spécification des besoins ;
- l'analyse du domaine ;
- la conception ;
- l'implémentation ;
- le test et la validation.

2. choix de la démarche suivie

Le processus de développement que nous avons suivi est le cycle de vie en cascade, ce modèle décrit par Royce en 1970 a été largement employé depuis, pour la description générale des activités liées aux logiciels [MUL 97] .Il décrit le cycle de vie d'un logiciel par une suite de phases qui s'enchaînent dans un déroulement linéaire (**Figure V.1**).

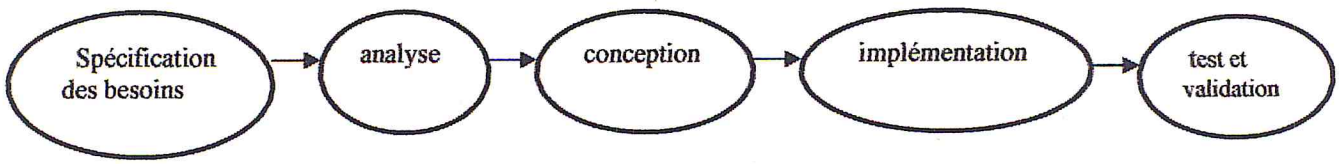


Figure V.1 : Cycle de vie d'un logiciel.

La notation unifiée UML que nous avons utilisée permet de couvrir le cycle de vie d'un logiciel dans l'étape spécification des besoins, analyse, conception et implémentation. Cette notation est de grande richesse, elle permet de définir :

- les besoins des utilisateurs de système, exprimés à l'aide de cas d'utilisation ;
- la spécification complète du système, sous forme de diagrammes couvrant les parties statiques et dynamiques du système ;
- la conception détaillée, jusqu' à un niveau très proche du code en langage objet de l'implémentation.

Nous représentons quelques notions de base d'UML, afin de faciliter la compréhension des différents diagrammes utilisés ultérieurement dans l'Annexe E.

Remarque

Les diagrammes d'UML présentées dans ce mémoire sont édités et validés par le logiciel Microsoft Visio 2003, et par conséquence la notation UML de ces diagrammes suit la notation de Microsoft.

3. La spécification des besoins

La finalité de cette étape est la description générale des fonctionnalités du système, en répondant à ces questions :

« Quelles sont les fonctions du système ? », « Quels sont les utilisateurs du système ? », et « Qu'attendent-ils du système ? ».

Cette phase étudie le comportement de système exprimé sous la forme des cas d'utilisation, le contexte de système, les acteurs et les scénarios.

3.1 La description des cas d'utilisations

Les cas d'utilisation ou "use case" sont utilisés pour décrire les besoins des utilisateurs. Qu'est ce qu'un utilisateur peut faire. Quelles sont les actions qui lui sont proposées ? De quoi a-t-il besoin ? Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.

- Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système.
- Ils permettent de classer les acteurs et structurer les objectifs du système.
- Ils servent de base pour mentionner les exigences d'un système dans un processus de développement intégrant UML.

Les différentes fonctionnalités offertes par le système forment un ensemble de *cas d'utilisation* UML proposé à travers les diagrammes des cas d'utilisation : pour répertorier et structurer les fonctionnalités que le futur système offrira à ses utilisateurs.

Notre système est destiné à des utilisateurs. Les entrées de système sont validées par l'utilisateur qui est un expert de domaine.

- Un expert de domaine (exemple développeur des sites Web) est un utilisateur chargé d'entrer les supports d'entrées qui sont des seuils importants pour l'exactitude des groupes (clusters).

Les cas d'utilisations recensés pour le système sont présentés dans la section suivante.

3.1.1 Cas d'utilisation « Système global »

Dans le cas d'utilisation « système global », l'utilisateur peut lancer plusieurs commandes et entrer plusieurs paramètres. L'utilisateur effectue les fonctions suivantes :

- gérer les entrées à travers l'interface graphique (les fichiers, les dossiers, ...) ;
- lancer l'analyse après que toutes les entrées sont effectuées ;
- lancer la commande de visualisation de fichier XML comme résultat ;
- lancer la commande d'enregistrement de fichier XML (sauvegarde de résultat) ;
- lancer la commande nettoyage pour entrer les nouvelles données.

Chapitre V

Démarche de développement

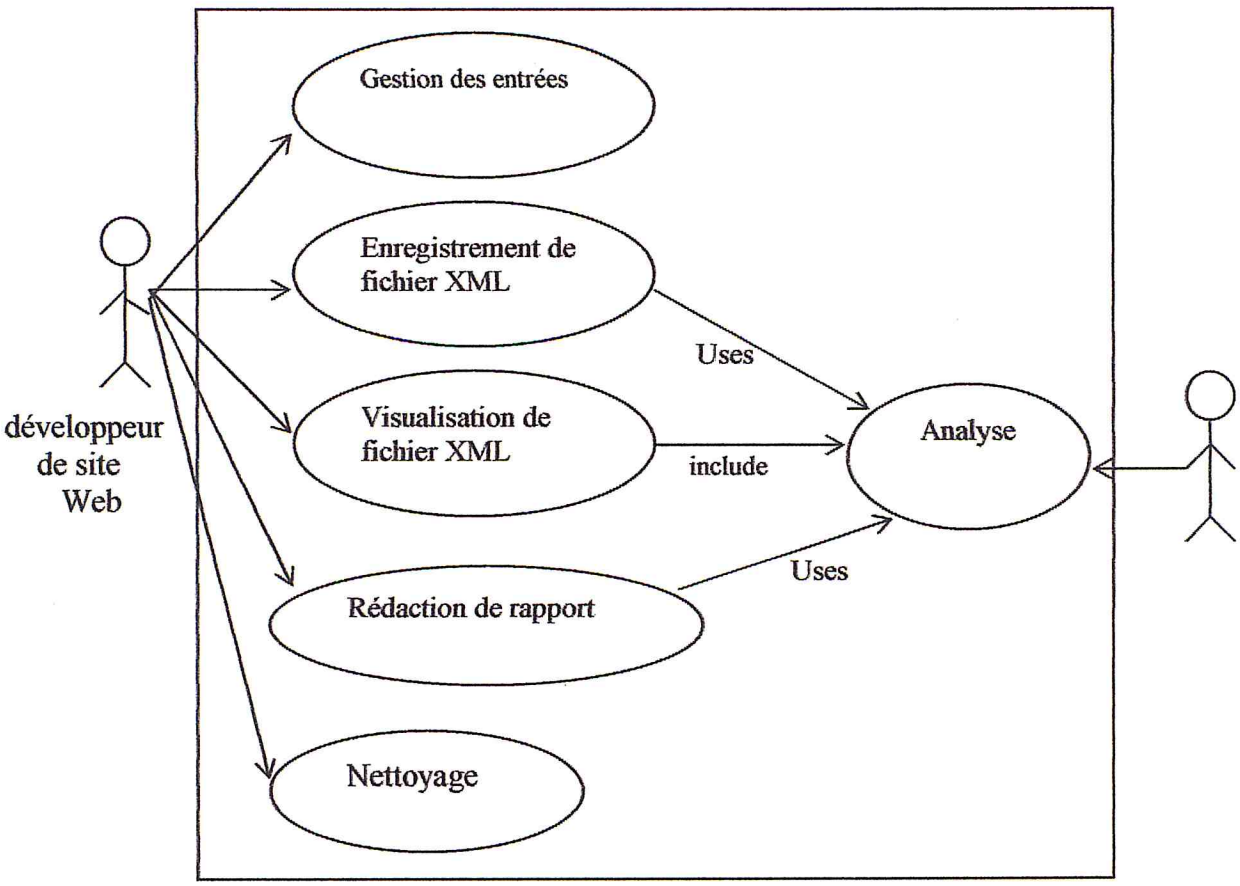


Figure V.2 : Cas d'utilisation « système global ».

Après que tous les paramètres d'entrées sont entrés, l'utilisateur lance le traitement à travers la commande « analyse », et le rapport de fonctionnement débute. A la fin de l'analyse l'utilisateur peut enregistrer le résultat qui est un fichier XML.

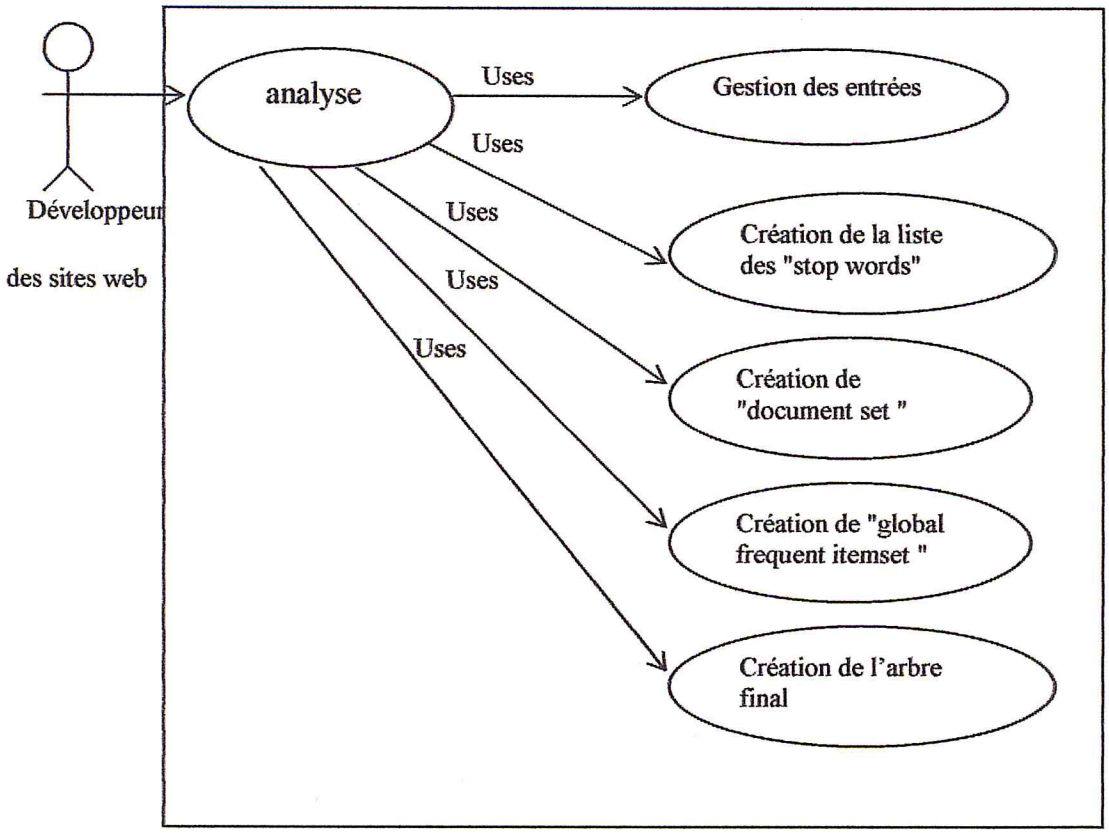
L'utilisateur peut à tout moment visualiser les résultats antérieurs (les fichiers XML obtenus précédemment). cela explique que les deux cas d'utilisations « rédaction de rapport »et « enregistrement de fichier XML » utilisent le cas « analyse » qui est exprimé dans le diagramme avec le stéréotype"Uses "; tandis que le cas d'utilisation « visualisation de fichier XML » inclut le cas « analyse » qui est exprimé par la relation "include".

3.1.2 Cas d'utilisation « analyse »

Le cas d'utilisation « analyse » utilise plusieurs autre cas d'utilisation qui sont:

- le cas « gestion des entrées » où tous les chemins des fichiers et dossiers sont entrées, en plus les supports ;
- le cas « créations de la liste des stop words », la liste des mots qui ne sont pas pris en considération pendant le traitement des fichiers ;
- le cas « création de documents set » où les fichiers et les dossiers sont traités ;
- le cas « création de l'arbre final »où l'arbre des clusters est construite.

Tous ce qui annoncé ci -dessous est représenté dans le diagramme par la relation "Uses".



FigureV.3 : Cas d'utilisation « analyse ».

3.1.3 Cas d'utilisation « Gestion des entrées »

Le cas d'utilisation « gestion des entrées » utilise trois autres cas : gestion des fichiers, gestion des dossiers et gestion des paramètres supplémentaires.

- gestion des fichiers inclut trois autres cas ; car l'utilisateur peut ajouter un fichier, supprimer un fichier ou supprimer tous les fichiers pour insérer d'autres.
- gestion des dossiers inclut aussi trois cas où l'utilisateur peut ajouter un dossier, supprimer un dossier ou supprimer tous les dossiers pour insérer d'autres.
- Gestion des paramètres supplémentaires qui sont :
  - minimum support ou seuil minimum où toute les fréquences des itemsets qui sont supérieures ou égales à ce support sont considérées comme fréquentes, tandis que tous les itemsets avec une fréquence inférieure à ce support sont pris comme infréquents ;
  - clusters support où *cluster support* x est le pourcentage des documents dans C_i qui contiennent x. (C_i est un cluster) ;
  - le nombre de clusters désiré dans le niveau 1 de l'arbre de clusters.

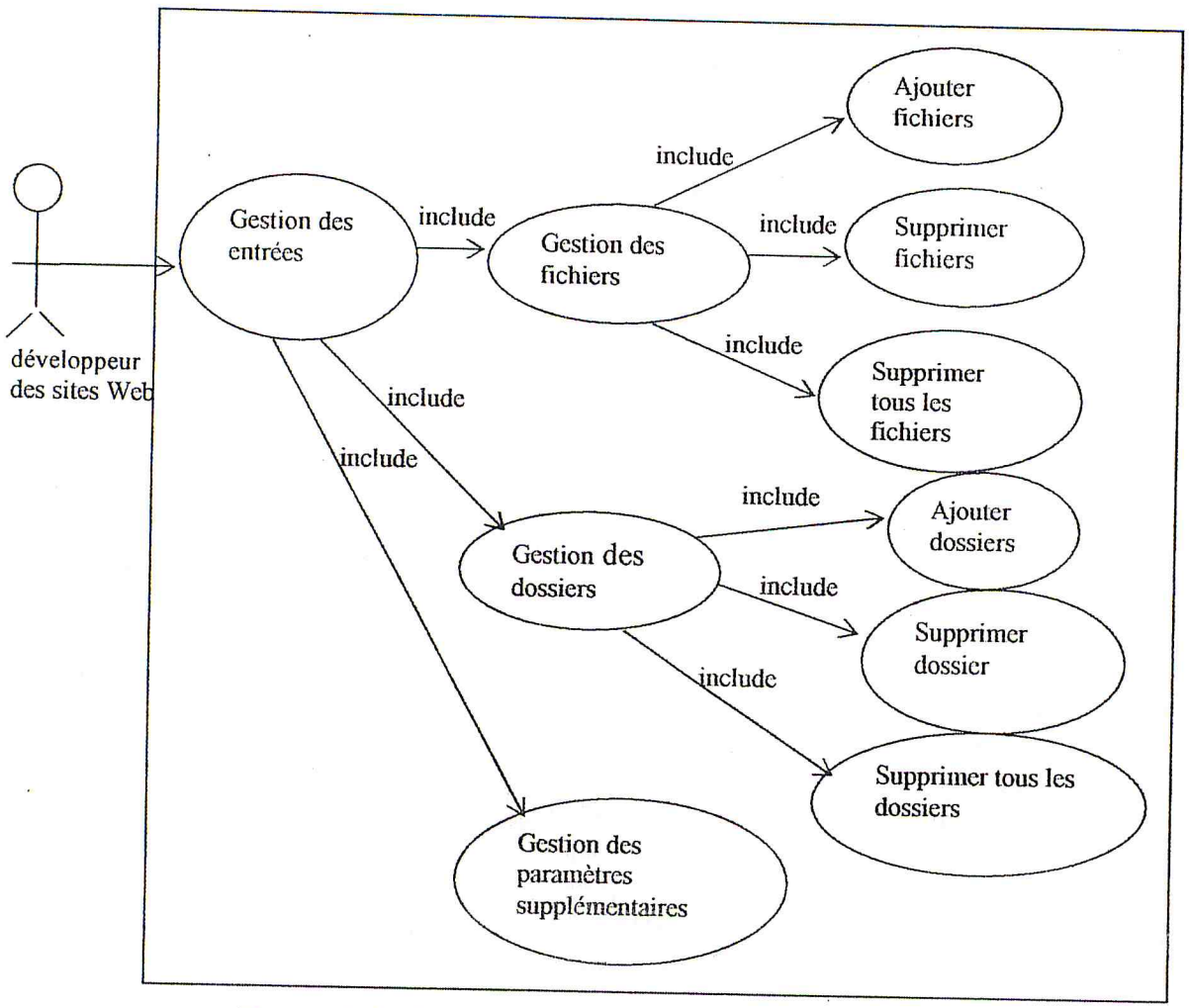


Figure V.4 : Cas d'utilisation « gestion des entrées ».

3.1.4 Cas d'utilisation « création de la liste des " stop words " »

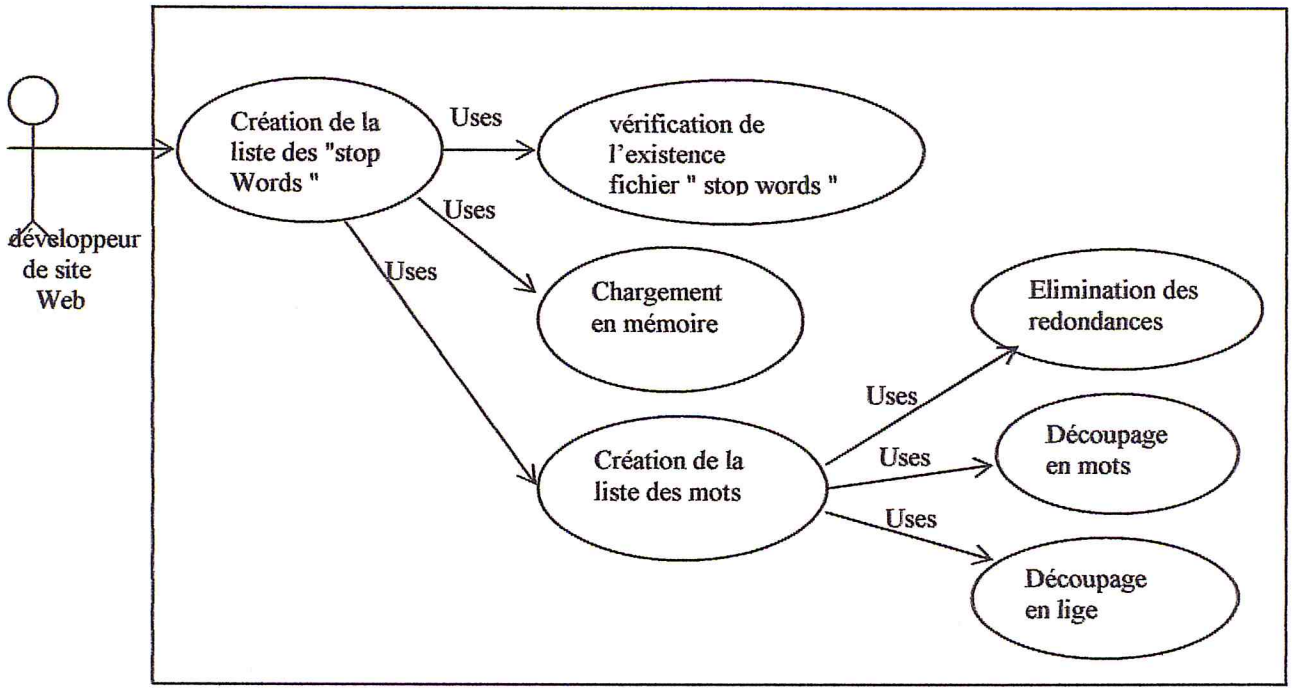
Le cas « création de la liste de "stop words "» utilise trois cas :

- vérification de l'existence qui vérifie le chemin de fichier "stop Words" ;
- chargement de fichier "stop Words "en mémoire ;
- création de la liste des mots à partir de fichier " stop Words " ;

En effet le cas « création de la liste des mots » utilise trois cas :

- transformer le fichier en une ligne ;
- découpage de la ligne en mots et insertion dans la liste des mots "stop words ";
- élimination des mots redondants.

Ces relations sont représentées par la relation "Uses" dans le diagramme.



FigureV.5 : Cas d'utilisation « Création de la liste de "stop word" ».

3.1.5 Cas d'utilisation « Création de "document set " »

Il est représenté dans la figure ci-dessous.

Ce cas d'utilisation s'intéresse au traitement d'un fichier, car il permet découper chaque fichier en un ensemble de mots qui sont regroupé dans une liste , qui donnent comme résultat une matrice d'occurrence de tous les mots trouvés dans tous les fichiers. Après ce découpage, nous appliquons une synthèse dont nous extrayions les mots fréquents.

Création de "document set " utilise deux cas, création de la matrice des occurrences, et synthèse.

1) Création de la matrice des occurrences utilise trois autres cas :

- création de la liste finale des fichiers après le parcours de tous les dossiers et les sous dossiers.
- vérification de l'existence des fichiers d'après les chemins mentionnés.
- chargement de fichier en mémoire afin de le traiter.
- création de la liste des mots cette liste est obtenue à partir d'un traitement de tous les fichiers, ce cas utilise aussi trois autre cas :
  - découpage en ligne d'un fichier.
  - découpage en mot de chaque ligne
  - élimination des mots redondants dans les différentes lignes de l'ensemble des fichiers.

2) Synthèse où la liste des mots est traités dont elle utilise trois autre cas :

- Création de la liste des mots fréquents (supérieure ou égale au minimum support).

- Création de la liste des mots fréquents répétés qui se répètent dans tous les fichiers.
- Création de la liste de tous les mots fréquents (supérieur ou égale au minimum support et qui se répètent dans tous les fichiers).

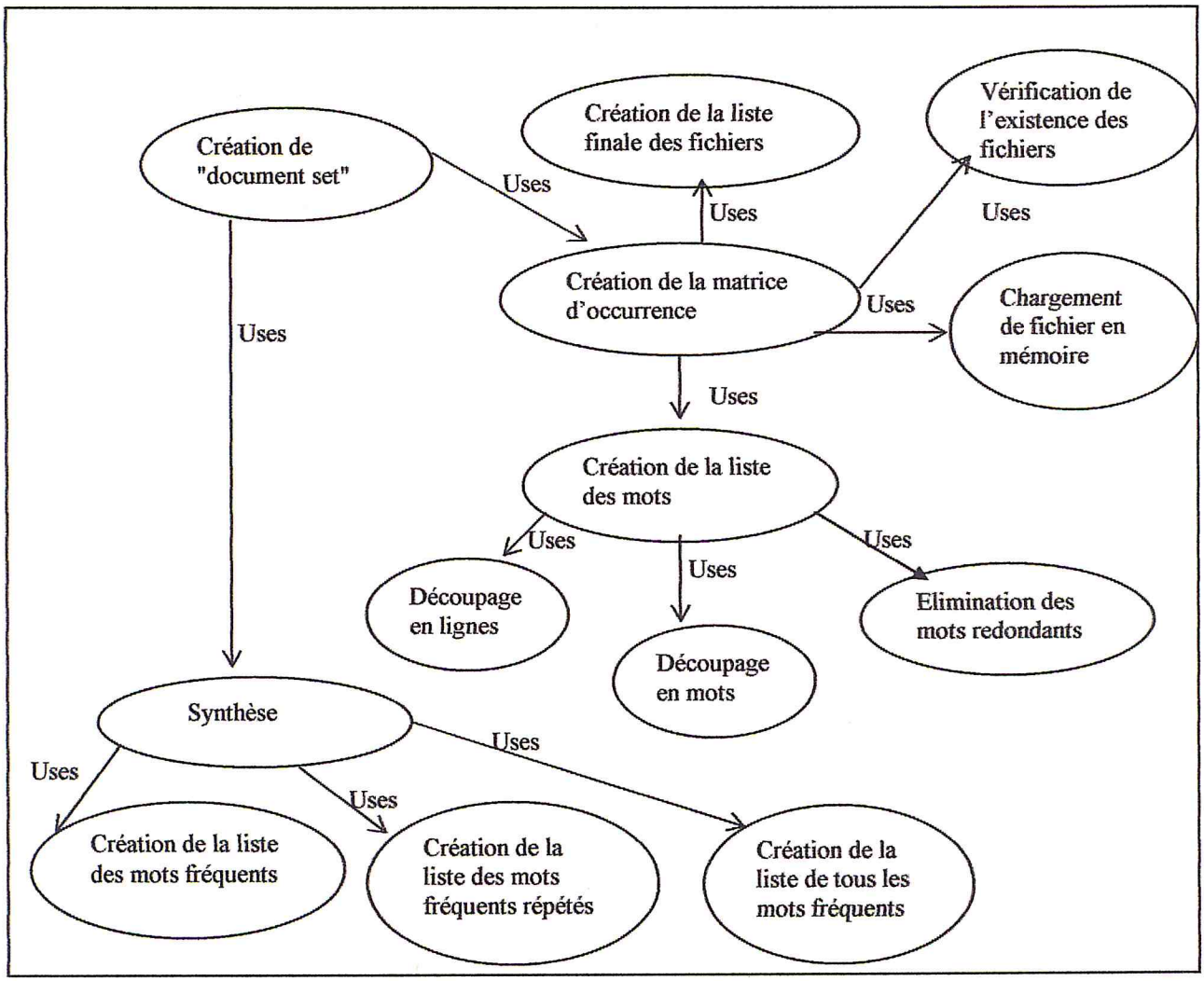


Figure V.6 : Cas d'utilisation « Création de "document set " ».

3.1.6 Cas d'utilisation « Création de " global frequent itemset "»

Pour créer la liste de " global frequent itemset " nous avons besoin de la liste de tous les mots fréquents, ceci utilise trois cas d'utilisation :

- création de la liste " 1-itemset " à partir de la liste des mots fréquents.
- création de la liste " K-itemset " qui utilise aussi le cas précédent (liste " 1-itemset ").
- création de la liste " K-itemset répétées" à partir de la liste des mots répétés.

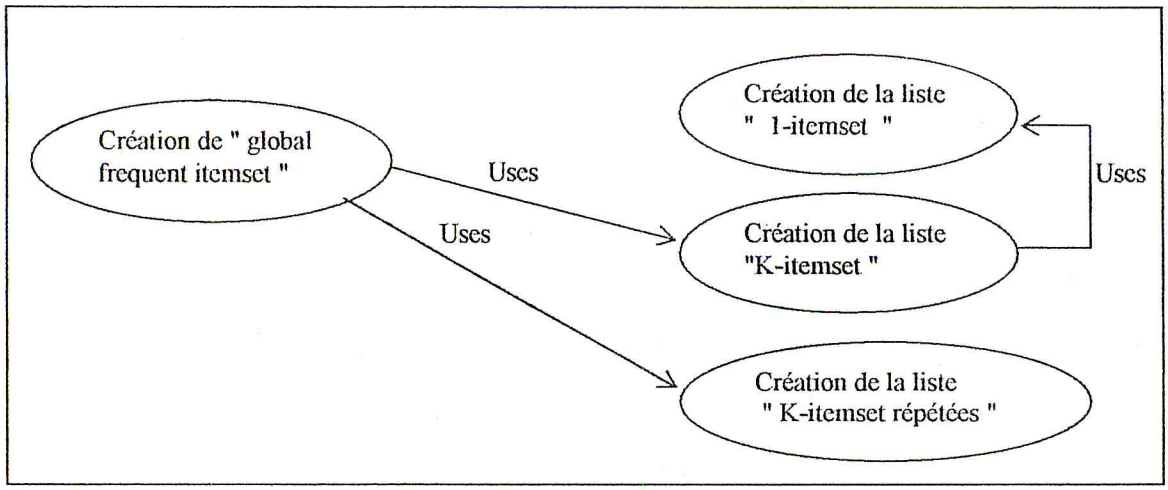


Figure V.7 : Cas d'utilisation « Création de "global frequent itemset" ».

3.1.7 Cas d'utilisation « Création de " l'arbre final " »

Le cas d'utilisation « création de l'arbre final » utilise trois cas : création de l'arbre initial, optimisation verticale et optimisation horizontale ; et il inclut le cas optimisation supplémentaire qui peut être demandé ou non par l'utilisateur ; cela est représenté par la relation " include" .

- Optimisation verticale utilise deux cas, calcul de la table d'inter-similarité et taille d'enfant.
- Optimisation horizontale utilise trois autre cas qui sont :
  - détermination des clusters similaires.
  - calcule de la matrice de similitude.
  - fusionnement des clusters regroupés dans les cas précédent.
- Optimisation supplémentaire utilise deux cas :
  - calcul de la matrice de similitude qui est utilisée pour le niveau 1 de l'arbre des clusters.
  - détection et élimination des clusters inutiles.

Ce cas d'utilisation est présenté dans la figure ci dessous.

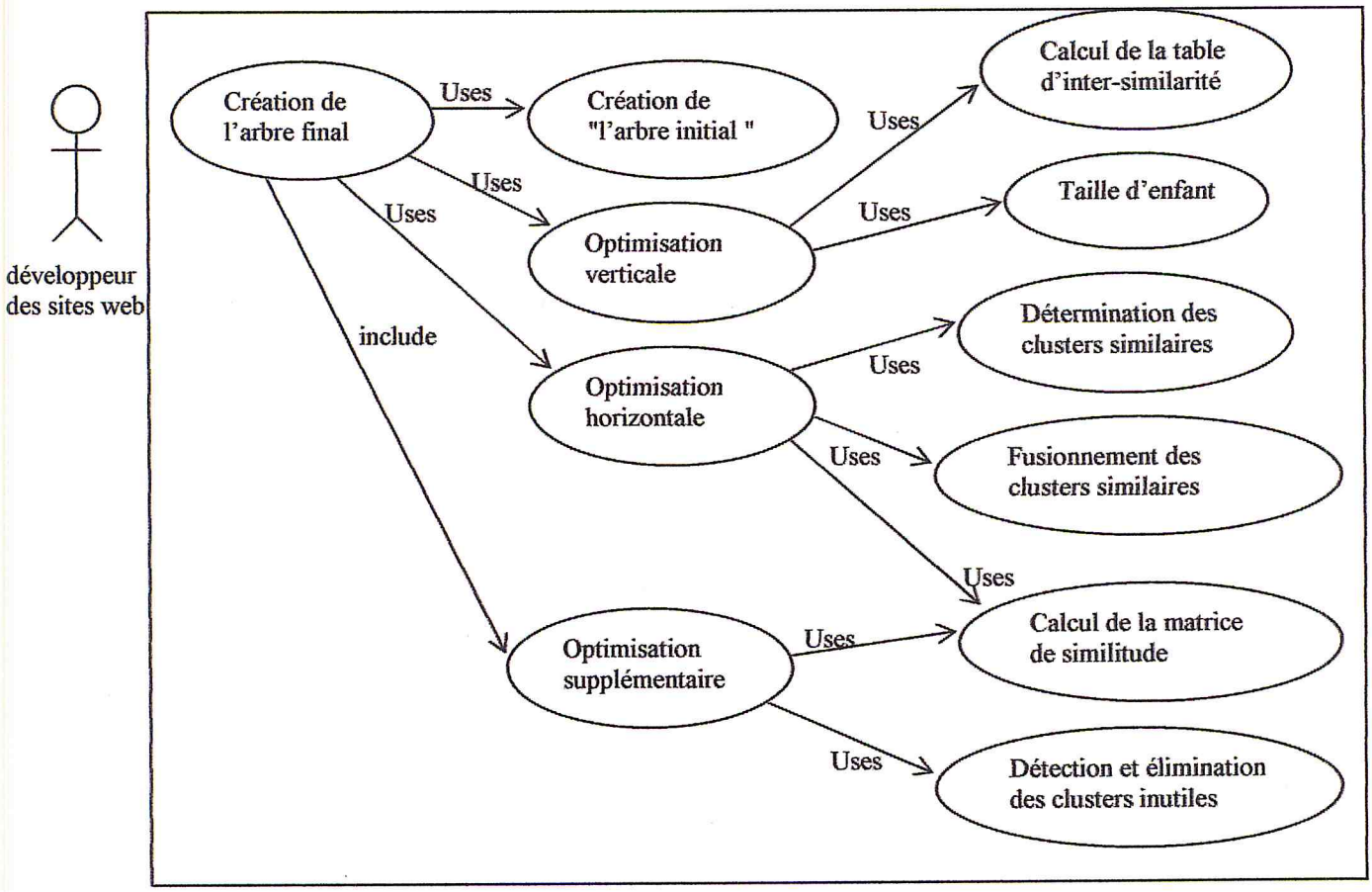


Figure V.8 : Cas d'utilisation « Création de " l'arbre final " ».

3.1.8 Cas d'utilisation « Création de " l'arbre initial " »

La « Création de " l'arbre initial " » utilise cinq cas :

- tri de la liste des "clusters" (d'après les " clusters labels").
- création des groupes de clusters.
- affectation père -fils.
- nettoyage.
- création de la liste " cluster disjoint".

Le dernier cas utilise trois autres cas :

- création de la liste "initiale clusters " qui utilise le cas « Calcul de la table " cluster support ".
- calcul de la table score.
- affectation fichier - cluster où les fichiers sont attribués aux meilleurs clusters.

