REPUBLIQUE ALGERIENNE DEMCRATIQUE ET POPULAIRE
Ministère de L'enseignement Supérieur et de la Recherche Scientifique
UNIVESITE de MOULOUD MAMMERI de TIZI-OUZOU
Faculté de Génie Electrique et Informatique
Département d'Informatique

Mémoire de fin d'études

En vue d'obtention du diplôme de master en Informatique.

Option : conduite de projets informatiques.

Thème

Extraction de règles d'associations entre les concepts biomédicaux.

Réalisé par :

M^{elle} : AFLIHAOU Samia

M^{r} : LEHAMEL Arezki

Dirigé par :

M^{me} : AMIROUCHE .F

Devant le jury d'examen composé de :

M. Hammache président
M. Amirouche examinateur
M. Habet examinateur
Mme Amirouche Promotrice

PROMOTION : 2012-2013

Remerciements

Nous tenons avant tout à remercier le bon dieu tout puissant de nous avoir donné de la volonté et de la détermination pour accomplir ce modeste et humble travail.

Que dieu soit témoin de nos grand remerciement pour Mme Amirouche.F qui a sacrifié de son temps précieux pour nous fournir l'aide qu'il nous faux pour mener à bien notre travail. Ainsi, tous les enseignants qui ont contribué par leur collaboration, disponibilité et sympathie à notre formation.

Notre considération pour toute personne ayant contribué de prés ou de loin à la réalisation de ce travail.

Nos remerciements les plus sincères et les plus profonds pour nos parents en reconnaissance de leurs sacrifices, aides, soutien et encouragement afin de nous assurer cette formation dans les meilleures conditions.

Enfin que les membres du jury trouvent ici l'expression de notre profonde gratitude pour l'honneur qu'il nous font en acceptant de juger notre humble travail étant une œuvre humaine qui n'est pas un modèle unique et parfait.

Dédicaces

Je dédie cet humble travail à ma grande mère, que dieu me la protège, elle est ma grande source d'inspiration dans le quotidien. A la mémoire de mes oncles défunts qui seront toujours gravées dans les mémoires.

A toute ma famille, mes très chers parents, je les remercie pour leur amour et leur affection, qui ont donné autant de mal pour mon éducation.
Mes frères et soeurs : Titem, Omar et
Tinhinane auxquels je souhaite une très longue vie pleine de bonheur.

A mes proches, mes grands parents maternels, mes tantes et oncles maternels et leurs familles, ma tante paternels et son mari et leurs enfants : Samy et Samia.

Sans oublier tous mes amis :Badia, Sorya, Ycine, Makhlouf, Massi nissa, Hocine, Omar, Samir, Rabah, Younes, Lyes, Fazia, Kamélia, Farid, Karima, touts mes camarades de classe et tout les autres qui nous ont aidé.
A toi ma binôme et amie Samia pour ta compréhension et ta sagesse.
A tous mes professeurs du département d'informatique de L'université Mouloud Mammeri de Tizi- Ouzou spécialement notre promotrice Mme Amirouche.

Arezki

Dédicaces

Je dédie cet humble travail à :

Mes très chères parents à qui je souhaite une longue vie et une bonne santé ;

que par leurs présence à mes côtés ont rendu chaque moment de ma vie

Un merveilleux passage dans le temps :

Mes adorables frères et sœurs

Ainsi je tien a remercier infiniment tous ceux qui ont contribué de prét ou de loin à la réalisation de ce travail :

Ainsi mes amis et toute ma famille.

Samia

Résumé :

Ce mémoire traite la problématique de l'extraction des règles d'association dans le corpus textuel extrait de MedeLine. En effet, avec l'avènement de l'informatique et l'augmentation de données stockées sur des supports informatiques, qui peuvent cacher des liens et des corrélations pertinentes, la conception et la mise en œuvre des outils de Text Mining deviennent nécessaire.

On introduisant le Data Mining qui est la source de Text Mining, la premier chapitre de ce mémoire est consacrée à la présentation et la définition de Text Mining et les différentes notions et techniques qui le constituent et qui interviennent dans le développement de notre projet.

Dans le deuxième chapitre, nous exposons les différentes caractéristiques des règles d'association et les algorithmes utilisés pour leur extraction. Et le troisième chapitre est consacré à la description de notre approche qui a comme objectif l'extraction de règles d'association à partir d'un corpus textuel extrait de MedeLine. Et enfin, le dernier chapitre est consacré à la présentation du système développé ainsi qu'à l'interprétation des différents résultats obtenus par l'analyse d'un ensemble de documents biomédicale.

De la manière la plus simple, la chaîne de traitement de notre système se déroule en deux phases. La première partie consiste à faire l'extraction des concepts on utilise cxtractor. Ensuite, nous utilisons les concepts obtenues dans une première partie pour générer des motifs fréquents et ça grâce à l'algorithme apriori et à partir de ses motifs on utilise l'algorithme de génération des règles d'association pour générer les relations entre les concepts.

Summary:

This thesis addresses the problem of extracting association rules in the textual corpus extract from Medeline. Indeed, with the advent of computers and increasing stored on computer media, which can hide the links and correlations relevant data, the design and implementation of text mining tools become necessary.

On introducing the Data Mining is the source of Text Mining, the first chapter of this thesis is devoted to the presentation and the definition of Text Mining and different concepts and techniques that are involved in and the development of our project.

In the second chapter, we discuss the different features of association and the algorithms used for their extraction rules. The third chapter is devoted to the description of our approach which aims extract association rules from a text corpus extract from Medeline. And finally, the last chapter is devoted to the presentation of the developed system as well as the interpretation of the different results obtained by the analysis of a set of biomedical documents.

In the easiest way, the processing chain of our system is divided into two phases. The first part is to extract the concepts using cxtractor. Then we use the concepts obtained in the first part to generate frequent patterns and that thanks to the Apriori algorithm and its grounds from the generation algorithm of association rules is used to generate the relationships between concepts.

Liste des figures

Liste des tableaux

Sommaire

Chapitre II : les règles d'association.

Chapitre III : Description de notre approche.

Chapitre IV: Implémentation de l'algorithme Apriori.

Conclusion générale

Introduction Générale

Introduction générale :

L'explosion des données informatiques sauvegardées sur des supports numériques ces dernières années, on poussé des chercheurs à développer des méthodes, outils et techniques pour traiter les données. Malgré, la mise en point de ces outils et techniques, des informations restent caché. Ce qui empêche d'avoir une vue complète sur les informations contenues dans ces données. Exploiter ces gigantesques masses de données cachées, constitue aujourd'hui un véritable défi pour les chercheurs en informatique.

La fouille de données sur support informatiques a été appréhendée selon deux angles principaux. On distingue nettement :

- ✓ Le Data Mining qui travaille sur des données fortement structurées et typées stockées dans des bases de données relationnelles.
- ✓ Le Text Mining qui travaille sur des données textuelles non structurées ou semi-structurées.

Du point de vue informatique, un texte est une suite de caractères alphabétiques, numériques, spéciaux. Manipuler le contenu des textes en informatique n'est pas une chose facile, il s'est avéré nécessaire de construire un model de représentation d'un texte afin de y pouvoir effectuer des opérations de fouilles. Cette étape constitue effectivement l'étape préalable à tout traitement informatique d'un texte.

Notre problématique générale est de construire un model de représentation des documents textuelles et d'en extraire les informations pertinentes afin d'adapter une technique de Data Mining (les règles d'association) pour extraire des connaissances Dissimulées.

Notre mémoire s'articule autour de l'étude de 4 chapitres suivant :

- Chapitre I: Text Mining.
- Chapitre II : Les règles d'association.
- Chapitre III : Description de notre approche.
- Chapitre IV : Réalisation de l'application et teste.

Chapitre I:

Le Text Mining

Introduction:

Dans les années 60 sont nées un ensemble de technologies qui visaient à exploiter les données sous formes tabulées (volume de vente, âge, fréquence d'achat, lieu, etc.), technologies que l'on a appelées « data mining ».Il s'agissait de reconnaitre des familles de données proches (ou au contraire éloignées), d'identifier des relations entre celle-ci (cause, conséquence, association forte,…), de repérer et de caractériser des tendances (croissance, décroissance, forte évolution…). Aujourd'hui ces technologies sont matures, largement diffusées et sont appliquées sur de grandes bases de données tabulées (données économiques, bases de données clients, données de production, etc.).

A la toute fin du siècle dernier, avec l'avènement d'internet notamment, sont nées de nouveaux ensembles de technologies qui visaient à étendre le champs d'exploitation des données aux données non tabulées et éparpillées dans les textes, technologies que l'on appelle aujourd'hui « text mining ».

I. Présentation du Data Mining :

En 1991, Piatetsky-Shapiro introduit comme titre de son ouvrage le terme de "Knowledge Discovery from Databases", abrégé par la suite en KDD et, dont l'équivalent français est extraction de connaissances à partir de Bases de Données (ECDB). Ce n'est que vers 1995 que l'usage des termes Knowledge Discovery from Databases et Data Mining se précise. Le but de Data Mining est d'explorer et d'analyser de grands volumes de données d'une part pour les rendre plus compréhensibles et d'autre part pour découvrir des corrélations significatives.

Le terme Data Mining employé aussi pour désigner l'ensemble des outils permettant à l'utilisateur d'accéder aux données de l'entreprise, de les analyser et de générer des informations riches à partir des données de l'entreprise, notamment des données historiques, de découvrir des modèles implicites dans les données. Ils peuvent permettre par exemple à un magasin de dégager des profils de client et des achats types et de prévoir ainsi les ventes futures. Il permet d'augmenter la valeur des données contenues dans le Data Werehouse, [Georges El Helou et Charbel Abou khalil ,2004].

On peut aussi définir le Data Mining comme suit:

- ✓ L'extraction de connaissances, non triviales, implicites, préalablement inconnues et potentiellement utiles, depuis des données stockées dans de larges bases de données.
- ✓ L'exploration et l'analyse de grandes quantité de données afin de découvrir des formes et des règles significatives en utilisant des moyens automatique ou semi-automatique."
- ✓ Un processus itératif par lequel on extrait des connaissances valides, nouvelles, potentiellement utiles et compréhensibles [Fayyad et al., 1995]
- ✓ L'analyse de bases de données (souvent très grandes) afin de découvrir des relations insoupçonnées et de résumer les données d'une manière à la fois compréhensible et utile [Hand et al. 2001].

Il existe également une branche spécialisée de la fouille de données qui est l'analyse de textes libres ou bien la fouille de données textuelles qui s'appelle « Text Mining ».

II. Introduction au text mining :

Le Text Mining est un ensemble de méthodes, de techniques et d'outils pour exploiter les documents non structurés que sont les textes écrits, comme les fichiers bureautiques de type word, les emails, les documents de présentation de type PowerPoint …..Etc. Autrement dit, l'objectif est le traitement de grandes quantités d'information qui sont disponibles sous une forme textuelle non structurée. (Feldman et al., 1998a).

"Nous définissons aussi le Text Mining comme étant le data mining sur les données textuelles. La fouille de textes est tout ce qui porte sur l'extraction de modèles et d'associations précédemment inconnus à partir de grandes bases de données textuelles".

Le Text Mining réfère ainsi à l'ensemble des techniques et méthodes du data mining, en vue de retrouver, dans les textes de documents de grandes bases de données textuelles, l'information pertinente, utile, et précédemment inconnue.

III. Le processus de fouille du texte (FDT) :

Le processus de Text Mining s'effectue en trois étapes suivantes (voir le schéma suivant):

(1) La modélisation du contenu des textes ;

(2) Les outils de fouille de données proprement dits ;

(3) Le module d'analyse des résultats et leur validation

La modélisation du contenu des textes permet d'extraire les données à partir des textes. Nous nous appuyons sur une représentation de type : un texte = {un ensemble de mots-clés}.

De la même façon que pour un processus d'ECBD, les outils de FdD constituent le module calculatoire d'un système de FdT. Les algorithmes de fouille de données qu'on peut utiliser et adapter montrent leurs intérêts par la capacité à traiter de grandes masses de données, ce qui permet d'envisager de traiter les données très volumineuses extraites des textes. Par conséquent, l'utilisation des techniques existantes semble pertinente dans le processus de FdT.

La contribution de l'analyste est indispensable pour les étapes d'analyse et la validation des connaissances potentielles extraites car ces deux étapes ne peuvent pas se faire

de façon automatique. Le processus de FdT est semi-automatique. Ce n'est qu'une fois les résultats validés qu'ils prennent le statut de connaissances. Ces connaissances peuvent alimenter une base de connaissances ou être exploitées à nouveau par le processus de FdT afin d'affiner la modélisation des textes. Nous appelons, cette base de connaissances l'ontologie du domaine. De notre point de vue, une ontologie du domaine est une hiérarchie de concepts d'un domaine de spécialité. Chaque concept est représenté par un terme. L'ontologie est une description valable pour une tâche ciblée et dans un domaine restreint. Une ontologie du domaine (Domain ontology) est différente de la définition classique d'une ontologie (top-level ontology) qui sert à représenter des structures conceptuelles et méta-structures applicables à des points de vues philosophiques et logiques de l'univers.

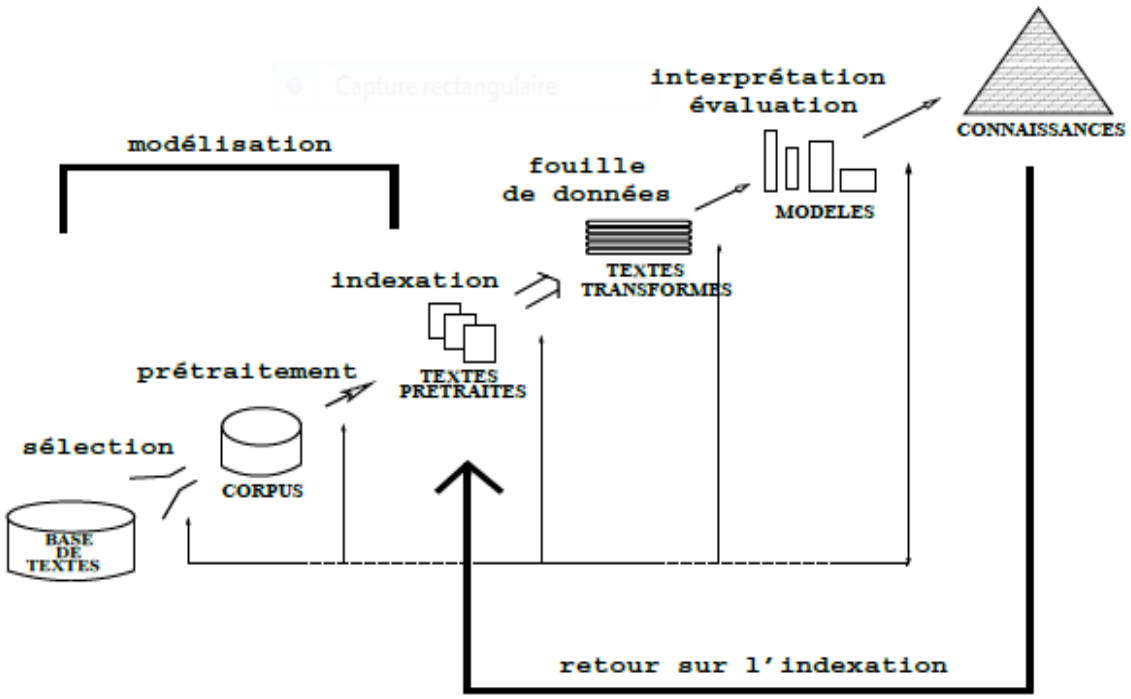


Fig.I.1: Le processus du text mining.

D'une manière plus simple et limpide, le text minig est un procédé consistant à synthétiser (classer, structurer, résumer, …) les textes en analysant les relations, les patterns, et les règles entre les unités textuelles (mots, groupes, phrases, documents).

IV. Etat de l'art :

IV.1. Les techniques de text mining :

Dans cette partie on va présenter un état de l'art sur les déférentes techniques qui opèrent sur le contenu des textes et vise à extraire et structurer des connaissances, et parmi ces techniques on trouve :

IV.1.1. Techniques de traitement de langage naturel (NLP):

Un traitement automatique est :

- ✓ une suite d'actions ou calculs à faire par la machine. Le Traitement Automatique des Langues a pour objectif de traiter des données linguistiques (textes) exprimées dans une langue dite "naturelle" [Delafosse, 1999].
- ✓ La Conception de programmes capables de traiter automatiquement des données linguistiques de type : textes écrits ; dialogues écrits ou oraux ; unités linguistiques (mots, phrases, énoncés, ...).

Les tâches impliquées dans cette technique peuvent inclure la tokenization, élimination des mots vides et filtrage de textes, Lemmatisation, La racinisation (ou troncature) :

IV.1.1.a.La tokenization (segmentation):

D'un point de vue informatique, un texte est un ensemble de mots eux même sont des chaînes de caractères. La tokenization découpe un texte en unités linguistiques de base (les mots). Le principe de la tokenization c'est que chaque chaine de caractères qui précède un séparateur linguistique est considérée comme token. La reconnaissance des tokens peut se faire selon les séparateurs choisis : tous les caractères non alphabétiques (espaces, apostrophes, tirets...) ou les espaces seulement ; et selon que l'on prend en considération les « mots composés » (« pomme de terre » = une unité) ou pas.

On peut aussi opter pour l'utilisation d'un ensemble d'automates choisit qui reconnaissent les tokens (mots), éventuellement en les typant (Lexème, Ponctuation, Nombre)

en parcourant les chaînes de caractère des textes en suivant certaines grammaires bien définit selon le besoin.

