

Projet d'intelligence artificielle :

Extraction de règles d'association : algorithme APriori

MERJA

DARNET Camille
VOURIOT Thierry
DESS Informatique

Java Environment for Mining Association Rules



sous la direction de Mr Jerzy KORCZAK : jjk@dpt-info.u-strasbg.fr

Sommaire :

Introduction.....	3
Problématique	3
Formalisation du problème	4
<i>Concepts</i>	4
<i>Décomposition du problème</i>	4
Recherche des items fréquents	5
<i>Algorithme APriori</i>	5
<i>Générations des candidats</i>	6
<i>Calcul des supports pour une transaction</i>	6
<i>Limites d'APriori</i>	6
Recherche des règles d'association	7
<i>Génération des règles</i>	7
Choix Algorithmique	8
<i>Discrétisation des données</i>	8
<i>Structure de données</i>	8
<i>Exploration de l'arbre</i>	10
Outils et environnement de programmation	11
Formats des fichiers	12
<i>Arff</i>	12
<i>Csv</i>	13
Manuel utilisateur	14
Conclusion.....	19

1. Introduction

Avec l'évolution des technologies, des quantités énormes de données sont recueillies dans diverses domaines : données sur les clients, numérisation de textes, images, vidéo, voix, catalogue en ligne...

Ces données sont en trop grandes quantités pour être traitées manuellement. Le nombre d'enregistrements se compte par million ou par milliard, les données ont aussi des dimensions importantes (trop de champs/attributs/caractéristiques).

Les nécessités économiques comme le e-commerce, le haut degré de concurrence, la personnalisation, la fidélisation de la clientèle a amené le monde informatique a développé des méthodes de fouille de donnée non supervisée.

Cela consiste à déterminer les valeurs associées parmi les données, par exemple l'analyse du panier de la ménagère, c'est-à-dire déterminer les articles associés dans les tickets de caisse :

Si un client achète du poisson et du citron, il achète aussi du vin blanc.

Si un client achète une télévision, il achètera un magnétoscope dans un an.

Ce type de recherche d'associations peut être utilisé par le monde de la grande distribution pour anticiper les comportements d'achats, par les services bancaires pour le diagnostic de crédit, mais aussi dans le domaine médical, par exemple les complications induites par des associations de médicaments ...

2. Problématique

A partir de base de données contenant les informations des différentes transactions, il faut un procédé qui permet d'extraire des règles d'association. C'est-à-dire trouver des associations entre les différents produits, par exemple : 98% des clients qui s'équipent en pneus et en accessoires pour automobiles se préoccupent également des services offerts pour leurs autos.

3. Formalisation du problème

Définition des concepts

Un *item* est un élément de la base de données.

Une *transaction* est constituée de plusieurs items.

Un *support* d'un ensemble d'items est le nombre de transactions qui contiennent cet ensemble.

Un ensemble d'items est *fréquent* si son support est supérieur au support minimum.

La *confiance* d'une règle de la forme $I_2 \Rightarrow I_1 - I_2$ avec I_1 et I_2 deux ensemble d'items et I_2 inclus dans I_1 se définit par le ratio $\text{support}(I_1) / \text{support}(I_2)$.

Une règle est *valide* si sa confiance est supérieure à la confiance minimum.

Décomposition du problème

La résolution du problème se découpe en deux parties :

- Recherche des items fréquents :

Trouver tous les ensembles d'items fréquents telles que leur représentation soit significative au sein de la base. Certaines règles d'association ne sont pas intéressantes car elles ne concernent qu'une faible part des objets.

- Recherche des règles d'association :

Une fois les items fréquents trouvés, les règles d'association peuvent être déduites. Les règles d'association doivent vérifier une proportion importante d'objets pour exprimer une association cohérente.

L'extraction ne va concerner que les règles d'association (valides) dont le support est supérieur ou égal au seuil (car les règles utiles concernent une population "importante") et dont la confiance est supérieure ou égale au seuil de confiance minimum (car on ne veut que les règles statistiquement significatives). Les seuils sont définis par l'utilisateur selon la nature des données analysées.