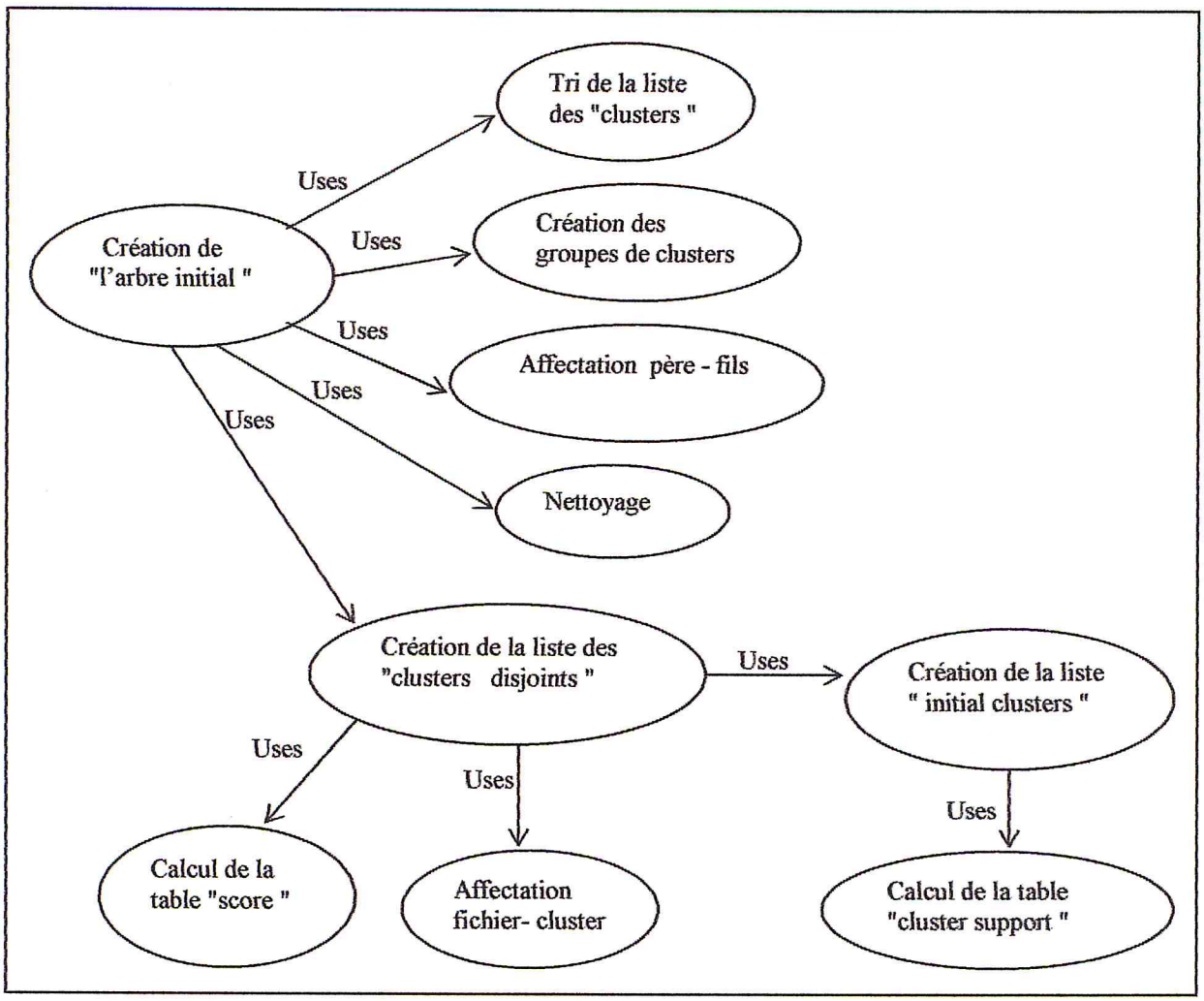


Figure V.9 : Cas d'utilisation « Création de " l'arbre initial" »

3.2 Diagrammes d'interaction

Un diagramme d'interaction permet de décrire le dialogue qui s'établit entre l'utilisateur et le système. Les cas d'utilisation de UML ont certes l'avantage d'être graphiquement très simples et donc faciles à appréhender. Malheureusement, cette simplicité ne va pas sans une certaine pauvreté sémantique [MUL 97], cependant les diagrammes de l'interaction (spécialement diagrammes de séquence) permettent de bien schématiser les scénarios des cas d'utilisation .

3.2.1 Gestion des entrées

1) Ajout d'un dossiers ou fichier (exemple "stop word")

Le scénario :

- → l'expert ajoute un nouveau fichier.
- → le système lui affiche la boîte d'ajout.
- → l'expert sélectionne le fichier.

Le diagramme de séquence suivant exprime le cas d'utilisation « Ajouter fichiers ».

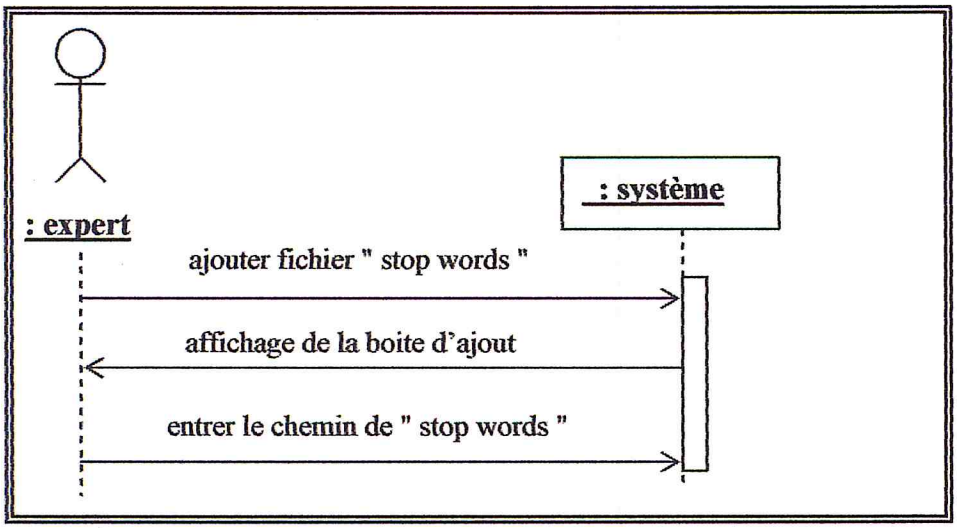


Figure V.10 : Le diagramme de séquence « Ajouter fichiers ».

2) gestion des paramètres supplémentaires

Le scénario :

- → l'expert entre le min-support.
- → l'expert entre le cluster-support.
- → l'expert coche la case d'optimisation.
- → le système active la case "nombre de clusters ".
- → l'expert entre le nombre de clusters désirés au niveau 1 de l'arbre de clusters.

Le diagramme de séquence suivant exprime le cas d'utilisation « gestion des paramètres supplémentaires ».

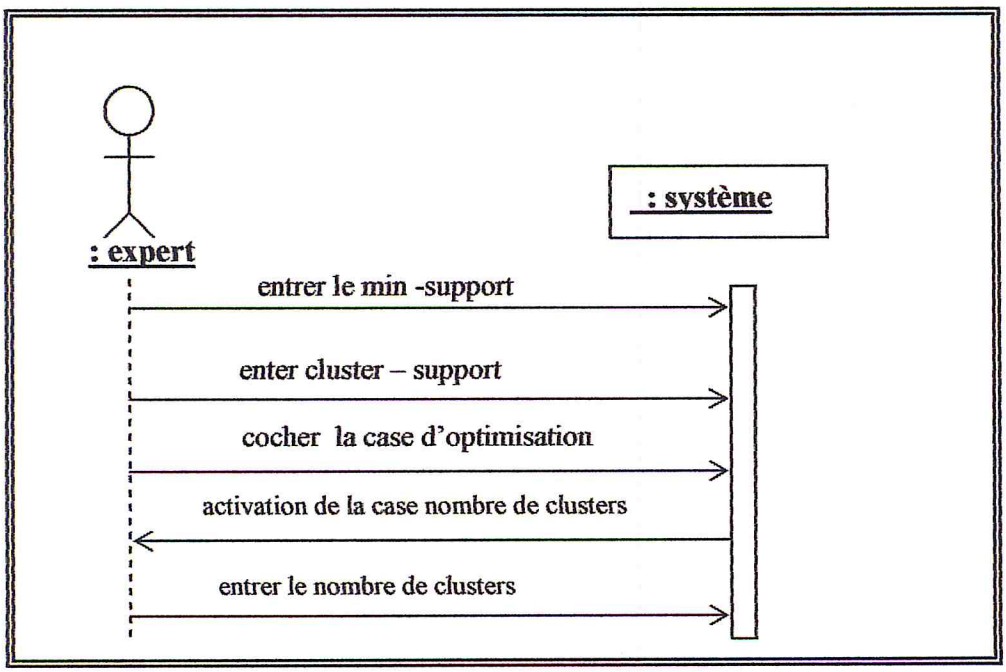


Figure V.11 : Le diagramme de séquence « gestion des paramètres supplémentaires ».

4. L'analyse du domaine

Lors de la phase d'analyse, également appelée phase de spécification *("requirements phase"," analysis phase"," definition phase")*, on analyse les besoins de l'utilisateur ou du système global et on définit ce que le logiciel devra faire [BOU 01]. L'objectif de cette phase est de déterminer les éléments intervenant dans le système à construire, ainsi que leur relation.

4.1 Propositions des solutions d'optimisation de la méthode FIHC

Avant de présenter les solutions d'optimisation, nous faisons quelque précisons [JAC 03]

☒ Processus et thread

Un processus est un flot « lourd » qui peut s'exécuter en concurrence avec d'autre processus, un « thread » est un flot « léger » qui peut s'exécuter en concurrence avec d'autres threads à l'intérieur de même processus.

Un processus est lourd dans le sens où il s'agit d'une chose connue du système d'exploitation qui s'exécute dans un espace d'adressage indépendant. Les processus ne sont jamais emboîtés les uns dans les autres.

Un thread est léger et peut être connu du système d'exploitation lui- même. La plupart du temps, il est caché dans un processus plus lourd et s'exécute à l'intérieur de l'espace d'adressage du processus englobant.

☒ Flot de contrôle

Dans un système séquentiel, il n'y a qu'un seul flot d'exécution, tandis que dans un système concurrent, il y a plusieurs flots de contrôle multiples, simultanés, existent chacun à la tête d'un processus indépendant ou d'un thread. Logiquement, si l'on prend un cliché d'un système conçurent, on voit des points d'exécution multiple.

☒ Un objet actif ()

Est un objet qui possède un processus ou un thread et qui peut déclencher une activité de contrôle .En UML un objet actif est un instance d'une classe active.

Nous allons dans ce qui suit présenté quelques solution.

Le facteur le plus important dans les méthodes de clustering est le facteur temps, car les méthodes classiques de clustering sont lentes surtout lorsque le nombre de documents est très important, ou la valeur des seuils est très petite. Donc nous avons choisi de développer les fonctionnalités principales qui sont décrites dans les « Use case » utilisant des structures qui permettent de minimiser le facteur temps , de maximiser l'exactitude des groupes(clusters)et de donner à l'utilisateur une vue globale des étapes de traitement des documents.

La question qui se pose est : comment l'utilisateur peut savoir l'avancement des étapes du processus d'analyse des documents ? Quelle est la durée de chacune sans interruption de traitement ?

Pour résoudre ce problème nous avons remédié au modèle « **producteur – consommateur**[23] » dans lequel nous utilisons la notion de « thread » où deux processus s'exécutent en parallèle.

Ces deux « thread » sont le processus **système** et le processus **analyse.** Les deux processus se communiquent à travers une file d'attente (section critique) dans la quel le processus" **analyse** " dépose les messages de chaque étape de traitement des documents, tandis que le processus " **système** " scrute cette file toutes les 5 millisecondes pour récupérer le message de l'entête afin de l'afficher dans l'interface de l'utilisateur. A travers ce modèle nous pouvons savoir le temps d'exécution de chaque étape de processus analyse sans interruption et exécution de programme d'interruption, en conséquence une amélioration de temps d'exécution de la méthode de clustering FIHC.

Un autre problème se pose : comment réaliser la structure de " **Document set** " ? Comme nous l'avons présenté dans le chapitre IV, la méthode FIHC utilise la structure " **features vectors** " (à la place "document vector") qui sont regroupés dans une structure " **Document set".** Cette structure regroupe à la fois l'ensemble de documents traités, les 1- itemset trouvés ainsi que leurs fréquences. Donc, nous avons utilisé deux structures :

- *Résume - mot* : regroupe l'ensemble des mots trouvés dans tous les fichiers avec leur fréquences.
- *Résume – fichier* : collecte l'ensemble de *Résume - mot* de chaque document.

A la fin, le résultat est une arborescence structurée dans un fichier XML exploitable pour un usage postérieur.

4.2 Les composants de système global

1) Les paquetages

D' après le cas d'utilisation de « système global » démontré dans la section précédente nous allons utilisé plusieurs classes organisés en paquetages pour le bon fonctionnement des différente étapes de notre application .Ces composant sont partagés en deux paquetages : le paquetage de processus « système » et le paquetage de thread « analyse ». La figure suivante montre le paquetage « système ».

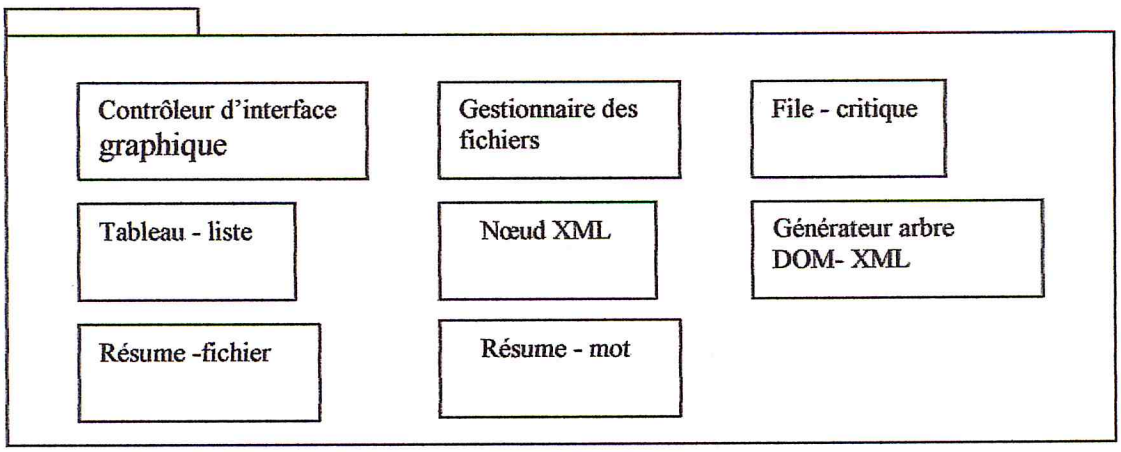


Figure V.12 : Le paquetage « système ».

[23] : Deux transactions (processus) T_1 et T_2 lancées en parallèle pouvant consulter et modifier simultanément un fichier commun F, T_1 effectue un dépôt et T_2 effectue un retrait.

Le paquetage « système » représente les outils utilisés par le processus « analyse » pour suivre les étapes de la méthode FIHC. La figure suivante montre le paquetage « analyse».

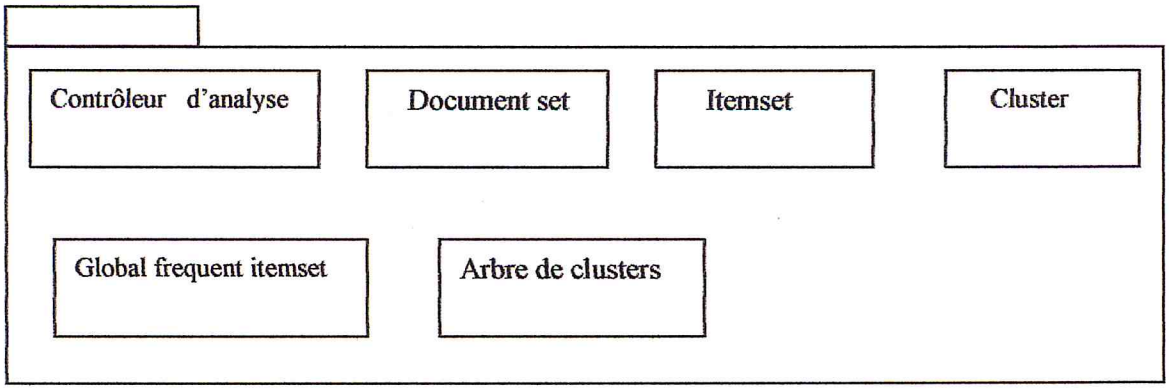


Figure V.13 : Le paquetage « analyse ».

Le paquetage « analyse » contient les classes qui représentent les concepts de base utiliser dans la méthode FIHC.

2) Diagramme de classes

Un diagramme de classes est une collection d'éléments de modélisation statiques (classes, paquetage ...), qui montre la structure d'un système.

Dans un diagramme de classes, on peut se focaliser sur :

- les classes qui participent à un cas d'utilisation ;
- les classes associées dans la réalisation d'un scénario précis ;
- les classes qui composent un paquetage.

Le diagramme de classe décrit dans la figure ci-dessus représente les éléments et les outils qui doivent être implémentés d'une manière synthétique.

Le diagramme comprend les classes des deux paquetages, les classes spécifique à la méthode FIHC qui sont : Global_frequent_itemset, DocumentSet, arbre de cluster, cluster, itemset. Les autres classes sont : Contrôleur d'analyse, File critique (une file où sont sauvegarder les messages), gestionnaire des fichiers, générateur arbre DOM-XML, Nœud XML, contrôleur d'analyse, interface graphique et la classe Tableau_liste dont toutes les autres classes héritent.

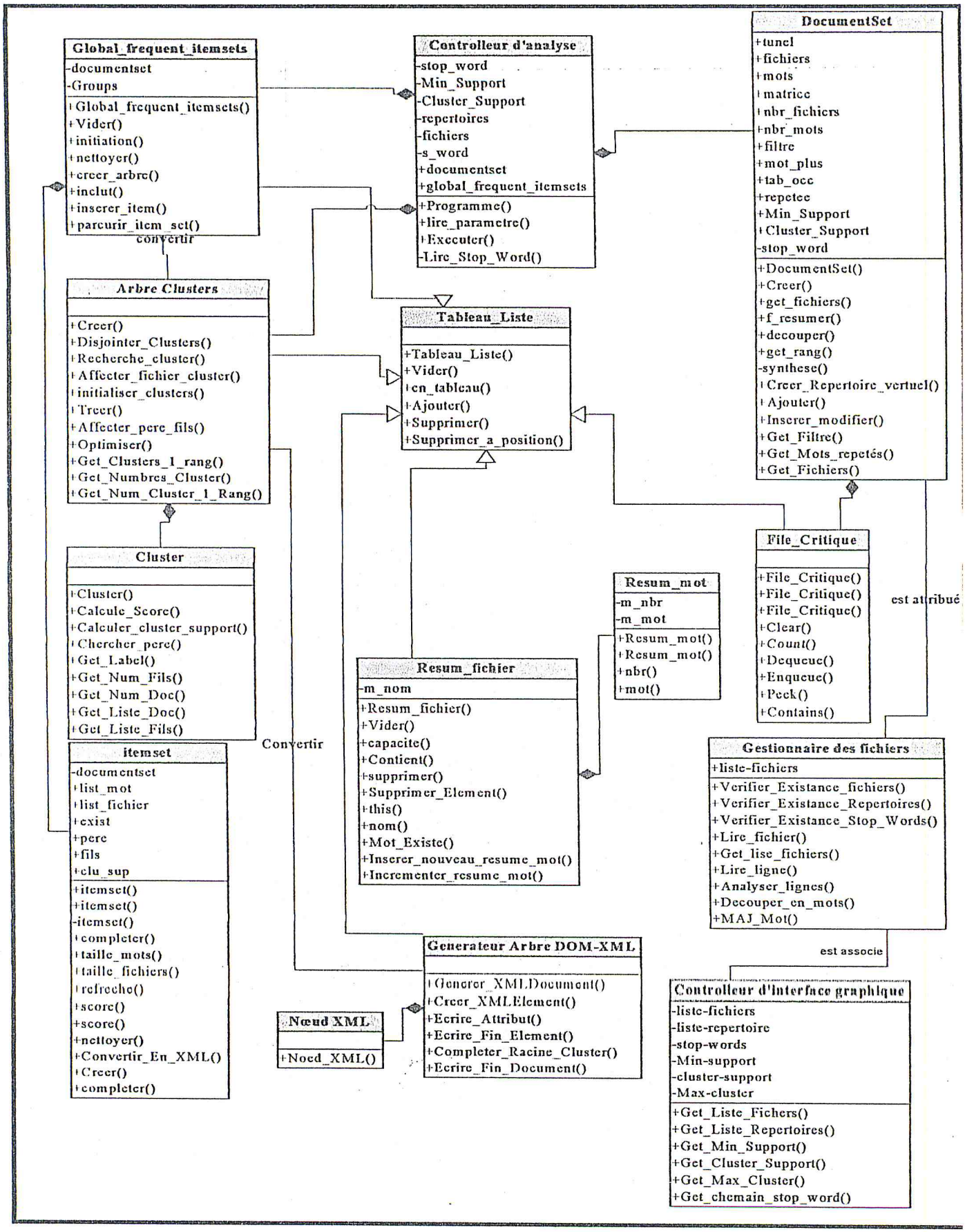


Figure V.14 : Diagramme de classes de système global.

5. La conception

La conception s'intéresse d'abord au « comment », à savoir la solution du problème énoncé. Elle commence par une conception dite « globale » qui décrit l'architecture du système, elle se poursuit par une conception détaillée.

5.1 Conception globale

La conception générale consiste en une description de l'architecture du système, c'est à dire obtenir à la fois une décomposition du système en un ensemble de modules précisant les interfaces et les fonctions de chaque module en interaction avec les autres et les structures de données utilisées. Dans la figure suivante nous présentons l'architecture de notre système global où sont illustrés les différents modules.

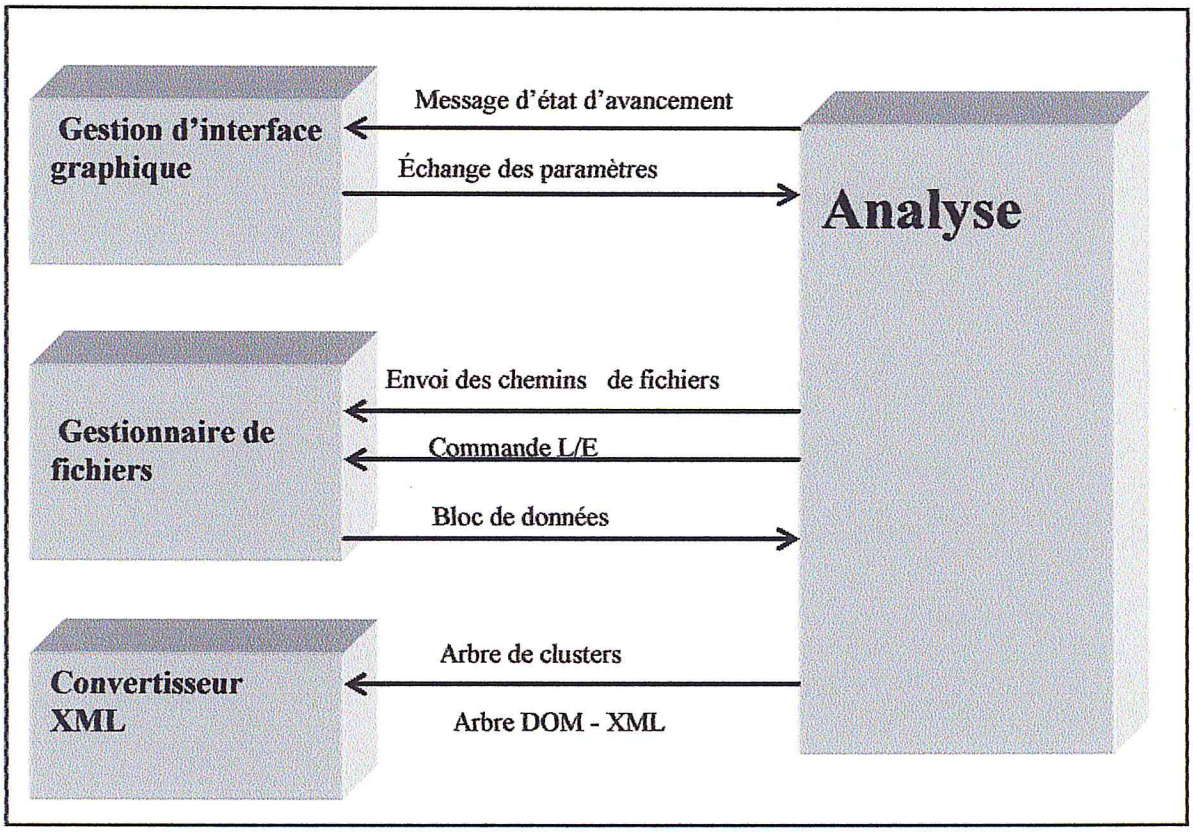


Figure V.15 : Architecture de système global.

Dans cette architecture, nous avons défini les grand modules de notre système Le module cet ensemble de modules assurent la communication, en permettant d'envoyer les donnée nécessaire et de récupérer les informations de fonctionnements.

Afin de mettre en œuvre notre système, nous avons défini une architecture logicielle composée de quatre modules :

- Module 1 : Gestion d'interface graphique.
- Module 2 : Gestionnaire de fichiers.
- Module 3 : convertisseur XML.
- Module 4 : Analyse.

A travers le module de gestion de l'interface graphique les différents paramètres sont entrés (documents, min-support, cluster support, nbr- clusters) et vont être transmis vers le module analyse ; c'est à ce niveau que les documents sont traités et regroupés dans des clusters.

L'état courant du processus analyse (étapes de la méthode de clustering FIHC) est envoyé vers le module gestion de l'interface graphique a fin d'informer l'utilisateur de la date de début et la date fin de chaque étape.

Le module analyse travaille en coopération avec le module gestionnaire de fichiers en envoyant les chemins des fichiers entrées ainsi que les commandes d'entrée /sortie à établir. Le module gestionnaire de fichiers reçoit les chemins de fichiers avec la commande E/S qui lui permet de localiser les données demandées.

A la fin du traitement, un arbre de clusters est construit. Cet arbre est transmis vers le module convertisseur XML où il est converti en un arbre DOM - XML , qui regroupe un ensemble de nœud XML (élément XML, pour plus d'information voir l'Annexe D). Ce flux XML est acheminé vers le module gestionnaire de fichiers afin de donner un fichier XML.

5.2 Conception détaillée

La conception détaillée fournit pour chaque module une description détaillée de la manière dont les fonctions du composants sont réalisées : algorithme, représentation du système [BOU 01]. La principale fonction dans notre application est " analyse", c'est pourquoi nous s'intéressons en premier à ce module.

5.2.1 Le module Analyse

C'est le module de traitement qui coopère avec les autres modules en échangeant les différentes données, messages et commandes.

5.2.1.1 les diagrammes d'activités

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités. Un diagramme d'activités est essentiellement un organigramme qui met en évidence l'activité qui a lieu dans le temps.

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles.

- Le passage d'une activité vers une autre est matérialisé par une transition.
- Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

1) diagramme d'activité « analyse »

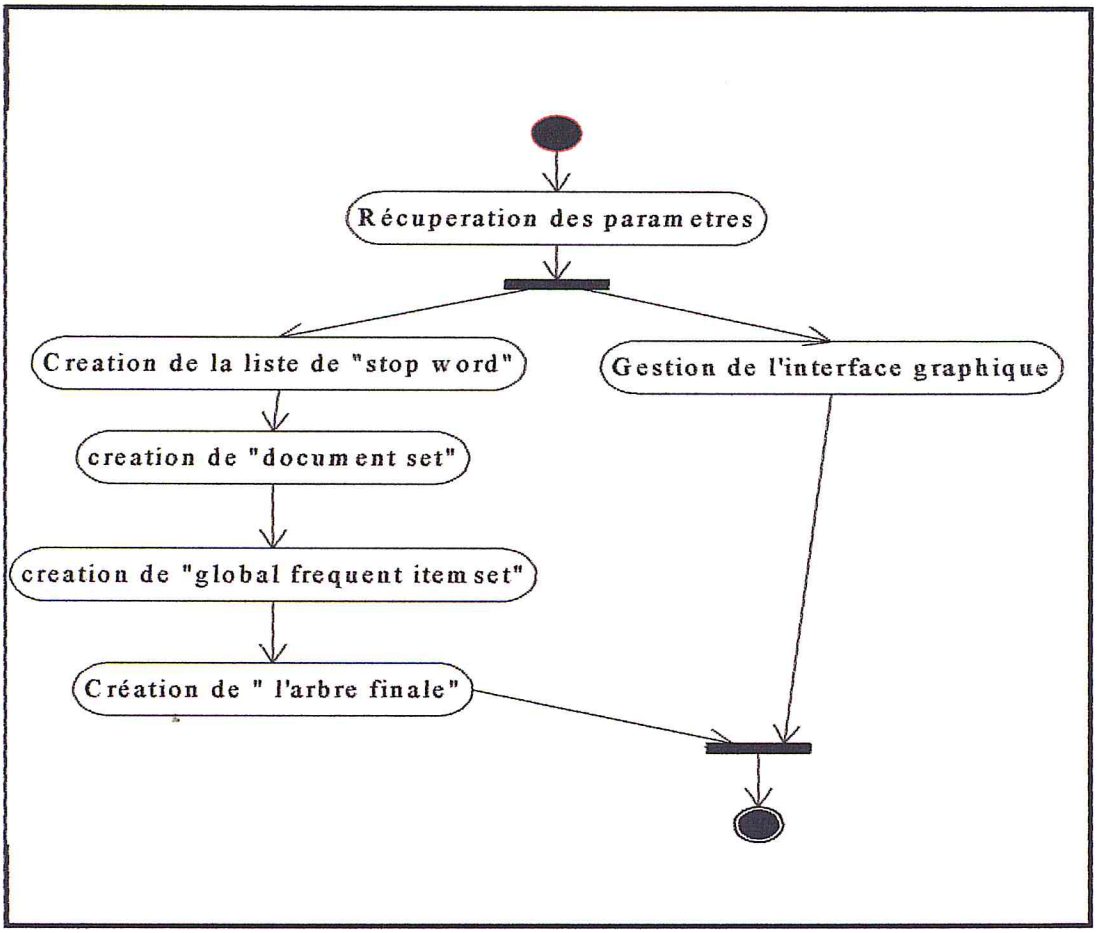


Figure V.16 : Le diagramme d'activité « analyse ».

Le diagramme d'activité ci -dessus décrit le déroulement du cas d'utilisation « analyse ».au début de diagramme et à la fin , on trouve deux barres en haut est appelée fourche concurrente, et en bas est appelée jonction concurrente . En UML, on utilise cette barre de synchronisation pour spécifier la division et le regroupement des flots de contrôle parallèle, généralement une barre de synchronisation est représentée par une ligne épaisse horizontale et verticale.

Ce diagramme illustre que le cas d'utilisation gestion de l'interface graphique s'exécute en parallèle avec « création de stop words, de document set, de global frequent itemset et de l'arbre final », c'est - à- dire que l'utilisateur peut utilisé l'interface graphique pendant que le processus analyse s'exécute.