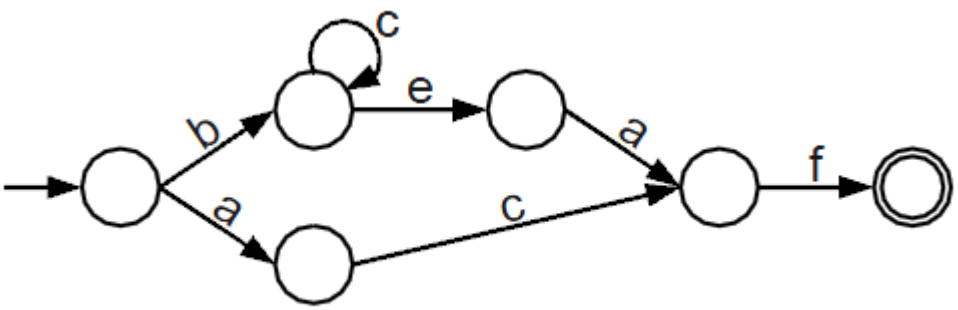


Fig.I.2. Graphe d'automate de l'expression régulière : (bc*ea|ac) f

Exemple 1 : tokenization :

Le collagène peut être employé comme système modèle pour étudier les effets du rayonnement sur une protéine. Les études quantitatives des fibrilles de collagène à partir des images d'électrons-optiques de matériel rayonné et non-rayonné peuvent produire des informations non seulement au sujet des effets structuraux produits par rayonnement, mais également sur la partie réelle de la fibrille étant affectée.

Tokenization

Le | collagène | peut | être | employé | comme | système | modèle | pour | étudier | les | effets | du | rayonnement | sur | une | protéine |. | Les | études | quantitatives | des | fibrilles | de | collagène | à | partir | des | images | d | ' | électrons | - | optiques | de | matériel | rayonné | et | non | - | rayonné | peuvent | produire | des | informations | non | seulement | au | sujet | des | effets | structuraux | produits | par | rayonnement | , | mais | également | sur | la | partie | réelle | de | la | fibrille | étant | affectée |.

IV.1.1.b. élimination des mots vides et filtrage de textes :

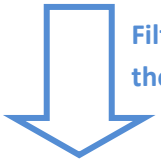
Après la tokenization, les documents sont filtrés en enlevant les mots sans importance (mots vides) qui sont généralement fréquents. Ils sont supprimés du contenu de documents en utilisant une liste qui contient tous les mots vides à supprimer (par exemple des pronoms, des déterminants, des prépositions et des conjonctions, des adverbes communs et des verbes non-instructifs (par exemple, être employé comme auxiliaire)) ou on utilisant les statistique (La pondération des termes) pour mesuré l'importance d'un terme dans un document. Les accents sont supprimés et les lettres majuscules sont remplacées par des minuscules et les caractères spéciaux, parenthèses, virgules, etc., sont remplacés par des espaces entre les mots dans les documents.

Le filtrage peut varier selon les besoins de traitement souhaité. Il peut aussi suivre un vocabulaire d'un domaine particulier (biomédical, politique) en gardant seulement les termes contenu dans ce vocabulaire. Pour le faire le filtrage s'appui sur un thésaurus.

Un thésaurus est un vocabulaire contrôlé qui regroupe un ensemble de concepts relatifs à un certain domaine. Il constitue un moyen de décrire ce domaine et de définir les concepts de ce domaine. Les grands thésaurus peuvent être fragmentés en micro-thésaurus couvrant chacun un sous thème particulier.

Exemple 2 : filtrage :

Le collagène peut être employé comme système modèle pour étudier les effets du rayonnement sur des protéines. Les études quantitatives des fibrilles de collagène à partir des images d'électrons-optiques de matériel rayonné et non-rayonné peuvent produire des informations non seulement au sujet des effets structuraux produits par rayonnement, mais également sur la partie réelle de la fibrille étant affectée.



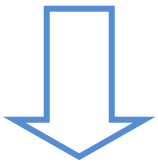
Filtrage en utilisant un thésaurus biomédical

Collagène électrons rayonnement protéines

IV.1.1.c. Lemmatisation:

La lemmatisation (stemming en anglais) c'est l'opération qui consiste à ramener les formes fléchies (conjuguées, plurielles) à des formes standard ou canonique (infinitif ou singulier).Dans cette phase un dictionnaire des formes canoniques des termes d'une langue est utilisé. Chaque terme rencontré dans un texte sera cherché dans ce dictionnaire. S'il y existe il sera remplacé par sa forme canonique sinon il sera inchangé. Dans la pratique cette technique est généralement évité parce qu'elle est très couteuses en temps.

Exemple 3 : si on essaye de lemmatiser le texte filtrer de l'exemple 2 dernier sa donnera :



Collagène électron rayon protéine

IV.1.1.d. La racinisation (ou troncature) :

La racinisation consiste à supprimer le suffixe (et plus rarement le préfixe) des mots d'un texte pour ne garder que les racines (stem en anglais) des mots. Ramener les termes d'un texte à leurs racines permit d'éliminer les différences dues à des formes particulières des mots. Il n'existe pas de manière unique de procéder et plusieurs algorithmes sont disponibles comme l'algorithme ''Porter'' pour l'anglais.

Exemple 4 : raciniser le texte filtrer de l'exemple 2 donnera le même résultat que l'exemple3.

IV.1.2.Les techniques d'extraction d'information (EI) :

L'EI est peut-être la technique la plus utilisée dans des opérations de prétraitement des textes. Elle consiste à définir et identifier l'information précise dans un texte. Sans les techniques d'EI, les systèmes de fouille de textes auraient des possibilités plus limitées de découverte de la connaissance. L'EI doit être distingué de la recherche documentaire (ou recherche d'information). La recherche documentaire renvoie les documents qui apparient à une requête donnée mais exige toujours de l'utilisateur de lire ces documents pour localiser l'information pertinente. L'EI, vise pour sa part, à indiquer exactement l'information pertinente et à la présenter dans un format structuré.

IV.1.3.La catégorisation (la classification supervisée) :

La classification de textes a pour objectif de regrouper les textes similaires, c'est à dire thématiquement proches, au sein d'un même ensemble. L'intérêt d'une telle démarche est d'organiser les connaissances de façon à pouvoir effectuer, par la suite, une recherche ou une extraction d'information efficace.

L'objectif d'un système de catégorisation est d'approximer la fonction de catégorisation exacte qui associe à chaque couple de (document / classe) une valeur (vraie / fausse), en fonction de l'appartenance ou non du document à la classe :

$$\varphi : D \times C \longrightarrow \{T, F\}$$

La classification de textes peut être un support pour différentes applications parmi lesquelles :

- Le filtrage : consiste à déterminer si un document est pertinent ou non (décision binaire),
- Le routage : consiste à affecter un document à une ou plusieurs classes parmi n,
- L'indexation automatique de textes : consiste à associer à chaque texte d'une collection un ou plusieurs termes parmi un ensemble prédéfini. l'objectif est de décrire le contenu de ces textes par des mots ou des phrases clés qui font partie d'un ensemble de vocabulaire contrôlé comme des classes, l'indexation de textes peut être alors vue comme une forme de classification de textes, [Tze 1993].

Parmi les méthodes de la classification supervisée on trouve :

IV.1.3.a .Arbre de décision :

Les arbres de décision sont les plus populaires des méthodes d'apprentissage. Les algorithmes connus sont ID3 et C4.5 . Ils sont également populaires pour la classification de documents.

Dans l'arbre de décision un texte est représenté par un vecteur caractéristique dont les composantes sont des couples de termes-clés du texte (les propriétés de l'objet) associés à leurs fréquences. Les classes prédéfinies correspondent aux feuilles de l'arbre de décision.

Chaque nouveau texte doit être assigné à une seule des classes déjà prédéfinies. La classe cible d'un nouveaux texte est celle dont la règle de décision est la plus proche du vecteur caractéristique du texte. [Apt 1998], [Mitc1997].

Exemple d'un arbre de décision :

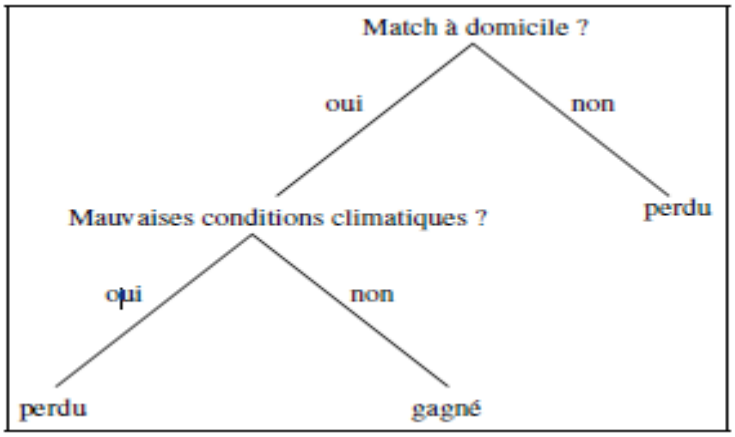


Fig. I.3.: exemple d'un arbre de décision.

IV.1.3.b. les règles de bayes :

Nous rappelons l'expression mathématique de la formule de bayes qui prend en considération la probabilité a priori d'apparition des documents des déférentes classes et de leurs distribution dans l'espace des descripteurs : [And 2000]

$$P(C_k|x) = \frac{\Pr_k \cdot f_k(x)}{\sum_{i=1}^{C} \Pr_i \cdot f_i(x)},$$

Ou C : est le nombre de classes.

Avec : $P(C_k|x)$: probabilité a posteriori qu'un document de coordonnées x appartienne à la classe k.

Pr_k : probabilité a priori qu'un document appartienne a la classe k.

$F_k(x)$: la densité de la probabilité de x c'est la classe est k.

Comme indiqué plus haut, la règle de décision de bayes consiste à choisir d'affecter l'individu à la classe dont la probabilité a posteriori (calculé par la formule de bayes ou par toute autre méthode) est la plus grandes. On démontre que cette décision minimise le risque d'erreur de classification.

IV.1.3.c. K plus proches voisins :

Plus connu en anglais sous le nom K-nearest neighbor (K-NN) ou encore Memory Based Reasoning. Pour classer un nouveau cas (où ranger un nouveau document), l'algorithme cherche les K plus proches voisins de ce nouveau cas et prédit la réponse la plus fréquente de ces K plus proches voisins. La méthode utilise donc deux paramètres : le nombre K et la fonction de similarité pour comparer le nouveau cas aux cas déjà classés.

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

Exemple :

Classer un nouveau document xq. Si on choisit K = 1, xq sera classé +. Si K = 5, le même xq sera classé -. On voit donc que le choix de K est très important dans le résultat.

Sur la figure de droite, on a représenté les exemples par des points. Chaque surface autour d'un exemple montre les positions possibles de nouveaux cas à classer où le résultat de la classification sera la classe de l'exemple si K=1. Cette figure est aussi connue sous le nom de diagramme de Voronoi.

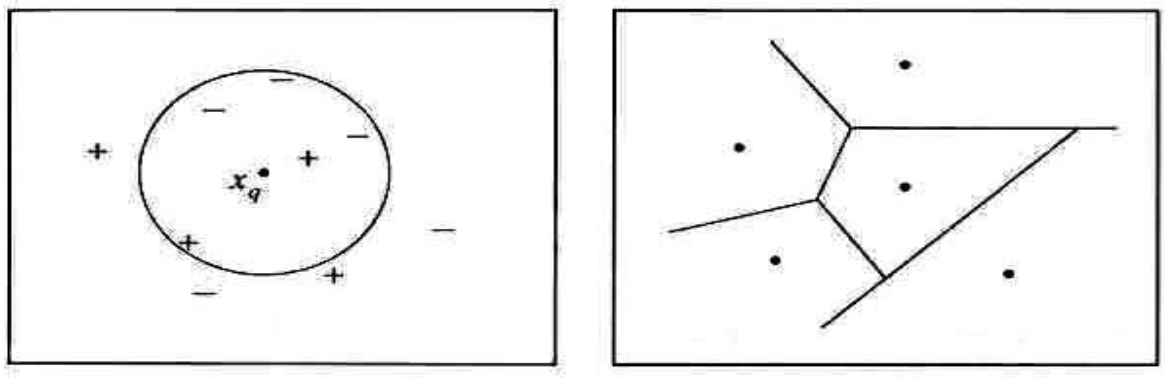


Fig. I.4 : le diagramme de Voronoi.

IV.1.3.d. Les réseaux de neurones :

Un réseau de neurone est un réseau d'unités, où les unités d'entrée représentent les termes, l'unité(s) de sortie représente la catégorie ou les catégories d'intérêts, et les poids sur les bords reliant les unités représentent les relations de dépendance. Pour classer un document de test dj, ses poids wkj sont chargés dans les unités d'entrée ; l'activation de ces unités se propage à travers le réseau, et la valeur de l'unité de sortie(s) détermine la décision du classement.

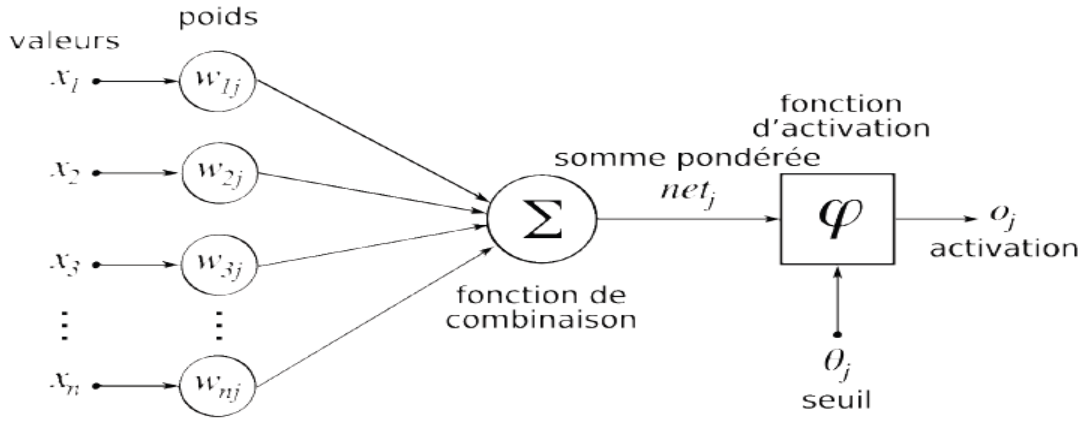


Fig. I.5: un réseau de neurone.

IV.1.4. Clustering (La classification non supervisé):

La classification non supervisée des documents ou clustering, c'est à dire la découverte de classes de documents sans à priori (on ne connaît pas les classes à l'avance). La différence entre le clustering et la classification est que dans le clustering il n'y a pas de variables sortantes. La tâche de clustering ne classifie pas, n'estime pas, ne prévoit pas la valeur d'une variable sortante. Au lieu de cela, les algorithmes de clustering visent à segmenter la totalité de documents en des sous groupes relativement homogènes. Ils maximisent l'homogénéité à l'intérieur de chaque groupe et la minimisent entre les différents groupes.

Nous présentons la classification non supervisée de textes par les méthodes suivantes :

IV.1.4.a. Les Réseaux bayésiens :

Un réseau bayésien est un graphe acyclique son but est de représenter les relations probabilisées entre un ensemble des variables. Il permet aussi d'apprendre la relation causale qui peut nous aider de faire des décisions.

On pourra définir un réseau bayésien comme suit :

• un graphe acyclique orienté G, G=G (V, E) où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G.

• un espace probabilisé fini (W, p).

• un ensemble de variables aléatoires associées aux nœuds du graphe et définies sur [W,p] tel que: P(V1,V2,….VN) = Π P(Vi \C(Vi)).i=1…n.

Avec C(Vi) est l'ensemble des parents de Vi dans le graphe. [Den 2000].

Exemple :

Chaque document est représenté par un vecteur de caractéristiques X(x1x,…. ;xmx). Les caractéristiques sont les termes qui peuvent bien identifier les classes de documents.

Le réseau bayésien utilisé est très simple il contient 2 couches, il se base sur une hypothèse que les caractéristiques de documents sont indépendantes. Dans ce réseau les nœuds Vi représentent les caractéristiques et les nœuds Cj représentent les classes :

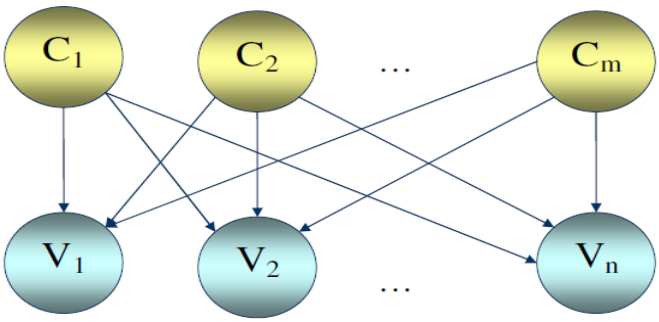


Fig. I.6: un réseau bayésien.

IV.1.4.b.La méthode K_means :

La méthode K_means a été proposée par [Mac 1967].Cette méthode suit une procédure simple de classification d'un ensemble de documents en un certain nombre K de clusters fixé à priori. Chaque cluster est caractérisé par son centre (prototype) qui se trouve être la moyenne des éléments composants le cluster. L'algorithme suit les étapes suivantes :

- Choisir K éléments initiaux « centre » des K clusters
- Placer les documents dans le cluster de centre le plus proche
- Recalculer le centre de gravité de chaque cluster
- Itérer l'algorithme jusqu'à ce qu'il ne soit plus possible de réaffecter les documents(les documents ne changent plus de cluster).