4. Recherche des items fréquents

But : Trouver les ensembles d'items fréquents : ceux qui ont un support supérieur à un seuil minimum. La recherche de ces ensembles d'items va se faire par l'algorithme APriori défini par Agrawal en 1993.

Algorithme APriori :

L'algorithme APriori est un algorithme itératif de recherche d'items fréquents.

Principe :

- Première passe :
 - recherche des 1-itemsets fréquents
 - un compteur par produits
- L'algorithme génère un candidat de taille k à partir de deux candidats de taille k-1 différents par le dernier élément
 - Génération des candidats
- Passe k :
 - comptage des k-itemsets fréquents candidats
 - sélection des bons candidats

L'étape la plus importante dans cet algorithme est la génération des candidats, pour réduire le nombre de sous ensemble créé à chaque itération voici deux règles qui permettent de ne pas générer les ensembles d'items qui ne seront pas valides :

- Règle 1 : *Si un ensemble d'items n'est pas fréquent alors tous ses sur ensembles ne sont pas fréquents.*
Si un ensemble n'est pas fréquent alors il est inutile de chercher des sur ensembles à partir de cet ensemble.
- Règle 2 : *Un ensemble d'items peut être fréquent si tous ses sous ensembles sont déjà fréquents.*
Lors de la création d'un ensemble d'items de taille k, cet ensemble sera créé si tous ces sous-ensembles sont déjà fréquents.

La phase de test de validité de chaque sous-ensemble se réduit par un parcours de la base de transaction qui permet l'énumération des ensembles candidats dans la base. Un ensemble candidat devient valide si son support est supérieur au support minimum.

Remarque :

La taille des ensembles d'items fréquents sera au plus la taille de la plus longue transaction.

Si lors de la génération il n'y a plus de candidats qui apparaissent ou que aucun candidat n'est valide alors on peut stopper le déroulement de l'algorithme

Génération des candidats

Les ensembles d'items candidats d'un niveau k sont générés à partir des ensembles d'items fréquents de niveau $k-1$. Cette génération consiste à faire une jointure en tous les items des ensembles fréquents de niveau $k-1$ en n'introduisant pas de redondance.

Avant que chaque ensemble d'items candidat soit réellement désigné comme candidat potentiel pour être ensemble d'items fréquents, il faut tester si tous les sous ensembles sont des ensembles d'items fréquents, ce test provient de la règle n°2.

Exemple

Soit quatre ensembles d'items fréquents de niveau 3
 $\{1\ 2\ 3\}$, $\{1\ 2\ 4\}$, $\{1\ 3\ 5\}$, $\{2\ 3\ 4\}$

Génération des candidats :

$\{1\ 2\ 3\ 4\}$, $\{1\ 3\ 4\ 5\}$

Suppression des candidats

Suppression de $\{1,3,4,5\}$ car $\{1,4,5\}$ n'est pas un ensemble d'items fréquents

Calcul des supports pour une transaction

Pour tous les ensembles d'items candidats on vérifie si ils appartiennent alors à des transactions. Ici le problème est posé différemment, on recherche l'existence des parties de la transaction comme des ensembles. Ce procédé de recherche sera développé dans le chapitre Exploration.

Limite d'APriori :

L'algorithme pêche surtout lors de la génération des candidats. Il génère beaucoup de candidats : 10^4 1-itemsets fréquent générant 10^7 2-itemsets candidats. Pour trouver les 100-itemsets on doit générer $2^{100} \approx 10^{30}$ candidats. Plusieurs "scans" de la base doit être fait pour générer les ensembles fréquents : $(n+1)$ scans, pour trouver les n -itemsets fréquents.

5. Recherche des Règles d'association

Génération des règles d'association

La génération des règles d'association se fait à partir des ensembles des items fréquents obtenus par la méthode précédente. Pour chacun de ces sous-ensembles, on renvoie des règles de la forme $a \Rightarrow (l - a)$ si le rapport support $(l)/\text{support}(a)$ vaut au moins la confiance minimum. Les sous-ensembles considérés ont une taille supérieure à 1. Tant que les sous-ensembles fréquents sont en mémoire dans l'arbre de hachage, le calcul des supports peut être réalisé efficacement.