2) Diagramme d'activité « création de la liste des " stop words "»

Ce diagramme décrit le cas d'utilisation « création de la liste des stop words »

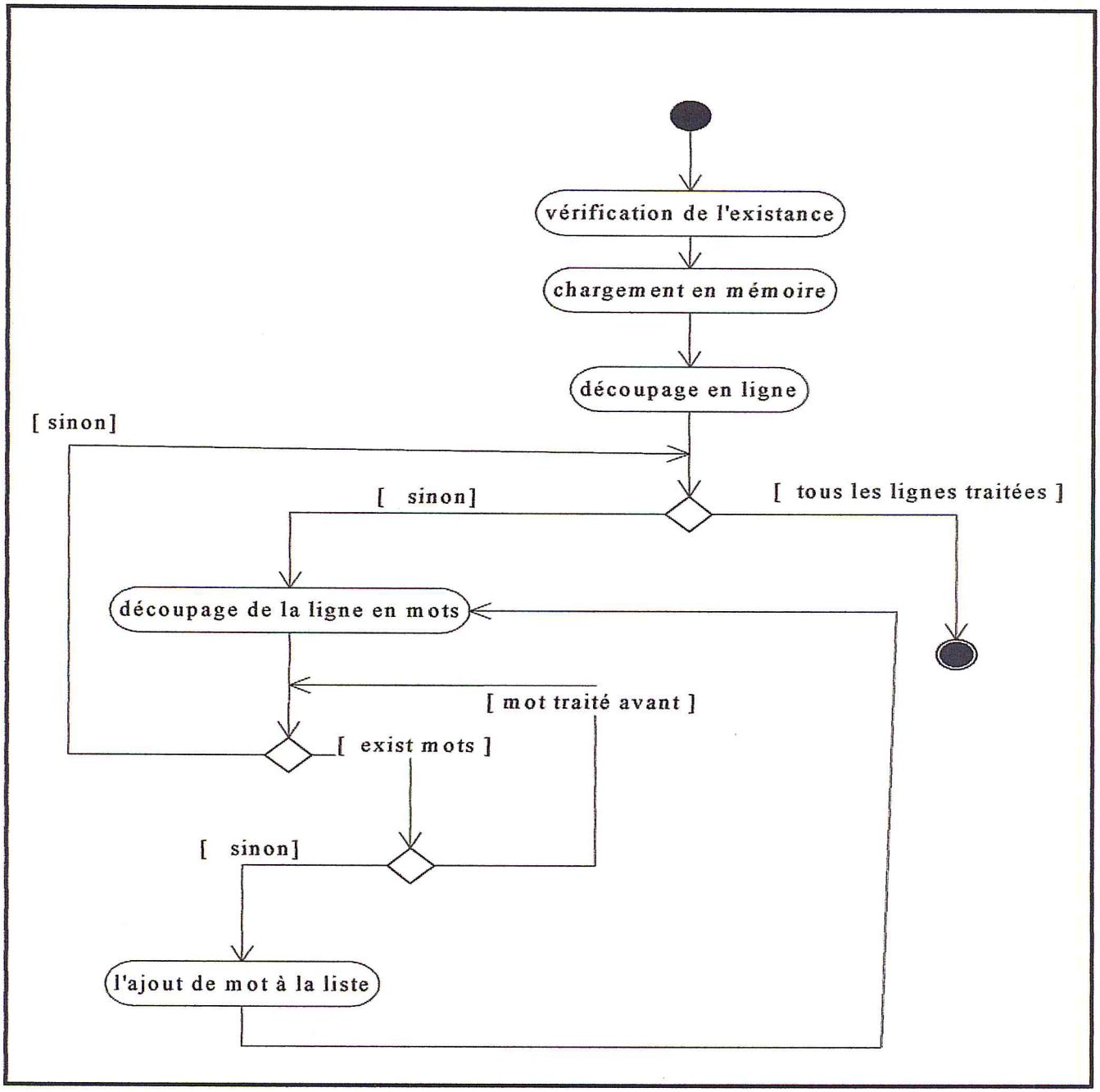


Figure V.17 : Diagramme d'activité « création de la liste des « stop word « ».

3) Diagramme d'activité de « création de Document set »

Le diagramme d'activité suivant décrit le déroulement du cas d'utilisation « création de document set ».Il représente son algorithme.

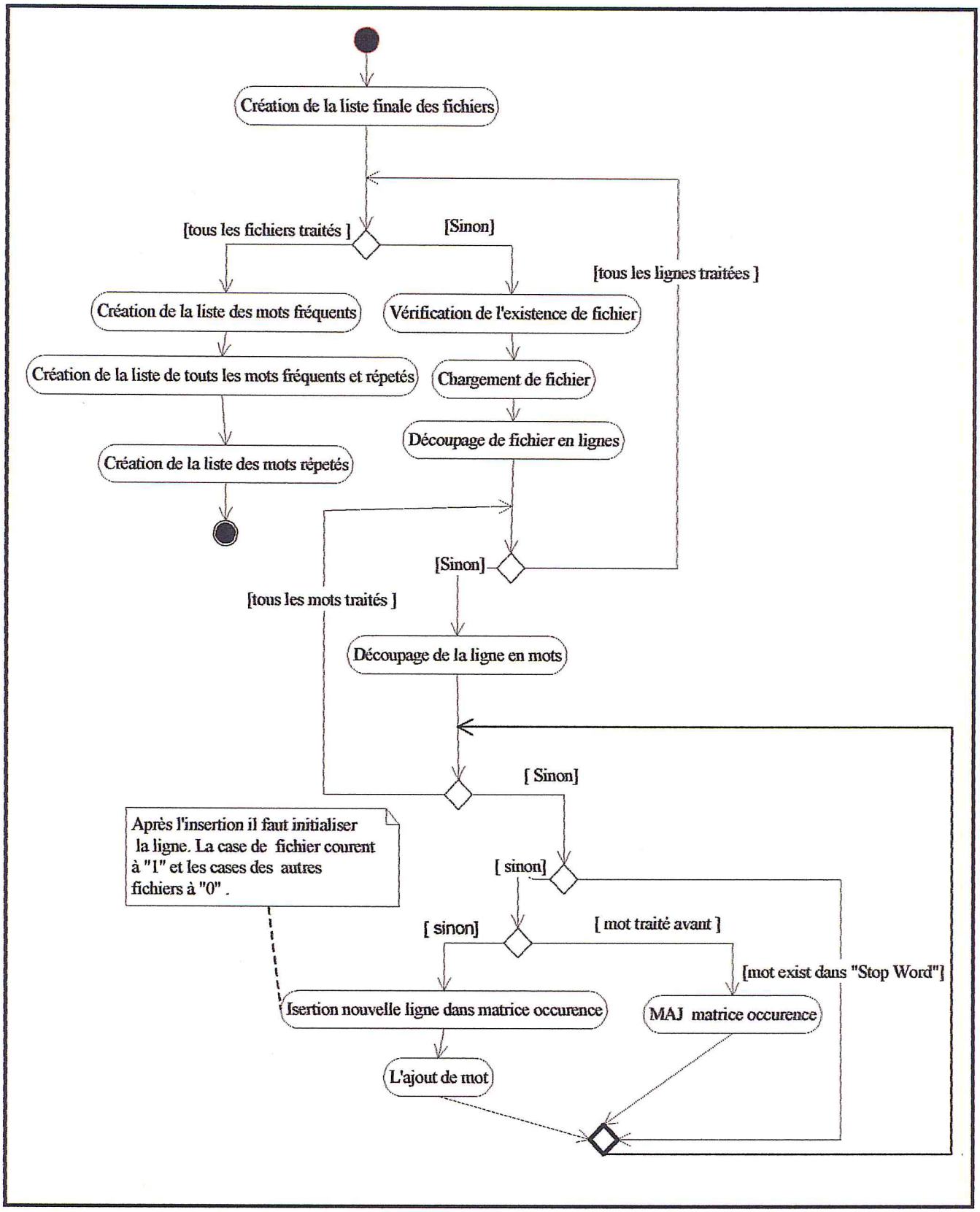


Figure V.18 : Diagramme d'activité de « création de Document set ».

4) Diagramme d'activité de « création de « global fréquent itemset » »

Le diagramme suivant représente le cas d'utilisation « création de "global fréquent itemset" » ainsi que son algorithme.

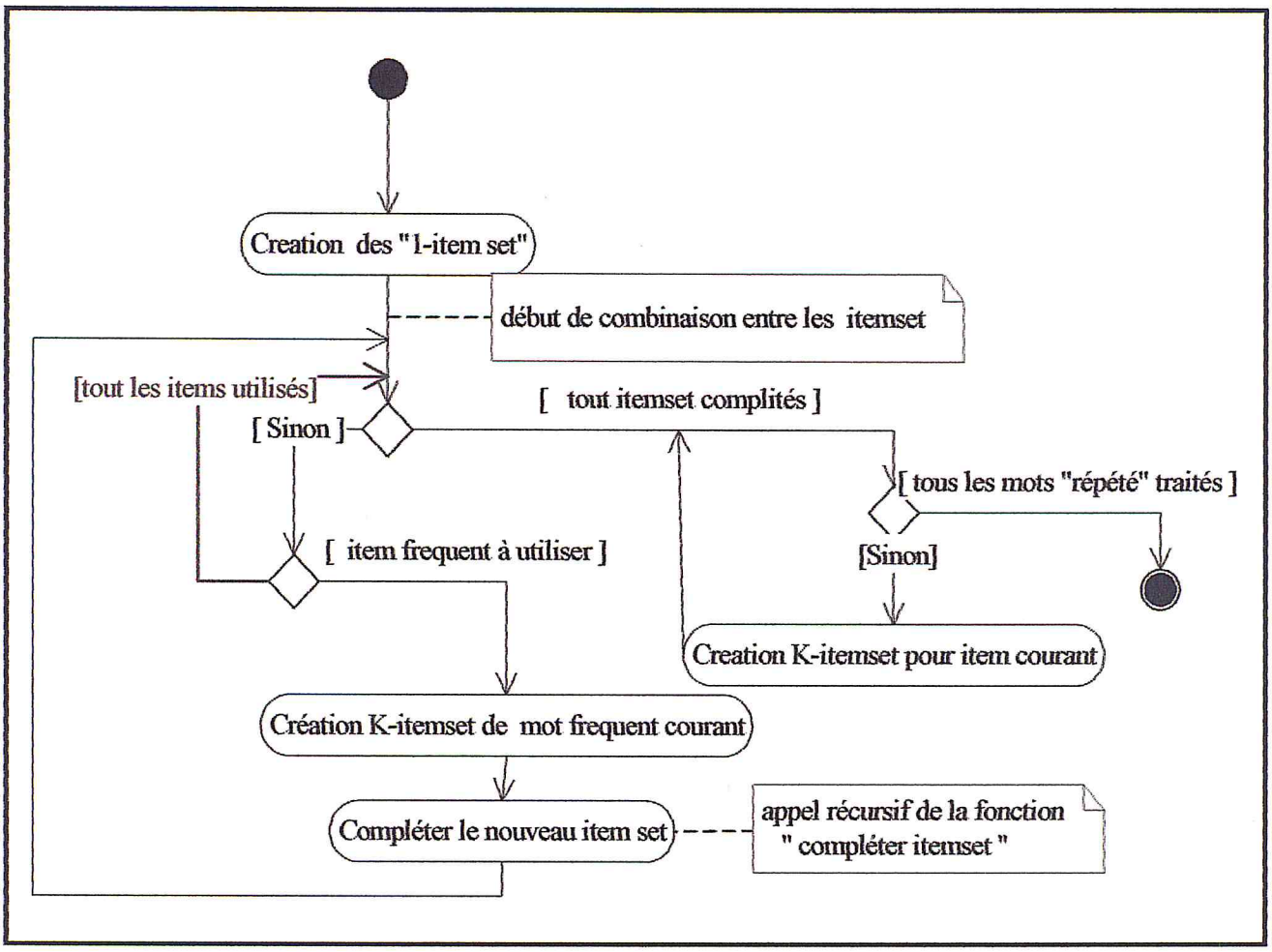


Figure V.19 : Diagramme d'activité « création de « global fréquent itemset « ».

Le diagramme de « création de "global frequent itemset "» décrit l'algorithme de création des 1-itemset .ensuite la combinaison de chaque item pour créer les K-itemset et ainsi de suite pour chaque nouveau item ceci est établie par un appel récursif de la fonction « compléter item » afin de donner tous les combinaisons possible des itemset.

5) Diagramme d'activité de « Création de "l'arbre initial "»

Le cas d'utilisation « analyse » utilise le cas « création de l'arbre final »qui utilise lui même le cas « création de l'arbre initial » dont nous définissons l'algorithme. Cet algorithme donne un arbre avec un nombre de niveau égal au nombre de mots qui existent dans le plus grand "global frequent itemset ". En d'autres termes, le niveau 1 comprend les clusters avec "cluster label" d'un seul mot, le niveau 2comprend les clusters avec "cluster label" deux mot et ainsi de suite jusqu'aux feuilles.

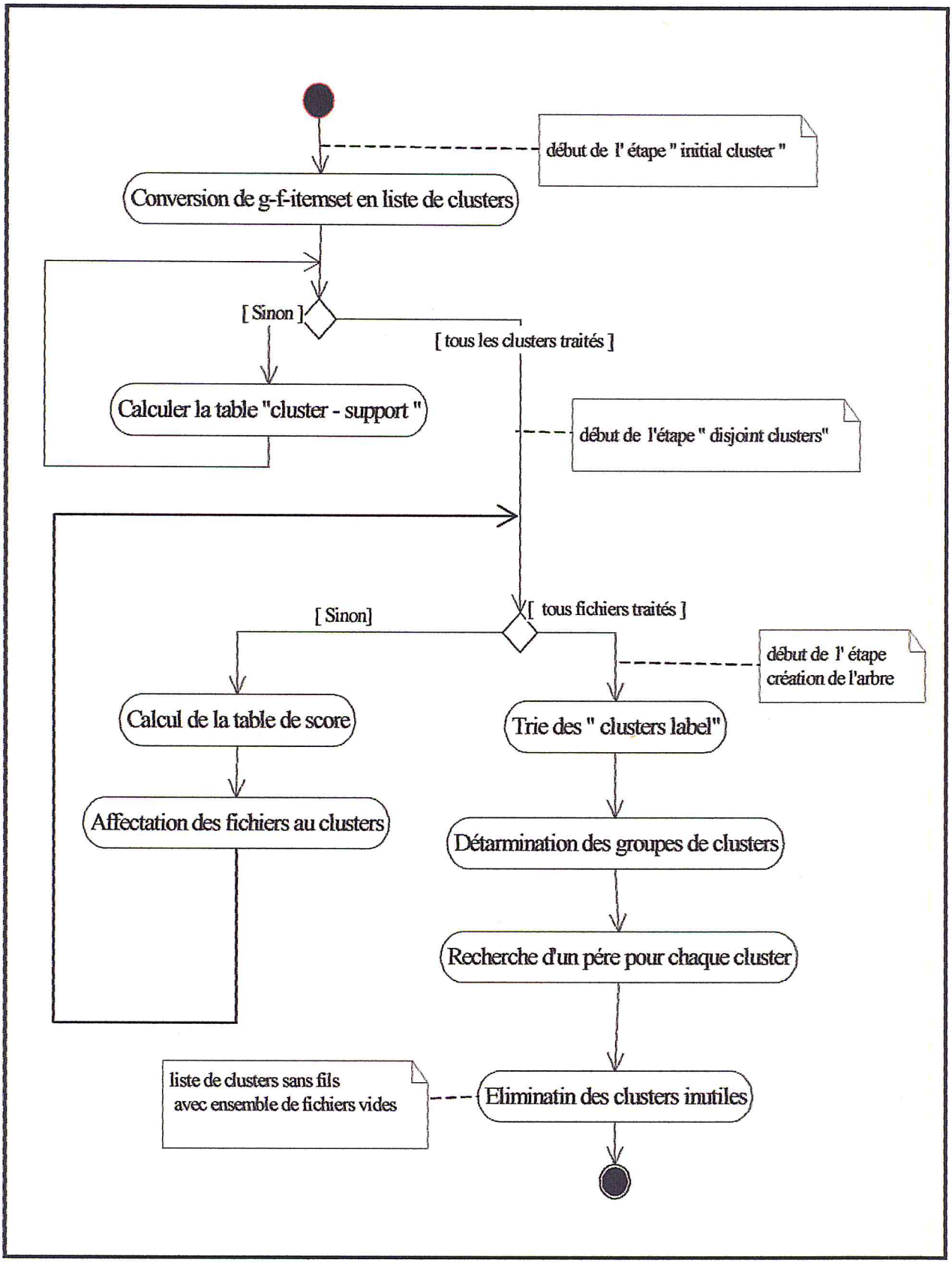


Figure V.20 : Diagramme d'activité « création de " l'arbre initial "».

6) Diagramme d'activité de « Création de "l'arbre final "»

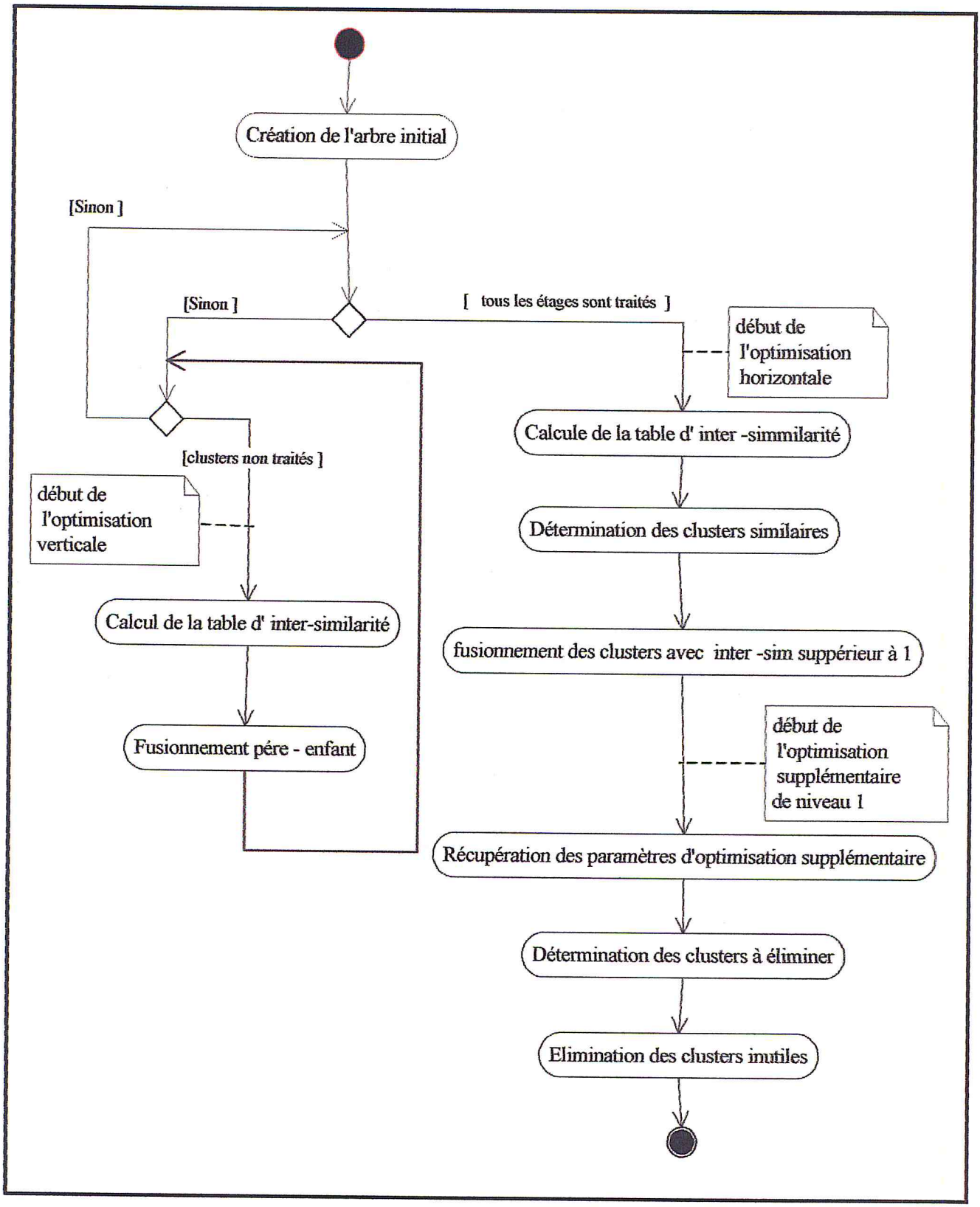


Figure V.21 : Diagramme d'activité « création de " l'arbre final "».

Le diagramme de « création de " l'arbre final " » illustre l'algorithme d'optimisation de l'arbre de clusters. La première technique est l'optimisation verticale ou, comme nous l'avons définie dans le chapitre IV " Taille d'enfant ". Elle

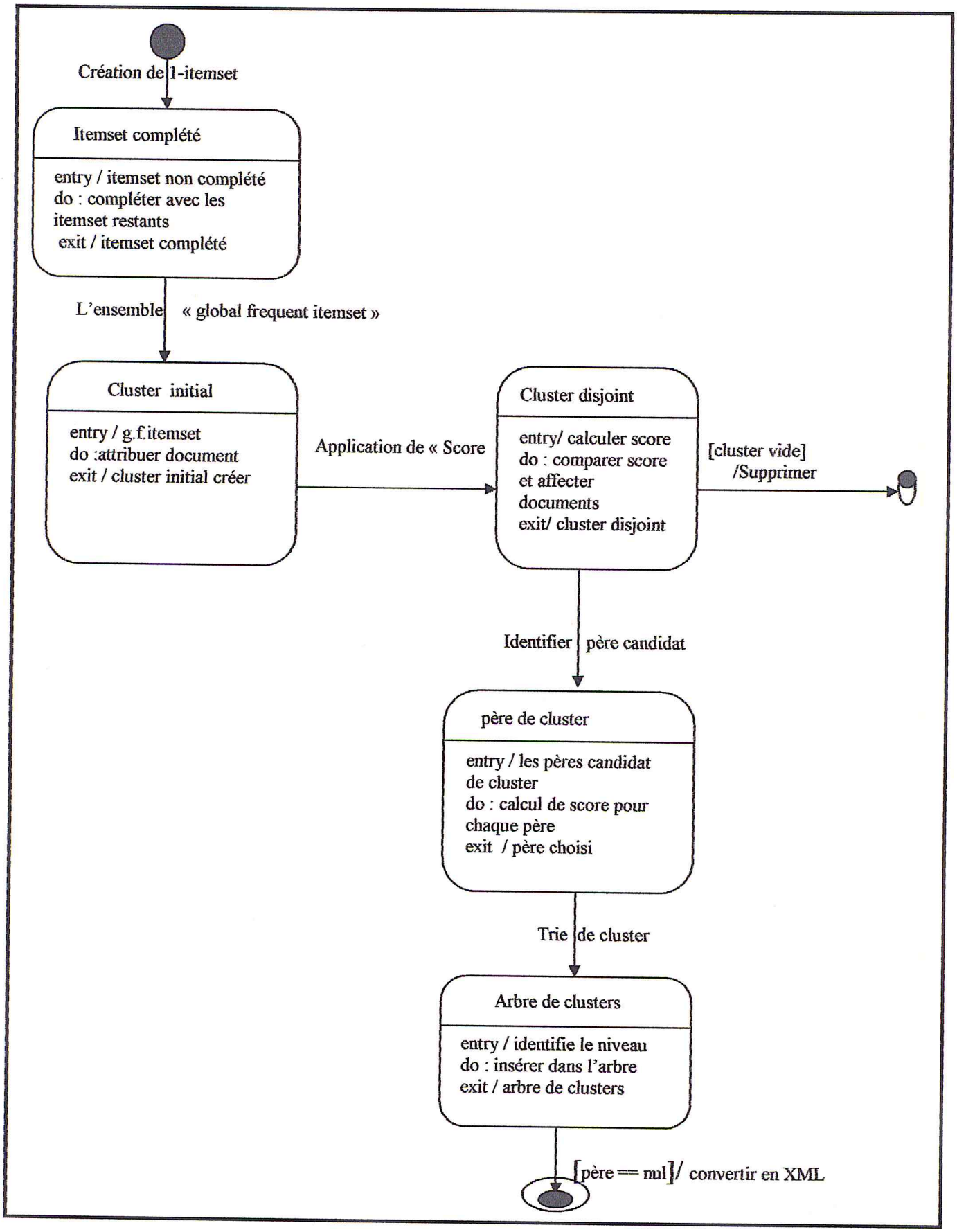


Figure V.22 : Le diagramme d'état de la classe « global frequent itemset ».

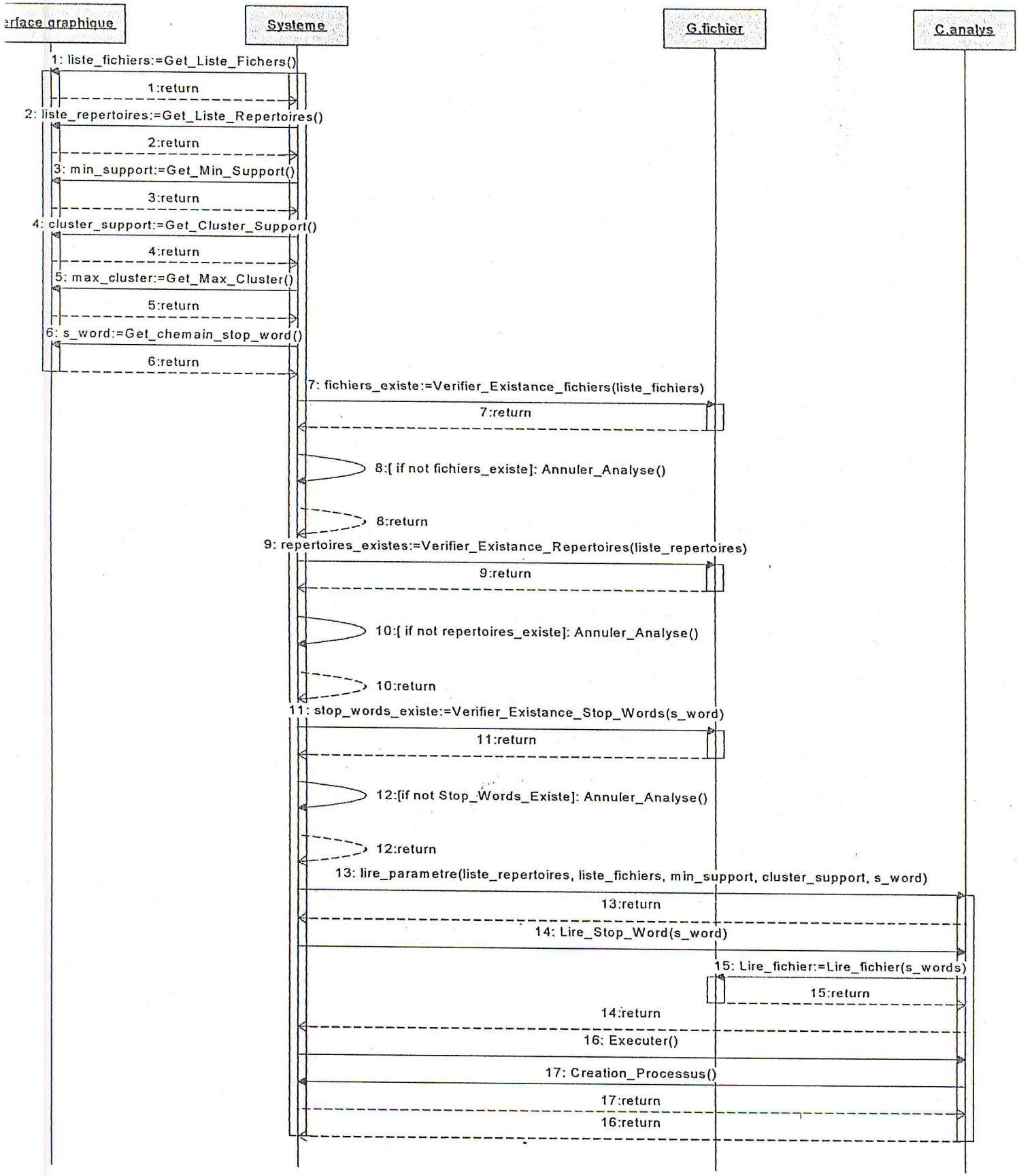


Figure V.23 : Diagramme de séquence de « création et lancement de processus analyse ».

permet de minimiser la profondeur de l'arbre en regroupant les parents avec leur fils lorsque la valeur d'inter- similarité est au-dessus de 1 .La deuxième technique est l'optimisation horizontale ou" le fusionnement d'enfants de même parent ", elle permet de réduire la largeur de l'arbre en fusionnant les clusters fils lorsque la valeur d'inter -similarité est supérieure à 1.

☒ Cette technique concerne seulement le niveau 1 de l'arbre, car l'application pour touts les niveaux est très coûteuse [FUN 03]. La troisième technique est l'optimisation supplémentaire. Cette technique est facultative, elle est validée par l'utilisateur, autrement dit c'est le nombre de clusters de niveau 1 de l'arbre désirés par l'utilisateur. Si elle est validée l'inter - similarité est calculée entre toutes les paires de niveau1 (avec une valeur évidemment inférieur à 1). Les clusters fusionnés sont les clusters qui ont la plus grande valeur, cette fonction est répétée jusqu'à l'atteinte du nombre de clusters désirés par l'utilisateur.

5.2.1.2 Les diagrammes d'états –transitions

Ce diagramme sert à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions. La portée du diagramme de séquence est limitée à un scénario. La portée du diagramme d'états s'étend à la vie entière de l'objet [PEN 02].

Il faut donc souvent plusieurs diagrammes de séquence pour construire un diagramme d'état.

Ce diagramme a pour but de présenter l'enchaînement des différents états que peut prendre une classe, à la suite de traitement particulier. Un diagramme d'état – transition concerne donc une classe particulière en présentant tous les états possibles que peut prendre cette classe et les événements(traitements) qui provoquent un changement d'état de la classe [BOU 01].

- Un état se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des valeurs des attributs d'un objet.
- Une transition représente le passage instantané d'un état vers un autre.

Nous utiliserons ce type de diagramme pour représenter les différents états que peuvent prendre la classe « global frequent itemset ».

Les états décrits ci-dessous sont les états pris par la classe « global frequent itemset ».

- **itemset complété :** cet état correspond à la création des K-itemset à partir de 1-itemset pour créer ensemble "global frequent itemset ".
- **cluster initial :** cet état décrit la phase de création des clusters initiaux.
- **clusters disjoint** : après l'application de score pour les documents.
- **père de cluster :** pour chaque cluster disjoint sont attribués des pères candidats. L'application de la fonction Score pour un cluster permet de choisir un.
- **arbre de cluster :** cet état permet de générer l'arbre de clusters suite à une organisation des clusters en niveau.

La figure suivante illustre ces états :

2) Diagramme de séquence de « **création et synthèse de document set** »

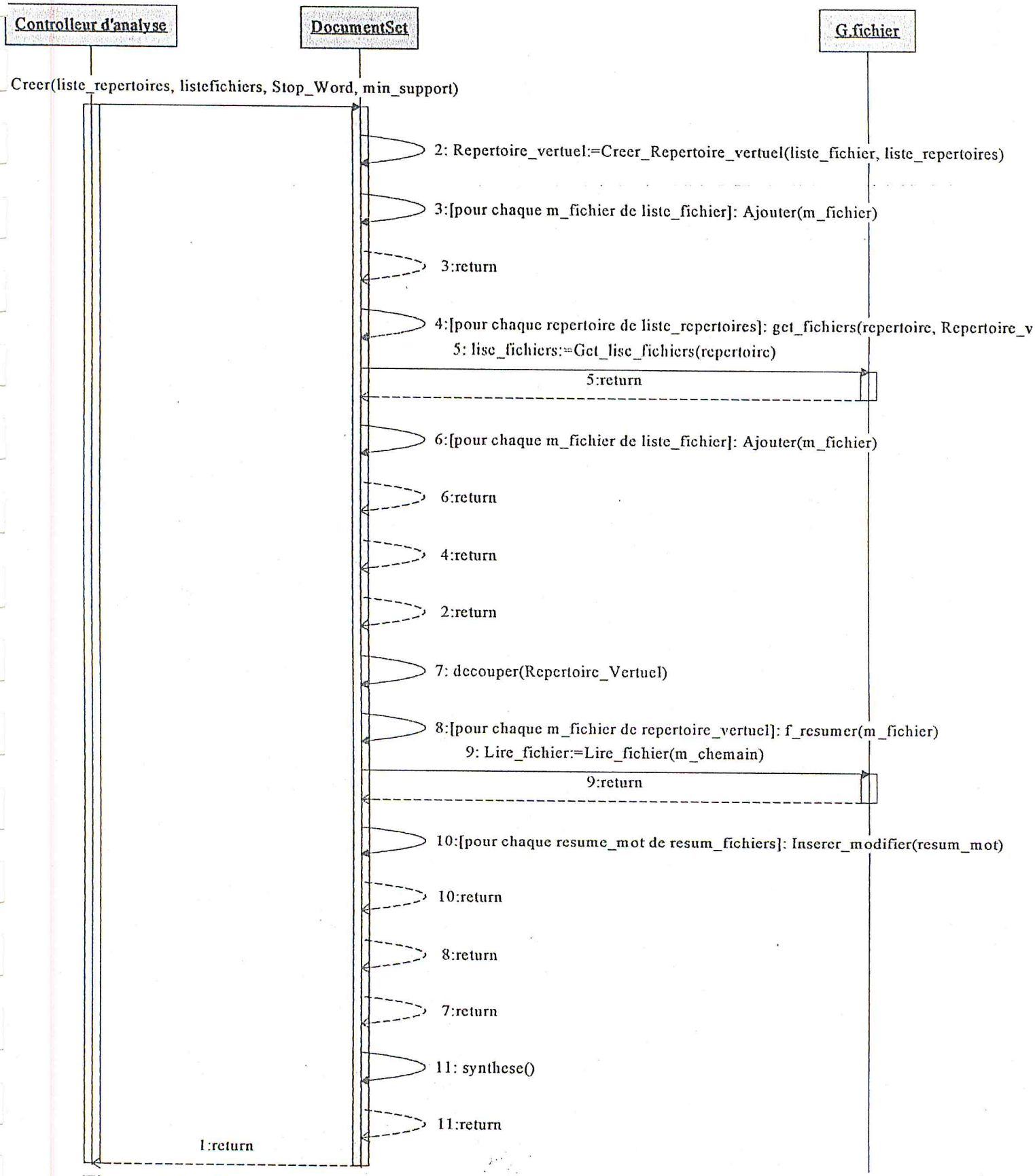


Figure V.24 : Diagramme de séquence de « création et lancement de processus analyse ».

Le diagramme ci- dessus présente la création de document set.

- L'objet document set permet de créer un répertoire virtuel où sont regroupés tous les fichiers et les dossiers dans le but d'établir la liste totale des fichiers à travers les dossiers et les sous-dossiers existant.
- Chaque fichier de répertoire virtuel est lu par le gestionnaire de fichiers auquel est associe un résume –mot (une structure où les mots et le nombre d'occurrence de chaque mot sont regroupés).
- Après le découpage de chaque fichier, une fonction synthèse est appliquée pour trouver les mots fréquents.