  Exemple :

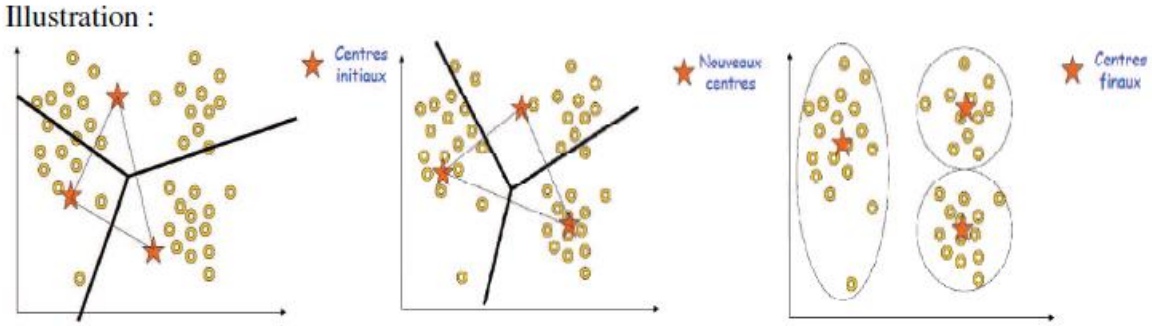


Fig. I.7: exemple de la méthode K_means.

Exemple :

Les objets sont à 1 dimension :

- A= {1, 2, 3, 6, 7, 8, 13, 15,17}. Créer 3 clusters à partir de A avec n=9 (9 objets à segmenter)
- On prend 3 objets au hasard. Supposons que c'est 1, 2 et 3. Cela donne C1= {1}, G1=1,

  C2= {2}, G2=2, C3= {3} et G3=3

- Chaque objet O de A est affecté au cluster dont le centre est le plus proche.

  6 est affecté à C3 car d (6, G3) < d (6, G2) et d (6, G3) <d (6, G1) on a donc après affectation des objets:

- C1= {1}, G1=1, C2= {2}, G2=2, C3= {3, 6,7, 8,13 ,15 ,17}, G3=69/7=9.86

  d (3, G2) <d (3, G3) _3 passe dans C2. Tous les autres objets ne bougent pas.

  C1= {1}, G1=1, C2= {2,3}, G2=2.5, C3= {6,7, 8,13, 15,17} et G3= 66/6=11

- d (6, G2) <d (6, G3) _6 passe dans C2. Tous les autres objets ne bougent pas.

  C1= {1}, G1=1, C2= {2,3, 6}, G2=11/3=3.67, C3= {7, 8,13, 15,17}, G3= 12

- d (2, G1) <d (2, G2) _2 passe en C1. d (7, G2) <d (7, G3) _ 7 passe en C2. Les autres ne Bougent pas.

  C1= {1,2}, G1=1.5, C2= {3,6 ,7}, G2=5.34, C3= {8, 13, 15, 17}, G3=13.25

- d (3, G1) <d (3, G2) _3 passe en 1. D (8, G2) <d (8, G3) _8 passe en 2

  C1= {1,2, 3}, G1=2, C2= {6, 7, 8}, G2=7, C3= {13,15 ,17}, G3=15

Après cela rien ne change

Nos clusters sont finalement :

C1= {1, 2,3}, C2= {6,7 ,8}, C3= {13, 15 ,17}.

IV.1.5.Association :

Une description formelle des règles d'association a été présentée pour la première fois dans les recherches sur le problème "du panier du marche ou panier de la ménagère". Elle est spécifiquement basée sur l'identification des ensembles fréquents. Dans la fouille de textes, les règles d'association ont été appliquées afin d'apprendre des relations de corrélations entre des éléments textuels, par exemple les termes constituant les mots-clés d'un texte.

IV.1.6.Méthodes statistiques :

Les méthodes statistiques génèrent des relations sous-catégorisées qui peuvent correspondre à plusieurs types de relations conceptuelles. Elles n'extraient pas véritablement de relations entre termes mais des nuages de points dans lesquels seul un expert peut retrouver des relations. Toutes ces hypothèses de relations doivent donc être validées systématiquement, sans recours au texte. Ce recours étant techniquement difficile.

Ces méthodes s'adressent donc plutôt à des experts qui les utilisent afin d'avoir une image rapide des textes étudiés.

IV.1.7.Les techniques de visualisation :

Les approches de visualisation pour la fouille de textes supportent généralement un ensemble de buts différents de ceux des interfaces classiques. Bien que les deux visent à rendre l'interaction avec les données possible, les outils de visualisation sont des interfaces graphiques plus sophistiquées incluant les hiérarchies de concepts, les graphes d'associations entre concepts, les histogrammes, les courbes, les graphes circulaires, les cartes a auto-organisation …

Conclusion :

Dans ce chapitre, nous avons essayé de donner une brève introduction au large champ de texte mining. Par conséquent, nous avons motivé ce champ de recherche, nous avons donné une définition plus formelle des techniques présentées ci-dessus et nous avons présenté une brève vue d'ensemble des méthodes de texte mining, de leurs propriétés et de leurs applications aux problèmes spécifiques.

Parmi les techniques du text mining on s'intéresse au règles d'association qui est l'objectif de notre mémoire et qu'on décrira dans le chapitre2.

Chapitre II :

Les règles d'association

I. Introduction :

Traiter un grand volume de données sauvegardé sur un support informatique ne nous permet pas d'avoir une vue complète sur les informations contenues dans ces données. Souvent, des informations restent cacher, invisibles pour l'utilisateur. Ces informations cachées sont la plus part de temps importantes aux yeux de l'utilisateur. Pour les exploiter, on utilise les méthodes de Data Mining.

La méthode de Data Mining la plus intéressante dans ce contexte est l'extraction de règles d'association. Cette méthode met en valeur les relations cachées qui existe dans une grande collection de données. L'exemple de règle d'association le plus connu : '' Dans super marché, un homme qui achète des couches pour bébés, achète aussi 2 packs de bières dans 65% des cas''. Dans cet exemple, cette règle pourra inciter le gérant du super marché à faire des réductions sur des achats des couches pour bébés et des packs de bières. Ce type de connaissances est généralement utile pour la prise de décision (exemple : Choisir les produits à mettre en promotion, organiser l'emplacement des produits, ...etc.).

L'extraction de règles d'association de Data mining sont aussi utilisées dans la fouille de texte pour analyser un texte ou une collection de textes afin extraire des connaissances dissimulées qui existe entre les termes. C'est cette technique de fouille données sera étudier durant toute la suite de notre mémoire.

II. Les règles d'association dans la Fouille de Texte :

L'extraction de règles d'association est une méthode qui est aussi utilisée pour l'analyse des textes. Son objectif est de découvrir des liens caché qui existent entre les termes des textes d'un corpus. Ces liens sont exprimés à travers des règles du type $A \rightarrow B$.

II.1. Définition d'une règle d'association :

On peut définir une règle d'association comme une règle de la forme $B \rightarrow H$ dans laquelle B et H sont des ensembles de terme. Une telle règle peut être interprétée de la façon suivante : **"**Les documents qui possèdent les termes de B possèdent également les termes de H**"**. B et H sont appelés des motifs (ou itemsets) [Agrawal et al. 1993].

II.1. Définition d'un motif :

Nous reprenons certaines définitions utilisées (dans [H. CHERFI, 2005], [Bastide, 2000] et dans [Pasquier et al. 1999]) déjà avant nous pour la définition des motifs et les motifs fréquents.

Soit ''T'' et ''D'' deux ensembles et R une matrice. 'T'' est un ensemble de termes et 'D'' est un ensemble de textes tel que:

$T = \{ a, b, c, d, e\}$ et $D = \{ d_1, d_2, \dots., d_6\}$

La matrice R représente la relation binaire qui existe entre l'ensemble T et D.

R	d_1	d_2	d_3	d_4	d_5	d_6
a	1	0	1	0	1	0
b	0	1	1	1	1	1
c	1	1	1	0	1	1
d	1	0	0	0	0	0
e	0	1	1	1	1	1

Tableau.II.1. Matrice Termes-documents

On appelle un motif tous les sous-ensembles de T. Un motif ''t'' est inclus dans un texte ''d_i'' si $\forall t \in T, R_{(t,d_i)} = 1$. Un motif de taille K est appelé k-motif. Par exemple, d_3 et d_5 contiennent le 4-motif $\{a, b, c, e\}$.

II.2. Définition d'une Image d'un motif :

L'image d'un motif ''T'' est l'ensemble des textes qui le contiennent. On le calcule en utilisant la fonction : $f(T) = \{ d_i \in D \mid d_i \text{ contient } T\}$. L'image d'un motif est appelée l'extension du motif.

Exemple : l'extension du 4-motif $\{a, b, c, e\}$ est l'ensemble de textes : $\{ d_3 , d_3 \}$.

$f(\{a, b, c, e\}) = \{ d_3 , d_3 \}$.

II.3. Définition de la fermeture d'un motif :

La fermeture d'un motif ''G'' est un motif ''F'' tel que ''F'' apparait dans les mêmes textes que ''G''. Pour la calculer on utilise deux fonctions de la Correspondance de Galois :

- f : associe à un motif les textes ou il apparait, elle est définit comme suit :

$$f(T) = \{d \in D \mid \forall t \in T \quad t \subset d\}.$$

- g : associe à un ensemble de textes les motifs qu'ils ont en commun, elle est définit comme suit :

$$g(D) = \{t \in T \mid \forall d \in D \quad t \subset d\}.$$

La fermeture d'un motif ''F'' est donc calculée de la manière suivante :

$$\text{fermeture}(F) = g \circ f(F).$$

II.4. Définition de Support d'un motif :

Le support d'un motif ''t'' est défini par sa fréquence d'apparition dans l'ensemble D. Lorsqu'il est exprimé en absolu, il représente le nombre de documents qui contiennent le motif ''t'' donc il est égal au cardinal de l'image de T, i.e. $\text{support}(T) = |f(T)|$. Sinon, il est égal à la fréquence d'apparition du motif dans l'ensemble de textes D, i.e. $\text{support}(T) = \frac{|f(T)|}{|D|}$.

II.5. Définition d'un Motif fréquent :

Si un motif ''t'' apparaît un nombre de fois et ce nombre est supérieur à un support minimal dans l'ensemble de textes D alors on dit qu'il est fréquent, i.e. $\text{support}(t) \geq \text{minsup}$ où minsup est le support minimal qui est donné par l'utilisateur.

Exemple :

Si $\text{minsup} = 3$ alors le motif $\{a, c, e\}$ n'est pas fréquent car $|f(\{a, c, e\})| = |\{d_3 d_5\}| = 2$.

Remarque : Nous notons l'ensemble des motifs fréquent de la façon suivante :

$F = \{T \subseteq T \mid \text{support}(T) \geq \text{minsup}\}$.

II.5.1. Propriétés des motifs fréquents :

Propriété 1 :

Soit deux motifs A et B tel que B. Si $A \subseteq B$ alors $\text{supp}(A) >= \text{supp}(B)$ car toutes les documents de D qui contiennent le motif B contiennent aussi nécessairement le motif A.

Exemple :

$A = \{b, c\}, B = \{a, b, c\}$. $A \subseteq B$ et $\text{supp}(A) = 4, \text{supp}(B) = 2 \rightarrow \text{supp}(A) > supp(B)$.

Propriété 2 : Si ''B'' est un motif fréquent et ''A '' est un sous-ensemble de ''B'' alors le motif ''A'' est un motif fréquent.

Propriété 3 : Si ''B'' est un motif non fréquent et ''A '' est un sur-ensemble de ''B'' alors le motif ''A'' est un motif non fréquent (Propriété anti-monotonie).

II.6. Définition d'un Motif fermé :

On dit qu'un motif ''F'' est « fermé » si et seulement si $\text{fermeture}(F) = F$.

II.7. Définition d'un Motif fermé fréquent :

FF est un motif fermé fréquent si et seulement si il est un motif fermé et sont support est supérieur à un seuil minimal de support [Pasquier 00, PBTL98]. Le nombre de motifs fermés fréquents est généralement bien inferieur au nombre d'ensembles de mots fréquents

II.8. Définition d'un Motif générateur :

Un motif générateur G_k d'un motif fermé F_k de taille ''K'' est un motif minimal dont la fermeture est égale à F_k, Plusieurs motifs peuvent générer le même motif fermé. Mais on dit que ''G'' est un motif générateur d'un motif fermé ''F'', si et seulement si :

$$\nexists\, k' < k, \qquad G_{k'} \subsetneq G_k \text{ tels que } \text{fermeture}(G_{k'}) = F_k.$$

II.8.1. Propriété des motifs générateurs :

- ✓ Si ''G'' est un motif générateur alors in n'existe aucun sous-motifs ''c'' de ''G'' tel que $fermetteur\ (c) = G$.
- ✓ Si ''G'' est un motif générateur fréquent alors tous sous-motifs ''c'' de ''G'' est un motif générateur fréquent ;
- ✓ Si ''G'' est un motif générateur non fréquent alors tous sous-motifs ''c'' de ''G'' est un motif générateur non fréquent ;

II.9. Définition des motifs maximaux :

Lors de l'extraction des motifs fréquents pour une collection de documents denses, on génère un très grand nombre de motifs fréquents. Parmi ces motifs extraits, ils existent des motifs longs dont la cardinalité est supérieur à celle des autres, ces motifs sont appelés motifs maximaux. On dit qu'un motif ''M'' est un motif maximal si quelque soit ''S'' est sur-ensemble de ''M'', ''S'' est non fréquent.

II.10. Définition de Support d'une règle d'association:

Le support d'une règle d'association $A \rightarrow C$ définit le pourcentage de documents qui contiennent A et C. Il représente le nombre de documents qui contiennent A et C ($\text{support}(A \cup C)$) divisé par le nombre total des documents:

$$\text{support}(A \rightarrow C) = \frac{\text{support}(A \cup C)}{|D|} \ .\ \text{support}(A \rightarrow C) \in [0,1]$$

II.11. Définition de la confiance d'une règle d'association :

La confiance est une mesure permettant d'évaluer la validité d'une règle d'association. La confiance d'une règle d'association $A \rightarrow C$, notée $\text{conf}(A \rightarrow C)$ représente la proportion de documents qui contient A et qui contient aussi C. Elle est définie comme suit :

$$\text{conf}(A \rightarrow C) = \frac{\text{support}(A \cup C)}{\text{support}(A)} \ . \ \text{conf}(A \rightarrow C) \in [0,1]$$

II.12. Les règle d'association informative :

Tout motif fermé est susceptible d'engendrer un grand nombre de règles d'association. Parmi ces règles, il y en a qui sont redondantes. Une règle est dite redondante par rapport à d'autres règles d'association, si l'information qu'elle apporte est exprimée dans d'autres règles. Une règle d'association redondante est donc inutile ou moins informative. Exemple :

1) $a \rightarrow b$.
2) $a \rightarrow bc$
3) $a \rightarrow bd$
4) $a \rightarrow bcd$

On remarque que les règles 1, 2, 3 n'apportent aucune information supplémentaire par rapport à la règle 4 qui est la plus générale. Elles sont redondantes donc inutiles. Les ensembles des règles informatives sont des ensembles de taille réduite qui minimisent le nombre de règles d'association générées tout en maximisant la quantité et la qualité des informations convoyées.

Une règle d'association $R : B \rightarrow H$ est informative, s'il n'existe pas de règle $R' : B' \rightarrow H'$ telle que :

- $\text{support}(R) = \text{support}(R')$.
- $\text{confiance}(R) = \text{confiance}(R')$.
- $B \supseteq B'$ et $H \subseteq H'$.

Soit B et H deux motifs particuliers de T pris tels que :

- $B \cup H = \{\text{fermeture}\} = f \circ g(B)$.
- $B = \{\text{générateur}\}$ l'ensembledes motifs générateurs,
- $H = \{\text{fermeture}\} \setminus \{\text{générateur}\}$ l'ensemble des fermeteur privées de générateurs.

Une règle d'association informative [Pasquier, 00] $R: B \rightarrow H$ est, en pratique, calculée par :

$$B \rightarrow \big((B \cup H) \setminus B\big) \text{ avec } \begin{cases} B \in \min[B \cup H] \text{ est un motif générateur fréquent} \\ (B \cup H) = \text{Max}[B \cup H] \text{est un motif fermé fréqquent} \\ B \subsetneq B \cup H \text{ inclusion stricte pour } H \neq \emptyset \end{cases}$$

II.13. Définition des règles maximales :

Les règles d'associations maximales ont été introduites pour éliminer le problème expliqué dans l'exemple suivant :

Soit D une collection de documents biomédicaux d'un hôpital. Supposant que A et B sont deux maladies et que X, Y, Z, W des médicaments. On suppose aussi que :

- ✓ X, Y, Z, W guérissent la maladie A et que X ou Y guérissent la maladie B (l'un ou l'autre pas les deux au même temps).
- ✓ La règle fréquence d'apparition de A est 70% et celle de B est de 15%

Si on cherche les règles d'association normales on trouvera la règle d'association $XYZW \rightarrow A$, mais ne trouvera pas les deux règles $X \rightarrow B$ et $Y \rightarrow B$ malgré qu'elles sont pertinentes. Ce problème est dut au fait qu'il y a beaucoup de liens qui relie A à X, Y, Z, W car le support de A est élevé, contrairement à B qui à un petit support et donc moins de liens avec X et Y.

Les règles d'association maximales représentent un moyen efficace pour contourner ce problème. Elles permettent d'extraire des relations moins intéressantes qui ne peuvent pas être extraite par des règles d'associations ordinaires. Pour trouver les règles $X \rightarrow B$ et $Y \rightarrow B$ on s'intéresse à capturer la notion suivante : à chaque fois que X (ou Y) apparait seul alors B apparait aussi.

On dit que X qui est d'une catégorie spécifique appelons la G (dans notre exemple, il est de la catégorie médicament) qu'il apparait seul dans un document D_i, si et seulement si $D_i \cap G = X$. Ce qui signifie que X est le plus grand élément de G contenu dans D_i, Alors X est maximal.