Pour réduire le nombre de règles à calculer, un principe relativement proche à celui de la réduction de la génération du nombre d'ensemble fréquents:

Règle : *Si une règle $a \Rightarrow (l-a)$ a une confiance insuffisante alors les règles $a - b \Rightarrow (l-a+b)$ auront également une confiance insuffisante.*

Cette règle va conditionner la façon dont les règles seront créées à partir des ensembles d'items fréquents. A partir d'un sous-ensemble fréquent F, les premières règles seront créées avec un seul élément dans la partie droite de la règle. Les autres règles seront générées par récurrence à partir de cette règle en passant un à un les éléments appartenant au membre de gauche dans le membre de droite. A chaque génération d'une règle, la règle, citée ci-dessus, devra vérifier la condition de génération explicitée ci-dessus.

Exemple :

Ensemble d'items fréquents 1,2,3,4

Les premières générées seront :

- 1,2,3 \Rightarrow 4
- 1,2,4 \Rightarrow 3
- 1,3,4 \Rightarrow 2
- 2,3,4 \Rightarrow 1

Supposons que les règles 1,2,4 \Rightarrow 3 et 1,3,4 \Rightarrow 2 ne soient pas valide alors la seule règle suivante générée sera :

3,2 \Rightarrow 1,4

6. Choix Algorithmique

Discrétisation des données :

Tous les items exprimés dans la base de données sont discrétisés lors de leur extraction de la base. On affecte à chaque item un identifiant unique correspond à un entier. Ce choix permet d'accélérer tous les traitements de comparaisons ainsi que réduire le stockage des données en mémoire. Le seul inconvénient est lors de la restauration des données, il faut transcrire les identifiants entre les valeurs réelles.

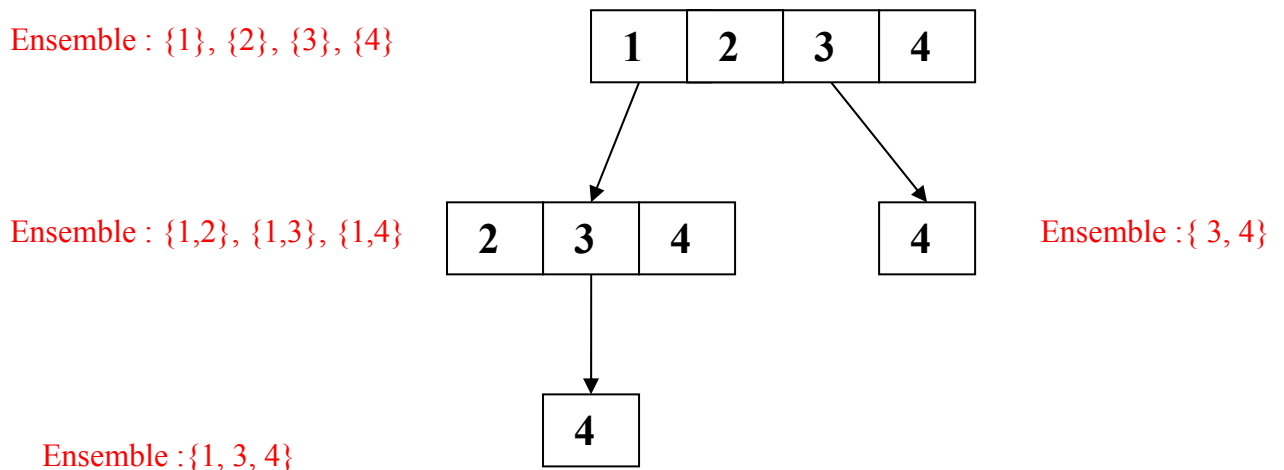
Structure de donnée utilisée :

Pour stocker les ensembles d'items fréquents, une structure en arbre est utilisée. Chaque nœud correspond à une série items. Chaque chemin allant de la racine à un nœud correspond à un ensemble d'items fréquents.

Les items appartenant à un même nœud sont triés par ordre croissant, ce qui permet de faire des recherches dichotomiques lors de la recherche d'un élément dans un nœud.