3) Diagramme de séquence « Création de " global frequent itemset " »

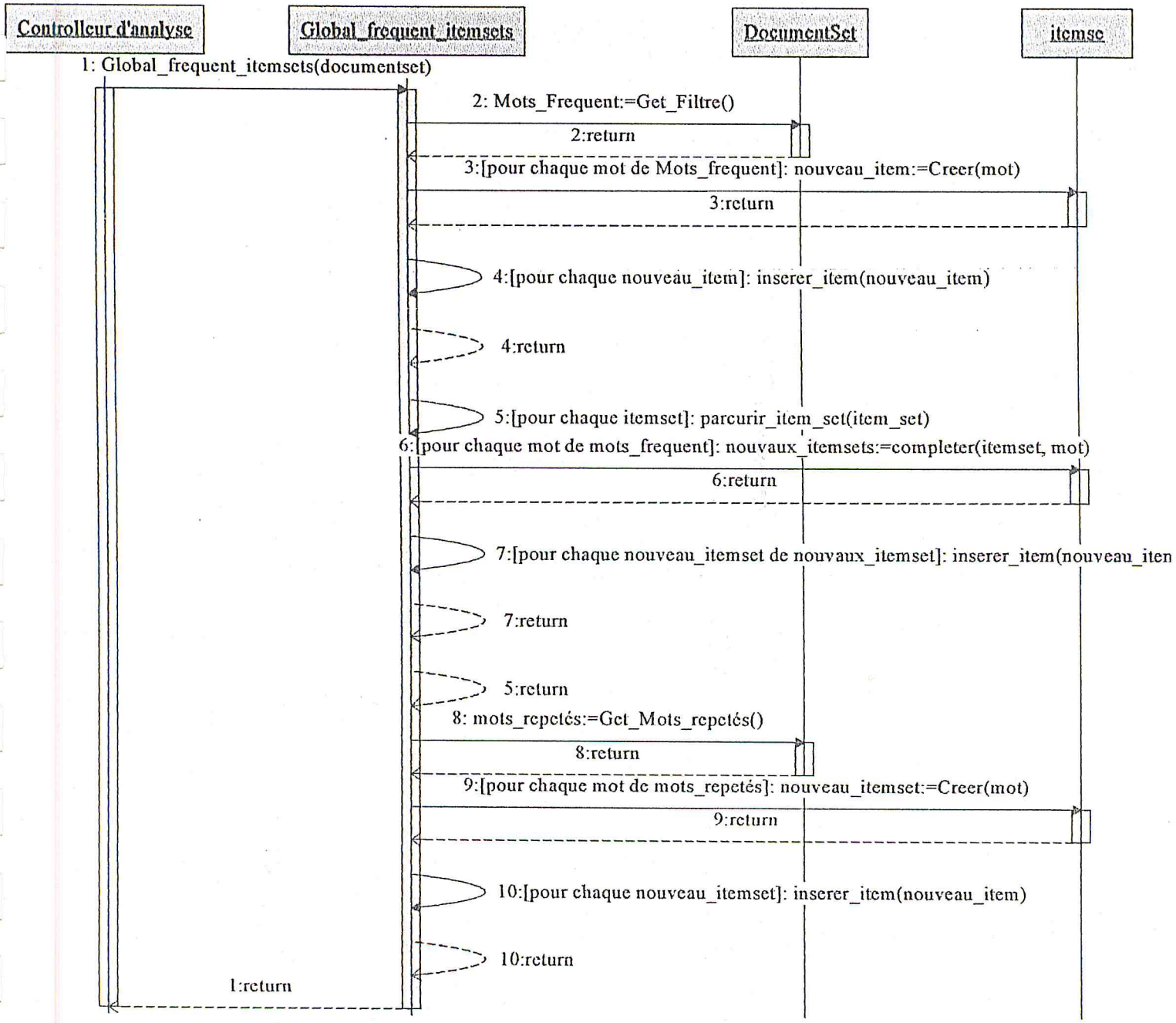


Figure V.25 : Diagramme de séquence « Création de " global frequent itemset " ».

4) Diagramme de séquence « **Conversion de " global frequent itemset " en arbre de cluster s** »

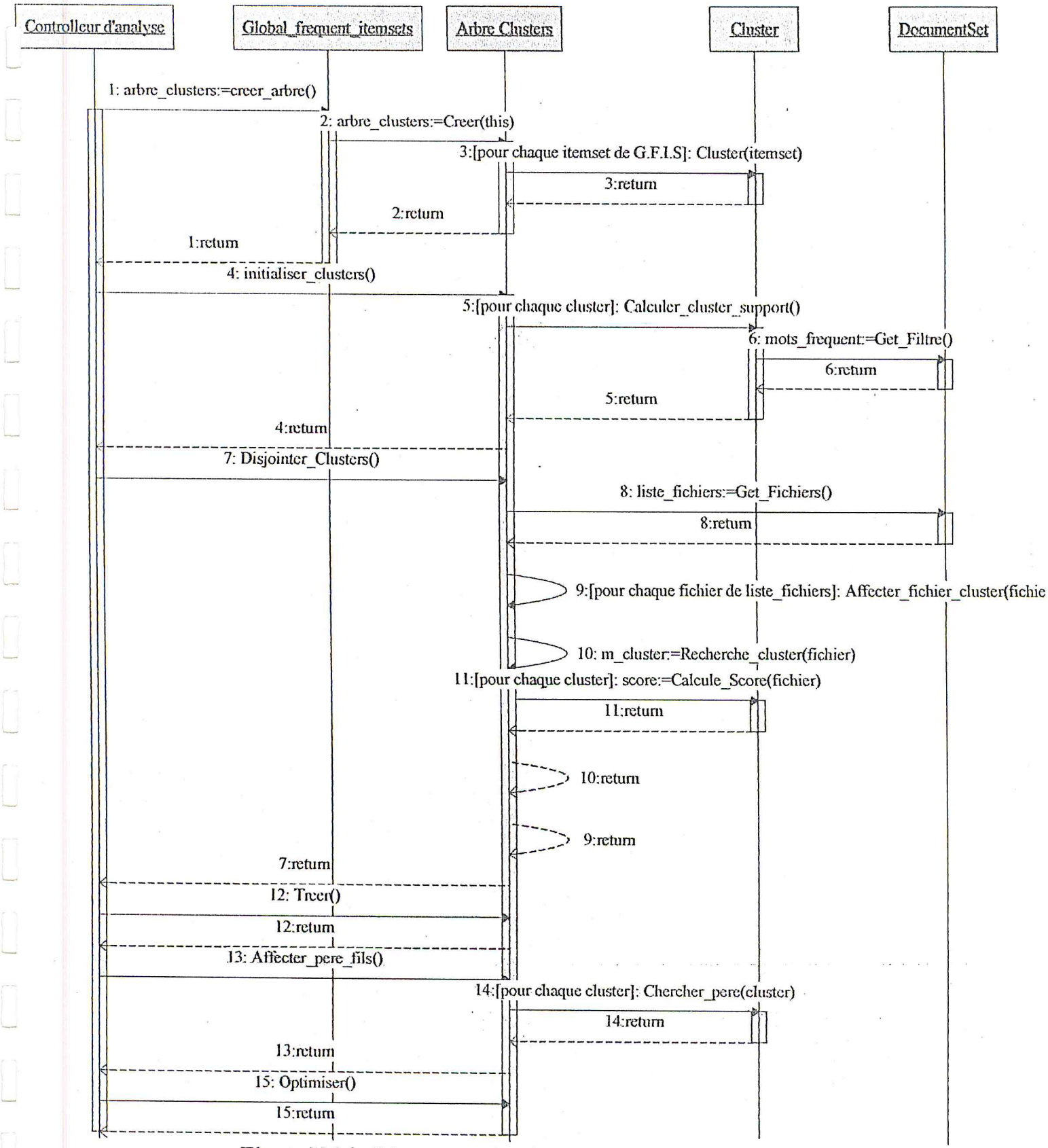


Figure V.26 : Diagramme de séquence « Conversion de " global frequent itemset "en arbre de clusters »

Le diagramme décrit ci-dessus représente la création de l'arbre final de clusters, il comprend :

- l'objet cluster qui permet la création d'un cluster pour chaque "global frequent itemset ".
- après la création des clusters initiaux ; un calcul de cluster - support est établi pour les "cluster frequents"(défini dans chapitre IV) de chaque cluster initial.
- l'affectation des fichiers aux meilleurs clusters utilisant la fonction " Score " et la création des clusters disjoints.
- le tri des clusters disjoints selon le " cluster label ".
- l'affectation d'un père pour chaque cluster et la création de l'arbre initial.
- l'optimisation de l'arbre initial pour produire l'arbre final.

Durant l'étape d'analyse le système permet la lecture des fichiers à travers le gestionnaire de fichiers et à la fin du processus, il convertit l'arbre de cluster construit en un arbre DOM - XML à travers le convertisseur XML.

5.2.2 Le module gestionnaire de fichiers

1) Diagramme de séquence de « Lecture de fichier texte »

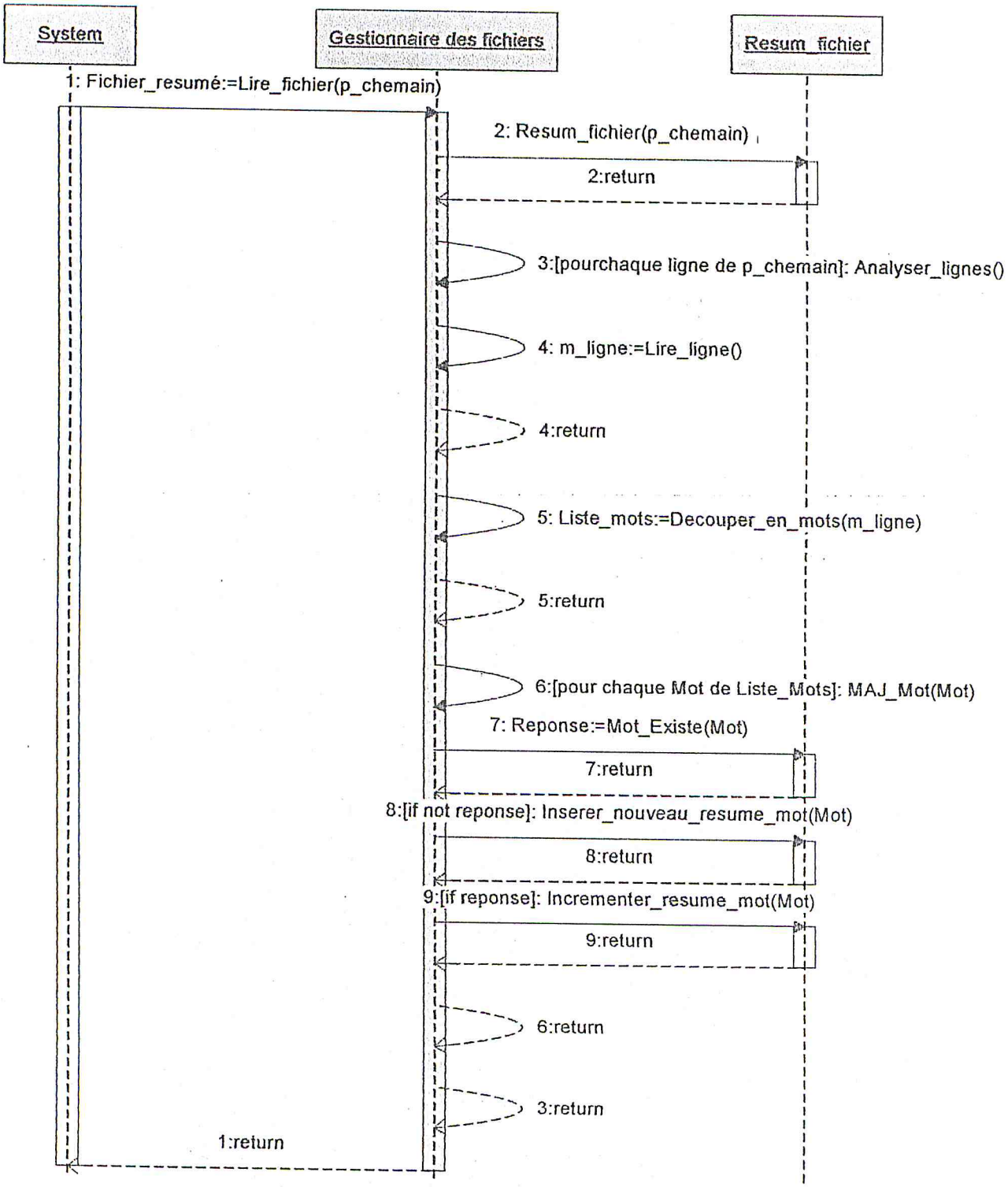


Figure V.27 : Diagramme de séquence de « Lecture de fichier texte »

Ce diagramme comporte les événements suivants :

- la lecture d'un fichier ; le résultat est mis dans une structure résume –fichier.
- le découpage de fichier en un ensemble de lignes.
- découpage de la ligne en une liste de mots.
- vérification du mot s'il n'existe pas déjà, il est ajouté à la liste des mots avec occurrence 1 dans résume –mot, sinon son nombre d'occurrence est incrémenté.

5.2.3 Le module convertisseur XML

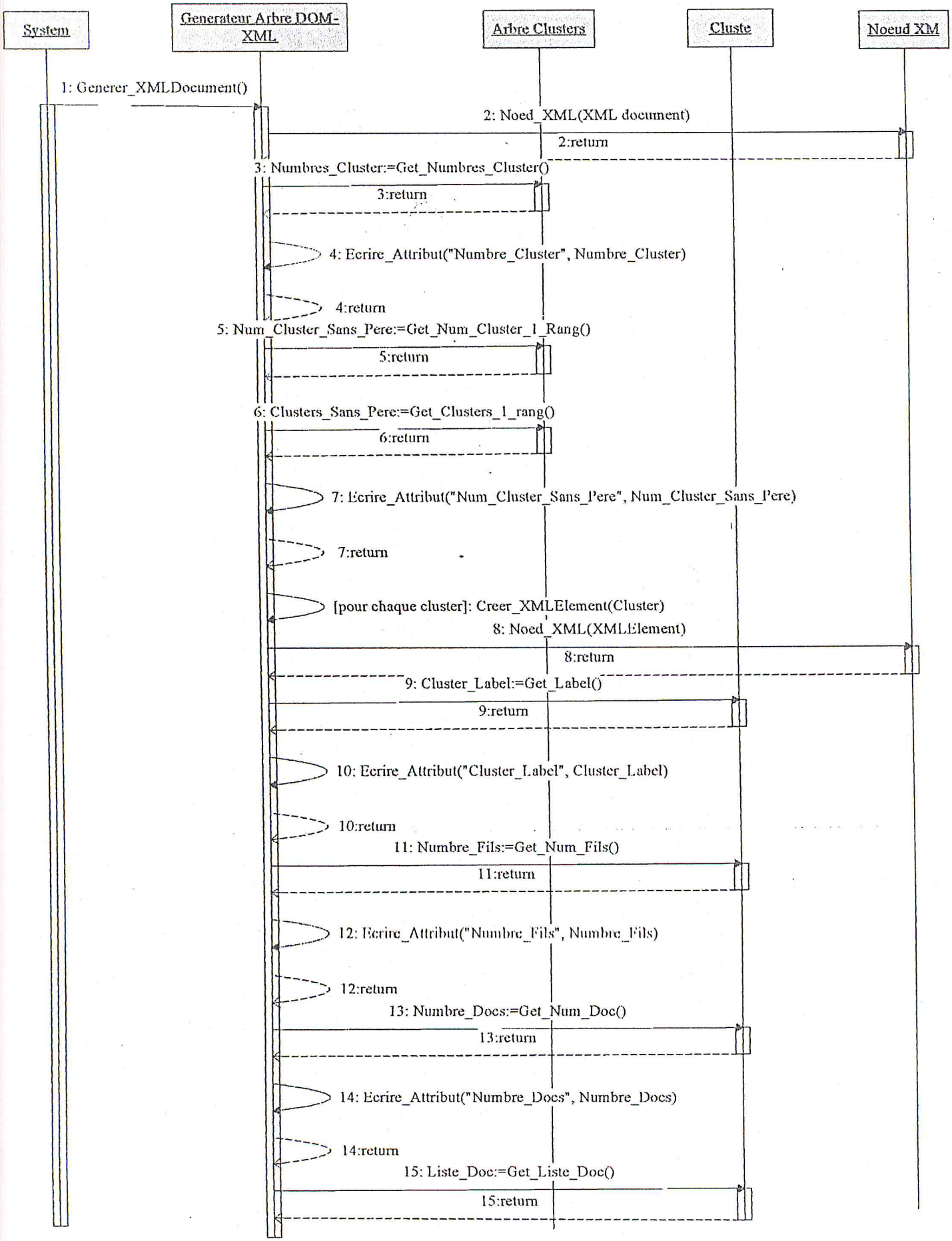
1) Diagramme de séquence « conversion de l'arbre de cluster en arbre DOM - XML »

L'arbre de clusters, résultat de la méthode FIHC est converti en une arbre DOM – XML (la définition de cet arbre est décrie dans l'étape implémentation).

La génération d'un document XML (l'arbre DOM) utilise un ensemble de " Nœud XML " .Ce noeud représente l'élément XML.

Le diagramme décrit en –dessous illustre l'ensemble de ces nœuds et les événements échangés entre les objets comme suit :

- ☒ création d'un noeud XML avec l'attribut (nombre de clusters de niveau 1),
- ☒ la création pour chaque cluster d'un nœud XML (élément) qui comprend les attributs suivant :
  - cluster - label,
  - nombre des fils,
  - nombre de documents de cluster.
- ☒ la création d'un noeud XML pour chaque liste de document d'un cluster avec l'attribut :
  - nombre de documents,
  - liste de documents.
- ☒ pour chaque liste des fils d'un cluster est affichée la liste de ces documents.


System
Generateur Arbre DOM-XML
Arbre Clusters
Cluste
Noeud XM
1: Generer_XMLDocument()
2: Noed_XML(XML document)
2:return
3: Numbres_Cluster:=Get_Numbres_Cluster()
3:return
4: Ecrire_Attribut("Numbre_Cluster", Numbre_Cluster)
4:return
5: Num_Cluster_Sans_Pere:=Get_Num_Cluster_1_Rang()
5:return
6: Clusters_Sans_Pere:=Get_Clusters_1_rang()
6:return
7: Ecrire_Attribut("Num_Cluster_Sans_Pere", Num_Cluster_Sans_Pere)
7:return
[pour chaque cluster]: Creer_XMLElement(Cluster)
8: Noed_XML(XMLElement)
8:return
9: Cluster_Label:=Get_Label()
9:return
10: Ecrire_Attribut("Cluster_Label", Cluster_Label)
10:return
11: Numbre_Fils:=Get_Num_Fils()
11:return
12: Ecrire_Attribut("Numbre_Fils", Numbre_Fils)
12:return
13: Numbre_Docs:=Get_Num_Doc()
13:return
14: Ecrire_Attribut("Numbre_Docs", Numbre_Docs)
14:return
15: Liste_Doc:=Get_Liste_Doc()
15:return

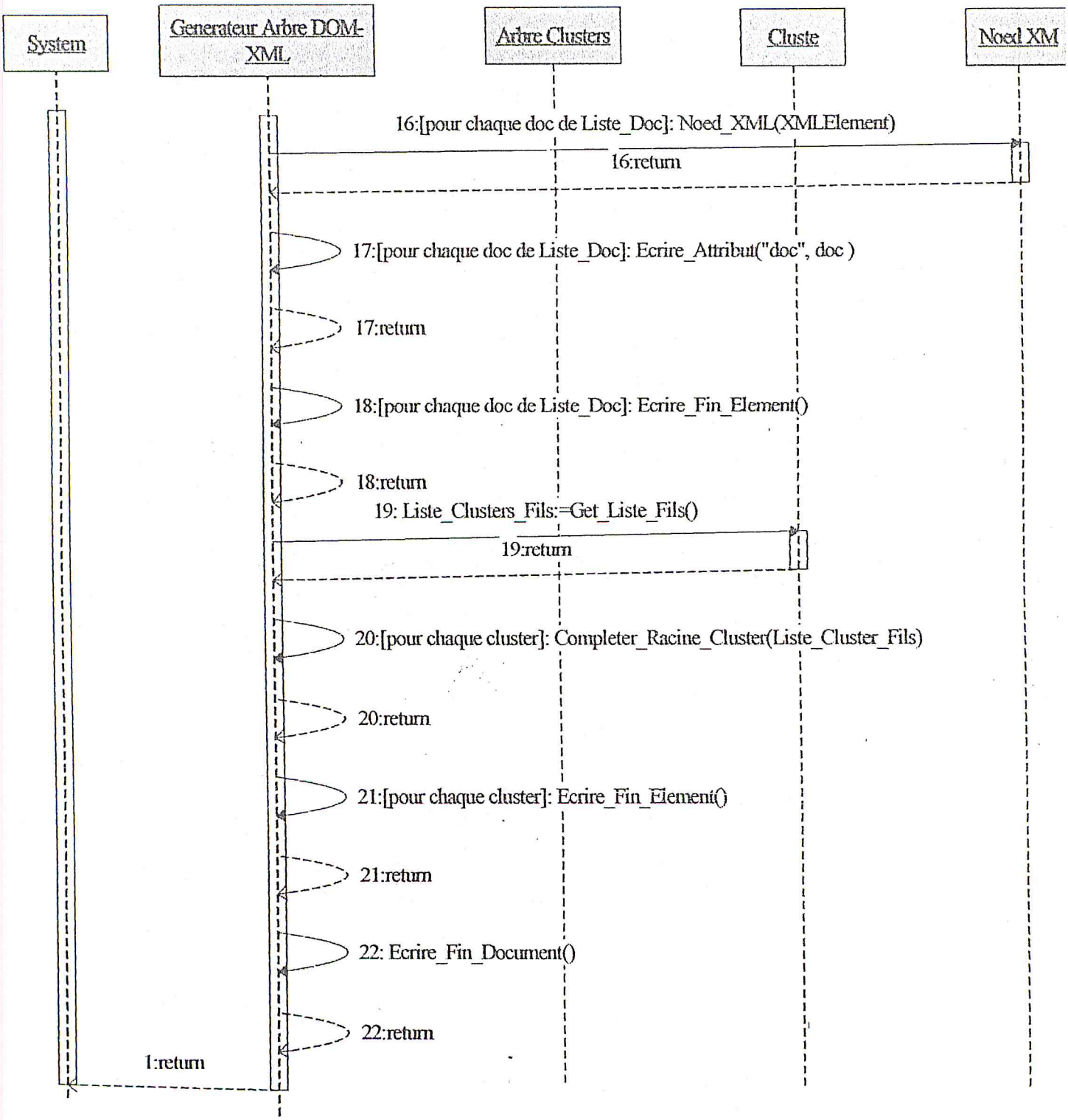


Figure V.28 : Diagramme de séquence « conversion de l'arbre de cluster en arbre DOM - XML »

6. L'implémentation

A ce niveau, il s'agit d'implémenter la solution retenue dans la phase de conception, c'est la phase au cours de laquelle les structures et les algorithmes définis pendant la conception sont traduits dans un langage de programmation.

6.1 Environnement de mise en œuvre

L'environnement choisi, pour l'implémentation des différentes méthodes est ***Windows*** utilisant l'architecture .Net, appelée à ses balbutiements NGWS (Next generation Web Service) .Elle consiste en une couche Windows, en fait une collection de DLL librement distribuable et qui sera incorporée dans les prochaines mises à jour de Windows et de Visual Studio. Le langage utilisé pour la réalisation des différentes méthodes est le langage ***C#.*** C'est le langage star de la nouvelle version de Visual Studio et de l'architecture .Net. C'est un langage dérivé de C++. C# qui peut être utilisé pour créer avec une facilité incomparable, des applications Windows et Web .C# ajout au C++ des techniques de construction de programmes sur la base de composants prêts à l'emploi avec propriétés et événements. [LEB 02].

6.2 Les mécanismes utilisés pour l'optimisation de FIHC

Dans notre implémentation, nous n'avons pas implémenté l'algorithme « A priori » présenté dans le chapitre II, qui est utilisé dans l'étape création de " global frequent itemset " de la méthode FIHC. L'algorithme « Apriori » qui souffre de certaines faiblesses présentées précédemment (chapitre II). Un des résultats de ses faiblesses est que le temps de recherche des itemset est exponentiel par rapport à la taille de transaction (accès à la base des documents).

Nous avons adopté le principe de l'algorithme « A priori » qui est :

- tout sous ensemble d'un ensemble fréquent est fréquent,
- tout sous ensemble d'un ensemble non fréquent est non fréquent.

Pour remédier nous avons utilisé quelque mécanisme d'optimisation :

1) dans l'étape de construction des itemset. Nous avons construit " *un seul* " item pour "*Un seule* " mot de l'ensemble de mots répétés (se sont les mots qui ont un support 100%, c'est - à – dire qu'il existe dans tous les documents). Nous argumentons ce mécanisme par un exemple illustratif.

Si on a trois clusters C1, C2, C3.

C1 avec étiquette (" cluster label ") : mot1 (répétés 100%),

C2 avec étiquette (" cluster label ") : mot 2 (répété 100%),

C3 avec étiquette (" cluster label ") : (mot1, mot2),

Si nous appliquons la fonction « Score » pour chaque cluster, pour l'attribution des fichiers aux meilleurs clusters, on trouve pour le fichier F1 :

Score (C1 , F1) = 0.60
Score (C2 , F1) = 0.60
Score (C3 , F1) = 0.60

Donc le fichier F1 est attribué automatiquement au premier cluster C1.

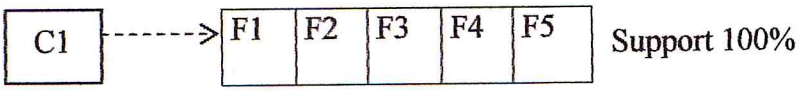
Si nous généralisons cette propriété pour tous les documents, nous remarquons que tous les documents sont assignés au premier cluster de l'étiquette (mot1) qui est de répété 100%. Comme résultat tous les calculs des autres itemsets sont inutiles prennent un temps considérable.

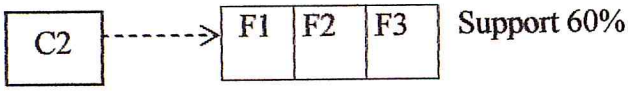
2) dans la combinaison des itemsets, nous n'avons pas pris le cas de combinaison d'un mot fréquent (avec support < 100%) et un mots répété (support est égale à 100%). Nous argumentons ce mécanisme par un exemple illustratif :

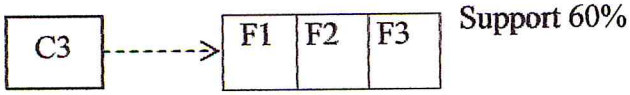
- mot 1 est avec un support 100%.
- mot 2 est mot fréquent avec un support de 60%.

L'attribution des fichiers aux clusters des étiquettes mot1 et mot3 est la suivante :

- C1 (mot 1) : le cluster associe au item " mot1",
- C2 (mot 2) : le cluster associe au item " mot2",
- C3 (mot1, mot 2) : le cluster associe au item " mot1, mot2",







Comme résultat les fichiers sont affectés au cluster de " *cluster label* mot2 " (C2) puisque c'est le premier qui possède les fichiers F1, F2, F3. Ce qui implique que le cluster (C3) est éliminé. Ce mécanisme permet d'améliorer considérablement le temps d'exécution.

3) la combinaison entre les mots répétés n'existe pas, car nous avons pris un seul item (comme il est mentionné dans 1), car le cluster de cet item prend tous les fichiers (le support est de 100 %).

4) Nous avons utilisé les indices des mots au lieu des mots, car il et plus facile de manipuler des entiers (" int ") que des chaînes de caractères (" string "). Pour la mise en œuvre, nous avons utilisé la classe « Array Liste » dérivé de la classe « Object ».

Cette classe implémente un tableau dynamique d'objet, la taille d'un tel tableau est automatiquement ajustée si nécessaire lorsque des éléments y sont ajoutés, ce qui n'est pas le cas pour les tableaux traditionnels [LEB 02].En conséquence, nous avons inspiré les structures de « Résume - fichier », « Résume - mot » et la matrice d'occurrence utilisée dans la classe « Document set » présentée dans la partie précédente.

5) la recherche des fichiers d'un cluster fils se fait dans les fichiers de cluster parent , car le "cluster label" de parent est un sous ensemble de " cluster label " parent ; ce mécanisme minimise la recherche dans tous les fichiers à chaque fois qu'est généré un nouveau itemset

Ces structures sont schématisées dans la figure suivante :

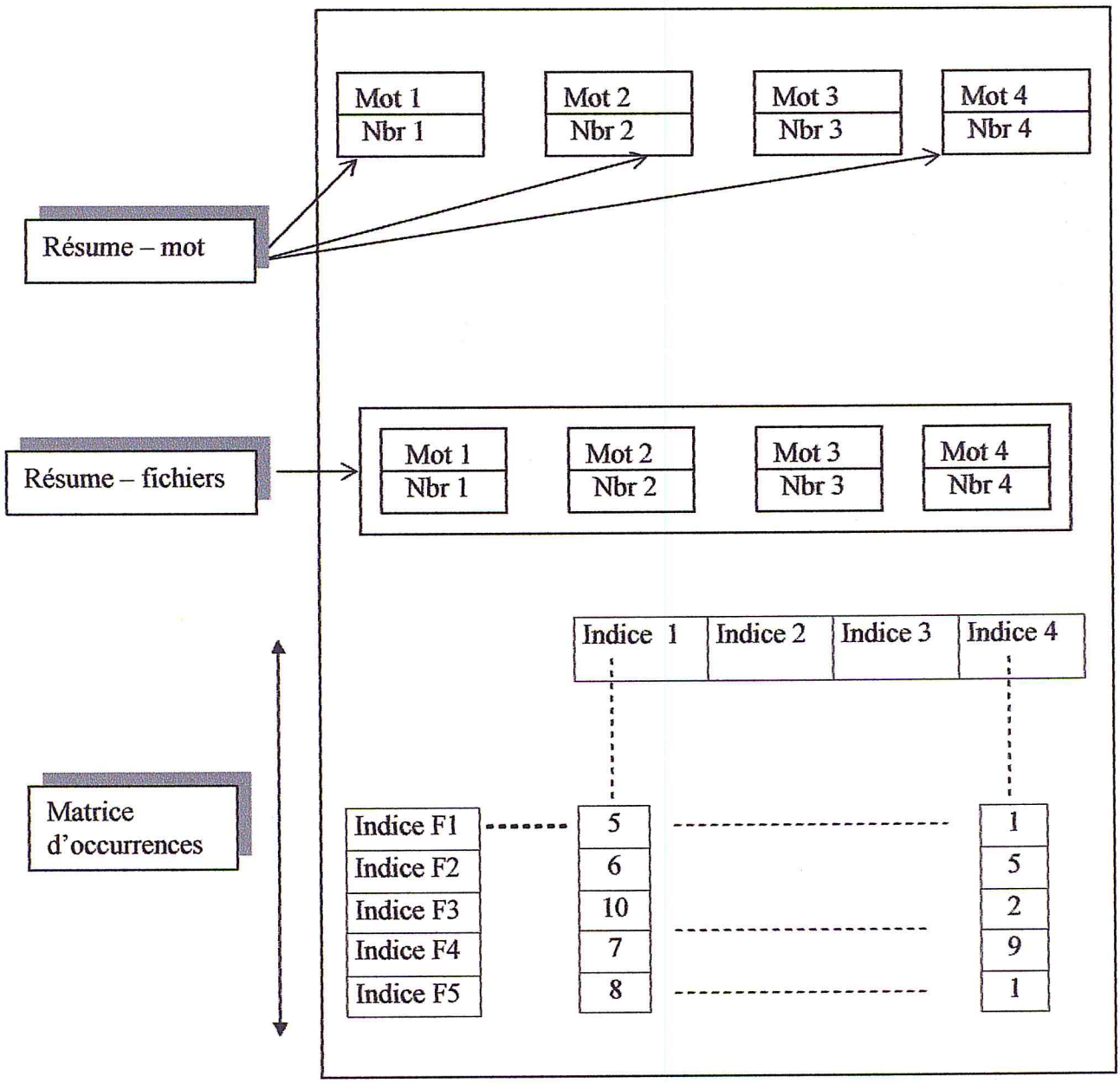
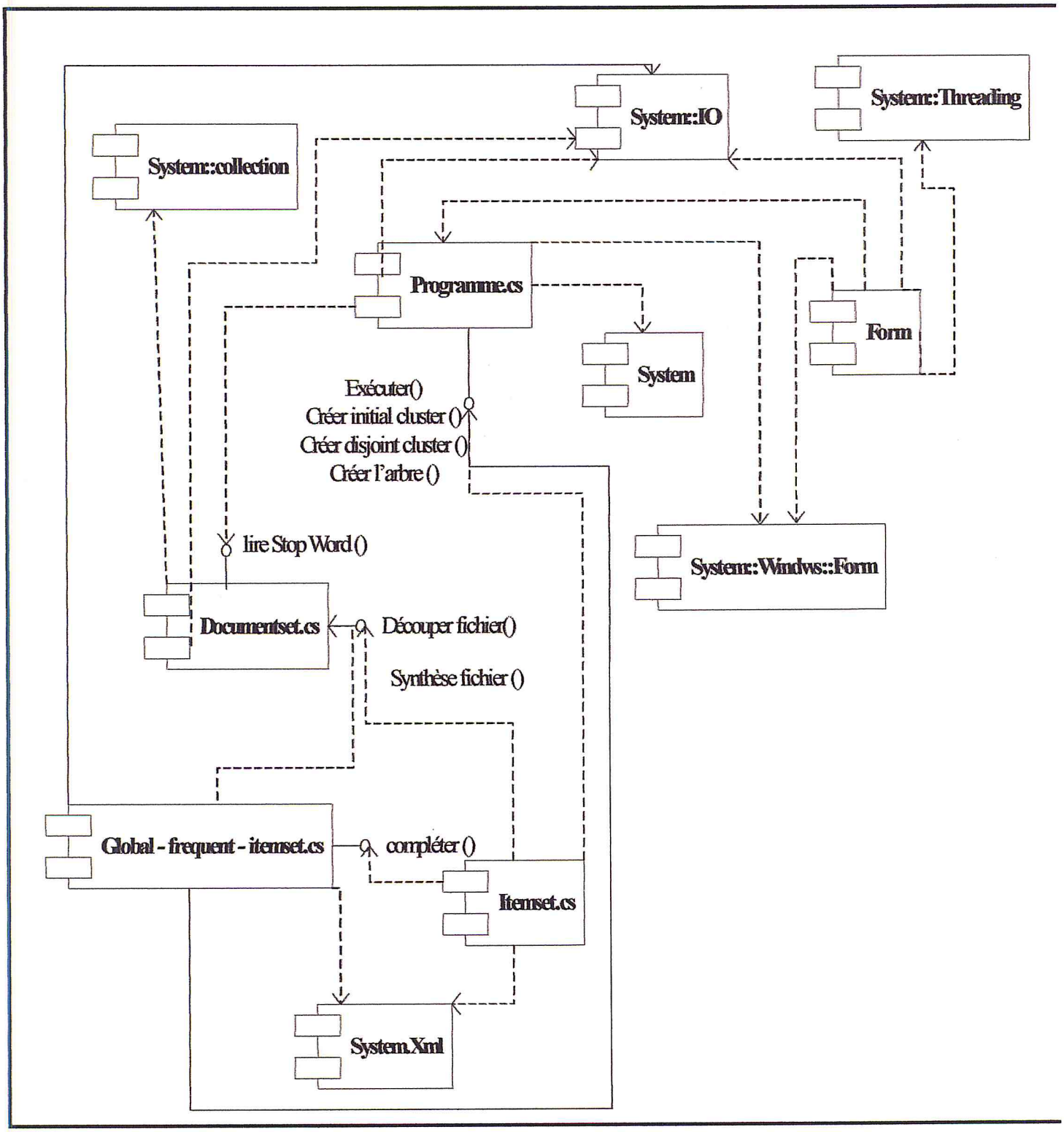


Figure V.29 : La structure de « Document set ».