Exemple :

Si on considère les 3 documents biomédicaux suivant :

$D_1 : X, A, Z$

$D_2 : Y, W, B$

$D_3 : X, A, B$

Tel que : X, Y, Z, W appartient à la catégorie médicament qu'on note G_1 et A et B à la catégorie maladie G_1 .

Si on prend en considération le motif X qui est de la catégorie G_1, on remarquera alors que ce motif :

- ✓ N'est pas seul dans D_1 car : $D_1 \cap G_1 = \{X, Z\} \neq \{X\}$
- ✓ Alors que, dans D_3 est seul par ce que $D_3 \cap G_1 = \{X\}$

Une règle d'association maximale à la même forme qu'une règle normale i.e. $X \rightarrow A$, avec X et Y deux motifs distinct qui appartiennent à deux catégories différentes.

II.13.1. Le support d'une règle d'association maximale :

Le support d'une règle d'association maximale $X \rightarrow A$ est le nombre de document qui contient le motif maximal X (le seul de sa catégorie) et le motif A. On note le support d'une règle d'association maximale par M-support.

II.13.1. La confiance d'une règle d'association maximale :

Pour calculer la confiance de la règle d'association maximale $X \rightarrow A$. On génère un sous-ensemble SD de la collection D, tel que SD qui contient X comme motif maximal et contient au moins un motif de la catégorie A. Elle est notée M-confiance.

La M-confiance est alors calculée de la manière suivante :

$$M - Confiance\ (X \rightarrow A) = \frac{\text{M} - \text{support}\ (X \rightarrow A)}{|SD|}$$

Exemple :

Prenons la collection suivante :

$D_1 : X, B$

$D_2 : Y, Z, B$

$D_3 : X, A, B$

Et les deux catégories G_1et G_2 :

$G_1 = \{X, Y, W, Z\}$

$G_2 = \{A, B\}$

Et on prend en considération la règle maximale suivante : $X \rightarrow A$. Le calcul de la confiance de cette règle se fait de la façon suivante :

- ✓ Calculer le support de la règle $X \rightarrow A$. Dans notre on aura : M-support ($X \rightarrow A$)=1car on a que le document D_3 qui contient X comme motif maximal et le motif A en même temps.
- ✓ Trouver la sous-collection SD qui contient au moins un motif de la catégorie de X et X comme motif maximal.

$$SD = \{D_1, D_3\}$$

- ✓ Calculer la M-confiance de la $X \rightarrow A$:

$$M - Confiance\ (X \rightarrow A) = \frac{\text{M} - \text{support}\ (X \rightarrow A)}{|SD|} = \frac{1}{2} = 0.5 = 50\%$$

III. Extraction de règles d'association :

La plus part des algorithmes de recherche de règles d'association (parmi eux : apriori) adopte une stratégie qui consiste à décomposer le problème en deux étapes :

1. Génération des ensembles d'items fréquents ;
2. Génération des règles d'association. Dont l'objectif est d'extraire toutes les règles de grande confiance à partir des ensembles d'items fréquents trouvés dans l'étape précédente [Guillaume, 2000].

III.1. Extraction des motifs fréquents :

Cette étape consiste à générer tous les motifs possibles à l'aide des termes d'une collection de documents textuels. Puis, calculer le support de chaque motif en parcourant la collection pour trouver le nombre de documents qui le contient, comparer la valeur trouvée avec la valeur de support minimal, et enfin ne garder que les motifs dont le support est supérieur au support minimal. Elle permet d'éliminer les motifs non fréquents qui sont sans intérêt pour générer les règles d'association. Les résultats de cette étape sont utilisés dans l'étape suivante, donc elle est primordiale.

III.2. Génération des règles d'association :

L'ensemble des motifs fréquents générer dans l'étape précédente est utilisé dans cette étape pour extraire une règle d'association. Pour cela, on appuie sur deux mesures : le support et la confiance d'une règle d'association. Ces deux mesures permettent de vérifier la validité des règles d'association en comparant leurs valeurs avec les valeurs de minsupp et minconf choisit pat l'utilisateur. Si leurs valeurs sont supérieurs alors la règles extraite et valides.

IV. Les algorithmes de génération de règles d'association :

Il existe plusieurs algorithmes de génération de règles d'association. Ils utilisent suivants les notions de support et de confidence pour déterminer la pertinence des règles d'associations. Parmi eux on site :

IV.1. Apriori :

Apriori est un algorithme classique de recherche de règles d'association introduit par Agrawal et Al. C'est le premier algorithme destiné à la recherche de règles d'association. Apriori génère les motifs fréquents puis les relie entre eux pour générer les règles d'association. Il se base essentiellement sur la propriété d'anti-monotonicité existant entre les motifs. Elle est utilisée à chaque itération de l'algorithme Apriori afin minimiser au maximum le nombre de motifs candidats à tester.

IV.2. Apriori-TID :

C'est une version améliorée de l'algorithme Apriori. Il permit d'éliminé l'un des grands inconvénients de Apriori qui le nombre de parcours important de la collection de documents. Apriori et Apriori-TID utilisent la même technique pour générer les motifs candidats mais ils déférent dans le calcul du support des motifs candidats. Apriori-TID génère un ensemble C_K de la forme $(D_i, \{c_K\})$ tel que : $\{c_K\}$ est l'ensemble qui contient tous les motifs de cardinalité K identifiée dans un document D_i. Grace à cet ensemble, le support d'un motif ''a'' est calculé en comptant sont nombre d'occurrence dans l'ensemble C_K. La collection est parcouru juste pour générer le ensemble C_{K+1} jusqu'à ce qu'il n y aura plus motifs candidats (i.e. $C_{K+1} = \emptyset$). Le plus inconvénients de cet algorithme c'est que, lors des premières itérations, C_K peut être très grand ce qui cause un problème de stockage.

IV.3. Partition :

L'algorithme Partition proposé par Savasere et Al. Cet algorithme à la particularité de faire que deux de parcours d'une collection de documents ''D'' on partitionnant (comme son nom l'indique) la collection en ''p'' partitions $D_1, \dots, D_p$. Lors de premier parcours de D, il calcule les motifs fréquents de chaque partition qui sont dit locaux par rapport à la partition. Si un motif local est fréquent alors l'algorithme Partition le considère comme un motif fréquent pour toute la collection D. L'algorithme se décompose en deux étapes :

- ✓ recherche des motifs fréquents locaux sur chaque Di,
- ✓ calcul pour chaque motif trouvé son support sur toute la base de transactions.

Les supports des motifs sur toute la base de transactions sont calculés par intersection des ensembles des documents qui contiennent les motifs. Par exemple, si le motif ''a'' est contenu dans 130 documents et le motif ''b '' est contenu dans 145 documents, alors on déduit le support de motif ''ab'' par $130 \cap 145 = 15$. Le nombre 15 représente le nombre de documents qui contient ''a'' et ''b''. L'algorithme Partition, contrairement à Apriori, a

l'avantage de n'effectuer que deux parcours de D. Par contre, il génère plus de motifs candidats, qui sont généralement peu fréquents, que l'algorithme Apriori.

IV.4. Eclat :

Eclat proposé par Zaki et Al. A une stratégie déférente que celle d'Apriori. Au lieu de chercher pour chaque document les motifs qu'il contient, on prend chaque motif et on cherche l'ensemble des documents qui le contiennent, c'est ce qu'on appel un format vertical de la collection de documents. Dans ce cas le support est calculé juste en effectuant des intersections entre les ensembles de documents, ce qui s'avère être un très grand avantage. Cette stratégie permit de réduire la taille de la collection puisque seuls les documents concernant un motif sont utilisés pour l'intersection.

Si deux K-motifs ont un préfixe commun de taille (k-1) on dit qu'ils appartiennent à une même classe d'équivalence. Par exemple ''ABC'' et ''ABD'' qui ont 3 comme cardinalité appartiennent à la même classe d'équivalence par ce qu'ils ont un préfixe de cardinalité 2 en commun qui est ''AB''. Eclat effectue une recherche des motifs fréquents en profondeur d'abord et se base sur le concept de classes d'équivalence.

Eclat n'utilise pas la propriété d'anti-monotonicité ce qui rend la réduction l'espace de recherche pauvre. Ceci est suffisant pour de petites collections de documents.

IV.5. FP-Growth (Frequent-Pattern Growth)

FP-Growth à fonctionnement complètement nouveau par rapport aux autres algorithmes de recherche de règles association qui sont presque tous basés sur Apriori.

L'objectif de cet algorithme est d'éviter les parcours répétés de la collection de documents. FP-Growth propose une méthode une méthode assez particulière qui consiste à générer les motifs fréquents sans passer par l'étape de génération des motifs candidats qui connu sous le nom FP-Growth ((Frequent Pattern growth). Cette méthode est composée de plusieurs étapes :

- ✓ Compresser la collection en une structure compacte appelée FP-tree (Frequent Pattern tree) ;
- ✓ diviser la FP-tree en sous projections de la collection de documents appelées bases conditionnelles. Tel que Chacune de ces projections est associée à un motif fréquent.
- ✓ Extraire les motifs fréquents sur chaque une des projections séparément.

Un FP-tree est une structure compacte constituée d'un :

- ✓ Arbre : dont la racine est null, chaque nœud de l'arbre contient les informations suivantes :
  - Le motif que représente ce nœud ;
  - sa fréquence ;
  - le nœud suivant dans l'arbre.

- ✓ Index : qui contient la liste des motifs fréquents. A chaque motif est associé un pointeur indiquant le premier nœud de l'arbre contenant ce motif.

L'avantage d'une telle représentation des données est qu'il suffit de suivre les liens qui existent entre les déférents nœuds pour connaître les occurrences d'un motif.

IV.6. Close :

Le principe de l'algorithme close est inspiré de l'algorithme Apriori. Il repose sur l'extraction des motifs fermés fréquents et des motifs générateurs [Pasquier et al. 1999b].

L'algorithme close procède au calcule des motifs fréquents fermés de la manière suivante :

1. Il initialise un ensemble des générateurs avec l'ensemble des singletons formés par les motifs du la collection de documents;
2. Puis, Calcul de la fermeture des générateurs de cardinalité k et de leur support ;
3. Ensuite, ajout les fermetures des générateurs à l'ensemble des ensembles de motifs fermés fréquents si le support de la fermeture est supérieur ou égale à un seuil de support (minsup) ;
4. Enfin, il génère les générateurs de cardinalité k + 1. ils sont obtenus de la même manière que dans l'algorithme Apriori, mais ceux appartenant à la fermeture d'un générateur de cardinalité k sont supprimés.

Cette méthode est très intéressante car des chercheur l'on trouvée fort utile pour la génération de règles d'association informatives (non redondantes).

IV.7. Pascal :

Introduit par Bastide et al. Cet algorithme vise à réduire le cout de calcul des supports des motifs. Pour le faire, il utilise la notion d'équivalence. On dit que les deux motifs ''A'' et ''B'' sont équivalent si le ensemble de documents contenant le motif ''A'' est égal à l'ensemble de documents contenant ''B''. Si deux motifs sont équivalents, ou si leurs supports respectifs sont égaux et dont l'un contient l'autre, alors ils appartiennent à la même classe d'équivalence. Deux motifs appartenant à une même classe d'équivalence ont nécessairement le même support.

Ainsi, il suffirait d'avoir le support d'un seul motif appelé motif clé de la classe d'équivalence pour déduire le support de tous les autres motifs de la même classe sans avoir à accéder à la collection de documents. Un motif est un motif clé s'il ne possède aucun sous-ensemble avec qui a le même support que lui.

V. Utilisation des règles d'association pour la fouille de textes :

L'intérêt des règles d'association dans la fouille de textes est multiple, on peut citer :

- ✓ Filtrage d'une terminologie issue de l'indexation pour la constitution d'un thésaurus
- ✓ Analyse de concepts formels
- ✓ la classification supervisée
- ✓ Extraction d'information
- ✓ Veille technologique et stratégique
- ✓ Recherche d'information (RI)

V.1. Filtrage d'une terminologie issue de l'indexation pour la constitution d'un thésaurus :

Le processus de fouille de textes est sensible à la phase d'indexation. Si un terme est absent de l'indexation d'un seul texte, cela peut altérer la précision des résultats de ce processus. A la fin d'une phase d'indexation manuelle, certain termes qui ne constitue pas des termes-clefs ne sont pas éliminés. On appelle ces termes « des termes bruits ». Les termes bruits ont généralement le même sens, comme les deux termes « syndrome de l'immunodéficience acquise » et « SIDA » qui représentent tous les deux la maladie de SIDA, alors une règle d'association entre ces deux termes de la forme « syndrome de l'immunodéficience acquise » $\Rightarrow$ « SIDA » peut inciter l'utilisateur à corriger ce problème.

En effet l'utilisateur peu remédier ça, on utilisant un processus itératif qui comprend les phases suivantes :

I- Filtrer un terme-bruit repéré dans une règle d'association en éliminant toutes les occurrences de ce terme dans la terminologie ;
II- Extraire les règles s'appuyant sur cette nouvelle terminologie;
III- Retourner au point (I).

A l'issu de ce processus, L'ensemble des termes présents en partie gauche ou droite des règles d'association constitue un thésaurus du domaine.

V.2. Analyse de concepts formels :

Les règles d'association permettent d'organiser des concepts dans une structure hiérarchique à partir de laquelle il est possible d'observer des corrélations entre les individus et leurs propriétés communes. En effet, on peut hiérarchiser les concepts on utilisant correspondance de Galois pour créer un graphe de concept muni d'une relation d'ordre entre les concepts. On appelle ce graphe un treillis de Galois. La construction d'un treillis de Galois permet de se donner une structure mathématique pour l'analyse de concepts issus d'un domaine.

V.3. Extraction d'information (EI) :

L'extraction de règles d'association permet de réaliser des tâches d'extraction d'information pour remplir des patrons. À ce titre, le système TEXTRISE [Nahm et Mooney, 2001] illustre l'application d'un processus de fouille de Texte pour l'EI. TEXTRISE apprend à remplir certains attributs de patrons pour de nouveaux textes à partir de règles d'association apprises sur d'autres patrons. Dans une notice bibliographique par exemple, un patron possède un attribut auteur inconnu aut? mais un attribut mots-clés complet $\{mc_1, mc_2, mc_3\}$. Si durant la phase d'apprentissage nous avons une règle d'association $(mc_1, mc_2 \Rightarrow aut_1)$ appelée *soft-matching rule*, ce texte est attribué, à un degré de confiance près, à aut_1. [HCENE CHERFI].

V.4. Veille technologique et stratégique :

La veille stratégique (appelée également business intelligence). C'est un processus de mise à jour périodique d'informations. Il offre une aide précieuse à la prise de décision pour les gérants d'entreprise. Les règles d'association révèlent des implications entre termes et permettent de faire des suivis scientifiques.

V.5. Recherche d'information (RI) :

La liste de documents pertinents qui constitue une réponse relative à une requête est fondée sur le lien de cooccurrence entre les termes de la requête et leur fréquence d'apparition ensemble dans les textes. L'utilisation des motifs fermés fréquents permet, par navigation dans le treillis de Galois correspondant, de répondre à une requête par les documents constituant l'extension d'un concept.

V.6. la classification supervisée :

L'utilisation de règles d'association est une méthode bien connue pour des tâches de classification supervisée. Les méthodes de classification à base de règles d'association sont nombreuses, populaires, et ont donné lieu à de nombreux travaux originaux pour optimiser le processus.

Ces méthodes utilisent les règles d'association de la forme X → c, où X est un ensemble d'attributs et c, une valeur de classe. Pour un objet à classer contenant les attributs de l'ensemble X, cette règle le met la classe c. Les classifieurs qui utilisant ces règles sont construits en extrayant les règles ayant un certain support et une certaine confiance issue d'un jeu d'apprentissage, puis en les faisant voter pour un nouvel objet à classer.

V. Conclusion :

La qualité des règles d'association extraites dépend des motifs fréquents extraits. L'extraction des motifs fréquents est une tâche très compliqué, très couteuse en temps de calcul et en espace mémoire comparativement à la génération des règles proprement dites. Des études ont été menées pour réduire le coût de la phase d'extraction des motifs fréquents c'est ce qui explique la diversité des algorithmes d'extraction de règles d'association. Ces études et cette diversité des algorithmes et tel qu'on s'ait contenté de représenter que les notions de bases des règles d'association.

Chapitre III : Description de notre approche

I. Introduction :

Appliquer les méthodes d'extractions de règles d'association au domaine biomédical permet de mettre en valeur tous liens qui existent entre les termes biomédicaux d'une collection de document biomédicaux. Cela peut avantager les médecins ou toute personne qui travaille dans ce domaine pour la prise de décision, avoir l'accès à l'information pertinente pour confirmer leur décision, ..., etc. Par exemple, une règle de la forme : $Maladie \rightarrow medicament$ avec un support et une confiance élevée permettra à un médecin d'administrer ce médicament pour guérir cette maladie.

L'objectif de notre approche est d'appliquer l'un des algorithmes d'extraction de règles d'association dans le domaine biomédical. Cet algorithme est l'algorithme apriori. On utilisera une collection de documents biomédicaux extraite à partir d'une source terminologique appelée MEDLINE (Medical Literature Analysis and Retrieval System Online).

Notre approche se compose de 4 étapes :

- ✓ Prétraitement de la collection documentaire ;
- ✓ Création d'une matrice Terme-Document ;
- ✓ Extraction des motifs fréquents ;
- ✓ Génération des règles d'association.

II. Prétraitement de la collection documentaire :

Pour prétraiter notre collection on utilise l'approche d'indexation proposée dans [Ba-Duy DINH, 2012] qui vise à améliorer la RI biomédical. Elle est basée sur l'utilisation des terminologies biomédicales. Cette approche de RI utilise deux méthodes d'extraction de concepts différentes : l'une consiste à extraire les concepts qui sont prédéfinis dans une terminologie, tandis que l'autre vise à extraire les concepts issus de plusieurs terminologies via huit modèles de vote. Ces concepts représentent les sujets sémantiques des documents. Les problèmes liés à la synonymie, à l'utilisation des abréviations ou des acronymes dans le document sont automatiquement résolus, en utilisant les termes préférés qui désignent les concepts.

Cette approche nous permettra de prétraiter notre collection en assurant les taches suivantes :

- ✓ Tokenizer les documents,
- ✓ Élimination des mots vides et filtrage de textes ;
- ✓ Lemmatisation des terems

A la fin de cette phase tous les termes non pertinents pour notre travail seront éliminés et il nous restera que les termes biomédicaux. Ils sont sauvegardés dans des documents qui serviront à la construction de la matrice terme-document.

III. Création d'une matrice Terme-Document :

Les documents prétraités restent comme même des données non structurées. Pour les structurer, on les place dans une matrice appelée **''Matrice termes-documents''** dans laquelle chaque colonne correspond à un terme et chaque ligne représente un document. Chaque cellule contient la fréquence d'apparition d'un terme ''T_i'' dans le document D_j.

Exemple 5 :

Soit la matrice termes-documents ''A'' qui contient les fréquences d'apparition de chaque terme dans chaque document d'une collection de documents donné.

Chaque cellule a_{ij} de la matrice ''A'' est égale à :

$$a_{ij} = \begin{cases} 1, si\ le\ terme\ j\ est\ présent\ dans\ le\ document\ i \\ 0, sinon \end{cases}$$

A	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
D_1	1	1	1	1	0	0	1	0	0
D_2	1	1	0	0	0	1	1	1	1
D_3	1	0	0	0	1	0	1	0	1

Tableau.III.1. Matrice Termes-documents

IV. Extraction des motifs fréquents :

La matrice Terme-document représente tous les documents de notre collection. Cette matrice nous servira comme donnée source structurée pour en extraire les motifs fréquents. Plusieurs Algorithmes assurent cette tâche. L'algorithme original de recherche des motifs fréquents est ''*Apriori* [Agrawal et Srikant, 1994]''. Il est destiné aux données structurées, il a été utilisé pour la première fois sur des données textuelles dans [Feldman et Dagan, 1995]. Pour notre approche, on a choisit d'utiliser cet algorithme pour générer les motifs fréquents et les règles d'association.

L'algorithme Apriori utilise une approche itérative pour générer les motifs fréquents. Il effectue à chaque itération, un passage dans le corpus D afin de calculer le support de chaque motif.

Dans ce qui suit, l'ensemble des k-motifs candidats (i.e. dont on ne connait pas encore le support dans D) sera dénoté par C_k et l'ensemble des k-motifs fréquents de taille k par F_k.

$$C_k = \{(\, c_k;\ \text{supp}(\, c_k))\,|\ \forall X \subsetneq c_k\,, X \neq \emptyset\,, \text{support}(X) \geq \text{supmin}\}$$

$$F_k = \{(\, l_k;\ \text{supp}(\, l_k))\,|\ l_k \text{ est un k} - \text{motif et support}(\, l_k) \geq \text{supmin}\}$$

L'ensemble D est tout d'abord parcouru afin de trouver C_1, l'ensemble des 1-motifs qui existe dans D. L'algorithme entame ensuite une boucle dans laquelle :

1. Il calcul le support de chaque motif ''c'' de l'ensemble des motifs candidats $''C_k''$.
2. Il vérifie si le support calculé est supérieur à minsup : si support (c) ≥ minsup alors le motif ''c'' est ajouté à l'ensemble des motifs fréquents '' F_k''.
3. Puis, il génère les (k+1)-motifs candidat en appelant la fonction Apriori-gen(F_k) qui prend l'ensemble F_k comme argument et retourne l'ensemble C_{k+1}. Elle fonctionne comme suit :
    a. Elle calcule tous les motifs candidats possibles de telle sorte que deux motifs forment un (k+1)-motifs si et seulement s'ils ont (k-1)-motifs en commun.
    b. En suite, elle s'assure que tous les motifs candidats générés respectent la règle d'anti-monotonicité en supprimant ceux qui ne la respectent pas.
4. En fin, il réitère les étape '1', '2', '3' jusqu'à ce que la fonction Apriori-gen() ne génère plus de motifs candidats.

Le corps de l'algorithme Apriori est donné dans la figure suivante :

Apriori(T; minsup)

- Entrée : T : corpus, minsup : entier
- Sortie : $\cup_k F_k$

Début :

$C_1 \leftarrow \{\text{singletons}\}$

$k \leftarrow 1$

Tant que $C_k \neq \emptyset$ **faire**

Pour chaque $c \in C_k$ **faire**

Pour chaque t ∈ T **faire**

Si $c \subset t$ **alors**

$Support(c) \leftarrow support(c) + 1$

Fin pour

$f_k \leftarrow \{ c \in C_k \mid support(c) \geq minsup\}$

Fin pour

$k \leftarrow k + 1$

$C_k \leftarrow \boldsymbol{Apriori - gen}(F_{k-1})$

Fin tant que

Retourner $\cup_k F_k$

Fin

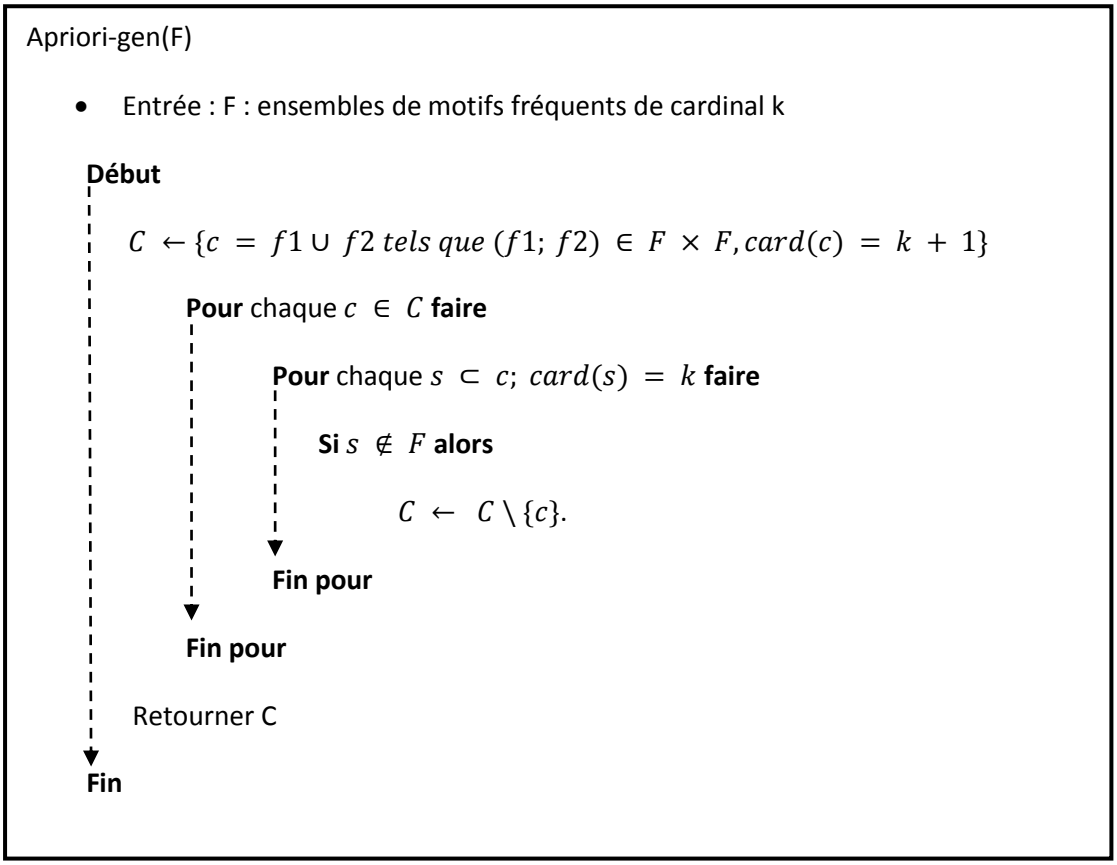


Fig.III.1. Code source de l'algorithme Apriori.

Exemple 6 : extraction des motifs fréquents avec l'algorithme Apriori **:**

Soit : T c'est l'ensemble de termes :

$$T = \{ a, b, c, d, e\}$$

D est l'ensemble de textes :

$$D = \{ d_1, d_2, \dots., d_6\}$$

Et R la matrice termes-documents qui représente la relation binaire entre les ensembles T et D.

Matrice termes-documents R :

R	A	b	C	d	E
d_1	1	0	1	1	0
d_2	0	1	1	0	1
d_3	1	1	1	0	1
d_4	0	1	0	0	1
d_5	1	1	1	0	1
d_6	0	1	1	0	1

Tableau .III.2. Matrice termes-documents

On notera C_k l'ensemble des motifs candidats et F_k l'ensemble des motifs fréquents. On veut extraire des motifs dont le support est supérieur ou égal à 2/6≈33%.

Apriori Génère les motifs fréquents de la manière suivante :

A la première itération de l'algorithme, chaque terme de T est un 1-motif de C_1.

Il parcoure l'ensemble D pour trouver le support de chaque 1-motif en calculant le nombre d'apparence de chaque 1-motif. Pour trouver tous les 1-motifs fréquents, il compare le support chaque 1-motif avec minsup et ne garde que les motifs dont le support est supérieur (support(d) =1/5 inférieur à 2/5 ==> le 1-motif « d » sera supprimé) en les mettant dans F_1.

Puis, il passe à la génération de l'ensemble des 2- motifs candidats en utilisant les motifs fréquents de F_1. Pour trouver l'ensemble C_2, une jointure $F_1 \times F_1$ est effectuée.

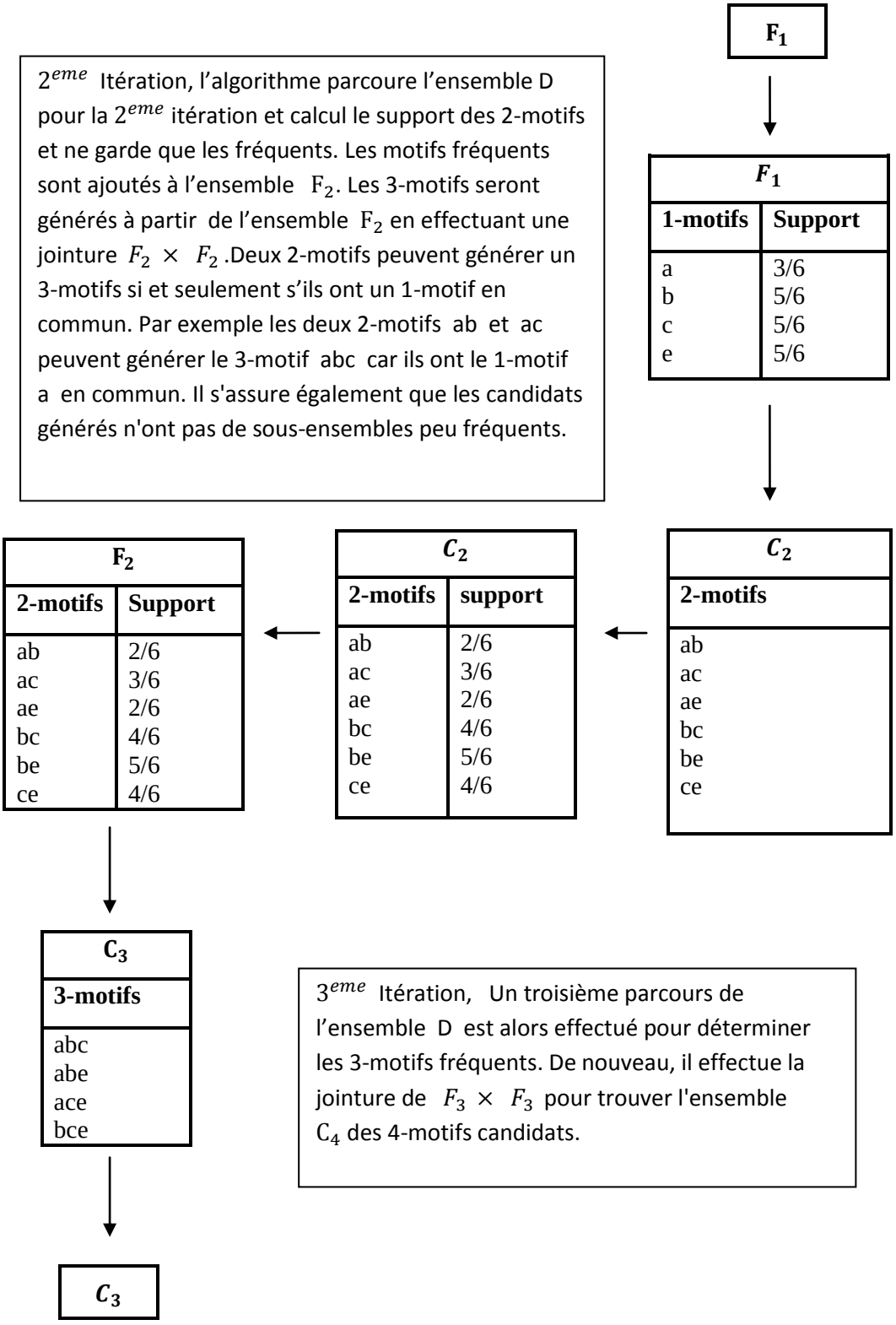
C_1
1-motifs
a
b
c
d
e

↓

C_1	
1-motifs	**Support**
a	3/6
b	5/6
c	5/6
d	1/6
e	5/6

↓

F_1



2^{eme} Itération, l'algorithme parcoure l'ensemble D pour la 2^{eme} itération et calcul le support des 2-motifs et ne garde que les fréquents. Les motifs fréquents sont ajoutés à l'ensemble F_2. Les 3-motifs seront générés à partir de l'ensemble F_2 en effectuant une jointure $F_2 \times F_2$ .Deux 2-motifs peuvent générer un 3-motifs si et seulement s'ils ont un 1-motif en commun. Par exemple les deux 2-motifs ab et ac peuvent générer le 3-motif abc car ils ont le 1-motif a en commun. Il s'assure également que les candidats générés n'ont pas de sous-ensembles peu fréquents.

3^{eme} Itération, Un troisième parcours de l'ensemble D est alors effectué pour déterminer les 3-motifs fréquents. De nouveau, il effectue la jointure de $F_3 \times F_3$ pour trouver l'ensemble C_4 des 4-motifs candidats.

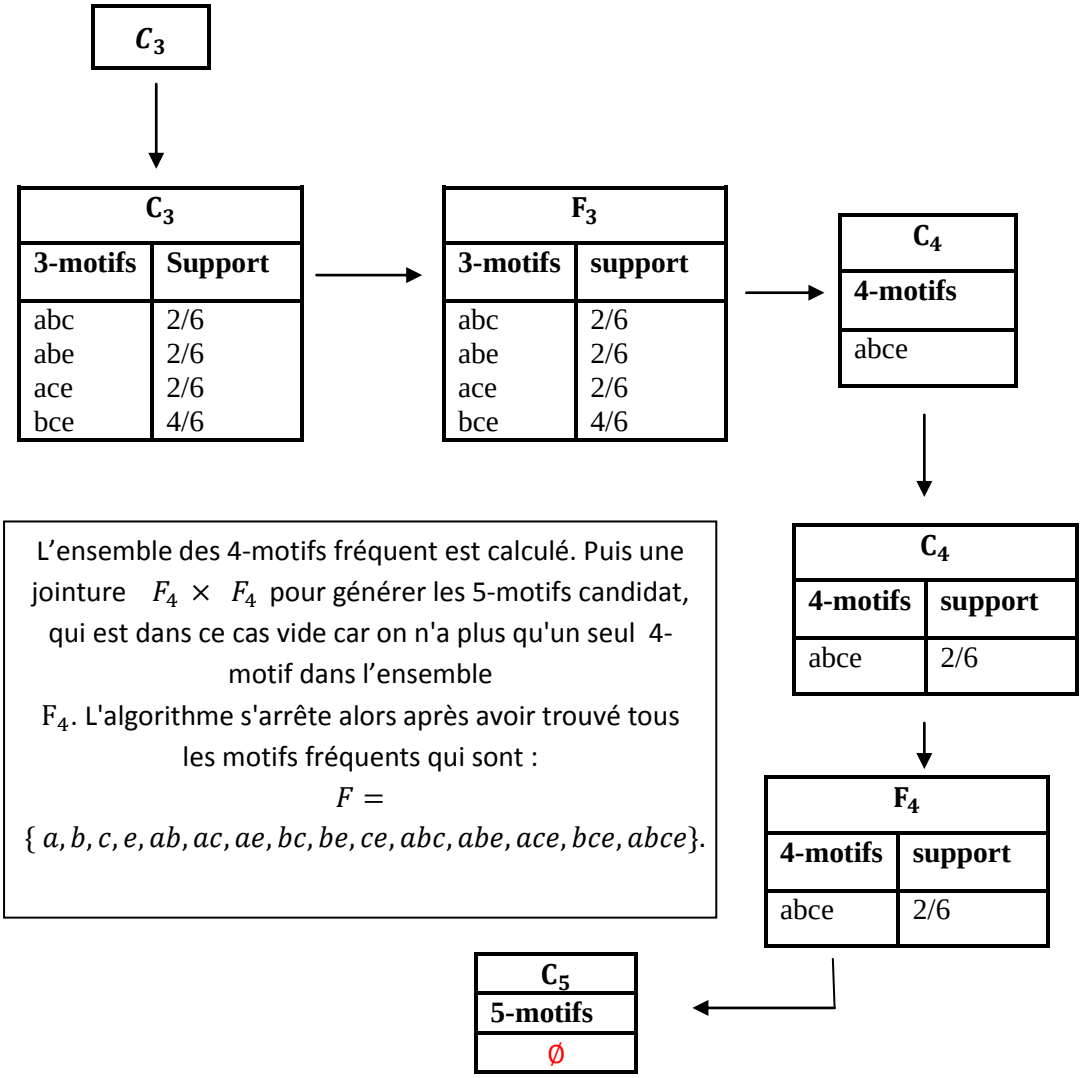


Fig. III.2. Schéma d'extraction des motifs fréquents

V. Génération des règles d'association :

L'ensemble des motifs fréquents générés vont servir pour générer les règles d'association. Pour générer les règles associatives, l'algorithme de génération de règles d'association d'Apriori s'appui sur les deux notions qui sont définis dans le chapitre précédent qui sont le support et la confiance d'une règle:

$$support(A \rightarrow C) = \frac{support(A \cup C)}{|D|} \ . \ support(A \rightarrow C) \in [0,1]$$

$$conf(A \rightarrow C) = \frac{support(A \cup C)}{support\ (A)} \ . \ conf(A \rightarrow C) \in [0,1]$$

L'algorithme de génération de règles d'association d'Apriori s'appel Gen-règles. L'algorithme Gen-règles utilise l'ensemble ''F'' des motifs fréquents trouvés en phase précédente pour générer les règles d'association. Pour chaque motif fréquent ''A'', on considère tous ses sous-ensembles (qui sont tous fréquents d'après la propriété d'anti-monotonicité). A partir de ces sous-ensembles fréquents, on génère toutes les règles dont le support et la confiance sont supérieurs ou égaux à un certain support et confiance minimaux définis par l'utilisateur. Des règles de la forme $(A - C) \rightarrow C$ de la manière suivante :

Soit : *''F''* l'ensemble de mots fréquents de cardinale supérieur ou égale à 2

Et *''R''* un ensemble de règles d'associations vide.