Diagramme de composants des entités implémentées



FigureV.30 : Diagramme de composants

6) La classe « Xml Document »

La classe Xml Document implémente ce que l'on appelle la norme W3C " Document Object Model " (DOM). Le DOM consiste en une représentation en mémoire et sous forme d'arbre d'un document XML (ce dernier peut provenir d'un fichier ou même d'une chaîne de caractères du programme).

Lors du chargement, les fonctions " Load " (pour le chargement d'un fichier) ou " Load Xml " (pour le chargement à partir d'une variable en mémoire) effectuent une vérification de syntaxe XML. Une exception est générée en cas d'erreur.

La classe « Xml Document » est dérivée de la classe « XML Node »et chaque nœud de l'arbre est lui même un objet « Xml Node » [LEB 02].

La figure suivante illustre un document XML.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <root num_clusters="269" label="null" num_children="30"
    num_docs="1560">
    <documents num_docs="0" />
  - <cluster label="advisori" num_children="0" num_docs="2">
    + <documents num_docs="2">
    </cluster>
  - <cluster label="arbitr" num_children="1" num_docs="9">
    - <documents num_docs="7">
        <document>Television.346</document>
        <document>Television.590</document>
        <document>Media.187</document>
        <document>Music.327</document>
        <document>People.1043</document>
        <document>Television.2</document>
        <document>People.867</document>
      </documents>
    - <cluster label="arbitr mechan" num_children="0" num_docs="2">
      - <documents num_docs="2">
          <document>People.666</document>
          <document>People.970</document>
        </documents>
      </cluster>
    </cluster>
    - <cluster label="argum" num_children="0" num_docs="4">
    - <documents num_docs="4">
        <document>Art.463</document>
        <document>Media.544</document>
        <document>Industry.1486</document>
        <document>Film.230</document>
      </documents>
    </cluster>
</root>
```

Figure V.31 : Exemple d'un arbre de clusters dans un document XML [FUN 03].

☒ Explorateur de fichier XML

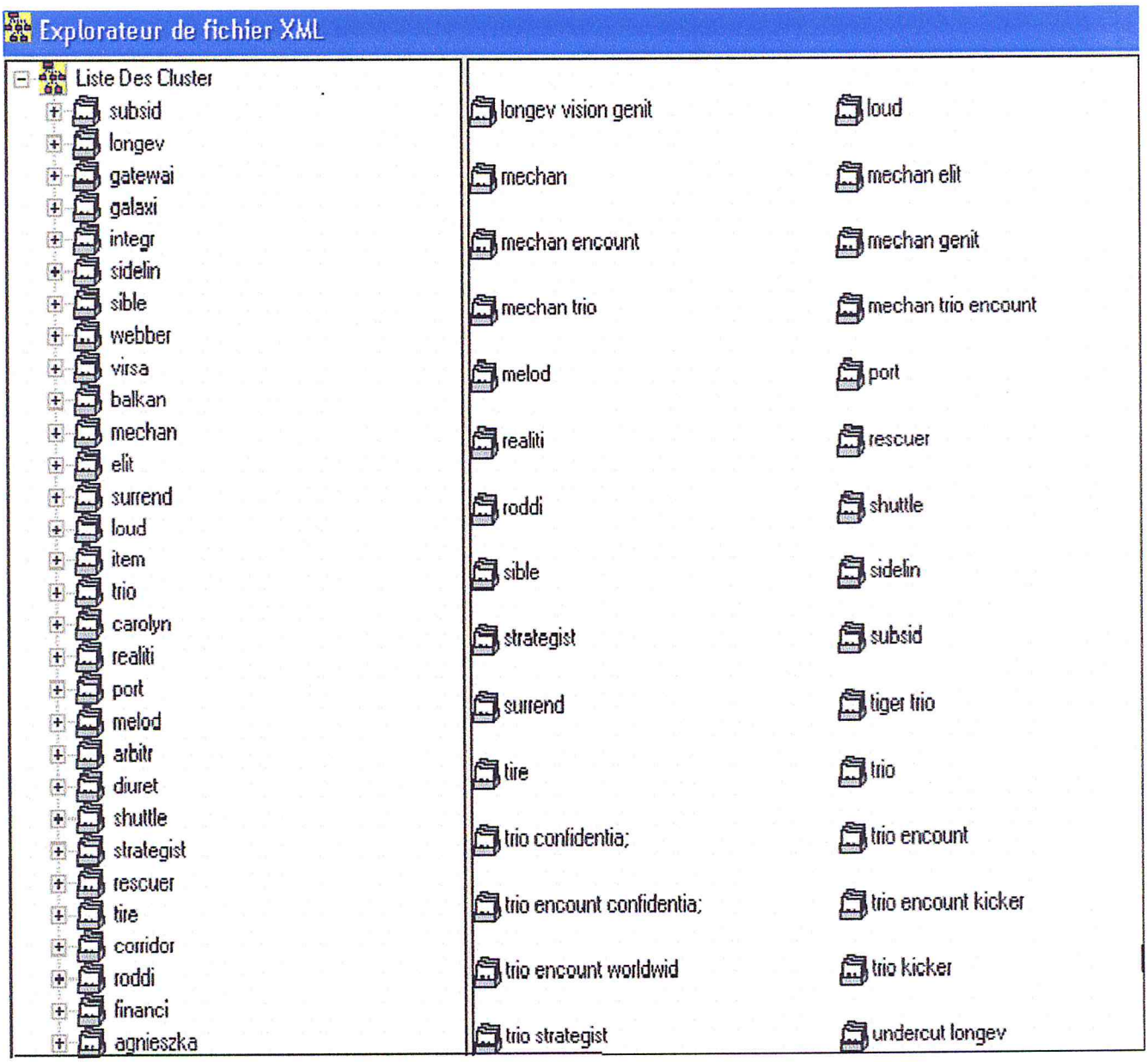


Figure V.34 : l'explorateur d'un fichier XML

☒ La liste des fichiers de chaque cluster

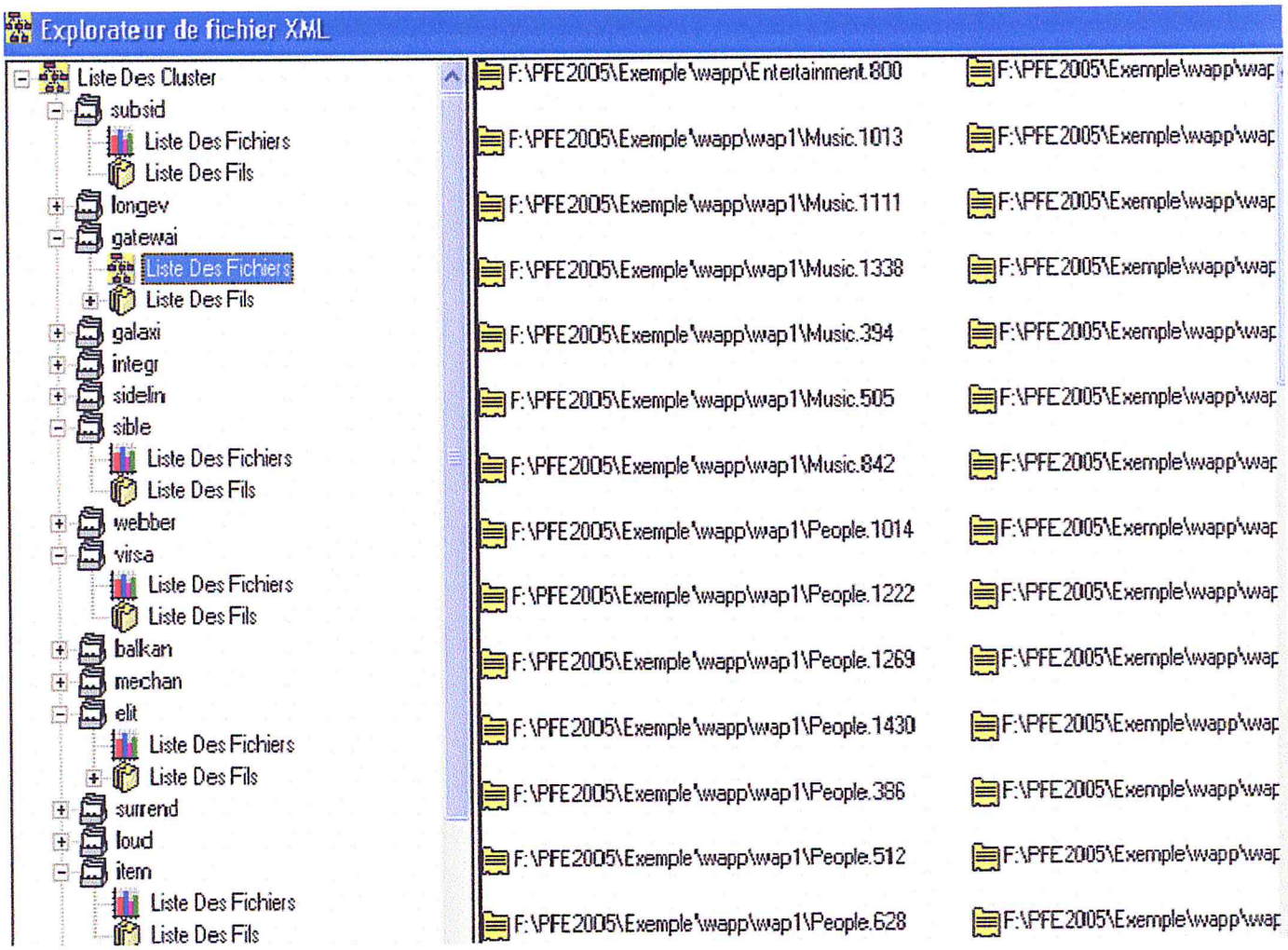


Figure V.35 : les listes des fichiers dans l'explorateur

- La liste des fils de chaque cluster

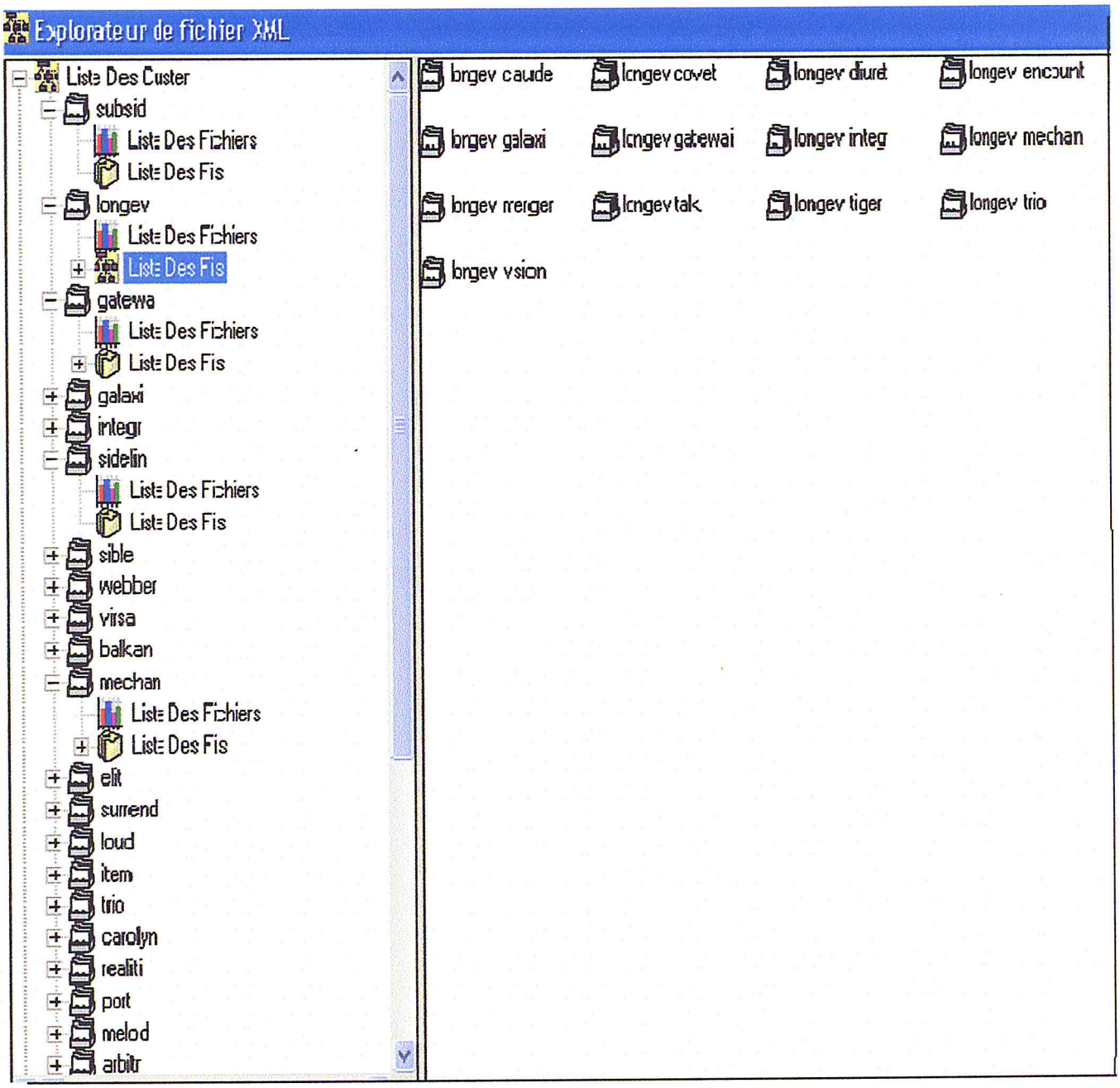


Figure V.36 : les listes des fils dans l'explorateur

6.3 Interface utilisateur

C'est l'interface graphique **(Figure V.32)** perceptible par l'utilisateur et qui sert d'intermédiaire entre l'utilisateur et l'application lui offrant tous les moyens pour une manipulation simple et aisée des différents modules du logiciel.

☒ Ajout d'un dossier

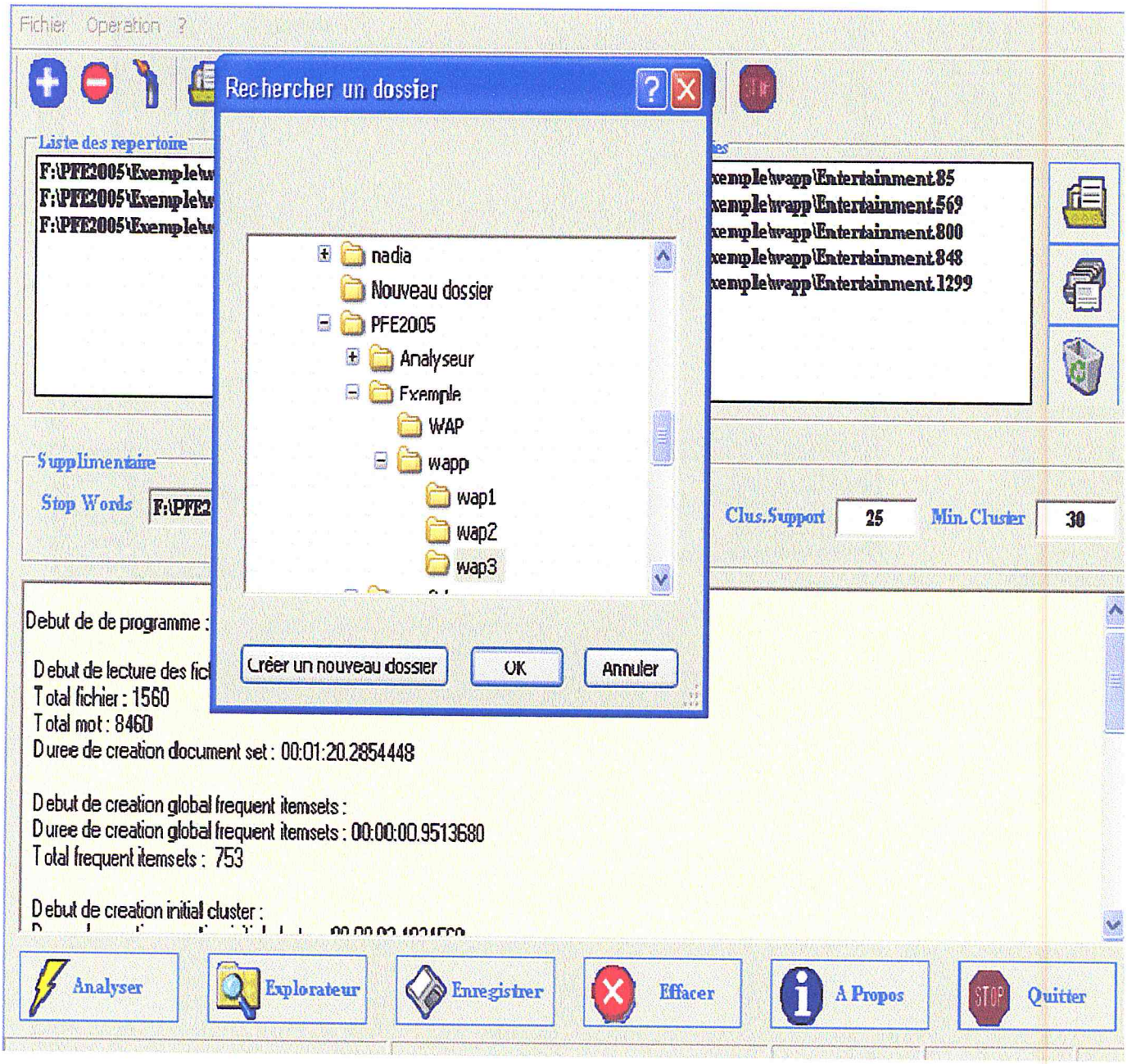


Figure V.32: l'ajout d'un dossier dans l'interface graphique

7. Tests et validation des résultats

Le test permet de réaliser des contrôles pour la qualité du système. Il s'agit de relever les défauts de la conception et de la programmation (revue de code, tests des composants de systèmes) [BOU 01]. Nous présentons dans cette section la dernière étape de la démarche qui est nécessaire pour valider les résultats.

L'évaluation de l'exactitude et de l'efficacité d'une méthode de clustering est effectuée par plusieurs fonctions de mesure. Les plus utilisée sont :

- « *entropy* » ou *entropie* utilisée pour mesurer la qualité des clusters, telle que la meilleure entropie obtenue quand chaque cluster contient un seul topic [STE 00].
- « *F-measure* » traite chaque cluster comme une transaction et chaque classe comme si c'était l'ensemble désiré de documents dans une transaction [FUN 03].

Nous allons présenter et étudier une évaluation de la méthode (FIHC) en utilisant "F-measure ", puis nous allons étudier le rapport de "F-measure" avec le paramètre d'entrée nombre de clusters (K). Nous étudions ensuite le rapport de " F-measure" avec le paramètre d'entrée "minimum support", et finalement nous présentons les résultats des tests de notre implémentation concernant le rapport de paramètre d'entrée minimum support et le temps d'exécution. Tous ces résultats sont obtenus en utilisant une base de documents ou un ensemble de documents qui est présenté ci-dessous.

7.1 L'ensemble de documents (« Data set »)

Dans cette évaluation, nous utilisons la base de document (« data set ») « wap » qui contient trois « data sets » (wap1, wap2, wap3) et d'autres documents divers. Cette base à l'origine est utilisée pour le projet **WebAce** [HAN 98], et pour les différentes recherches de clustering [KAR 02].

Cette base et ses data sets sont hétérogènes en terme de taille de document,de taille de cluster, de nombre de classes (topic) et de distribution des documents. Elle est caractérisée par différentes propriétés ou attributs tels que le nombre de documents, le nombre de classes, la taille de classes, la taille moyenne de classe et le nombre total des mots.

A chaque data set est associé un nombre de classes ou bien « topic », qui regroupe les principaux mots communs. Ces classes sont utilisées dans la méthode d'évaluation pour mesurer l'exactitude des résultats de clustering.

Nous présentons ci-dessous un tableau qui caractérise la base de documents « wap », suivi d'un autre qui présente les différentes caractéristiques de chacun de ses data sets.

Data set	Nombre de documents	Nombre de classes	La taille de classes	La taille moyenne de classe	Nombre de mots
wap	1560	20	5 - 341	78	8460

Tableau V.1 : Description récapitulative de la base de documents « wap ».

Data set	Nombre de documents	Nombre de classes	La taille de classes	La taille moyenne de classe	Nombre de mots
wap1	1128	8	13 - 341	41	8110
wap2	189	5	11 - 76	13	4547
wap3	238	6	29 - 65	6	5279

Tableau V.2 : Description récapitulative de différents data sets de « wap ».

7.2 La méthode d'évaluation « F-measure »

« **F-measure** » est une méthode standard pour évaluer les différentes méthodes de clustering. Elle mesure l'exactitude produite par ces dernières, en produisant une mesure équilibrée de "***precision*** "et "***recall***". Elle traite chaque cluster comme le résultat d'une requête, et chaque classe comme un ensemble de documents approprié pour une requête.

L'algorithme de calcul de « F-measure » pour une classe K_i et un cluster C_j est représenté dans le pseudo code dans la figure suivante :

Algorithme