1. Générer les sous-ensembles $e \ de \ F \ tel \ que \ e \neq \emptyset \ et \ e \neq F$.
2. Pour chaque sous-ensemble e :
   a. Générer l'ensemble H des singletons de e $(i.e.card(H) = 1)$.
   b. Pour chaque élément $h \in H$ calculer :

$$conf((\mathrm{e} - \mathrm{h}) \rightarrow \mathrm{h}) = \frac{support(\ e)}{support\ (e - h)}\ .$$

   Si $conf((\mathrm{e} - \mathrm{h}) \rightarrow \mathrm{h}) \geq minconf$ alors ajouter la règle $(\mathrm{e} - \mathrm{h}) \rightarrow \mathrm{h}$ à l'ensemble ''R'' sinon enlever h de l'ensemble H.
   c. Générer un nouvel ensemble H d'ensembles de cardinal $card(h) + 1$.
   d. Ré-exécuter les phases 'b' et 'c' jusqu'à avoir $card(h) = card(E)$.

Gen-règles(E; minsup; minconf)

. Entrée : E : ensemble de motifs (fermés) fréquents, minsup; minconf : entier

Début

Pour chaque $e \in E;\ card(e) \geq 2$ **faire**

$m \leftarrow 1.$

$H \leftarrow \{singletons\ de\ e\}.$

Tant que $m < card(e)$ **faire**

Pour chaque $h \in H$ **faire**

$Confiance \leftarrow support(e)/support(e - h)$

Si $confiance > minconf$ **alors**

$R \leftarrow R \cup \{(e - h) \rightarrow h\}$

Sinon $H \leftarrow H \setminus \{h\}.$

Fin pour.

$H \leftarrow \boldsymbol{Apriori - gen(H)}.$

$m \leftarrow m + 1.$

Fin tant que.

Fin pour.

Retourner R

Fin.

Fig.III.3. Code source de l'algorithme Gen-règles.

Les règles d'association sont générées en considérant d'abord tous les 2-motifs, puis les 3-motifs, etc. Les 2-motifs fréquents ont permis de générer les règles d'association de la forme : $a \rightarrow b$ .i.e. un seul élément prémisse et un seul conséquent.

Le 3-motifs abc permet de générer les règles d'association, d'abord avec un conséquent à un seul élément puis des règles avec un conséquent de deux éléments. D'une manière générale un k-motif permet de générer une règle avec un conséquent à seul élément puis 2 puis 3, ..., Jusqu'à k-1 éléments.

Exemple 7 : extraction des règles d'association avec l'algorithme Apriori **:**

Prenons l'ensemble des motifs fréquent générés dans l'exemple 6 :

$$F = \{a, b, c, e, ab, ac, ae, bc, be, ce, abc, abe, ace, bce, abce\}.$$

Le tableau suivant donne toutes les règles d'association générées dont la confiance est supérieur ou égal à minconf= 45% et un support supérieur ou égal à minsup=33%:

Motif	Règle	Confiance	Support	Gardé ?
Ab	$a \rightarrow b$ $b \rightarrow a$	2/3=66.66% 2/5=40.00%	2/6=33,33% 2/6=33,33%	Oui Non
ac	$a \rightarrow c$ $c \rightarrow a$	3/3=100.0% 3/5=60.00%	3/6=50% 3/6=50%	Oui Oui
Ae	$a \rightarrow e$ $e \rightarrow a$	2/3=66.66% 2/5=40.00%	2/6=33,33% 2/6=33,33%	Oui Non
bc	$b \rightarrow c$ $c \rightarrow b$	4/5=80.00% 4/5=80.00%	4/6=66,66% 4/6=66,66%	Oui Oui
be	$b \rightarrow e$ $e \rightarrow b$	5/5=100.0% 5/5=100.0%	5/6=83,33% 5/6=83,33%	Oui Oui
Ce	$c \rightarrow e$ $e \rightarrow c$	4/5=80.00% 4/5=80.00%	4/6=66,66% 4/6=66,66%	Oui Oui
Abc	$ab \rightarrow c$ $ac \rightarrow b$ $bc \rightarrow a$ $a \rightarrow bc$ $b \rightarrow ac$ $c \rightarrow ab$	2/2=100.0% 2/3=66.66% 2/4=50.00% 2/3=66.66% 2/5=40.00% 2/5=40.00%	2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33%	Oui Oui Oui Oui Non Non
Abe	$ab \rightarrow e$ $ae \rightarrow b$ $be \rightarrow a$ $a \rightarrow be$ $b \rightarrow ae$ $e \rightarrow ab$	2/2=100.0% 2/2=100.0% 2/5=40.00% 2/3=66.66% 2/5=40.00% 2/5=40.00%	2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33%	Oui Oui Non Oui Non Non
Ace	$ac \rightarrow e$ $ae \rightarrow c$ $ce \rightarrow a$ $a \rightarrow ce$ $c \rightarrow ae$ $e \rightarrow ac$	2/3=66.66% 2/2=100.0% 2/4=50.00% 2/3=66.66% 2/5=40.00% 2/5=40.00%	2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33%	Oui Oui Oui Oui Non Non
Bce	$bc \rightarrow e$ $be \rightarrow c$	4/4=100.0% 4/5=80.00%	4/6=66,66% 4/6=66,66%	Oui Oui

	$ce \rightarrow b$ $b \rightarrow ce$ $c \rightarrow be$ $e \rightarrow bc$	4/4=100.0% 4/5=80.00% 4/5=80.00% 4/5=80.00%	4/6=66,66% 4/6=66,66% 4/6=66,66% 4/6=66,66%	Oui Oui Oui Oui
Abce	$abc \rightarrow e$ $abe \rightarrow c$ $ace \rightarrow b$ $bce \rightarrow a$ $ab \rightarrow ce$ $ac \rightarrow be$ $ae \rightarrow bc$ $be \rightarrow ac$ $ce \rightarrow ab$ $bc \rightarrow ae$ $a \rightarrow bce$ $b \rightarrow ace$ $c \rightarrow abe$ $e \rightarrow abc$	2/2=100.0% 2/2=100.0% 2/2=100.0% 2/4=50.00% 2/2=100.0% 2/3=66.66% 2/2=100.0% 2/5=40.00% 2/4=50.00% 2/4=50.00% 2/3=66.66% 2/5=40.00% 2/5=40.00% 2/5=40.00%	2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33% 2/6=33,33%	Oui Oui Oui Oui Oui Oui Oui Non Oui Oui Oui Non Non Non

Tableau .III.3. Règles d'association extraites

Dans cet exemple, les règles écrites en rouge ne seront pas sélectionnées. Soit par ce qu'elles ont un support ≤ minsupp ou une confiance ≤ minsonf.

VI. Une vue générale sur notre application :

Notre application consiste à implémenter l'algorithme Apriori. La première tâche de notre application est de construire la matrice termes-documents en passant par les étapes suivantes :

1. L'extraction des concepts biomédicaux en utilisant cxtractor en suivant l'exemple dernier,
2. Créer l'ensemble des documents en parcourant le répertoire qui contient les documents résultants de l'étape 1,
3. Créer l'ensemble des concepts en parcourant tous les documents,
4. Créer une matrice tel que : matrice [0] [y]= l'ensemble des concepts et matrice [x] [0]= l'ensemble des documents,
5. Remplir la matrice tel que :

- Pour chaque document de vecteur matrice [x] [0]

-Pour chaque concept de vecteur matrice [0] [y]

-Si matrice [x] [0] contient matrice [0] [y] alors matrice [x] [y]=1

-Sinon matrice [x] [y]=0

Ces tâches sont réalisées par quatre classes :

- ✓ List_doc : Elle Crée l'ensemble des documents ;
- ✓ List_concepts : Elle Crée l'ensemble des concepts ;
- ✓ Matrice : Elle Rempli la matrice termes-documents ;
- ✓ API : Elle Crée la matrice termes-documents et appel ma méthode de remplissage de la classe Matrice pour remplir la matrice terme-document.

Une fois que la matrice est créée l'application passe à l'exécution de l'algorithme Apriori, qui est implémenté dans la classe Apriori, pour extraire les motifs fréquents en utilisant la matrice termes-documents pour éviter les accès à la collection des documents pour gagner en temps d'exécution. Elle procède comme suit :

1. Initialiser le support minimal,
2. Créer un ensemble des motifs candidats et l initialiser avec l'ensemble des concepts,
3. Pour chaque motif candidat qui contient 'n' concepts {matrice [0] [$concept_1$], matrice [0] [$concept_2$],..., matrice [0] [$concept_n$]} tel que n<= à l'ensemble des concepts, elle parcoure tous les vecteurs : matrice [x] [$concept_1$], matrice [x] [$concept_2$], ..., matrice [x] [$concept_n$]

   -Si matrice [x] [$concept_1$]=1et matrice [x] [$concept_2$]=1et ... et matrice [x] [$concept_n$]=1 alors support= support+1 ;
4. Si le support calculé et supérieur au support minimal alors le motif en question est ajouté à l'ensemble des motifs fréquents,
5. Générer l'ensemble des motifs candidats de cardinalité supérieur en utilisant l'ensemble des motifs fréquents,

L'étape qui vient après la génération des motifs fréquents c'est la génération des règles d'association en utilisant l'ensemble des motifs fréquents qui est l'algorithme Gen-Règles d'Apriori. L'algorithme Gen-Règles est implémenté dans la classe Règles qui assure la génération des règles d'association.

Cette classe manipule des objets nommés association qui ont 4 champs distincts :

- ✓ Prémice de la règle d'association ;
- ✓ Conséquence de la règle d'association ;
- ✓ Sa confiance.
- ✓ Et sont support

VI. Conclusion :

Cette approche nous permettra de toucher à notre objectif qui est d'extraire les règles d'association qui existent dans une collection de documents biomédicaux extraites à partir de la source terminologique MEDLINE. Elle nous servira comme partie conception, dont laquelle on mettra en évidence la démarche à suivre pour extraire les règles d'association à partir des documents textuels, pour réaliser notre application.

Chapitre IV : Réalisation de l'application et teste.

I. Introduction :

Les documents de la collection d'articles de MEDLINE qu'on utilise dans notre cas ont un format (.text). On les converti en format TREC dans l'unique objectif d'utiliser un logiciel pour extraire les termes biomédicaux, ce logiciel se nome Cxtractor. Il réalise les tâches la première étape de notre approche i.e. La tokenization (segmentation), l'élimination des mots vides et filtrage de textes et la lemmatisation ou La racinisation (troncature).

Une fois l'extraction est achevée, les termes extraits serviront de documents source pour notre application. Elle est programmé en java en utilisant le logiciel éclipse, pour généré les règles d'association.

Dans ce chapitre nous allons présenter notre application, notre plate forme de développement et les outils utilisés pour mener à bien la réalisation de notre application ainsi que, quelques interfaces de l'application.

II. Le langage Java:

On peut faire remonter la naissance de Java à 1991. À cette époque, des ingénieurs de chez SUN ont cherché à concevoir un langage applicable à de petits appareils électriques (on parle de code embarqué). Pour ce faire, ils se sont fondés sur une syntaxe très proche de celle de C++, en reprenant le concept de machine virtuelle déjà exploité auparavant par le Pascal UCSD.

L'idée consistait à traduire d'abord un programme source, non pas directement en langage machine, mais dans un pseudo langage universel, disposant des fonctionnalités communes à toutes les machines. Ce code intermédiaire, dont on dit qu'il est formé de byte-codes, se trouve ainsi compact et portable sur n'importe quelle machine ; il suffit simplement que cette dernière dispose d'un programme approprié (on parle alors de machine virtuelle) permettant de l'interpréter dans le langage de la machine concernée.

En fait, ce projet de langage pour code embarqué n'a pas abouti en tant que tel. Mais ces concepts ont été repris en 1995 dans la réalisation du logiciel HotJava, un navigateur Web écrit par SUN en Java, et capable d'exécuter des applets écrits précisément en byte codes.

Les autres navigateurs Web ont suivi, ce qui a contribué à l'essor du langage qui a beaucoup évolué depuis cette date, sous forme de versions successives : 1.01 et 1.02 en 1996, 1.1 en 98 et 1.2 (finalement rebaptisée Java 2) en 1999, 1.3 en 2000, 1.4 en 2002, 5.0 en 2004 (toujours appelées Java 2). Ainsi parle-t-on du J2SE 1.4 (Java 2 Standard Edition 1.4) basée

sur le JDK 1.4 (Java Development Kit 1.4), plus récemment du J2SE5.0 (JDK 5.0) ou encore de Java 5. En revanche, la dernière version s'intitule JSE 6 (le 2 a disparu !), ou plus simplement Java 61.

On notera que, au fil des différentes versions, les aspects fondamentaux du langage ont peu changé (ils ont quand même été complétés de façon substancielle par Java 5, notamment par l'introduction de la programmation générique et du remaniement des "collections"). En revanche, les bibliothèques standards (API) ont beaucoup évolué, à la fois par des modifications et par des ajouts. Il en va d'ailleurs de même des ensembles de spécifications accompagnant chaque version standard de Java (J2EE jusqu'à la version 4 et JEE depuis la version 5). [Claude DELANNOY]

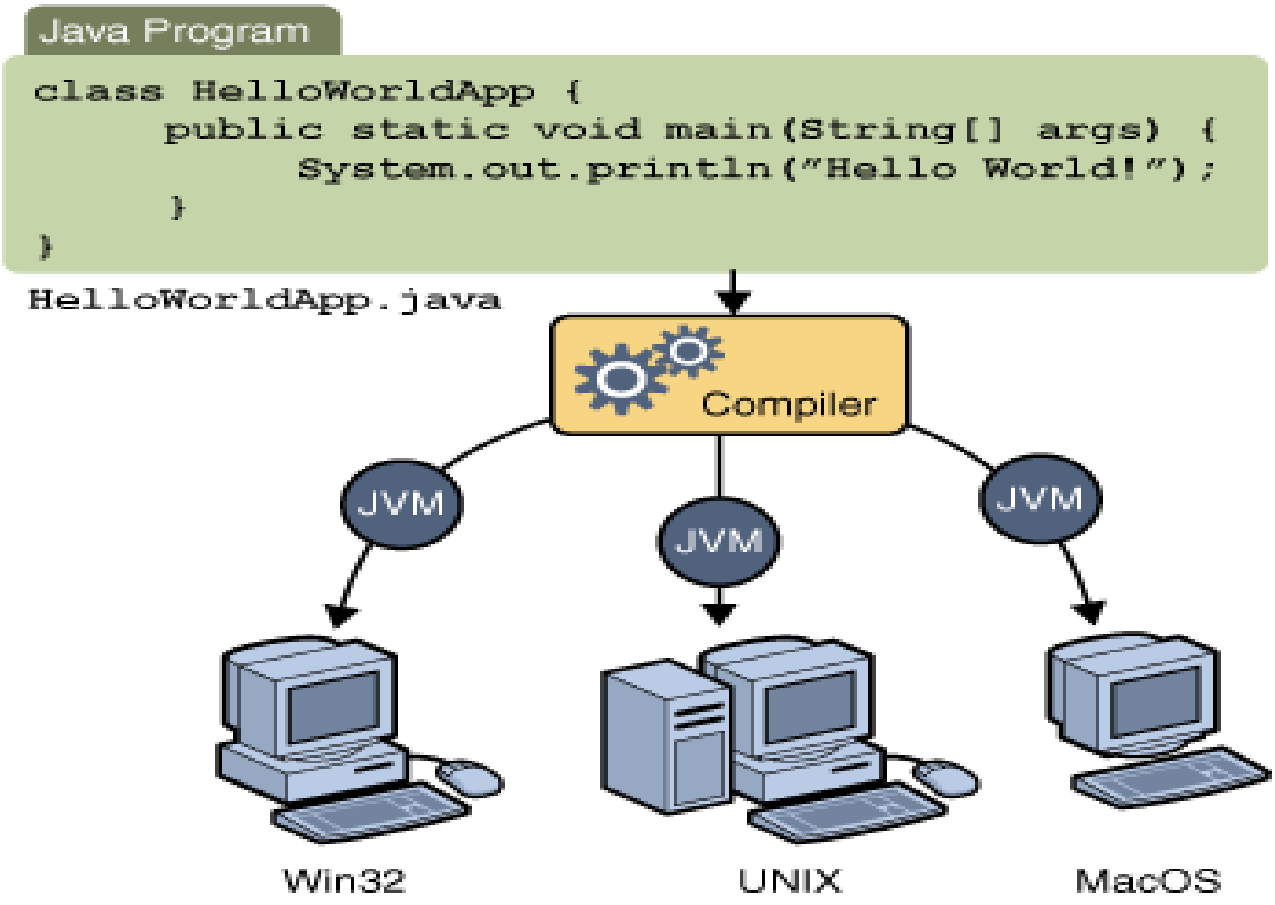


Fig.IV.1. Indépendance d'un programme en java de toute plate forme.

II.1. Java et la portabilité

Dans la plupart des langages, on dit qu'un programme est portable car un même code source peut être exploité dans des environnements différents moyennant simplement une nouvelle compilation.

En Java, la portabilité va plus loin. En effet, comme nous l'avons évoqué précédemment, la compilation d'un code source produit, non pas des instructions machine, mais un code intermédiaire formé de byte code. D'une part, ce code est exactement le même, quel que soit le compilateur et l'environnement concernés. D'autre part, ces byte codes sont exécutables dans toute implémentation disposant du logiciel d'interprétation nommé machine virtuelle1 ou, parfois, système d'exécution Java.

De surcroît, Java définit exactement les caractéristiques des types primitifs servant à représenter les caractères, les entiers et les flottants. Cela concerne non seulement la taille de l'emplacement mémoire, mais aussi le comportement arithmétique correspondant. Ainsi, quelle que soit la machine, une valeur de type float (réel) aura exactement même taille, mêmes limites et même précision. Java est ainsi le premier langage qui assure qu'un même programme, exécuté dans des environnements différents, fournira les mêmes résultats. [Claude DELANNOY]

II.2. Java et la programmation orientée objet

La P.O.O. (programmation orientée objet) possède de nombreuses vertus universellement reconnues désormais. Notamment, elle ne renie pas la programmation structurée (elle se fond sur elle), elle contribue à la fiabilité des logiciels et elle facilite la réutilisation de code existant. Elle introduit de nouveaux concepts, en particulier ceux d'objets, d'encapsulation, de classe et d'héritage. [Claude DELANNOY]

II.3. Java et la programmation événementielle

Actuellement, on peut distinguer deux grandes catégories de programmes, en se fondant sur leur interface avec l'utilisateur, c'est-à-dire sur la manière dont se font les échanges d'informations entre l'utilisateur et le programme :

• les programmes à interface console;

• les programmes à interface graphique.

II.3.1. Les programmes à interface console (ou en ligne de commande)

Historiquement, ce sont les plus anciens. Dans de tels programmes, on fournit des informations à l'écran sous forme de lignes de texte s'affichant séquentiellement, c'est-à-dire les unes à la suite des autres. Pour fournir des informations au programme, l'utilisateur tape des caractères au clavier (généralement un "écho" apparaît à l'écran).

Avec une interface console, c'est le programme qui décide de l'enchaînement des opérations : l'utilisateur est sollicité au moment voulu pour fournir les informations demandées. [Claude DELANNOY]

II.3.2. Les programmes à interface graphique (G.U.I.)

Dans ces programmes, la communication avec l'utilisateur se fait par l'intermédiaire de composants tels que les menus déroulants, les menus surgissant, les barres d'outils ou les boîtes de dialogue, ces dernières pouvant renfermer des composants aussi variés que les boutons poussoirs, les cases à cocher, les boutons radio, les boîtes de saisie, les listes déroulantes...

L'utilisateur a l'impression de piloter le programme, qui semble répondre à n'importe quelles demandes. D'ailleurs, on parle souvent dans ce cas de programmation événementielle, expression qui traduit bien le fait que le programme réagit à des évènements provoqués (pour la plupart) par l'utilisateur. [Claude DELANNOY]

III. Outils de développement :

Pour réaliser notre application, nous avons opté pour l'utilisation de la plate forme Windows avec son système d'exploitation Windows 7, le logiciel Java éclipse et cxtractor.

III.1. Eclipse :

Eclipse IDE est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. [Wikipedia]

III.1.1. Lancement et « workspace » :

Lors du premier lancement d'Eclipse, l'environnement vous demande de spécifier un « workspace » (voir Figure suivante). Les workspaces (espaces de travail) d'Eclipse sont des répertoires dans lesquels sont regroupés les projets de développement. Ainsi, il est possible d'avoir plusieurs espaces de travail, dédiés à des langages de programmation différents, des buts différents, ou par exemple des enseignements différents. [Stéphane Huot]

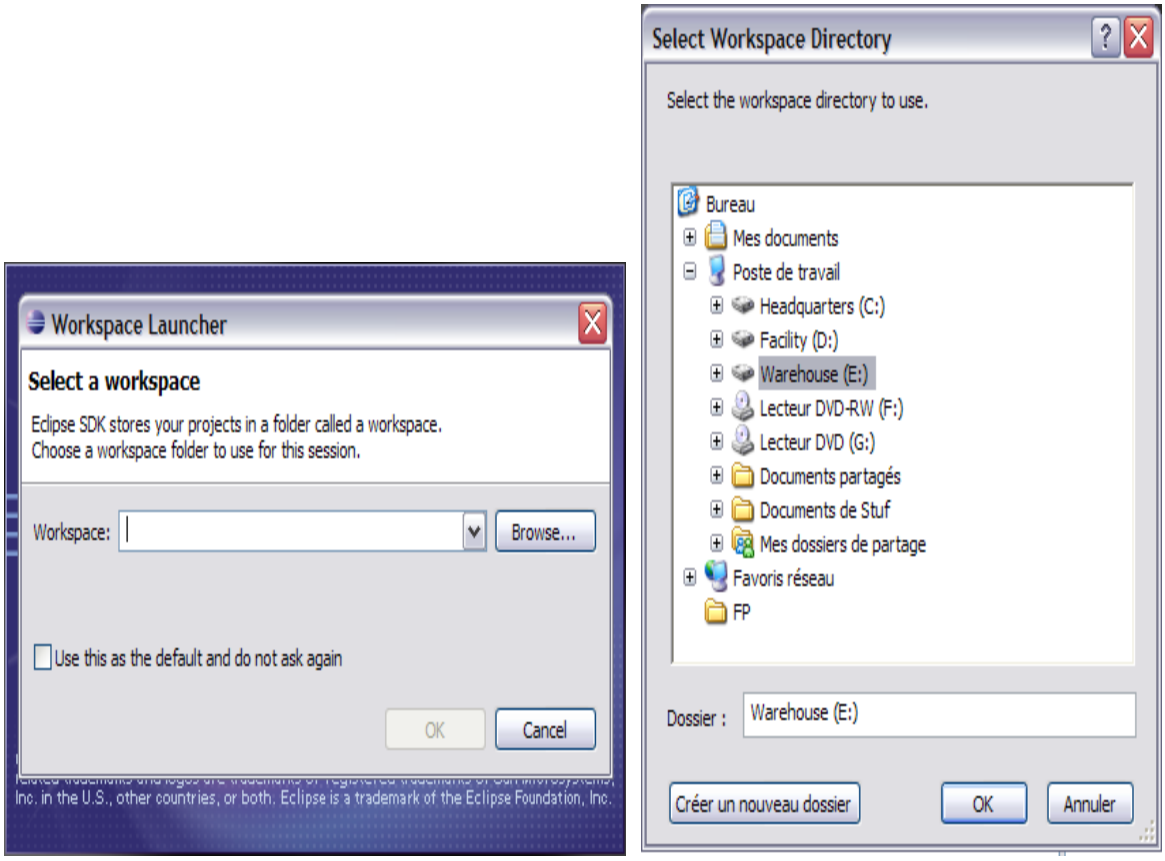


Fig.IV.2. Dialogue de sélection et de création d'un workspace.

III.1.2. Projets

Au sein des workspaces, Eclipse regroupe le code en **projets**. Un projet peut regrouper le code d'une application, d'une librairie, d'un module d'une application, etc. [Stéphane Huot]

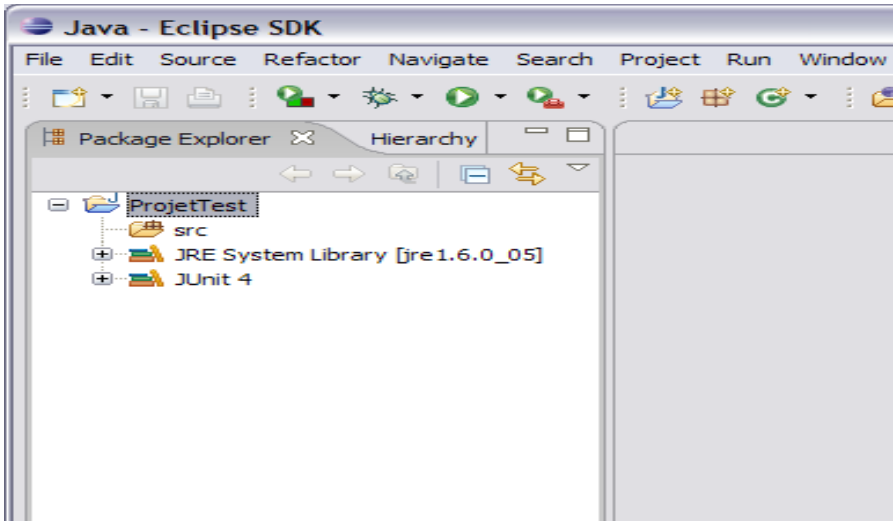


Fig.IV.3. Exemple de projet sous eclipse.

III.1.3. Compilation, tests et exécution

Un des avantages d'Eclipse est de fournir un environnement intégré qui facilite la création, la compilation, les tests et l'exécution de projets java pouvant contenir de nombreux fichiers de code, repartis dans différents paquetages et référençant de nombreuses librairies (cas complexes qui nécessite des lignes de commandes longues et complexes ou des fichiers « batchs »). [Stéphane Huot]

III.1.3.1. Compilation :

Par défaut, lorsqu'un projet est créé, Eclipse active l'option de « compilation automatique ». Ainsi, le projet est compilé en temps réel, dès que du code est tapé et sauvegardé ou que l'exécution est lancée. Eclipse fournit en plus de nombreux mécanismes pour présenter les erreurs de compilation. Par exemple signalant dans la fenêtre de droite de (Figure suivante) des croix rouges sur des fichiers, signifiant que ces derniers contiennent des erreurs de compilation. Ces erreurs sont mises en valeur et même expliquées dans l'éditeur de code. [Stéphane Huot]

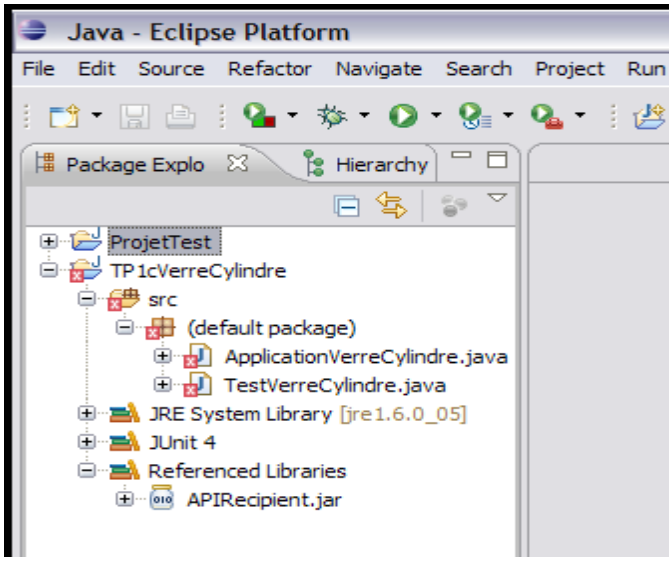


Fig.IV.4. Exemple de signalisation des erreurs.

III.1.3.2. Tests:

Eclipse permet de lancer interactivement des tests JUnit écrits par l'utilisateur afin d'effectuer les tests unitaires des classes des projets utilisateur. [Stéphane Huot]

III.1.3.3. Exécution:

Enfin, pour lancer l'exécution d'un programme écrit, il suffit de faire un clic droit dans l'onglet « Package Explorer » sur la classe contenant la méthode « Main » de programme et de choisir « Run as/Java Application » ou de cliquer sur le bouton dans la barre d'outils, après avoir sélectionné la classe contenant la méthode « Main ». [Stéphane Huot]

IV. Le logiciel cxtractor :

Cxtractor est un projet Open Source développé en Java visant à intégrer dans la plateforme BioSIR des algorithmes d'extraction de concepts. Les algorithmes d'extraction de Cxtractor peuvent être exécutés en ligne de commande. Ces algorithmes implémentent plusieurs méthodes d'extraction de concepts à partir des documents biomédicaux, à savoir les méthodes d'extraction basées sur un modèle de RI (e.g., TF_IDF, BM25, etc.) ainsi que la mesure statistique de Spearman permettant de modéliser la corrélation entre chaque concept candidat et un texte donné grâce aux positions des mots communs entre eux.

V.1. Plateforme BioSIR :

BioSIR (**Bio**medical **S**emantic **I**nformation **R**etrieval) plateforme permet d'extraire des concepts biomédicaux qui sont définis dans les terminologies (e.g., MeSH, SNOMED, GO ou UMLS) à partir des textes biomédicaux, d'indexer les documents biomédicaux avec les concepts biomédicaux qui représentent les sujets sémantiques du document et/ou de la requête de l'utilisateur, de sélectionner de l'information biomédicale pertinente en réponse à une ou plusieurs requêtes de l'utilisateur ainsi que d'évaluer les performances des modèles de RI biomédicale. [Ba-Duy DINH]

V.2. Les entrés/Sorties de logiciel Cxtractor :

V.2.1. Les entrés :

Les documents d'entrée de logiciel cxtractor peuvent avoir un des formats suivants : .txt, .html, ou les formats de TREC (c'est ce format qui sera utilisé dans notre cas). Le choix de format se fait en reconfigurant le contenu de fichier « configuration/config/settings.properties.sample ». Lors de l'exécution, les paramètres de configuration sont chargés automatiquement. Un document TREC contient des balises particulières, par exemple :

- ✓ **DOC** représente le document,
- ✓ **DOCNO** représente l'identifiant unique du document,
- ✓ **TITLE** correspond au titre du document,
- ✓ **ABSTRACT** correspond au résumé du document.

Exemple de document TREC:

<DOC>

<DOCNO> 11567884</DOCNO>

<ABSTRACT>

We studied the effect of pilocarpine hydrochloride, a parasympathicomimetic agent, on major salivary gland output and subjective responses in 40 patients with salivary hypofunction. Pilocarpine increased salivary output or gave significant symptomatic relief in 21 of the 40 patients. The women fared better than the men. Side effects were uncommon, were generally mild, and caused no treatment interruption. These results indicate that pilocarpine is effective in relieving the signs and symptoms of postradiation xerostomia.

</ABSTRACT>

</DOC>

Fig.IV.5. Exemple de document TREC.

V.2.2. Les sorties :

Les documents de sortie de logiciel cxtractor ont le format .ker. Un document de format .ker contient :

- ✓ Le nom de document traité précédé par la balise <DOCNO> comme entête;
- ✓ Chaque ligne contient un concept biomédical extrait en spécifiant :
  - ➢ Son ordre dans le document .ker ;
  - ➢ Son son rang dans l'arborescence de MeSH ;
  - ➢ Le nom de concept ;
  - ➢ Son poids ou sont score calculé en se basant sur sont degré de pertinence dans le document .TREC ;

Exemple :

<DOCNO> DOC1

Ordre | CUI | Nom de concept | le score de concept |

Ordre | CUI | Nom de concept | le score de concept |

.....

Ordre | CUI | Nom de concept | le score de concept |

<DOCNO> DOC2

Ordre | CUI | Nom de concept | le score de concept |

Ordre | CUI | Nom de concept | le score de concept |

.....

Fig.IV.6. Exemple de document Ker.

VI. Exemple d'extraction de concepts biomédicaux en utilisant cxtractor :

Pour extraire les concepts biomédicaux de document TREK de l'exemple de document TREK dernier en utilisant cxtractor il faut suivre la procédure suivante :

- ✓ mettre le document dans le répertoire ...\cxtractor-1.0.3\cxtractor\tests\trec.
- ✓ lancer l'invite de commande et saisir les requêtes suivantes :
  - `C:> cd ...\cxtractor-1.0.3\cxtractor ;`
  - `C:\...\ cxtractor-1.0.3\cxtractor> java -jar cxtractor-1.0.3.jar -r -c -d tests/trec/nom_de_document_TREK ;`

Après l'exécution de ses commandes, on aura comme sortie dans le répertoire ...\cxtractor-1.0.3\cxtractor\tests\trec. un fichier de format .Ker suivant :

```
<DOCNO>  11567884
0|C0043352|Xerostomia|17,7404|C07.465.815.929
1|C0031923|Pilocarpine|17,1976|D03.132.672
2|C0043210|Women|16,0868|
3|C0036098|Salivary Glands|15,3924|
4|C0030705|Patients|10,4221|
5|C0025266|Men|9,5162|
6|C0039796|Therapeutics|8,0145|
```

Fig.IV.7. Exemple d'extraction de concepts biomédicaux en utilisant cxtractor.