```
// Recall et Precision sont calculés pour chaque classe dans un cluster
Pour chaque cluster Cj dans la liste des clusters
 {
  Pour chaque classe Ki dans la liste des classes
  {
   Recall (Ki, Cj) = nij / | Ki | ;
   Precision (Ki, Cj) = nij / | Cj | ;
  // nij est le nombre de documents de la classe Ki dans le cluster Cj.
   F (Ki , Cj )= (2* Recall (Ki , Cj )* Precision (Ki , Cj) )/( Recall (Ki , Cj )+
   Precision (Ki, Cj));
   //F (Ki, Cj) représente la qualité d'un cluster Cj en décrivant la class Ki. Lorsqu'on calcule
   // F (Ki, Cj) dans une structure hiérarchique, tous les documents appartenant aux sous arbres
   //de Cj sont considérés comme des documents de Cj.
  }
}
// F(C) est « overall F-measure » la somme du maximum F-mesure de toutes les classes et
clusters
 maximum = max Cj appartient à C {F (Ki, Cj)} ;
// Le maximum de F (Ki, Cj) peut être vu comme le choix du cluster qui peut mieux décrire
//une classe
F(C)=0
Pour chaque classe Ki de l'ensemble de classes K
 {
  F(C)= F(C) + (| Ki |/ | D |) ;
  // | D | présente le nombre total des documents dans un data set
 }
F(C) = F(C) maximum;
```

Figure V.37 : Le pseudo code de la fonction « F-measure » [FUN 03].

☒ Ajout d'un fichier

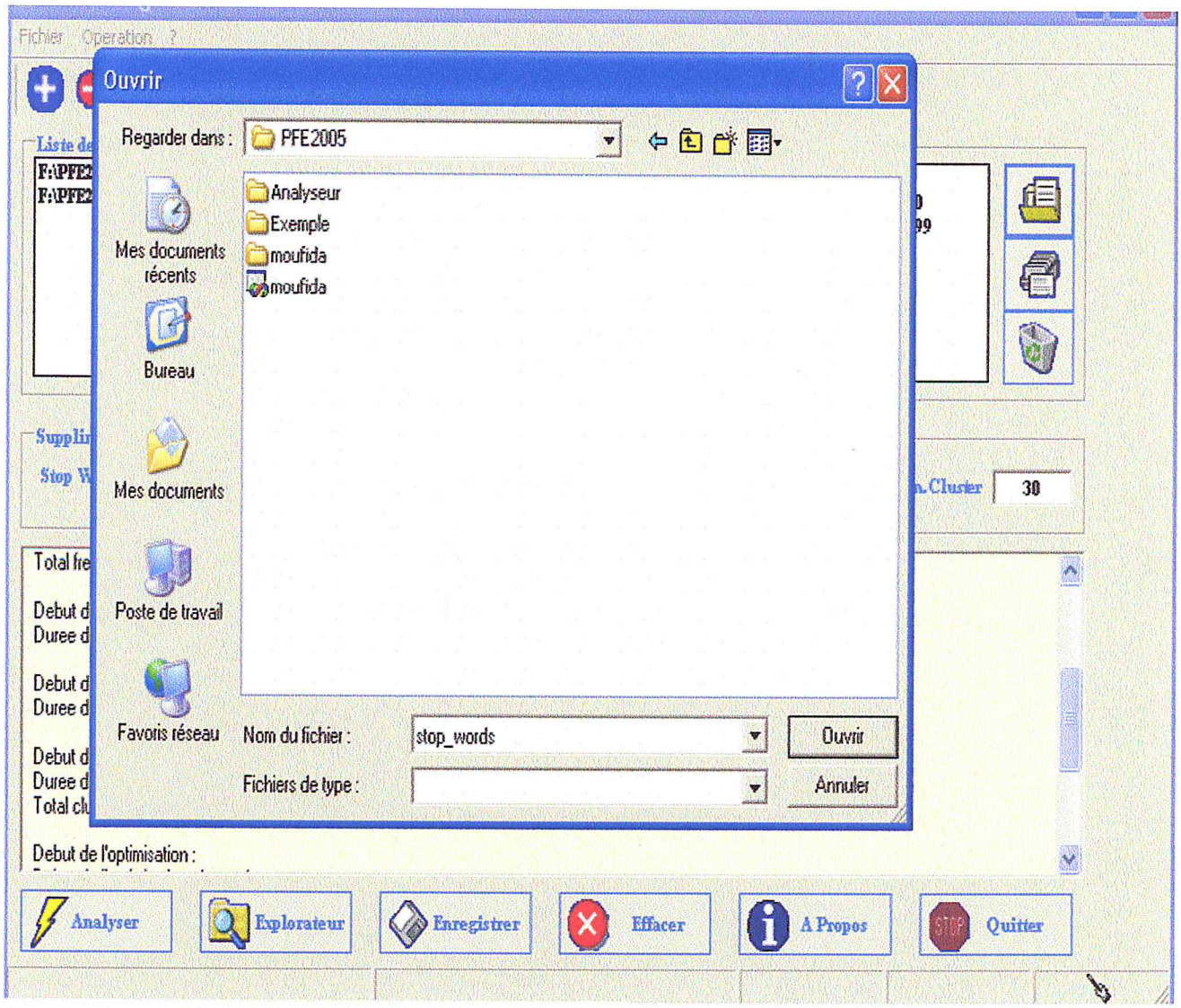


Figure V.33 : l'ajout d'un fichier dans l'interface graphique

Remarque

Lorsque la valeur de F(C) est plus grande, cela indique une exactitude élevée de clustering.

L'application de F-measure à la base « Wap »

La base « Wap » contient 1560 documents (documents écrit en anglais) et 20 classes ou matières qui sont cités comme suit :

Television, industry, technology, culture , stage, cable, variety, online, people, music, art, review, film, media, sports, entertainment, politics, business, health, multimedia.

L'application de l'algorithme « FIHC » pour trois clusters donne :

Clusters	Les classes	Nombre de classes
Cluster1	Health, people, industry, politics, multimedia, culture, television, music, film, sport, stage.	11
Cluster2	Politics, culture, television, people, business, sport, entrainment, music, media, film.	10
Cluster3	Television, industry, technology, culture, stage, cable, variety, online, people, music, art, review, film, media, sports, entrainment, politics, business, health, multimedia.	20

Tableau V.3: La description des clusters et leurs différentes classes.

Le calcul de F (K_i, C_1)

Pour la classe « health » (K_1)

Recall (K_1, C_1) = nombre de documents de la classe « health » dans C_1/ total documents dans « health » =331/ 341=0.97

Precision (K1, C1) = nombre de documents de la classe « health » dans C_1/ total documents dans « C1» = 331/351=0.94

Et ainsi de suite pour toute les classes restantes pour calculer à la fin F (Ki, C1). La même chose pour tous les clusters pour calculer en finalité F(C) pour les trois clusters.

Les résultats obtenus sont illustrés dans le tableau suivant :

Data set	Nombre de clusters	F-measure
wap (20)	3	0.40
	15	0.56
	30	0.57
	60	0.55
	Moyen	0.52

Tableau V.4 : « F-measure » de la méthode FIHC [FUN 03].

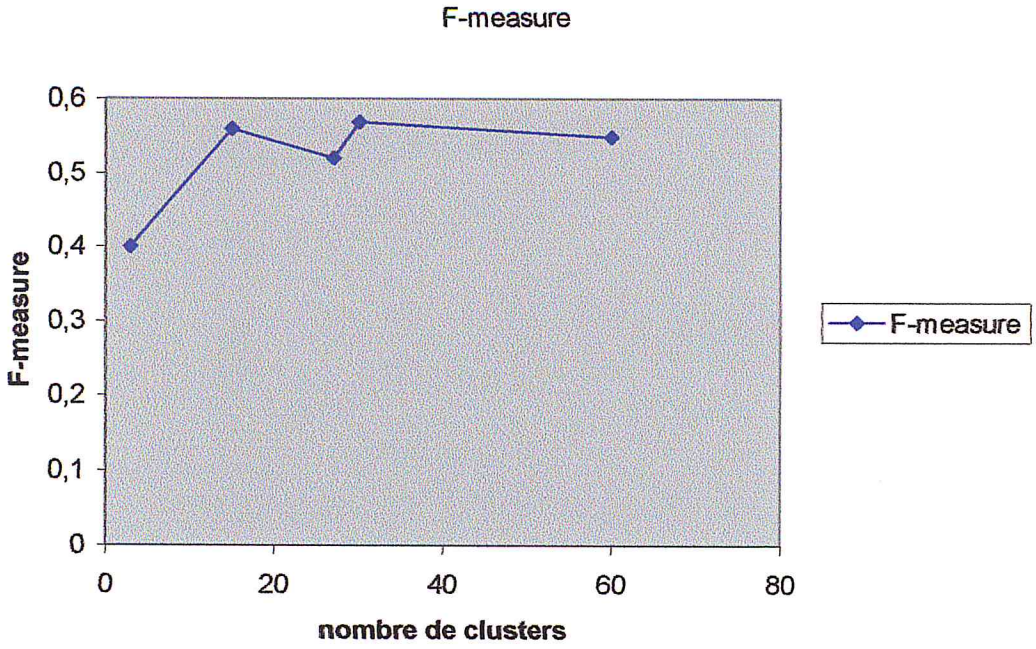


Figure V.38 : La sensibilité de FIHC au nombres de clusters [FUN 03].

D'après le graphe de la figure V.38 nous remarquons que lorsque le nombre de clusters augmente l'exactitude de FIHC augmente jusqu'a l'obtention d'une valeur presque équivalente au nombre de classes de la base de documents utilisés.

Alors, nous remarquons d'après ces résultats que l'exactitude de FIHC est liée au nombre de clusters fournis, et spécialement à une valeur équivalente au nombre de classes c'est-à-dire que FIHC atteint son exactitude optimale lorsque chaque cluster contient une seule classe de la base de documents.

7.3 Résultats expérimentaux

Nous allons dans cette section étudier et analyser quelques résultats expérimentaux ensuite nous étudions nos propre résultats en utilisant un PC Intel (R) Pentium IV avec un micro processeur de 2.40 GHZ et une RAM de 254 Mo.

7.3.1 FIHC et la sensibilité aux paramètres (« Minimum support »)

Le " minimum support" permet à l'utilisateur d'ajuster la forme de l'arbre de clusters en éliminant tous les items au dessous de ce support. Si la valeur de "minimum support" est petite, alors l'arbre est large et profond et vice versa.

Les expériences ont montré qu'une très grande exactitude est obtenue lorsque le "minimum support" est entre 3% et 9 % et lorsque l'arbre de clusters a de deux à quatre niveaux [FUN 03]. Elles ont aussi montré que lorsqu'on a un arbre de moins de 5000 documents le résultat est meilleur avec le "minimum support" entre [3% - 5%], ce qui est illustré dans le graphe ci dessous [FUN 03], qui fait apparaître le rapport de l'exactitude de FIHC avec le "minimum support" en utilisant la base « wap ».

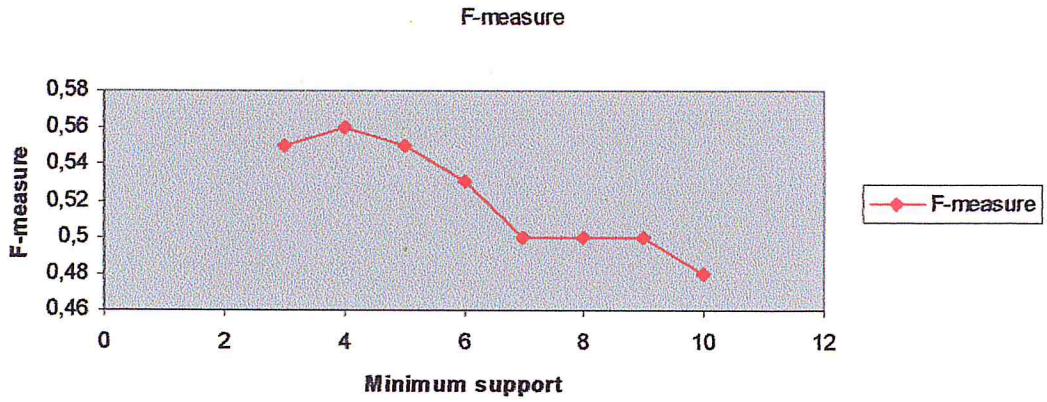


Figure V.39 : La sensibilité de FIHC au minimum support avec la spécification du nombre de clusters [FUN 03].

Nous remarquons que l'augmentation de minimum support a une influence négative sur « F-measure » et l'exactitude de l'algorithme FIHC.

Nous remarquons aussi que « F-measure » atteint sa valeur optimale dans l'intervalle de minimum support [3%, 5%].

7.3.2 FIHC et le temps d'exécution

En utilisant la base de documents « wap » contenant 1560 documents, un cluster support de 25% et un nombre de clusters de 30, nous obtenons les résultats affichés dans le tableau suivant :

Le minimum support (%)	Temps d'exécution (secondes)	Nombre total de clusters	Le nombre des niveaux
4	196	419	6
5	115	276	6
6	82	190	5
7	66	179	5
8	58	123	5
9	50	89	4
10	47	80	4
11	49	66	4
12	46	62	4
13	43	52	4
14	45	50	4
15	42	43	4
20	44	33	3
30	44	9	3

Tableau V.5 : Les résultats d'exécution de FIHC obtenus avec « wap ».

Voici le tableau présentant le graphe du rapport entre le temps d'exécution et le « minimum support ».

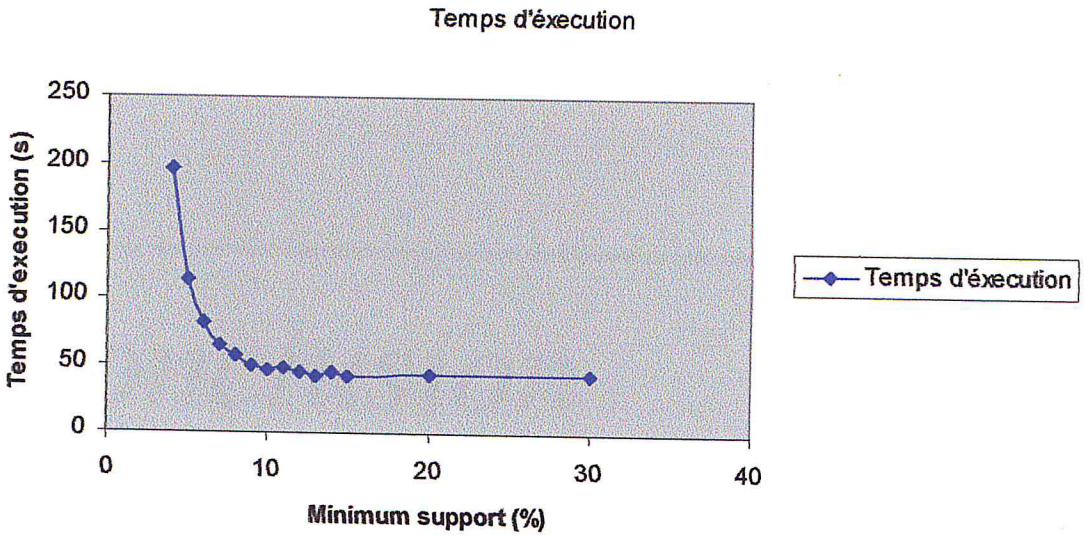


Figure V.40 : Le rapport entre le minimum support et le temps d'exécution de FIHC.

Nous remarquons que le temps d'exécution est inversement li au « minimum support », à l'augmentation de l'un entraîne la diminution de l'autre

Nous remarquons aussi que :

- l'augmentation de minimum support permet de diminuer le nombre total de clusters ;
- l'augmentation de minimum support permet de diminuer le niveau de l'arbre de clusters.

Remarque

Le cluster support est un pourcentage qui distingue si un item est « cluster frequent » sa valeur est donnée habituellement à 25% d'après les différents résultats des expériences.

8. Conclusion

Dans ce chapitre nous avons présenté, les étapes de développement de notre application, en utilisant une démarche dans un environnement orienté objet .Dans le développement de notre système le modèle en cascade représente la démarche suivie, et l'utilisation de la notation UML montre l'environnement.

Notre développement, commence par une spécification des besoins qui décrit les fonctionnalités du système global, présenté par les diagrammes de cas d'utilisation et les diagrammes d'interaction. Par la suite, une analyse présente des propositions trouvées ainsi que les composants du système global. L'étape suivante est entamée, c'est la plus importante dans le développement, elle repose sur une conception globale qui décrit l'architecture du système global et est suivie d'une autre détaillée. Puis vient but l'implémentation de notre solution. La validation est enfin présentée après un test.

Deuxièmement, la plupart des algorithmes de clustering des documents, y compris FIHC, considèrent un document comme un sac de mots, tandis que les rapports sémantiques entre ces mots ne sont pas pris en considération. Donc la méthode FIHC peut être améliorée en utilisant la signification des mots ainsi que les synonymes, autre recherche possible.

On pourrait aussi examiner la possibilité dans FIHC, d'utiliser une base de données avec des documents en plusieurs langues (anglais, français, espagnol, ...).

Pour finir, nous insistons sur l'importance d'une méthode efficace de classement, dans un domaine promis à de grands développements, en raison de la masse quotidienne de données non structurées.

Glossaire

Agent

Entité physique ou abstraite capable d'agir sur son environnement, de communiquer, de se reproduire et de poursuivre un objectif individuel.

Analyse des données

En statistique, méthodes développées dans les années 1930 et dont le succès actuel est du à l'essor des ordinateurs qui permettent d'automatiser les calculs. Les méthodes d'analyse des données permettent une étude globale des individus et des variables en utilisant généralement des représentations graphiques suggestives.

Analyse statistique

Elle permet, grâce aux méthodes et aux représentations graphiques du statistique traditionnelle, de décrire les données textuelles (fréquences, corrélations, dispersion, co-occurrences, etc).

Algorithme

Procédure informatique qui prend des valeurs en entrées et produit un jeu de valeurs en sortie.

Algorithmes génétiques

Algorithmes de fouille de données permettant de rechercher une solution à un problème par mutation des attributs (variables) de la population étudiée. Utilisés par exemple dans une étude d'optimisation d'un réseau de points de vente ou d'agences (quelles sont les variables qui expliquent la réussite de telle ou telle agence ? en modifiant telle variable de telle agence améliore-t-on son résultat ?).

Arbres de décision

Algorithmes de fouille de données permettant de répartir une population selon des variables en fonction de l'atteinte d'un objectif. L'intérêt de ce type d'algorithmes est qu'ils sont intuitifs et assez simples d'utilisation et qu'il n'est pas nécessaire de fixer a priori les variables à étudier : l'algorithme détecte lui même les variables les plus discriminantes pour l'objectif fixé. Ces algorithmes sont également utilisés pour définir mesure de risque.

Attribut

Élément descriptif d'un exemple.

CBR (Case Base Reasoning)

Algorithme qui Associe un nouveau cas à un ensemble de cas déjà traités et rencontrés.

Classification

Description des données en procédant à une réduction du nombre des individus par leur regroupement en classes homogènes.

Clustering

Le regroupement des document similaires dans des clusters, appelé aussi classification non supervisée.

Classification hiérarchique

Classification qui produit des suites de classes emboîtées qui définissent une hiérarchie. A chaque étape, les deux classes les plus proches sont recherchées et fusionnées...jusqu'à ce qu'il n'y ait plus qu'une seule classe. Cette méthode consiste à fournir un ensemble de partitions plus ou moins fines, obtenues par les regroupements successifs de parties.

Classification non hiérarchique

Méthode de classification conduisant à une partition de l'ensemble de départ en un nombre n de classes de même niveau. Elle permet de traiter des ensembles d'effectifs assez élevés en optimisant des critères pertinents.

Classification par les centres mobiles

Méthode de classification non hiérarchique décomposant un ensemble d'individus en un nombre n de classes choisies à priori, et ce par un processus itératif convergeant de sélection des représentants de chaque classe (un par classe) qui peut être initialisé au hasard ou par l'utilisateur de la méthode.

Classification par partition

À la différence de la CAH, cette méthode n'est pas hiérarchique (pas de classes imbriquées, pas d'arbre) et elle décompose l'ensemble en un nombre de classes fixé à priori, initialisé par un ou plusieurs représentants. Elle utilise des algorithmes de classification non hiérarchique (le calcul de l'inertie interclasse et intra classe, le regroupement autour des centres mobiles et la méthode des nuées dynamiques). Elle est basée sur l'existence d'un critère global qui mesure la distance entre les individus et par la même, la qualité d'une partition. Les résultats sont présentés sous forme de cartes factorielles.

« Cluster » ou groupe

En scientométrie désigne une classe de mots entre lesquels il existe des liens forts. Ensemble de documents formant un graphe connexe (c'est-à-dire tel qu'il existe toujours un chemin entre deux mots donnés). La technique de classification utilisée est celle par lien simple.

Corpus

Ensemble d'informations issu de l'interrogation sur un thème donné d'une ou plusieurs bases. Ensemble d'énoncés écrits ou enregistrés dont on se sert pour la description linguistique.

Corrélation

Coefficient de corrélation linéaire : mesure l'intensité de la liaison entre deux caractères quantitatifs.

Data mart

Sous- ensemble logique et physique d'un data warehouse.

« Data Mining » ou Fouille de données

Est la découverte et l'extraction, à partir de bases de données, de l'information implicite, non triviale, préalablement non connue et potentiellement utile pour l'utilisateur.

Découverte de connaissance

Processus permettant de trouver des connaissances dans les données.

Déduction

Approche qui consiste à partir des connaissances acquises pour émettre des hypothèses qui seront soumises à l'épreuve des faits.

Dimension

Entité indépendante dans le modèle qui sert de point d'entrée ou de mécanisme de découpage des agrégats.

Donnée textuelle

Donnée alphanumérique issue de la littérature (mot-clé, code, liste des auteurs, texte libre,...).

Entrepôt de données ou « Data Warehouse »

Un entrepôt de données organise et mémorise les données nécessaires pour des traitements d'analyse sur une période de temps longue. C'est une collection de données orientée sur un sujet, intégrée, variable dans le temps, non volatile, utilisée en support d'un processus de prise de décision.

Entropie

Mesure mathématique du désordre d'un attribut multivalué.

HTML (Hypertext Markup Langage)

Langage de description de pages hypertexte élaboré par le W3C (World Wide Web Consortium) pour la présentation de données multimédias d'un serveur Web accessible via l'Internet.

Indexation automatique

Indexation effectuée automatiquement par analyse de texte en se référant à un thésaurus spécifique du domaine.

Induction

Processus d'apprentissage à partir des exemples.

Intelligence artificielle

Domaine de l'informatique traitant de la façon dont les ordinateurs peuvent reproduire ou simuler certaines fonctions généralement associées à l'intelligence humaine, telles que la vision, la reconnaissance de la parole, le raisonnement et le traitement des connaissances, la possibilité d'apprendre à partir d'une expérience passée et celle de faire des déductions raisonnables à partir d'informations incomplètes ; on parle aussi d'informatique avancée ou de système à base de

connaissances, le traitement portant sur des connaissances symboliques et non sur des données numériques.

Java

Langage de développement informatique portable conçu par Sun, en passe de standardisation sur Internet et plus généralement dans monde applicatif.

Lemmatisation

En lexicographie automatique : opération consistant à regrouper les formes occurrentes d'un texte ou d'une liste sous des adresses lexicales. La première étape concerne le regroupement des formes fléchies sous la forme type leur servant d'adresse lexicale ou lemmatisation à proprement parler. La seconde étape consiste en la séparation des formes servant d'adresses lexicales quand elles sont homographes.

Modèle

Représentation aussi fidèle que possible, mais toujours simplifiée, d'une réalité.

Mot-clé

Dans une notice bibliographique, mot qui décrit le contenu du document.

Nettoyage des données : opération qui consiste à exclure les données incomplètes, manquantes ou erronées.

Niveau de confiance

Étant donné l'association $X \Rightarrow Y$, le niveau de confiance est le pourcentage d'enregistrement qui contiennent X ou Y rapporté au nombre d'enregistrement qui contiennent.

Niveau de support

Étant donné l'association $X \Rightarrow Y$, le niveau de support est le pourcentage d'enregistrement qui contiennent X ou Y.

Occurrence

Toutes les fois qu'un élément linguistique (type) figure dans un texte, on parle d'occurrence (token). L'apparition du terme "socialisme" dans un texte analysé du point de vue linguistique sera une occurrence du mot "socialisme".

OLAP (OnLine Analytical Processing)

Désigne une catégorie d'outils d'exploitation de donné qui permettent de visualiser des valeurs dans plusieurs dimensions.

Recherche de règle d'association

Traitement mettant en œuvre différents algorithmes visant à extraire des dépendances entre différentes variables représentatives de données.

Règle de production

Mode d'expression déclaratif et procédural des connaissances sous forme de règles logique du type «si…, alors…».

Conclusion Générale

Réseaux bayésiens

Algorithme d'apprentissage qui cherche à construire un modèle de classification en utilisant la théorie des probabilités conditionnelles de bayes.

Réseaux de neurones

Algorithme d'apprentissage composé d'une structure de neurones connectés associant des entrées à une ou plusieurs sorties.

Réseau sémantique

Formalisme de représentation des connaissances tenant compte de leur sens, sous forme de graphes. Les nœuds du graphe représentent des concepts (objets ou événements) et les arcs, des relations entre ces concepts.

Segmentation

Opération visant à découper une population hétérogène en sous-ensemble plus homogène disposant de caractéristique commune.

Sélection

Processus utilisé dans les algorithmes génétiques permettant de sélectionner les effectifs qui donnent naissance à une génération suivant.

Text mining ou « Fouille de données textuelles »

Analyse de grands volumes de données textuelles. Est un prolongement de la fouille de données.

Traitement automatique des données

Ensemble des opérations réalisées par des moyens automatiques, relatif à la collecte, l'enregistrement, l'élaboration, la modification, la conservation, la destruction, l'édition de données et, d'une façon générale, leur exploitation.

Traitement des données

Séries d'opérations visant à valoriser un corpus de données (tri des informations, évaluation, analyse, élaboration de synthèse).

Annexe A

Concepts de Base de Data Mining

1. Apprentissage automatique

En général, l'apprentissage est défini comme étant l'action d'acquérir intentionnellement des connaissances par expérience. L'apprentissage automatique est l'automatisation de cette action.

L'apprentissage automatique examine des exemples précédents et leurs résultats et apprend comment reproduire ces derniers et faire des généralisations au sujet de nouveaux cas. [PCC XX]

On distingue deux types d'apprentissage, selon qu'on dispose d'une variable à expliquer ou non: *Apprentissage supervisé et Apprentissage non supervisé.*

A) Apprentissage supervisé *et* Apprentissage non supervisé [LSP 03]

Distinguons deux types de problèmes : la présence ou non d'une variable Y à *expliquer*. Dans le premier cas il s'agit bien d'un problème de modélisation ou apprentissage supervisé ; en utilisant une fonction " _ " susceptible, au mieux selon un critère à définir, de reproduire Y ayant observé X.
Y = _(X) + " où " symbolise le bruit ou erreur de mesure.

Dans le cas contraire, en cas d'absence d'une variable à expliquer, il s'agit alors d'apprentissage dit non supervisé. L'objectif généralement poursuivi est la recherche d'une typologie ou taxinomie des observations : comment regrouper celles-ci en classes homogènes mais les plus dissemblables entre elles. C'est un problème de classification (*clustering*).

B) Data Mining et Apprentissage Automatique (AA) :

D'après la définition de l'apprentissage automatique, on observe une certaine ressemblance entre le Data Mining et l'apprentissage automatique. Cependant, ils diffèrent sur certains points :

- ✓ le Data Mining est concerné par des bases de données très grandes et réelles, alors que l'Apprentissage Automatique est généralement concerné par de petits ensembles de données [PCC XX] ;

- ✓ l'apprentissage automatique est un domaine large qui inclut l'apprentissage à partir d'exemples, mais aussi, l'apprentissage renforcé, l'apprentissage avec un expert et

Conclusion générale

Nous avons utilisé à travers ce mémoire, une méthode (FIHC) pour résoudre le problème du clustering (classification non supervisée) qui est une des tâches de « Data Ming». Résoudre un problème de " clustering des documents " consiste à partitionner un ensemble de documents en groupes homogènes, sans aucune connaissance préalable du nombre des clusters.

La présence d'un processus de développement formalisé, bien défini, est un facteur de réussite d'un projet, c'est pourquoi nous avons suivi le modèle en cascade utilisant UML comme langage de modélisation.

Au cours de notre projet, nous avons présenté et implémenté la méthode FIHC qui est basée sur les itemsets fréquents. Cette méthode surpasse plusieurs méthodes traditionnelles en terme d'exactitude. En effet, l'utilisation de la fonction " Score " permet d'affecter les documents aux meilleurs clusters ; en plus les techniques de taille utilisées améliorent cette exactitude. La solution résultant de cette méthode est une arborescence de clusters, où chaque cluster a une étiquette qui récapitule avec concision les membres de son groupe. Le rendement de cette méthode est un fichier XML qui peut être échangé avec d'autres outils d'extraction de texte (texte mining).Cette méthode a comme avantage de produire toujours le même résultat pour le même jeu de documents, c'est-à-dire que l'ordre des données d'entrée ne peut affecter le résultat groupant.

Cette étude très enrichissante, nous a donné l'occasion de nous plonger dans le problème de clustering, ce qui nous a permis d'affiner les connaissances acquises. Nous avons eu la chance d'être intégrées à un projet important impliquant un grand nombre de personnes et faisant appel à des technologies novatrices.

Le début été difficile, ce qui est principalement dû à la multiplicité des concepts utilisés dans le clustering, d'autant plus que nos seules sources documentaires étaient en anglais. Précisons en outre, qu'il existe une combinaison des différentes méthodes et une grande variété de données à manipuler.

Ce projet nous a aussi permis de progresser en termes de rigueur, et de méthodologie, aussi bien concernant la conduite du projet que la programmation. Il nous a de plus, permis d'approfondir nos connaissances sur le « Data mining »et ses différentes techniques, sur le langage UML, sur le langage C# et sur l'architecture. Net.

Au cours de un certain nombre de perspectives nous sont apparues :

Premièrement, l'arbre de clusters peut être mis à jour lorsqu'un nouveau document arrive. Cette tâche peut être accomplie en assignant le nouveau document au plus semblable cluster, mais un inconvénient peut être apparaître c'est que l'exactitude groupante peut être dégradée dans le temps parce que l'ensemble " global frequent itemset " ne peut pas refléter l'état actuel des documents dans la base. Cela peut faire l'objet d'un prolongement de recherche.

bien d'autres, alors que le Data Mining n'utilise que l'apprentissage à partir d'exemples.

2. OLAP

Le terme OLAP (*On-Line Analytic Processing*[1]) se rapporte de la technologie qui permet à des utilisateurs des bases de données multidimensionnelles de produire des sommaires descriptifs ou comparatifs ("vues") des données.

La technologie OLAP peut être intégrée dans les grandes entreprises qui utilisent des systèmes de base de données pour permettre à des analystes et des directeurs de surveiller le déroulement des affaires (par exemple, les divers aspects du processus de fabrication ou les nombres et les types de transactions réalisées à différents endroits) ou du marché [STA 02].

OLAP et Data Mining [TWO 99]

OLAP fait partie de la gamme des outils de support de décision. Les outils traditionnels de requêtes et d'états décrivent ce qui est dans une base de données. OLAP va plus loin; il est utilisé pour répondre *à* "pourquoi" certaines choses sont vraies. L'utilisateur forme une hypothèse au sujet d'un rapport et la vérifie avec une série de requêtes. Par exemple, un analyste pourrait vouloir déterminer les facteurs qui mènent aux défauts des prêts dans une banque. Il pourrait au commencement présumer que les gens qui ont de bas revenus et une dette élevée sont de mauvais clients et analyser la base de données avec OLAP pour vérifier (ou réfuter) cette prétention. En d'autres termes, l'analyste OLAP génère une série de configurations et de rapports hypothétiques et utilise des requêtes pour les vérifier ou pour les réfuter.

Le Data Mining est différent d'OLAP parce que, plutôt que de vérifier les configurations hypothétiques, il emploie les données elles-mêmes pour découvrir de telles configurations. C'est essentiellement un processus inductif. Par exemple, supposons que l'analyste qui a voulu identifier les facteurs de risque pour un prêt a utilisé un outil Data Mining. L'outil pourrait découvrir que les gens avec une dette élevée et des bas revenus sont de mauvais clients (comme ci-dessus), mais il pourrait aller plus loin et découvrir que l'expérience (professionnelle) est également une cause déterminante de risque.

3. Les statistiques

Les statistiques sont le moyen naturel pour associer des données et explorer les relations entre elles et leur importance. L'idéal est d'arriver à prendre en compte le facteur temps et de permettre ainsi la prévision du comportement d'un élément donné.

Les statistiques et Data Mining

Bien que le Data Mining soit le descendant des statistiques et, selon certains, le successeur des statistiques et bien qu'ils aient le même but, modéliser de manière compréhensible la réalité. En théorie le Data Mining est différent des statistiques. En effet, les algorithmes de Data Mining sont exploratoires, alors que les statistiques sont

[1] : *Traitement Analytique En ligne*

confirmatoires, c'est -à -dire qu'elles interviennent pour confirmer une hypothèse [LEF 98].

Les statistiques sont également limitées par le nombre de variables à analyser : dans de grandes bases de données, le nombre possible de relations entre variables est trop important pour que l'outil puisse les tester une à une. Il est donc essentiel d'utiliser des techniques de recherche intelligentes. Les techniques statistiques seront toujours utilisées pour s'assurer de la force de la relation.

Les techniques de D*ata Mining* apportent un gain énorme tant en performance qu'en maniabilité ou en temps de travail. La possibilité de réaliser ses propres modèles statistiques sans besoin de sous-traiter ou de se concerter avec un statisticien apporte une grande liberté aux utilisateurs opérationnels [PMS XX].

Annexe E

Le Langage UML

Bref historique d'UML

Les langages de modélisation orientés objet ont fait leur apparition entre le milieu des années soixante-dix et la fin des années quatre-vingt, quand les spécialistes de méthode confrontés à un nouveau genre de langages de programmation orientés objet et à des applications de plus en plus complexes, ont commencé à expérimenter de nouvelles approches d'analyse et de conception .En 1989 et 1994 le nombre de méthodes orientées objet est passé de moins de 10 à plus de 50.La majorité des utilisateurs ne trouvaient pas de langage de modélisation qui répondre entièrement à leur besoins ; ils ont alors déclenché la "guerre des méthodes ". Basées sur l'expérience acquise, de nouvelle générations de ces méthodes sont apparues .Certaines d'entre elles se sont nettement détachées du lot et plus spécialement la méthode BOOCH, OOSE (Object-Oriented Software Engineering) de Jacobson et OMT (Object Modeling Techenique) de Rumbaugh. Chacune de ces méthodes constituait une méthode complète mais présentait aussi des avantages et des inconvénients.

Certaines idées déterminantes ont émergé progressivement au milieu des années quatre vingt dix lorsque Grady Booch, Ivar Jacobson et James Rumbaugh se sont associés pour créer un langage de modélisation unifié et ont donné naissance à UML.

UML (Inified Modeling Langage) est un langage standard conçu pour l'écriture de plans d'élaboration de logiciels. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à forte composante logicielle.

UML est adapté à la modélisation de système, depuis les systèmes informatiques d'entreprise jusqu'aux applications distribuées basées sur le Web, en passant par les systèmes en temps réel embarqués. C'est un langage très expressif qui couvre toutes les perspectives nécessaires au développement puis au déploiement d'un tel système.

Le modèle conceptuel d'UML

Le modèle conceptuel implique l'assimilation de trois éléments essentiels : les briques de base d'UML, les règles qui déterminent la manière d'assembler ces éléments et quelques mécanisme généraux qui s'appliquent à UML dans son ensemble.

Les briques de base

Les briques de base d'UML sont :

- les éléments ;
- les relations ;
- les diagrammes.

Les éléments sont les abstractions essentielles à un modèle. Les relations constituent les liens entre ces éléments ; les diagrammes les regroupent en des ensembles dignes d'intérêt.

Annexe B

Algorithmes D'extraction des Règles associatives

L'algorithme Apriori TID [GAR 00]

Comme nous l'avons vu dans le chapitre II , l'algorithme Apriori nécessite k passes sur la base, k étant la taille du plus grand ensemble susceptible d'être fréquent, ce qui signifie une utilisation d'un énorme espace de mémoire.

Une optimisation possible consiste à générer en mémoire, pendant la première passe, les identifiants (TID) des transactions pour chaque 1-itemset fréquent. Dans la suite , les listes de TID correspondant à chaque k-itemset sont gardées, mais son calcul se fait toujours à partir des deux (K-1)-itemsets contenant un élément de moins ; le comptage se fait simplement par intersection des deux listes de TID des deux (k-1)-itemsets sources.

L'utilisation de l'algorithme Apriori avec cette optimisation permet l'apparition d'un nouveau dérivé de l'algorithme Apriori proposé par Agrawal et Srikant en 1994 [AGR 94], qui est Apriori TID ; son pseudo code correspondant apparaît dans la figure B.1.