Et enfin, les règles d'association extraites sont visualisées en utilisant une interface graphique (voire la figure suivante) qui met en valeur toutes les relations extraites. Cette interface graphique est programmée dans la JFrame ''Main''.

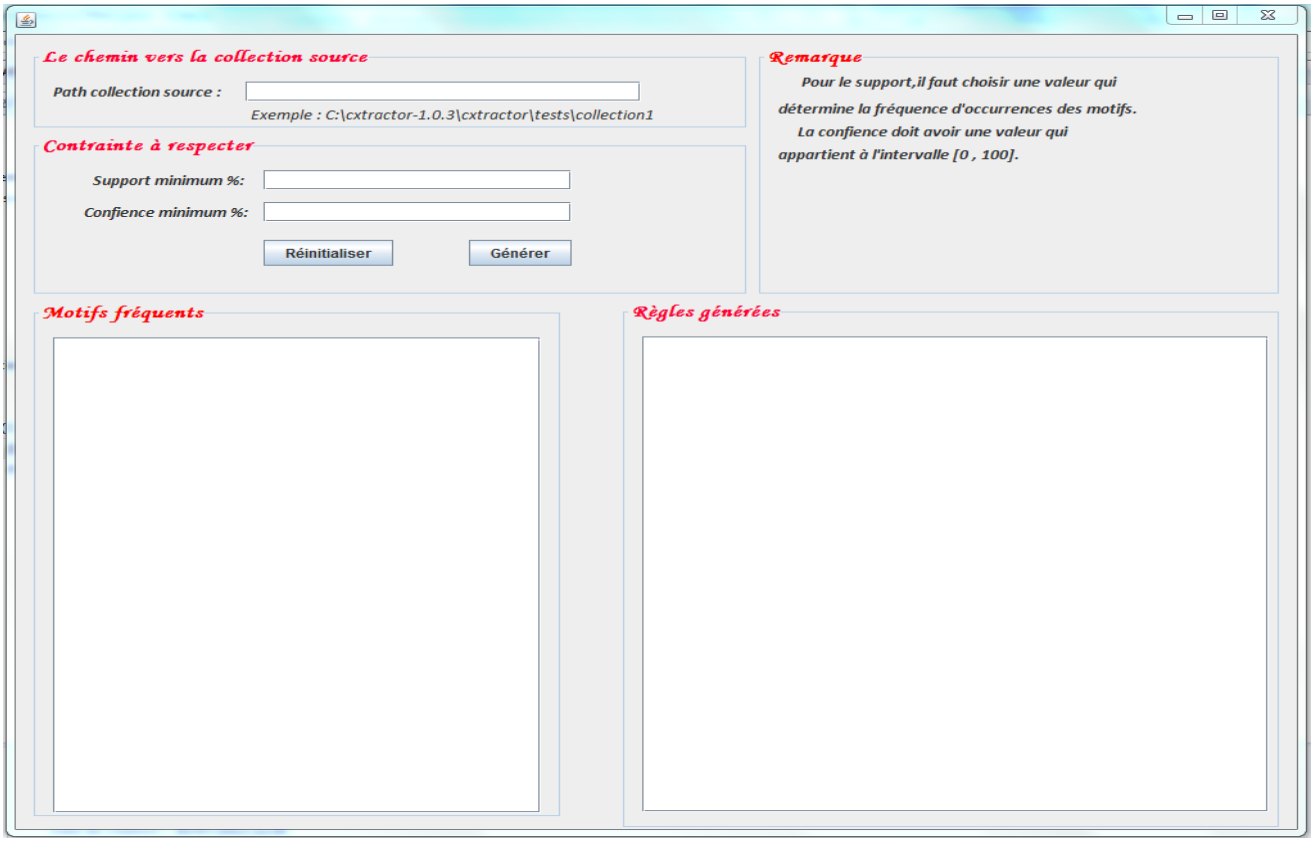


Fig.IV.8. Interface graphique de notre application.

Notre interface graphique est composée de 5 panels :

- Le panel '' Le chemin vers la collection source'' : ce panel contient le champ de saisit de lien vers le dossier à tester. Le dossier source à tester doit contenir des documents enregistrés sous le format (.ker) sinon ça ne marche pas. Si le dossier source n'existe pas un message sera affiché au dessous de ce champ de saisie ;
- Le panel ''Contrainte à respecter'' : il contient deux champs de saisie, le premier est le support minimal des motifs et le deuxième est la confiance minimal des règles d'associations. Ainsi que deux boutons : un pour lancer la génération des règles d'association et l'autre pour réinitialiser les champs de saisie;
- Le panel ''Remarque'' : donne des indications sur les intervalles des valeurs qui peuvent être saisies dans les champs support minimal et confiance minimal.
- Le panel ''Motifs Fréquents'' : Permet d'afficher les motifs fréquent extraits. Chaque motif est affiché sous la forme suivante :

  [Motif], support = xxx.xx%
- Le panel '' Règles générées'' : Permet d'afficher les règles d'association extraites. Chaque règle extraite est affichée de la manière suivante :

  [prémice] ==> [conséquence], confiance= xxx.xx%, support= xxx.xx%

Pour assurer la portabilité de notre application, on l'a exportée sous format .jar. Ce format permet d'exécuter l'application, indépendamment de l'environnement de développement eclipse, dans toutes plateformes contenant un JDK 7 et une JRE.

VII. Test de notre application :

Pour faciliter la vérification de bon fonctionnement de l'application on a créé une collection qui ressemble à la collection étudiée dans l'exemple 5 donné dans le chapitre III (page 41). Cette collection contient six documents qui contiennent à leur tour les motifs a, b, c, d, e en suivant la fréquence d'apparaissions exprimée dans la matrice de l'exemple. Grace à cette collection notre application va créer la même matrice qui lui servira pour générer les motifs fréquents et les règles d'association.

Dans l'exemple 5 (chapitre III) étudié on a trouvé 15 motifs fréquents en prenant 33% comme support, et dans l'exemple 6 (chapitre III) étudié on a trouvé 37 règles d'associations valides en prenant 33% comme support minimal et 45% comme confiance minimal;

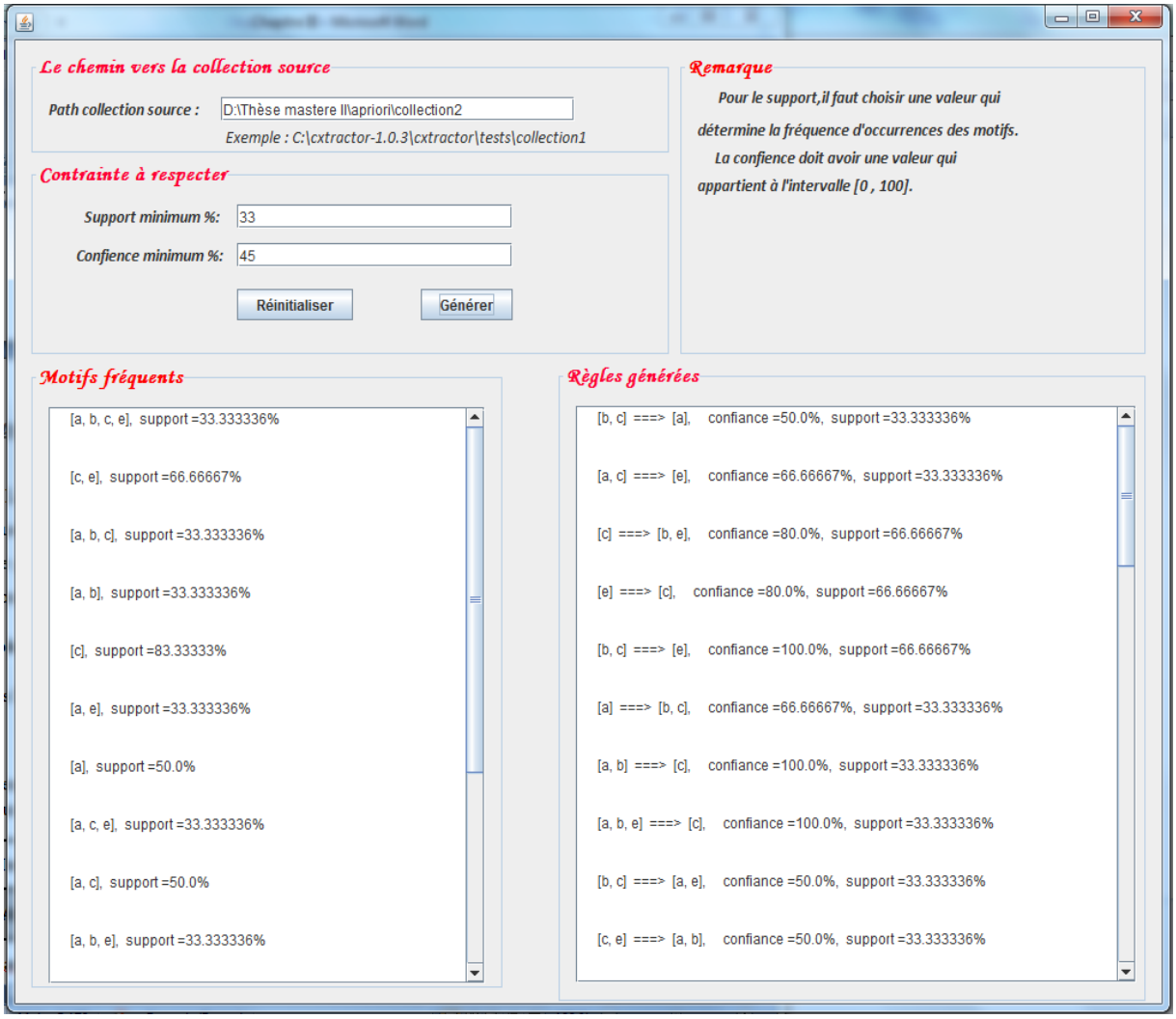


Fig.IV.9. Exemple d'exécution de notre application.

Après l'exécution de notre application avec un support minimal égal à 33% et une confiance minimal égale à 45% on retrouve les mêmes 15 motifs et les même 37 règles d'association de l'exemple étudié.

Conclusion :

Notre application vise à implémenter l'algorithme d'extraction de règles d'association Apriori en utilisant le langage JAVA. Les deux logiciels que nous avons définis dans ce chapitre et qui nous ont servi pour programmer notre application sont:

- Cxtractor qui est un extracteur de concepts. Ce logiciel nous a permis de filtrer notre collection de sorte qu'à ne garder que les concepts biomédicaux.
- **Eclipse IDE** nous a servi comme environnement de développement.

On a parlé aussi brièvement sur le domaine biomédical, de sa littérature, de Thésaurus MeSH qui regroupe tous les concepts biomédicaux et de méta thésaurus UMLS. Et enfin, nous avons donné une présentation générale sur le fonctionnement de notre application et un exemple de son fonctionnement.

Conclusion Générale

Conclusion Générale :

Suite à une explosion vertigineuse de la quantité d'informations stockées dans les collections de texte, il est nécessaire de recourir à des techniques et à des outils d'exploitation automatique de ces textes, afin d'extraire l'information renfermée dans ces collections. Les techniques du Text Mining ont pour objectif de satisfaire cette nécessité. L'algorithme utilisé dans notre approche (recherche des de règles d'associations) réalisent plusieurs passes sur la collection de document pour la recherche des motifs fréquents et le calcule de leurs support. Une fois que, tous les motifs fréquents sont trouvé et que leurs supports sont déterminé l'algorithme de notre approche (apriori) passe au calcule des règles d'association entre les éléments de l'ensemble des motifs fréquents.

Ce mémoire à pour objectif principal de proposé une approche pour appliqué un algorithme d'extraction de règles d'association sur des données textuels. Ainsi nous avons commencé ce travail (premier chapitre) par introduire le Text Mining, ses intérêts et ses différents objectifs.

Les notions de bases des règles d'association en Text Mining sont étudiées dans le deuxième chapitre en introduisant quelques concepts utilisé par les principaux algorithmes de découvertes des règles associatives, parmi les quels on a choisit « Apriori », qui est étudié dans le troisième chapitre suivie par une illustration de sont fonctionnement.

Le quatrième chapitre est consacré à l'implantation de l'algorithme Apriori en langage JAVA. Cette étape est précédée par l'extraction des termes biomédicaux à partir d'une collection de document extraite à partir de MEDLINE. Cette extraction repose sur l'utilisation de logiciel Cxtractor qui utilise le thésaurus MeSH pour l'extraction des termes biomédicaux.

Perspectives et travaux futurs :

Nous avons étudié les règles d'associations et un algorithme d'extraction de règles d'associations ''Apriori''. Nous avons met en évidence le principe de cet algorithme en illustrant sont fonctionnement avec des exemples. Nous avons aussi proposé une approche, qui consiste à structurer des documents textuels biomédicaux pour y appliquer l'algorithme Apriori afin d'extraire les règles d'association. Il est intéressant de faire des recherches pour :

- ✓ Trouver le nombre de règles d'association qu'un motif est susceptible de générer et proposer une formule mathématique plus générale;
- ✓ Mettre évidence les avantages et les inconvénients de l'algorithme Apriori;
- ✓ Dans notre cas, beaucoup de règles d'association générer sont redondantes. Donc on peut enrichir notre approche en proposant une solution pour le problème de règles d'association redondantes ;
- ✓ Notre application est couteuse en temps d'exécution et en espace mémoire, il est intéressant de proposé une optimisation des calculs et une minimisation d'espace mémoire.
- ✓ En fin, étendre notre approche en étudiant les règles maximales pour extraire des règles avec une confiance moins élevé mais qui peuvent renfermer des informations intéressantes.

Bibliographie

Bibliographies:

[Agrawal et al., 93] Rakesh Agrawal, Tomasz Imielinski And Arun Swami : Mining Association Rules Between Sets Of Items In Large Databases. In Proc. Of The ACM SIGMOD International Conference Management Of Data, pp. 207-216, Washington, D.C., 1993

[Apt 1998]: C .Apté."Data mining:An Industrial Research Perspective"Dans IEEE Computational Science and Engineering,April-June,pp 6-9,1997.

[Bastide, 2000] Y. Bastide. Data mining : algorithmes par niveau, techniques d'implantation et applications. Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II, 2000.

[Ba-Duy DINH, 2012] Accès à l'information biomédicale : vers une approche d'indexation de recherche d'information conceptuelle basée sur la fusion de ressources termino-ontologiques.2012

[Claude DELANNOY] Programmer en Java.

[Delafosse, 1999] : pdf : TALN Informatique, Brigitte Bigi,CLIPS - Equipe GEOD,ITC - Avril 2006.

[Den 2000].L.Denoue . « cours de classification supervisée de documents ».Unnivesité de Savoie,PH.Erikson,Les rituels du dialogue,Editions de l'Harmattan 2000.

[Fayyad et al., 1995] : pdf de titre : Découverte de connaissances à partir de données

[Feldman et al., 98] : R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler Et O. Zamir. Text Mining At The Term Level. Dans Proc. Of The 2nd Eur. Symp. On Principles Of Data Mining And Knowledge Discovery (PKDD'98), J. M. Zytkow Et M. Quafafou Editors, Volume 1510. Lecture Notes In Artificial Intelligence – LNAI, Pages 65–73, Nantes, 1998.

[Feldman et Dagan, 1995] R. Feldman et I. Dagan. Knowledge Discovery in Textual Databases (KDT). Dans Proc. of the 1st Int'l Conf. on Data Mining and Knowledge Discovery, rédacteurs U. M. Fayyad et R. Uthurusamy, pages 112–117, Montréal, Canada, 1995. AAAI Press.

[F.SOUAM, 2007] :''Découverte d'associations par expansion de Trasaction à partir de la Hiérarchie de concepts d'un document textuel

[Georges El Helou et Charbel Abou khalil] : Data Mining ''*Techniques d'extraction des connaissances''*, 16 février 2004.

[H. CHERFI, 2005], Etude et réalisation des d'un système d'extraction de connaissance à partir de textes, 2006.

[Hand et al. 2001] : :pdf de titre : Découverte de connaissances à partir de données, (Datamining ou KDD), R. Duda, P. Hart, D. Storck (2000). Pattern Classification. Wiley.

[Mac 1967] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. 1967.

[Mitc 1997]:T.M Mtchell."Machine learning".dans computer Science. McGrow_hill, New_ York.1997.

[Pasquier, 2000]. N. Pasquier. -Extraction de bases pour les règles d'association à partir des itemsets fermées fréquents. In : Proc. INFORSID'2000, pp. 56-77, 2000.

[PBTL98]. Pasquier (N.), Bastide (Y.), Taouil (R.) et Lakhal (L.). - Pruning closed itemset lattices for association rules. Actes des 14èmes journées Bases de Données Avancées BDA'98, 1998, pp. 177-196.

[Pasquier et al., 1999] N. Pasquier, Y. Bastide, R. Taouil et L. Lakhal. Efficient mining of association rules using closed itemset lattices. Information Systems, 24(1) :25–46, 1999.

[Stéphane Huot 2008]. Stéphane Huot. –Cour Java. Introduction à Eclipse, 2008.]

[TZE 1993]:K.Tzeras et S.Hartmann"Automatic indexing based on Bayesian inference net works"dans R.Korf.hage,E.Rasmusen and P.willeth,editors proce.dugs of SFGIR-93,16th

[Vap 1995]:V.Vapnik. « The nature of satistical learninig Theoriy » .Springer,New york, 1995.Berkeley, University of California Press. Volume 1, pp 281-297.

Les sites:

[Wikipedia] http://fr.wikipedia.org/wiki/Eclipse_%28projet%29#Eclipse_Platform

http://www.nlm.nih.gov/mesh/