```
L1= {1-itemset fréquent};
C1chap= base de données D;
Pour (k=2; Lk-1≠∅ ; k++) faire
   Ck= Apriori- gen (Lk-1); // la génération de nouveau candidats
   Ckchap= ∅ ;
   Pour tout entries t ∈ Ck-1chap faire // determiner tous les candidats itemsets dans Ck
   contenus dans la transaction avec un identifiant t.TID
      Ct = {c ∈ Ck | (c - c[k]) ∈ t.set-of-itemsets ∩ (c - c [k – 1]) ∈ t.set-of-itemsets};
      Pour tout candidat c∈ Ck faire
         c.count++ ;
      Fin
      Si ( Ct≠∅) alors
         Ckchap+ =< t.TID, Ct>;
   Fin
   Lk={c∈ Ck | c.count ≥ minsup};
Fin
Answer = ∪k Lk ;
```

Figure B.1 : L'algorithme Apriori TID [AGR 94]

La construction de la liste des TID est faite après avoir déterminé les 1-itemsets fréquents, ce qui est plus efficace en taille mémoire mais nécessite deux passes. La première passe permet d'éliminer les items non fréquents, et réduit donc les listes de TID en mémoire construites dans une deuxième passe.

La génération des listes de TID en mémoire en parallèle dès la première passe est plus efficace. Cependant, il n'est possible d'éliminer des listes qu'après le comptage total de la base, lorsqu'on sait qu'un 1-itemset ne sera pas fréquent.

L'algorithme Apriori partitionné [GAR 00]

Cet algorithme proposé par [SAV 95], permet de résoudre le problème de l'espace mémoire de l'algorithme Apriori. Son principe consiste à diviser la base en Q partitions, de sorte que chaque partition tienne en mémoire, et que chaque partition soit traitée indépendamment et les itemsets fréquents découverts pour chaque partition.

La validité de l'algorithme est basée sur le lemme suivant : pour qu'un itemset soit globalement fréquent, il doit être fréquent dans au moins une partition. Ayant obtenu les itemsets fréquents sur chaque partition, il suffit dans une passe finale d'explorer l'union des itemsets fréquents sur la base, ce qui est fait par un comptage simple sur la base avec l'élimination de tous les itemsets non globalement fréquents, mais localement fréquents dans une partition.

L'avantage de cet algorithme est qu'il nécessite deux passes au plus. Il est aussi facilement parallélisable, les partitions pouvant être traitées indépendamment. Dans ce cas, il faut cependant beaucoup de mémoire pour tenir les partitions et les listes de TID en mémoire.

L'algorithme « Count Distribution » [GAY 04]

Proposé par R. Agrawal et J.C Shafer en 1996 [AGR 96], cet algorithme est une version parallélisée de l'algorithme Apriori. Chaque processeur traite sa portion locale de la base de transactions et chacun d'eux calcule des candidats "locaux", il minimise le coût des communications, mais est freiné par la structure de données utilisée qui, lors du traitement d'une base de données de grande taille, impose de coûteuses opérations d'entrée/sortie. Seuls les supports des candidats sont transmis lors d'une opération de Broadcast (*"All to all")* par un processeur aux autres qui calculent chacun le support global de l'itemset.

Les performances des algorithmes *Apriori* et *Count Distribution* sont limitées, même en parallèle, pour diverses raisons. Tout d'abord, ces différents algorithmes (qui pour rappel ont la même base) nécessitent des lectures de la base entière à chaque itération. En outre, ils énumèrent tous les itemsets candidats autant de fois qu'ils sont trouvés dans la base de données même si les transactions sont identiques.

En plus de tout ceci, la base de données de transactions est considérée comme ayant une conception horizontale : une transaction possède un identifiant, suivi des items qu'elle contient. Il apparaît que ce type d'organisation de données n'est pas très adapté pour la phase d'évaluation du support de ces algorithmes.

L'algorithme « Data Distribution » [MAH 04]

Il s'agit encore d'un algorithme issu d'Apriori et proposé par R. Agrawal et J.C. Shafer en 1996 [AGR 96], il diffère de "Count Distribution" dans le sens où les processeurs se partagent le travail non plus par portion de base de données mais par candidats.

Chaque processeur traite un ensemble de candidats différents, ce qui impose d'accéder aux portions de la base de transaction des autres processeurs, ce qui augmente (drastiquement) la charge des communications. Il s'avère meilleur que "Count Distribution" lorsque la base contient beaucoup d'items distincts et lorsqu'on choisit un support minimum peu élevé.

Afin de réduire la charge en communication de cet algorithme, une version considérant les processeurs comme étant sur un anneau logique a été créée, c'est l'algorithme "Intelligent Data Distribution".

L'algorithme bitmap [GAR 00]

Deux algorithmes basés sur des index bitmap plutôt que sur des listes d'identifiants ont été proposés dans [GAR 98]. L'intérêt des bitmaps est d'être plus compacts que les listes et de pouvoir ainsi tenir en mémoire, car l'union et l'intersection de vecteurs binaires sont aussi beaucoup plus rapides à calculer que l'union et l'intersection de listes, d'où un remarquable gain en CPU.

Un index bitmap est un tableau de bits de dimension K*P, où K est le nombre de lignes de la table indexée et P le nombre de valeurs possibles pour l'attribut indexé. Chaque colonne du tableau correspond à une valeur possible de l'attribut indexé. Le bit (i, j) est à 1 si la ligne i possède la valeur P_j correspondant à la colonne j du tableau.

Dans le cas du panier de la ménagère, on indexe la table des transactions sur les produits. Mais dans le cas d'une table imbriquée, on obtient l'index représenté dans la **figure B.1** pour l'exemple indiqué. Notez que la construction d'index bitmap sur des attributs à grands nombres de valeurs possibles est difficile. Par exemple, un index bitmap sur le prix nécessite tout d'abord un découpage des prix possibles en intervalles significatifs.

Ménagère	Produits	Prix
• 1	$\{P_1, P_3, P_5\}$	120
• 2	$\{P_2, P_3\}$	70
• 3	$\{P_4\}$	150
• 4	$\{P_2, P_5\}$	110
• 5	$\{P_3, P_4, P_6\}$	220

	P_1	P_2	P_3	P_4	P_5	P_6
•	1	0	1	0	1	0
•	0	1	1	0	0	0
•	0	0	0	1	0	0
•	0	1	0	0	1	0
•	0	0	1	1	0	1

Figure B.2 : Exemple d'index bitmap pour un fichier de 5 transactions

L'algorithme bitmap naïf [GAR 98] est une adaptation de Apriori aux bitmaps. Comme lui, il réalise la génération d'un k-itemset fréquent à partir de deux (k-1)-itemsets fréquents, et le support d'un k-itemset est calculé par comptage du nombre de bits dans l'intersection des vecteurs de la bitmap associés aux composants.

Afin de gagner de l'espace mémoire, il est préférable de comprimer les bitmaps. Par exemple, avec 1 million de transactions et 1 000 produits, il faut 1 000 000 000 bits, soit 125 mega octets. Ceci peut tenir en mémoire, mais il n'est pas rare qu'un supermarché gère beaucoup plus de références.

Une technique de compression dérivée de l'encodage de monotonie par des longueurs ("run lengh encoding") consiste à coder alternativement les longueurs des séquences de bits à 0 puis à 1, mais un tel codage n'est guère efficace. Une autre technique est d'indexer la bitmap par une bitmap de niveau supérieur et d'éviter de mémoriser les bitmaps des k-itemsets pour k>1. En effet, il est facile de compter un k-itemset à partir des 1-itemsets sources en généralisant à k itemsets (ensembles) la procédure précédente.

L'algorithme « Eclat » [MAH 04] [GAY 04]

Introduit par M. J. Zaki en 1997 [ZAK 97], cet algorithme repose sur le découpage de la base en classes d'équivalences et distribution de la charge de travail sur tous les processeurs. On considère que deux itemsets sont dans la même classe d'équivalences, désignés par les items qu'ils contiennent dans l'ordre lexicographique, ils possèdent un préfixe commun. Par exemple, les itemsets ABC et ABD sont dans la classe d'équivalence AB.

Au lieu de transmettre des supports locaux ou des portions de base de données comme dans les principaux algorithmes dérivés d'Apriori, cet algorithme fonctionne en transmettant les listes de transactions correspondant à chaque classe d'équivalence au processeur qui s'occupe de celle-ci. En plus de cet avantage, l'algorithme Eclat ne fait que deux parcours seulement de la base de données. Il parcourt une première fois la base pour construire les 2- itemsets puis une seconde fois pour la mettre sous forme verticale. L'algorithme Eclat est composé de trois phases principales :

- la phase d'initialisation, qui consiste à la construction globale des 2-itemsets ;
- la phase de transformation, qui permet le partitionnement de l'ensemble des 2-itemsets fréquents et la distribution de ces partitions aux autres processeurs, c'est une transformation verticale de la base ;
- la phase asynchrone, qui permet la construction des k-itemsets fréquents.

Le pseudo code se présente ainsi :

Phase d'initialisation
Scanne la partition locale de la base.
Calcul des comptes locaux des itemsets de taille 2.
Construction des comptes globaux de L_2.
Phase de transformation
Partitionnement de L_2 en classes d'équivalences.
Distribution de L_2 sur les processeurs par classe d'équivalence.
Transformation de la base locale en base verticale.
Envoi des listes de transaction aux autres processeurs.
L_2 **local** = listes de transaction des autres processeurs.
Phase asynchrone
Pour chaque classe d'équivalence E_2 dans L_2 **local**
Construction (E_2) ;
Phase finale de réduction
Regroupement des résultats et calcule des associations.

Figure B.3 : L'algorithme Eclat [MAH 04]

La tâche la plus coûteuse de l'algorithme Eclat est en général la transmission des listes de transactions de chaque item. En effet, sur des bases où les items sont répartis de manière homogène, chaque processeur doit transmettre des listes de taille importante à tous les autres, ce qui est considéré comme le point faible de l'algorithme Eclat .

Annexe C

Les Bases de L'analyse de Données

L'évaluation des similarités entre entités textuelles est un des problèmes centraux dans plusieurs disciplines traitant de documents, comme l'analyse de données textuelles, la recherche documentaire ou l'extraction de connaissances à partir de données textuelles. Dans chacun de ces domaines, des notions de similarité sont en effet utilisées pour une large variété de traitements.

- En analyse de données textuelles (ADT), les similarités sont utilisées pour la description et l'exploration de données, pour l'identification de structures cachées et pour la prédiction.
- En recherche documentaire (RD), l'évaluation des similarités entre documents et requêtes est utilisée pour identifier les documents pertinents par rapport à des besoins d'information exprimés par les utilisateurs.
- En " text mining " (TM), les similarités sont utilisées pour produire des représentations synthétiques de vastes collections de documents, dans le cadre de procédures d'extraction d'information à partir de données textuelles.

Cette annexe a pour objectif de rappeler certaines techniques statistiques élémentaires. Qui sont les fondations des outils de Data Mining. Elles consistent toutes plus ou moins à utiliser des données, à regrouper ou à relier les éléments qui se ressemblent et à séparer ceux qui diffèrent.

Nous précisons dans ce qui suit la manière dont se construisent les notions de ressemblance et de différence, à partir des concepts de similarité, de distance et de probabilité.

La notion de similarité

La similarité sur des variables disjonctives [LEF 98]

On dit que deux objets A et B, décrits par p attributs, sont similaires si le maximum d'attributs sur les p attributs sont identiques entre eux. Ainsi, si l'on effectue une comparaison entre une voiture à moteur, une diligence et une calèche sur les cinq variables suivantes : roues, plancher, portes, moteur et toit, on construit le tableau suivant :

	Voiture	Diligence	Calèche
Présence de roues	Oui	Oui	Oui
Présence d'un plancher	Oui	Oui	Oui
Présence de portes	Oui	Oui	Non
Présence d'un moteur	Oui	Non	Non
Présence d'un toit	Oui	Oui	Non

Tableau C.1 : Un exemple de la similarité des variables disjonctives [LEF 98].

Ce tableau permet de constater de manière intuitive que la diligence est plus proche de la voiture que la calèche. Il est facile de se rendre compte que la voiture et la diligence ont quatre points communs alors que la calèche et la voiture n'en ont que deux. En statistique, la notion de point commun est dénommée *coïncidence*. Les coïncidences permettent de construire une mesure quantitative de la similarité entre des objets.

Il existe deux type de coïncidences : les coïncidences positives et les coïncidences négatives, selon que les deux objets présentent ou non la même caractéristique. La matrice suivante illustre les différents types de coïncidences :

Valeur de l'attribut pour l'objet A	Valeur de l'attribut pour l'objet B	Coïncidence
Oui	Oui	Positive
Oui	Non	Non-coïncidence
Non	Oui	Non-coïncidence
Non	Non	Négative

Tableau C.2 : La coïncidence positive et la coïncidence négative [LEF 98].

La somme des coïncidences et des non- coïncidences est égale au nombre de variables de comparaison et détermine un indice de similarité qui peut varier entre 0 et 1 : 0 signifie que les éléments n'ont aucun point commun, 1 signifie qu'ils sont identiques en tout point.

Selon la manière de prendre en compte des coïncidences négatives, on obtiendra différentes formules, et donc différentes valeurs de similarité. L'approche la plus restrictive, celle dite de Russel, n'accorde aucun poids aux coïncidences négatives, parce qu'elle considère comme le seul élément comparatif fiable les coïncidences positives sur le nombre de variables de comparaison. L'approche la plus extensive accorde le même poids aux coïncidences positives et aux coïncidences négatives, soit la somme de toutes les coïncidences sur le nombre de variables de comparaison. Cet indice, l'indice de Sokal, est plus difficile à utiliser et exige des critères de comparaison des objets valables. Une approche intermédiaire consiste à accorder un poids moins important aux coïncidences négatives qu'aux coïncidences positives.

Le choix du bon indice de coïncidence ne peut s'effectuer qu'après une analyse des variables de comparaison et une étude de la distribution des valeurs.

On constate que la similarité dépend fortement de l'indice choisi. Le choix du bon indice conditionne les résultats et souligne l'importance de la sélection des variables préalables à l'analyse des données.

Cette première notion de similarité construit des indicateurs uniquement sur des données de type disjonctif (oui /non), ce qui limite fortement leur utilisation.

La similarité sur des variables quelconques [LEF 98]

Compte tenu de l'hétérogénéité des variables, il s'agit ici de déterminer un indice composite de toutes les similarités sur différents critères :

- la similarité sur des variables disjonctives (oui /non) est égale à 1 si les deux objets présentent la caractéristique (coïncidence positive) ;
- la similarité sur des variables qualitatives (bleu, vert, rouge) est égale à 1 si les deux objets présentent la caractéristique ;

- la similarité sur des variables quantitatives (franc, mètre, âge, gramme) mesure l'écart entre les objets de manière relative par rapport à l'étendue de la distribution de la variable.

Prenons comme exemple un couple qui souhaite sélectionner une station de sports d'hiver pour ses prochaines vacances. Il détermine dans un premier temps une grille de sélection qui correspond à ses critères. Il recherche ensuite, parmi trois stations, celle qui se rapproche le plus de ces critères de choix.

	Station cible	Station A	Station B	Station C
Prix forfait	1500 F	1800 F	2100 F	1400 F
Altitude	1800 m	1500 m	1800 m	2300 m
Garderie	Oui	Non	Oui	Non
Piste dominante	Verte	Bleue	Rouge	Verte

Tableau C.3 : Exemple de la similarité de variable quelconques [LEF 98].

La mesure de la similarité entre la station cible et la station A est déterminée de la façon suivante :

- Pour le critère Prix forfait, il faut déterminer l'étendue de la distribution. celle-ci est égale à la valeur maximale (2100 F) moins la valeur minimale (1400 F), soit 700 F. La similarité entre la station cible et la station A est notée :

  S (Cible, A, Prix forfait)= 1- ([1500 F-1800 F]/ 700 F)= 1-0.428 = 0.572.

- Pour le critère Altitude, l'étendue est de 2300 m- 1500 m= 800 m et la similarité est :

  S (Cible, A, Altitude) = 1- ([1800 m-1500 m]/ 800 m) = 1-0.375 = 0.625.

- Pour le critère Garderie, S (Cible, A, Garderie)= 0 car la station A n'a pas de garderie.

- Pour le critère Piste dominante, S (Cible, A, Piste dominante)=0 car Bleue est différent de Verte.

La similarité entre la station cible et la station A est égale à (0.572 + 0.625 + 0 +0) / 4, soit 0.299. La même démarche sur les stations B et C donne 0.535 et 0.486.

Cette introduction sur les similarités montre qu'il est facile de transformer des données hétérogènes (disjonctive, qualitative et quantitative) en un indicateur synthétique. Elle souligne également qu'une analyse de la signification des variables et de l'objectif recherché peut profondément modifier les résultats d'une mesure de similarité.

La notion de distance [LEF 98]

Compte tenu de l'hétérogénéité des types de variables exploitées dans une analyse de Data Mining, il est fréquent de procéder à des transformations préalables pour positionner les individus dans un espace multidimensionnel.

La notion de similarité trouve son complément (si ce n'est que la similarité, contrairement à la distance, n'est pas nécessairement symétrique) dans la notion de distance, qui mesure l'écart dans cet espace.

La distance s'écrit Distance (A, B)= 1-Similarité (A, B). Dans l'exemple précédent, les distances deviennent donc :

- Distance (Cible, A)=1- 0.517 = 0.483 ;
- Distance (Cible, B)=1- 0.476 = 0.524 ;
- Distance (Cible, C)=1- 0.833 = 0.167.

Deux objets similaires ont donc entre eux une distance nulle ; en revanche, la distance maximale sépare deux objets différents. Cette transformation de la similarité en distance permet de donner une représentation graphique du choix de couple de skieurs.

Il s'agit d'une première approche permettant de positionner des objets dans une espace. Plus les points sont proches, plus les individus sont similaires. Ce prédicat est la base des techniques de classification. Celles-ci utilisent ce même principe de distance pour construire la classification des objets en groupes. Un groupe s'obtient par l'agrégation de *n* objets proches. Par itération de proche en proche, ce processus de regroupement finit par classifier l'ensemble de population.

Nous allons présenter quelques techniques de base, qui illustrent la multiplicité des critères de regroupement possibles selon le sens de la classification (ascendante et descendante).

→ La notion de distance et la classification hiérarchique

Il existe de multiples façons de calculer des distances ; nous nous intéresserons ici à la distance la plus commune, celle qui fait le principe des cours de géométrie du cycle secondaire : la distance euclidienne.

Comme exemple, nous prenons les notes suivantes attribuées à cinq produits de grande consommation sur l'esthétique et la mémorisation de l'accroche publicitaire (la note 1 signifie faible esthétique ou faible mémorisation).

	Esthétique	Mémorisation
Produit A	1	1
Produit B	1	2
Produit C	4	3
Produit D	4	5
Produit E	2	4

Tableau C.4 : Le tableau des notes des produits [LEF 98].

D'après la représentation graphique citée dans [LEF 98], nous remarquons que la notion de distance fait intuitivement référence à l'éloignement entre les points. Le graphe nous permet de constater que les produits A et B sont très proches et que la distance est égale à 1, soit (2-1) sur l'axe Mémorisation. Tandis que la distance entre les produits A et D se calcule en utilisant les propriétés des triangles rectangles et du théorème de Pythagore, selon lesquelles le carré de l'hypoténuse est égal à la somme des carrés des deux autres côtés.

La distance entre B et E, notée d (B, E) est telle que :

$d(B, E)^2 = d(B, F)^2 + d(F, E)^2$.

Dans notre exemple, d (B, E) vaut donc $\sqrt{(4-2)^2+(2-1)^2}$, soit 2.24. La représentation des distances entre les différents produits est synthétisée dans la matrice suivante :

	A	B	C	D	E
A	-	1.00	3.61	5.00	3.16
B	1.00	-	3.16	4.24	2.24
C	3.61	3.16	-	2.00	2.24
D	5.00	4.24	2.00	-	2.24
E	3.16	2.24	2.24	2.24	-

Tableau C.5 : Matrice des distances [LEF 98].

La matrice des distances est totalement symétrique. En effet d (A, B)=d (B, A). En conséquence, nous ne reporterons que la partie supérieure de la matrice.

Les algorithmes de classification regroupent pas à pas les points les plus proches pour former un nouveau groupe. Tous les regroupements faits sont cités dans les matrices ci-dessous à partir des distances les plus grandes.

	AB	C	D	E
AB	-	3.61	5.00	3.16
C		-	2.00	2.24
D			-	2.24
E				-

Tableau C.6 : Matrice représentant le premier regroupement [LEF 98].

Puis on regroupe C avec D, ce couple ayant la distance la plus courte (2).

	AB	CD	E
AB	-	5.00	3.16
CD		-	2.24
E			-

Tableau C.7 : Matrice représentant le deuxième regroupement [LEF 98].

On regroupe CD et E, qui ont entre eux la distance la plus courte (2.24).

	AB	CDE
AB	-	5.00
CDE		-

Tableau C.8 : Matrice représentant le troisième regroupement [LEF 98].

Le travail de regroupement est terminé et permet de construire l'arbre de classification à partir des distances de regroupement. Ce graphique est appelé dendogramme.

Après avoir introduit les notions de similarité et de distance, et d'après les domaines cités auparavant, nous allons détailler quelques similarités et distances utilisées dans ces domaines.

Comme la similarité à base de cosinus qui est souvent utilisée dans la recherche documentaire, la distance du Chi-deux utilisée en analyse des documents textuelles (ADT) et la distance angulaire utilisée en recherche documentaire (RD). Ces distances et similarités se basent sur l'utilisation des vecteurs conceptuels des documents.

Similarités à base de cosinus [DEL 02]

En recherche documentaire, le problème principal est d'évaluer les similarités entre les éléments stockés dans une base documentaire. Dans le cadre du modèle vectoriel classique, les approches utilisant des métriques à base de cosinus sont les plus fréquentes (Salton et Buckley, 1990). Différentes variations de cette approche ont été implémentées dans le système SMART, bien connu dans le domaine (Salton et Buckley, 1988).

On définit une matrice T dont les lignes sont :

$D_i=(w_{i,j})_j$, avec $w_{i,j} = 0{,}5(1+ p_{i,j}/\max_l (p_{i,j})).\ \log (N/n_j)$ si $p_{i,j}>0$
Et $w_{i,j} = 0$ sinon

$w_{i,j}$ est le poids du terme T_j dans le document D_i ;

$p_{i,j}$ est la fréquence relative de T_j dans D_i ;

N représente le nombre total de documents dans la base documentaire e ;

n_j le nombre de documents contenant le terme T_j.

Les mesures utilisées dans cette application portent en réalité sur la dissimilarité. Ce sont :

1) atn $(D_i , D_{i'}) = D_i \bullet D_{i'}$, où $\bullet$ est le produit scalaire.
2) atc $(D_i, D_{i'}) = \cos (D_i, O, D_{i'})$, où O est l'origine de l'espace de représentation.

Les propriétés du produit scalaire font que sous les conditions :

$\text{Max}(D_i \setminus D_{i'}) < \text{Max}(D_i / D_{i'})$ et $\text{Max}(D_{i'} / D_i) < \text{Max}(D_{i'} \setminus D_i)$,

où $D_i \setminus D_{i'}$ est la restriction de D_i aux parties nulles de $D_{i'}$ et $D_i / D_{i'}$ est la restriction de D_i aux parties non nulles de $D_{i'}$.

La dissimilarité atn n'est sensible qu'aux parties partageant les profils lexicaux des entités textuelles comparées. Elle est de ce fait bien adaptée pour le calcul de similarité dans les cas où les similarités entre parties de documents sont suffisantes pour entraîner les similarités entre les documents pris dans leur ensemble. En simplifiant, la mesure atn est sensible au « nombre» de mots communs entre les documents comparés.

Dans le domaine de la recherche documentaire, la dissimilarité atn peut être utilisée pour rechercher de l'information « à l'intérieur » des documents, par exemple : dans une partie ou une phrase de ces derniers. Pour comparer des textes de manière plus générale, la dissimilarité atc, plus sensible à la « proportion » de termes communs, est mieux adaptée.

La distance du Chi-deux [DEL 02]

L'analyse de données textuelles s'intéresse essentiellement à l'évaluation de similarités entre documents. Usuellement, chaque document est représenté par son profil lexical : un "tuple" ou un document D_i qui contient les fréquences des unités textuelles dans le document.

Le corpus est alors représenté par une matrice T dont la i ème ligne est la représentation du i ème document. La similarité entre les documents est mesurée par une distance, appelée " la distance du chi-deux ", très proche de la distance euclidienne, mais avec une pondération (probabilité) ($1/f_j$) associée à chacun des termes de la somme.

$$d_{X^2}(D_i, D_{i'}) = \sum_j \frac{1}{fj} \left(\frac{fi,j}{fi,} - \frac{fi',j}{fi',} \right)^2, \text{ où } fi, = \sum_j fi,j \text{ et } fi', = \sum_j fi',j$$

Si on refond la matrice pour intégrer les pondérations, on obtient pour chaque ligne:

$D_i = (w_{i,j})_j$, *avec* $w_{i,j} = f_{i,j} / (f_{i,} * \sqrt{f,j})$.

Alors la distance s'exprime par :

$$d_{X^2}(D_i, D_{i'}) = | D_i - D_{i'} |$$

Notons que le rapport $f_{i,j} / f_{i,}$, noté $p_{i,j}$, est la fréquence relative du j éme terme dans le i éme document.

L'une de ses propriétés importantes est que les distances entre les lignes (resp. colonnes) restent inchangées lors de la fusion de deux colonnes (resp. lignes) de même profil. Cette propriété d'invariance induit une certaine stabilité des résultats pour les analyses textuelles : en effet, deux textes ayant le même profil lexical pourront être indifféremment considérés comme une seule entité ou deux entités distinctes sans que cela n'affecte les autres distances.

L'autre propriété est que la distance du chi-deux est une mesure de proximité particulièrement sensible aux « différences hors intersection » (pour les termes n'apparaissant pas dans l'un des textes). La sensibilité aux différences n'a bien sûr rien de surprenant puisqu'une distance est, par définition, une dissimilarité et est de ce fait une fonction qui croît lorsque les différences entre les entités comparées augmentent. En revanche, il est notable que les « différences hors intersection » jouent un rôle important dans le calcul de la valeur de la dissimilarité. La conséquence de cette propriété est que la distance du chi-deux est a priori peu adaptée aux situations où les tailles des entités textuelles comparées sont fortement différentes. Cela est par exemple souvent le cas en recherche documentaire lors de l'évaluation de similarités entre courtes requêtes et longs documents.

La distance angulaire [DEL 02]

Les vecteurs conceptuels offrent une véritable mesure sémantique globale, ou restreinte à des sous parties de textes. De manière analogue aux mesures à base de cosinus, l'espace de représentation présente des propriétés scalaires. On peut donc introduire la mesure de similarité (ou dissimilarité) déjà observée :

$$\text{Sim}(X, Y) = \cos(X, Y) = X \bullet Y / (\|X\| * \|Y\|)$$

On en déduira la distance angulaire D_A, qui possède au sens formel les véritables propriétés d'une distance :

$$D_A(X, Y) = \arccos(Sim(X, Y))$$

Remarque

Après l'étude des principales techniques de similarité et distance nous dégageons une première propriété importante : qu'elles sont toutes définies en termes de fréquences relatives en non pas en termes de fréquences absolues. Cela entraîne la propriété intéressante de garantir qu'un document composite obtenu par concaténation d'un nombre quelconque de copies d'un même document élémentaire sera, pour ce qui est des similarités, strictement équivalent au document élémentaire lui-même.

Une seconde propriété des deux premières distances mentionnées est qu'elles tiennent compte des dimensions de l'espace de représentation, par le biais des facteurs :

- $\log (N/n_k)$ pour les dissimilarités utilisées en recherche documentaire ;

et

- $1/f_j$ pour la distance du chi-deux en analyse des données textuelles.

Annexe D

Le Langage XML

Introduction

L'apparition de l'Internet et son développement rapide ont permis l'apparition de plusieurs milliers de documents et données non structurés, d'où la nécessité d'un modèle d'échange représentant le contenu de l'information , et non pas sa représentation. Ainsi, **XML** (eXtensible Markup Language) définit un standard de structuration de données, mais pas un langage de programmation. On ne peut pas faire de tests, ni inclure un fichier dans un autre. En très gros, XML sert uniquement à stocker des données.

Dans ce qui suit nous allons définir le langage XML,

Définition du langage XML

Le groupe de travail du Consortium W3C en charge du Web a proposé un nouveau standard appelé XML comme format d'échange standard sur le Web. Comme HTML, XML est un langage à balises (ou " tags "), les balises représentant une définition de structure intégrée aux données [GAR 00].

La norme XML est un ensemble de spécifications désignant les règles à adopter afin de créer des documents universels mais libérés de toute contrainte. Il est conçu comme un sous ensemble de SGML (Standard Generalized Markup Language), norme internationale de documents structurés beaucoup utilisée dans l'industrie ; est plus permissif que celui-ci mais beaucoup moins que HTML.[JEX 01].

XML est donc né de la volonté d'intégrer le savoir faire résultant d'une décennie d'expérience de SGML, en ne retenant que les caractéristiques essentielles, indispensables à un usage et une implémentation facile et surtout en mettant fin à l'amalgame entre données et présentation que faisait HTML. D'ailleurs, l'utilisateur d'HTML ne trouve pas beaucoup de difficulté car la syntaxe de XML est proche de HTML [JEX 01].

Présentation de XML [JEX 01]

Un document XML se présente sous la forme d'un simple fichier texte contenant des balises de type <balise>. On trouve cinq types de balises en XML: les éléments (<balise>), les entités (é), les commentaires (<!-- commentaire -->), les instructions de traitement (<?xml ?>, <?php ?>) et les sections ignorées. La principale difficulté pour le débutant en XML est de bien comprendre qu'en réalité cette norme ne définit aucun élément par défaut.

Il existe deux types de documents XML. Nous atteignons ici la principale différence entre les deux types de documents XML pouvant exister. D'un côté nous avons les documents "bien formés" (c'est - à -dire que les balises sont fermées correctement, sans croisement, que les règles de syntaxe sont respectées, par exemple que les valeurs d'attributs sont bien entre guillemets doubles, etc...). De l'autre nous avons les documents valides, bien formés. Un document est déclaré valide s'il est accompagné d'un second document nommé DTD (Document Type Definition) qui définit les éléments constituant le document.

Chaque document XML doit posséder une première ligne qui est de la forme :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Cette ligne définit le numéro de version (ici 1.0), et l'encodage des caractères. Pour l'encodage, **ISO-8859-1** est le plus simple pour écrire un document en Français car les lettres accentuées ne prennent qu'un caractère (octet) ; et **UTF-8** est un Format international qui permet d'écrire dans n'importe quelle langue, car chaque caractère à un codage unique (Unicode),et les caractères latins (a-z et A-Z), chiffres arabes (0-9) et quelques autres caractères de ponctuation française sont codés sur un seul octet, mais les autres caractères sont codés sur plusieurs octets (la taille peut varier de 2 à 4 caractères, peut-être plus), par exemple les caractères accentués français prennent deux octets [STI 03][DEV 05].

Pour écrire en XML, il existe quelques règles de base à respecter :

- un document XML ne contient qu'une balise racine (parent) ;
- une balise doit être refermée.

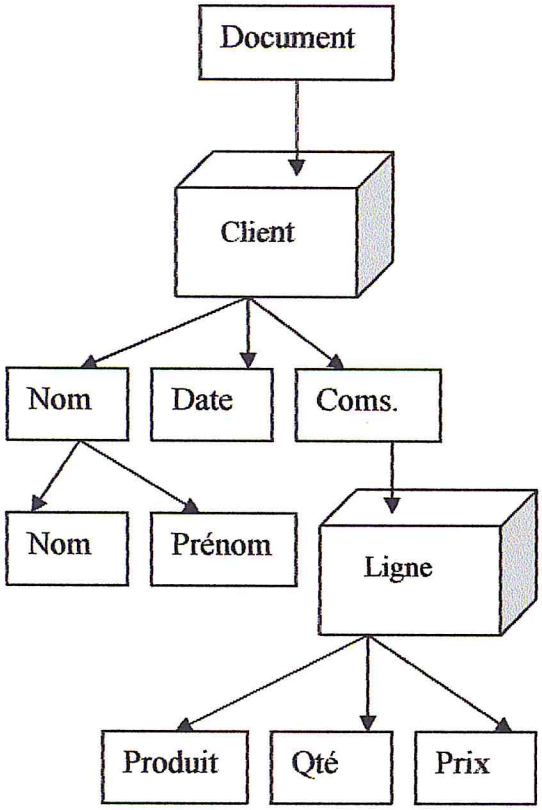
Dans la figure suivante, nous donnons un exemple simple de document XML. La première ligne définit la version de XML utilisée et le codage. La seconde ligne permet d'accéder à la DTD. Le document est ensuite constitué d'un titre, d'une section 1 avec un titre et d'une figure imbriquée à la fin de la section1.

```
<?XML version= "1.0 " standalone= 'yes' encoding= "ISO-8859-1"?>
<!doctype doc SYSTEM " http://www.myorg.org/dtds/doc.dtd">
<document>
   <titre> Le data Web: L'intégration des BD au Web </titre>
   <section1>
      <titre> L'emergence de XML </titre>
      <figure id= " fig1" sysid= " figures/ fig1.cgm"/>
   </section1>
</document>
```

Figure D.1 : Premier exemple de document XML [GAR 00].

La notion de DTD existe déjà en SGML. Une DTD SGML est à la fois la définition d'une syntaxe (grammaire), qui doit impérativement être connue pour permettre l'analyse d'un document, et la définition d'un modèle de données abstrait (schéma), qui définit des règles de structure génériques pour une classe de documents. Au contraire, XML sépare ces deux notions : la syntaxe est fixe pour l'analyse (il n'y a plus besoin de grammaire) ; la vérification de conformité au modèle spécifié par la DTD est optionnelle et s'effectue après l'analyse [GAR 00].

Nous proposons dans la figure (Figure D.2) un exemple de DTD, puis dans la figure D.3 un document XML correspondant à cette DTD. En quelque sorte, la DTD ressemble au schéma d'une base de données dont le document est une instance.




```
< ?XML version= "1.0 " ?>
<! DOCTYPE  DOCUMENT [
<! ELEMENT DOCUMENT (CLIENT)*>
<! ELEMENT CLIENT (NOM, DATE,
COMS)>
<! ELEMENT NOM (NOMF, PRENOM)>
<! ELEMENT NOMF (# PCDATA)>
<! ELEMENT PRENOM  (# PCDATA)>
<! ELEMENT DATE (# PCDATA)>
<! ELEMENT COMS (LIGNE)*>
<! ELEMENT LIGNE (PROD, QTE,
PRIX)>
<! ELEMENT PROD (# PCDATA)>
<! ELEMENT QTE  (# PCDATA)>
<! ELEMENT PRIX (# PCDATA)>
]>
```

Figure D.2 : Un exemple de DTD representant une hiérarchie [GAR 00].

```
<DOCUMENT>
  <CLIENT>
    <NOM>
      <NOM> Dupont </NOM>
      <PRENOM> Albert </PRENOM>
    </NOM>
    <DATE> 22 septembre 1998</DATE>
    <COMS>
      <LIGNE>
        <PROD> Salade</PROD>
        <QTE> 10 </QTE>
        <PRIX> 22 F </PRIX>
      </LIGNE>
      <LIGNE>
        <PROD> Salade</PROD>
        <QTE> 10 </QTE>
        <PRIX> 22 F </PRIX>
      </LIGNE>
    </COMS>
  </CLIENT>
</DOCUMENT>
```

Figure D.3 : Exemple de document XML [GAR 00].

Caractères spéciaux et CDATA [STI 03]

Etant donné que les balises XML sont délimités par des crochets : '<' et '>', on ne peut pas écrire ces caractères directement. Il faut les écrire, respectivement, < et >. Pour s'en rappeler, lt est l'abréviation de "lower than" (plus petit que), et gt l'abréviation de "greater than" (plus grand que).

Comme autre solution, on peut écrire le code d'un caractère en décimal. Exemple: & représente le caractère &, avec le code 38. Nous présentons ici quelques codes utiles:

- 38 : représente le caractère '&' (et commercial).
- 60 : représente le caractère '<' (inférieur à).
- 62 : représente le caractère '>' (supérieur à).
- 160 : représente le caractère ' ' (espace). Ce caractère est représenté par la balise .

Mais tout cela est assez contraignant, plus particulièrement pour écrire du source (en langage C par exemple) qui est rempli de ces caractères. On peut alors utiliser des sections CDATA. Il suffit d'ouvrir une section par la balise <![CDATA [, et la refermer par]]>. A l'intérieur d'une section CDATA, on peut utiliser les caractères spéciaux sans problème. Par contre on ne peut pas y écrire]]> qu'il faut écrire]] &gt Exemple de CDATA :

```
<p>Un peu de texte <b>HTML</b> qui ne sera pas mis en forme
Car il est placé dans une section CDATA ;-)</p>
```

XML et les langages de programmation [JEX 01]

Le XML prend une place de plus en plus importante dans l'industrie du logiciel. Nombreuses sont les productions professionnelles faisant appel à cette technologie. Citons par exemple *Netscape 6*. Pour cela les sites Internet également utilisent de plus en plus XML, couplé à XSL. Dans ce qui suit, nous allons voir le rapport de XML avec deux langages de programmation Java et Visual Basic, ainsi que les parsers XML liés à chaque langage.

Il existe plusieurs parsers XML pour Java sur Internet. Citons en vrac celui **XML4J** d'IBM, **ProjectX** de Sun Microsystems, **Ælfred** de Microstar ou encore **TinyXML** de Tom Gibara. Tous ces parsers, bien que capables de réaliser les mêmes choses, ne s'utilisent pas forcément de la même manière.

Nous allons, quant à nous, opter pour le parser Ælfred. Celui-ci possède en effet des caractéristiques séduisantes: léger (40 ko compilé), support de l'API SAX et simplicité de programmation. Malheureusement, Ælfred n'est pas un parser dit "validant" bien qu'il sache employer les DTD.

Pour l'autre langage, la création et la lecture d'un fichier XML nécessite l'utilisation des mentions DOM (*Document Object Model*) et SAX (*Simple API for XML*) désignant deux méthodes utilisées pour manipuler des documents XML.

La méthode DOM est réellement très simple. Son principe consiste à décrire tout document XML sous forme d'arborescence d'objets appelés *Nodes*. Chaque *Node* représente une balise du fichier XML. Un objet *Node* possède donc des attributs, des fils et un parent. En employant à bon escient les caractéristiques parents et enfants, on peut aisément parcourir l'ensemble du document XML.

L'avantage de la méthode DOM est sa souplesse. En effet, le programmeur n'a à traiter qu'un seul et unique type d'objet. Généralement, le document XML est créé dans un objet nommé *DOMDocument*. Cependant, cet objet n'est qu'un objet *DOMNode*. De la sorte, l'ensemble des fonctions de manipulations XML de programme pourra traiter un document XML à n'importe quel niveau, et ce sans aucune modification du code.

La méthode SAX permet d'aboutir au même résultat que la méthode DOM mais de façon très différente. Le schéma de fonctionnement de la méthode DOM est le suivant:

- Création de l'objet DOMDocument ;
- DOMDocument.loadXML *(fichier)* ;
- Parcours du document grâce aux *Nodes.*

Si la gestion du document XML peut s'avérer parfois fastidieuse (il n'est pas toujours pratique de parcourir un arbre), cette méthode permet de conserver un objet contenant l'arborescence du document XML. SAX au contraire facilite le traitement des données XML mais ne donne pas accès à un objet décrivant le document. En effet, un parser de type SAX posséde trois manipulateurs (ou *handlers*):

- "Contents Handler" ;
- "DTD Handler" ;
- "Error Handler".

Chacun de ces manipulateurs correspond à une classe héritant d'une interface de type *SAXHandler*. Le parser, en parcourant le document XML, appellera les méthodes requises dans ces manipulateurs.

Les applications de XML [JEX 01]

XML est presque exclusivement associé à Java car l'universalité de ces deux technologies les rend très complémentaires l'une de l'autre. La majeure partie des logiciels tirant parti de XML sont donc écrits en Java. Parmi les applications de XML on notera la génération de pages HTML depuis des documents XML et leur feuilles de styles associées (feuilles XSL, écrites en... XML), mais aussi description d'interfaces graphiques, bases de données, descriptions hiérarchicales (par exemple pour des scènes graphiques modélisées en 3D), propriétés utilisateurs, etc...

Avantages et inconvénients de XML [DEV 05]

Apres avoir fait un tour dans XML, nous allons maintenant citer les principaux avantages et inconvénients de cette norme.

A) Avantages

1. Un document XML est hiérarchisé sous forme d'arbre. Cette représentation est logique et permet de faire des recherches très pointues à l'intérieur d'un document XML.

2. Le nom des balises, leur quantité, ainsi que les attributs ne sont pas limités. Si on veut les limiter, on peut passer par les DTD ou XML Schema.
3. Un document XML est très portable et facilement lisible. Il existe des outils XML pour tous les langages courants (C/C++, Java, PHP, etc.).

B) Inconvénients

- La forme est tellement libre qu'on peut avoir des formats incompatibles. Par exemple: une information peut être stockée sous forme de balise, ou alors sous forme d'attribut dans deux documents différents. On peut utiliser les DTD ou XML Schema pour empêcher ces problèmes.
- Le XML utilisé sous forme de fichier peut générer des fichiers très gros et difficilement éditables à la main. Pour des grands nombres d'enregistrement, un découpage en plusieurs fichiers ou l'utilisation d'une base de donnée est nécessaire.

Conclusion

Le format XML va être de plus en plus utilisé parce qu 'aujourd'hui, ce qui devient important c'est de retrouver une information dans une multitude de documents, ce qui fait du format XML un début de solution.

D'un côté le XML est très souple, de l'autre il est très portable et sa forme est définie selon des règles strictes. Il permet l'interopérabilité dans un environnement très hétérogène.

Annexe E

Le Langage UML

Bref historique d'UML [JAC 03]

Les langages de modélisation orientés objet ont fait leur apparition entre le milieu des années soixante-dix et la fin des années quatre-vingt, quand les spécialistes de méthode confrontés à un nouveau genre de langages de programmation orientés objet et à des applications de plus en plus complexes, ont commencé à expérimenter de nouvelles approches d'analyse et de conception .En 1989 et 1994 le nombre de méthodes orientées objet est passé de moins de 10 à plus de 50.La majorité des utilisateurs ne trouvaient pas de langage de modélisation qui répondre entièrement à leur besoins ; ils ont alors déclenché la "guerre des méthodes ". Basées sur l'expérience acquise, de nouvelle générations de ces méthodes sont apparues .Certaines d'entre elles se sont nettement détachées du lot et plus spécialement la méthode BOOCH, OOSE (Object-Oriented Software Engineering) de Jacobson et OMT (Object Modeling Techenique) de Rumbaugh. Chacune de ces méthodes constituait une méthode complète mais présentait aussi des avantages et des inconvénients.

Certaines idées déterminantes ont émergé progressivement au milieu des années quatre vingt dix lorsque Grady Booch, Ivar Jacobson et James Rumbaugh se sont associés pour créer un langage de modélisation unifié et ont donné naissance à UML.

UML (Inified Modeling Langage) est un langage standard conçu pour l'écriture de plans d'élaboration de logiciels. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à forte composante logicielle.

UML est adapté à la modélisation de système, depuis les systèmes informatiques d'entreprise jusqu'aux applications distribuées basées sur le Web, en passant par les systèmes en temps réel embarqués. C'est un langage très expressif qui couvre toutes les perspectives nécessaires au développement puis au déploiement d'un tel système.

Le modèle conceptuel d'UML [JAC 03]

Le modèle conceptuel implique l'assimilation de trois éléments essentiels : les briques de base d'UML, les règles qui déterminent la manière d'assembler ces éléments et quelques mécanisme généraux qui s'appliquent à UML dans son ensemble.

Les briques de base

Les briques de base d'UML sont :

- les éléments ;
- les relations ;
- les diagrammes.

Les éléments sont les abstractions essentielles à un modèle. Les relations constituent les liens entre ces éléments ; les diagrammes les regroupent en des ensembles dignes d'intérêt.

A) Eléments d'UML

Il existe quatre types d'élément dans UML :

- les éléments structurels ;
- les éléments comportementaux ;
- les éléments de regroupement ;
- les éléments d'annotation.

➢ Éléments structurels

Les éléments structurels sont représentés par des noms dans les modèles UML ; ce sont les parties les plus statiques d'un modèle, ils représentent les éléments conceptuels ou physiques .Au total, il existe sept types d'élément structurels.

1) une classe représente un ensemble d'éléments qui partagent les mêmes attributs, les mêmes opérations, les mêmes relations et les mêmes sémantiques . Une classe implémente une ou plusieurs interfaces. Elle est représentée par un rectangle qui contient son nom, ses attributs et ses opérations.

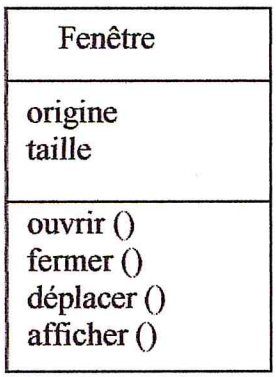


Figure E.1 : Une classe.

2) *une interface* est un ensemble d'opération qui définissent la fonction d'une classe ou d'un comportement .Elle définit les spécifications de ses opérations (c'est -à-dire la signature). Une interface est représentée par un cercle avec son nom. Elle est généralement liée à la classe ou au composant qui réalise l'interface.

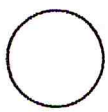


Figure E.2 : Une interface.

3) *Une collaboration* définit une interaction entre divers éléments qui travaillent ensemble pour fournir un comportement coopératif .Elle représente une implémentation des parties qui structurent un système. Une collaboration est représentée par une ellipse en pointillés qui en règle générale contient seulement son nom.

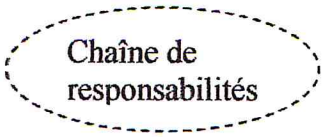


Figure E.3 : Une collaboration.

4) *un cas d'utilisation* est la description d'une séquence d'actions exécutées par un système pour produire un résultat qui peut être constaté par un acteur particulier. Il est représenté par une ellipse entrait plein qui contient son nom.

Figure E.4 : Un cas d'utilisation.

5) *une classe active* est une classe dont les objets possèdent un ou plusieurs processus ou threads et qui peut donc lancer une activité de commande. Elle est représentée comme suit :

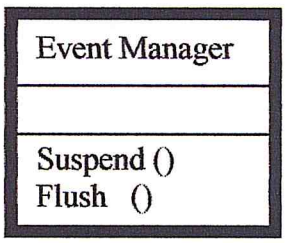


Figure E.5 : Une classe active.

Les deux éléments restant (composant, nœud) sont encore différents. Ils représentent des éléments physiques, alors que les cinq éléments précédents représentent des éléments conceptuels ou logiques.

6) *un composant* est une partie physique remplaçable d'un système qui se conforme à un ensemble d'interfaces. Il représente l'enveloppe physique d'éléments de nature logique. Il est représenté comme suit :

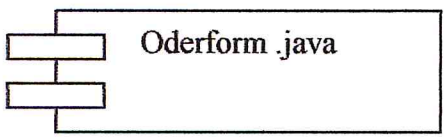


Figure E.6 : Un composant.

7) *un nœud* est un élément physique qui intervient lors de la phase d'exécution ; il représente une ressource de calcul et dispose généralement au moins d'un peu de mémoire et souvent d'une capacité de traitement. Un ensemble de composants peut résider sur un nœud, mais peut également se déplacer d'un à l'autre .Un noeud est représenté par un cube qui contient son nom.

Figure E.7 : Un nœud.

➢ Eléments comportementaux

Les éléments comportementaux représentent les parties dynamiques des modèles d'UML. Il existe deux types fondamentaux d'éléments comportementaux.

1) *une interaction* est un comportement qui comprend un ensemble de messages échangés, au sein d'un groupe d'élément, dans un contexte particulier pour atteindre un but bien défini. Une interaction implique un certain nombre d'éléments y compris des messages, des séquences d'actions, des liens (relation entre éléments). Un message est représenté comme suit :

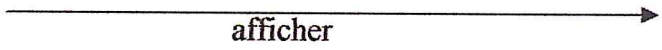


Figure E.8 : Messages.

2) *un automate d'état finis* est un comportement qui précise les séquences d'état d'un élément ou une interaction on réponse à un événement. Un état est représenté par un rectangle avec des angles arrondis.

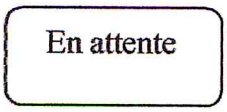


Figure E.9 : Etat.

Ces deux éléments, interaction et automate à état fini constituent les éléments comportementaux de base représentés dans le modèle UML.

➢ Eléments de regroupement

Les éléments de regroupement représentent les parties organisationnelles des modèles UML. Le seul type fondamental d'élément de regroupement est le « paquetage ».

Un paquetage est un mécanisme général qui permet de regrouper des éléments. Des éléments structurels, des éléments comportementaux et même d'autres éléments de regroupement peuvent être rangés dans un paquetage. À la différence des composants (qui existent lors de la phase d'exécution), un paquetage est purement conceptuel (il existe seulement lors de la phase de développement). Un paquetage est modélisé par un dossier étiqueté, il contient seulement son nom, mais parfois son contenu.

4) La réalisation est relation sémantique entre deux éléments. Les relations de réalisation apparaissent à deux occasions : entre les interfaces et les classes ou les composants qui les réalisent et entre les cas d'utilisations et les collaborations qui les réalisent .Une relation de réalisation est représentée par le mélange d'une généralisation et d'une relation de dépendance .

Figure E.13 : Réalisation.

C) Les diagrammes dans UML

Un diagramme est la représentation graphique d'un ensemble d'éléments qui constituent un système .La plupart du temps, il se présente sous la forme d'un graphe connexe[1] où les sommets correspondent aux éléments et les arcs aux relations. Les diagrammes servent à visualiser un système sous différentes perspectives et sont donc des projections dans le système.

Pour englober un système avec forte composante logicielle, UML comprend neuf diagrammes.

- **Les diagrammes de classes,** représentent un ensemble de classes, d'interfaces et de collaboration, ainsi que leur relations. Ce sont les diagrammes les plus fréquent dans la modélisation des systèmes orientés objet. Ils représentent la vue de la conception statique d'un système.

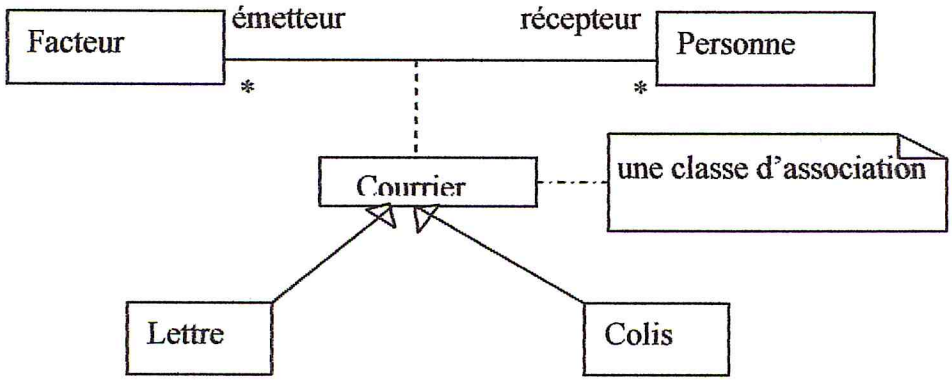


Figure E.14 : Un exemple d'un diagramme de classe.

- **Les diagrammes d'objets** représentent un ensemble d'objets et leurs relations .Ce sont des vues statiques des instances des éléments qui apparaissent dans les diagrammes de classes. ils représentent la vue de processus statique d'un système comme les diagramme de classe, mais à partir de cas réels ou de prototypes.
- **Les diagrammes de cas d'utilisation** représentent de cas d'utilisation et d'acteur (sorte de classe particulières) et leurs relations. Ils présentent la vue statique des cas d'utilisation d'un système et sont particulièrement importants dans l'organisation et la modélisation des comportements d'un système.

[1]

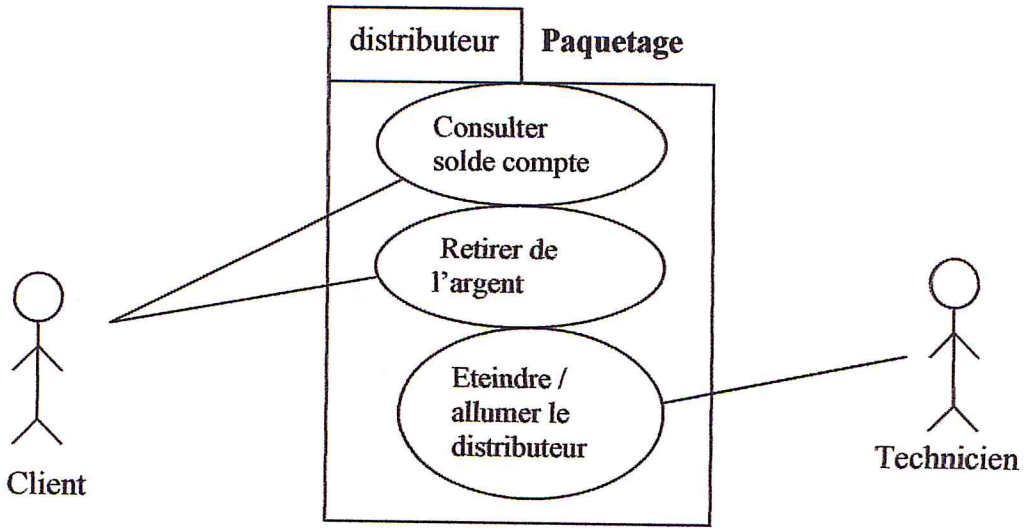


Figure E.15 : Un exemple d'un diagramme de cas d'utilisation.

❖ **Les diagrammes de séquences et les diagrammes de collaboration** sont deux types de diagrammes d'interaction. Ces derniers représentent une interaction, c'est - à- dire un ensemble d'objets et leurs relations, y compris les messages qu'ils peuvent s'échanger. Ils présentent la vue dynamique d'un système. Les diagrammes de séquence sont des diagrammes d'interactions qui mettent l'accent sur le classement chronologique des messages alors que les diagrammes de collaboration sont des diagrammes d'interactions qui mettent l'accent sur l'organisation structurelle des objets qui envoient et reçoivent des messages.

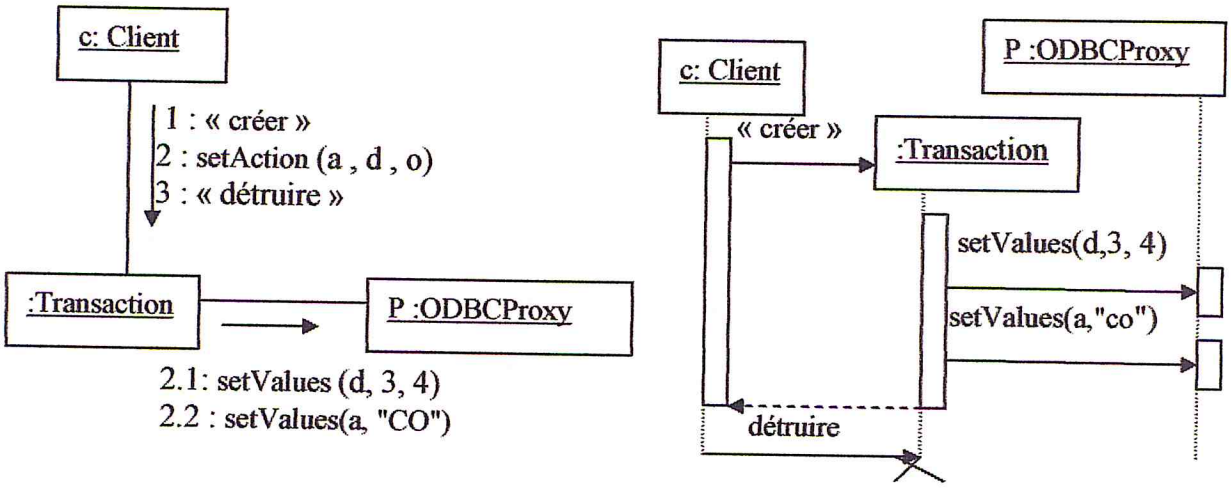


Figure E.15 diagramme de collaboration et diagramme de séquence.

❖ **Les diagrammes d'état–transitions** sont des automates à états finis, composés d'états, de transitions, d'événements, d'activités. Ils présentent la vue dynamique d 'un système et sont particulièrement importants dans la modélisation du comportement d'une interface, d'une classe ou d'une collaboration, ce qui est utile dans la modélisation des systèmes réactifs

❖ **Les diagrammes d'activités** sont un type particulier de diagramme état- transition qui décrit la succession des activités au sein d'un système .Ils présentent la vue dynamique d'un système ; ils sont particulièrement importants dans la modélisation de la fonction d'un système et mettent l'accent sur le flot de contrôle entre les objets.

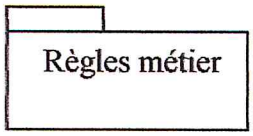


Figure E.10 : Paquetage.

➢ **Eléments d'annotation**

Les éléments d'annotation représentent les parties explicatives des modèles d'UML. Ce sont les commentaires qui peuvent accompagner tous éléments dans un modèle, à des fins de description , d'explication et de remarque .Il existe un type fondamental d'éléments d'annotation , appelé « note ». Une note est représentée par un rectangle écorné qui contient un commentaire textuel ou graphique.

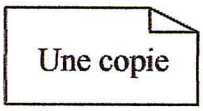


Figure E.11 : Note.

B) Les relations dans UML

Il existe quatre relations dans l'UML :

1) la dépendance est une relation sémantique entre deux éléments selon la quelle un changement apporté à l'un (élément indépendant) peut affecter la sémantique de l'autre (élément dépendant). Une dépendance est représentée par une ligne en pointillés s qui peut être fléchée ; elle comprend parfois une étiquette.

Figure E.12 : Dépendance.

2) l'association est une relation structurelle qui décrit un ensemble de liens. Un lien constitue une relation entre différents objets. Une association est représentée par une ligne qui peut être fléchée ; elle comprend parfois une étiquette, et souvent la multiplicité et les noms de rôles.

Figure E.13 : Association.

3) la généralisation est une relation de spécialisation /généralisation selon laquelle les attributs de l'élément spécialisé (l'enfant) peuvent se substituer aux attribut de l'éléments généralisé (le parent).Une relation de généralisation est modélisée par une flèche dont le trait est plein et dont la pointe vers creuse est dirigée vers le parent.

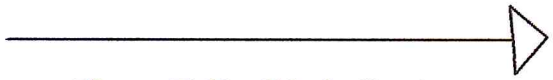


Figure E.13 : Généralisation.

Bibliographies

[AGR 93] R. Agrawal, T. Imielinski, A. Swami, «Mining association rules between sets of items in large databases »; In Proc, of the ACM SIGMOD Conference on Management of Data, Washington, D.C.; Mai 1993.

[AGR 94] R. Agrawal R. Sikrant ; « Fast algorithms for mining association rules »; Proc. 20th Int. Conf. Very Large Data Bases VLDB; 1994.

[AGR 96] R. Agrawal J.C. Shafer ; « Parallel mining of association rules »; IEEE Transactions on Knowledge and Data Engineering; Décembre 1996.

[BIC 99] Bichindaritz .I et Mille.A et Napoli.A (1999) « Raisonnement à partir de cas » Edition Hermes science publications.

[BOU 01] Bruno Bouzy ; « cycle de vie d'un logiciel » ; Eyrolles ; juin 2001.

[BRE 97] Christophe Le Bret ; « Connaissez-vous le Data Mining » ; Science Tribune ; Octobre 1997.

[CLE 04] Guillaume Cleuziou ; « Organisation des données et apprentissage non- supervisé » ; LIFO, Laboratoire d'Informatique Fondamentale d'Orléans ; 2004.

[COM 01] Soft. Computing; « Le datamining »; Memo technique (Février 2001).

[CRO 99] Two Crows Corporation; « Introduction to Data Mining and Knowledge Discovery »; third edition; 1999.

[CUT 93] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey; « Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections »; 2000.

[DEL 02] Jean-Michel Delorme, proposé par Mme Maguelonne Teisseire ; « L'apport de la fouille de données dans l'analyse de texte » ; Examen probatoire de CNAM : Conservatoire National des Arts et Métiers : Centre régional de Montpellier, le 24 Avril 2002.

[DUV XX] Béatrice Duval ; « Extraction de connaissances à partir des données (ECD) (Knowledge Discovery In Databases. KDD.) » ; Laboratoire d'Informatique, Université d'Angers.

[FAY 96] Usama M. Fayyad, George Piatetsky-Shapiro, Padhraic Smith, and Ramasamy Uthurusamy; « Adanced in Knowledge Discovery and Data Mining »; 1996.

[FAY 99] Usama Fayyad, Ramasamy Uthurusamy; « Data Mining and knowledge discovery in databases: Introduction to the special issue »; Communications of the ACM, *39(11), November 1999.*

[FRE 02] Alex A. Freitas; « Data Mining and Knowledge Discovery algorithms »; Computing laboratory, Université de Kent; édition Springer 2002.

[FUN 03] Benjamin C. M. Fung, Ke Wang, and Martin Ester; « Hierarchical Document Clustering Using Frequent Itemsets »; Dans la conference SIAM 2003 international en Data Mining (SDM 03), San Francisco, CA; 13 Mai 2003.

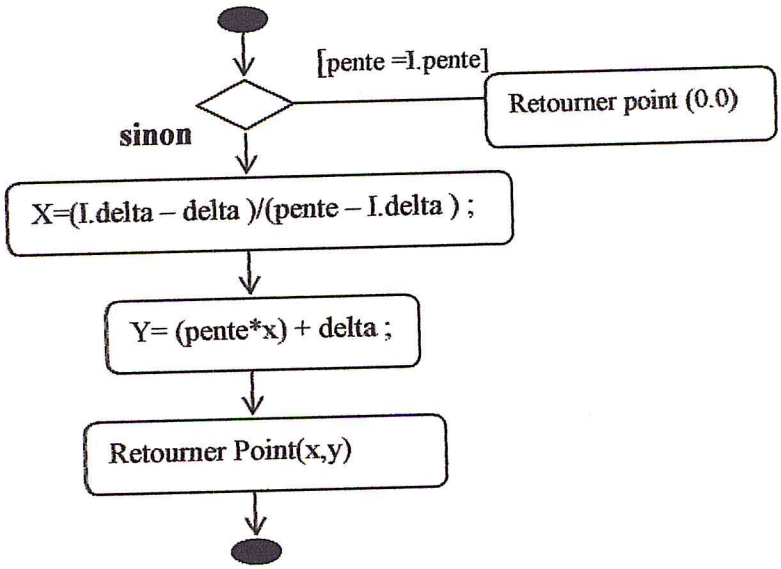


Figure E.16 : Un diagramme d'activité.

- **Les diagrammes de composants** représentent l'organisation et les dépendances. Au sein d'un ensemble de composants .ils représentent la vue d'implémentation statique d'un système et sont liés aux diagrammes de classe dans le sens où un composant correspond généralement à une ou plusieurs classes, interfaces ou collaborations.

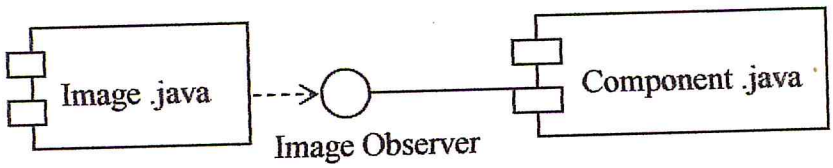


Figure E.17 : Un exemple d'un diagramme de composant.

- **Les diagrammes de déploiement** représentent la configuration des nœuds de processus en phase d'exécution ainsi que les composants qui y résident .Il présentent la vue de déploiement statique d'une architecture et sont liés aux diagramme de composants , dans le sens où un nœud renferme un ou plusieurs composants .

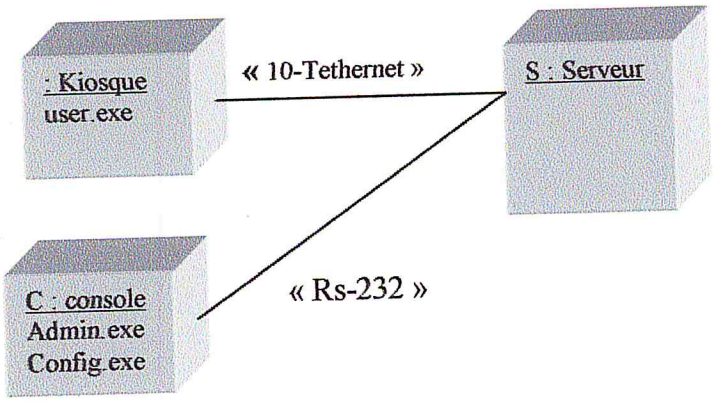


Figure E.18 : Un exemple d'un diagramme de déploiement.

Cette liste n'est pas exhaustive. Des outils peuvent utiliser UML pour produire d'autres sortes de diagrammes, même si les neufs sortes précédemment citées sont les plus fréquemment utilisées.

[GAR 98] Gardarain G., Pucheral Ph., Wu F., « Bitmap Based Algorithms For Mining Association Rules », Proc. of 15th Workshop on "Bases de Données Avancées", Tunis; octobre1998.

[GAR 00] Gardarain G. ; « Internet/intranet et baes de données : Data Web, Data Media, Data Warehouse et Data Mining » ; Édition Eyrolles ; Paris 2000.

[GAY 04] Jean-Sébastien Gay ; « Application des arbres de radicaux a des algorithmes parallèles du datamining », stage de DEA encadré par C. Cérin et M. Koskas, LARIA : Université de Picardie Jules Verne, le 22 juillet 2004.

[HAN 00] J. Han, J. Pei, and Y. Yin; « Mining frequent patterns without candidate generation »; In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00), Dallas, Texas, USA; Mai 2000.

[HAS 03] Salima HASSAS, Jérémy CLECH; « Web Mining Système Multi-Agents » ; EGC: Extraction et Gestion des Connaissances ; Lyon ; Janvier 2003.

[HEB 03] Georges Hébrail ; « Fouille de données pour la gestion de la relation client », ENST Paris ; Séminaire EGSH - 20/03/2003 : Gestion de la Relation Client : Information et Intermédiation.

[HOU 93] M. Houtma, A. Swami; « Set-oriented mining of association rules »; Research Report RJ 9567, IBM Almaden Research Center, San Jose, California; Octobre 1993.

[JAC 03] Ivar jacobson, Grady Booch ; « le guide de UML» ; Eyrolles, édition 2003.

[JEM 04] Mohamed Jemni ; « Méthodologie de parallèlisation : conception, algorithmique et programmation & Elaboration d'un système adaptatif pour le e-learning » ; PhD thesis, Université de Versailles, 2004. Présentation des travaux pour obtenir l'habilitation à diriger des recherches.

[KIN 01] King-Ip Lin, Ravikumar Kondadadi; « A SIMILARITY-BASED SOFT CLUSTERING ALGORITHM FOR DOCUMENTS » ; Department of Mathematical Sciences, The University of Memphis, USA ; 2001.

[KOD 97] Kodratoff.Y ; « L'extraction de connaissances à partir de données : un nouveau sujet pour la recherche scientifique » ; bulletin de l'AFIA, N° 30 ; 1997.

[KOD 01] Kodratoff.Y, Napoli, Zighed ; « ECBD ou encore Fouille de données » ; Bulletin de l'AFIA ; 2001.

[LAU XX] Anne Laurent ; « Fouille de données » ; LIRMM – POLYTECH, Université MONTPELLIER 2.

[LEB 02] Gérard Leblanc ; « C# et .NET » ; EYROLLES, 2 ème édition 2002.

[LEF 98] René Lefébure, Gilles Venturi ; « Le Data Mining » ; Édition Eyrolles ; Paris, 1998.

[LIN 97] Michael J.A. Berry & Gordon Linoff ; « Data Mining : techniques appliquées au marketing, à la vente et au service client » ; 1997.

[LSP 03] Laboratoire de statistique et probabilités ; « Data Mining : Modélisation statistique et apprentissage » ; 2003.

[MAH 04] Gaël Le Mahec encadré par C. Cérin et M. Koskas ; « Utilisation des arbres de radicaux pour les algorithmes de Data-Mining sur grille de calcul » ; Stage de DEA en Informatique Parallèle Répartie et Combinatoire ; Laboratoire de recherche en informatique d'Amiens-Université de Picardie Jules Verne – le 24 juillet 2004.

[MUL 97] Pierre-Alain Muller ; « modélisation objet avec UML » ; Eyrolles ; 1997.

[PAD 04] Padova; « a fuzzy map clustering method for steroarray data analysis »; Italy FMC; ITS 2004.

[PAS XX] Philippe Pasquier et Rolf Schmidt ; « Text Mining et techniques connexes ».

[PEN 02] Tom Penders ; « Introduction à UML » ; OEM ; 2002.

[POL 00] Xavier Polanco ; « Convergence de produits logiciels et d'information en Fouille de Données (Data Mining) et Extraction de Connaissance à partir de Bases de Données (Knowledge Discovery in Databases) » ; URI-INIST-CNRS ; Séminaire ADEST - 15 février 2000.

[POL XX] Xavier Polanco ; « TEXT MINING ET INTELLIGENCE ECONOMIQUE : AUJOURD'HUI ET DEMAIN » ; Unité de Recherche et Innovation : Institut de l'Information Scientifique et Technique : Centre National de la Recherche Scientifique.

[SAV 95] Savaeste A., Omseinski F., Navatgr S.; « An efficient algorithm for Mining Association Rules in large Databases »; Proc of the 21 th Intl. Conf. on Very Large Data Bases, p. 432-444, Zurich, Switzerland, 1995.

[SCH XX] Gabriella Schoier ; « Introduction au Data Mining » ; Département de Science Economique et Statistique : Faculté d'Economie - Université de Trieste.

[STE 00] Stehael Steinbach, George Karypis and Vipin Kumar; « A Comparison of Document Clustering Techniques »; 23 Mai 2000.

[STI 03] Vector Stinner; Introduction à XML; date de publication: 16/09/2003.

[SUN 00] Sung-Chien Lin: « Document Clustering »; Department of Library and Information Studies, Shih-Hsin University; 16 Mars 2003.

[ZAK 97] Mohammed Javeed Zaki, Srinivasan Parthasarathy, and Wei Li; « A localized algorithm for parallel association mining »; In ACM Symposium on Parallel Algorithms and Architectures, pages 321–330, 1997.

Bibliographies Internet

[AND XX] www.anderson.ucla.edu; what is data mining.

[ARG XX] www.argamon.com ; Unsupervised Disambiguation.

[BAS XX] www.bashful.pub.ac.uk; Data Mining, An introduction.

[CSE 99] www.cse.iitb.ac.in ; Agglomerative Algorithm for Complete- Link Clustering; 1999.

[DEV 05] www.developpez.com; XML; 2000-2005. .

[HAI 98] www.cs.haifa.ac.il ; Bootstrapping; 1998.

[JEX 01] www.jext.org; Le XML; dernière mise à jour le 09/09/2001.

[KAR 02] wwwusers.cs.umn.edu/~ karypis/cluto; G. Karypis; Cluto 2.0 clustering toolkit, Avril 2002.

[ORL 04] www.univ-orleans.fr; regroupement (clustering); 2004

[PCC XX] www.pcc.qub.ac.uk; Data Mining.

[PMS XX] www.pmsi.fr; le Data Mining.

[STA 02] www.statsoftinc.com; Data Mining 1984-2002.

[TEX XX] www. Text Mining Research Group at the University of Waikato.htm « what is Text Mining? ».

[UTE XX] www.cs.utexas.edu; Text Clustering.