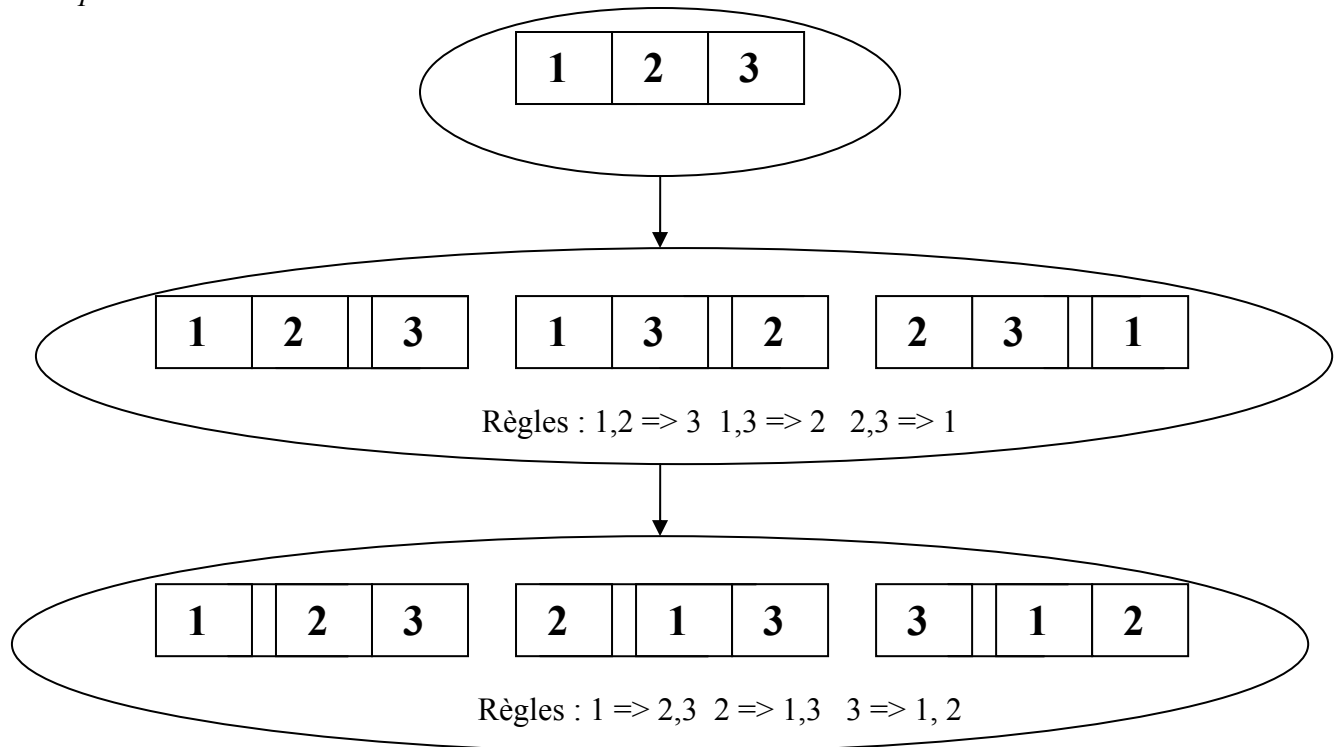
De plus, les items dans une même branche sont classés par ordre croissant, c'est-à-dire que tous les items d'une branche l à niveau k sont "supérieurs" aux items d'un niveau $n < k$. Tous les ensembles utilisés sont triés par ordre croissant, cette contrainte n'est pas difficile à satisfaire car tous les ensembles de l'arbre ont déjà cette contrainte. Seule les transactions provenant de la base doivent être triées lors de leur extraction de la base.

Exemple:



Les règles d'association sont stockées dans un arbre se basant sur le même principe.

Exemple:



Phase d'exploration :

L'exploration dans l'arbre des ensembles d'items fréquents est souvent utilisé lors des deux grandes phases de l'algorithme : recherche d'un support, existence, récupération des résultats, génération des candidats, calcul du support d'une transaction. Le principe d'exploration va être développé ci-dessous, l'existence d'un ensemble va être pris comme illustration.

L'existence d'un ensemble dans l'arbre se réduit à l'existence d'un chemin entre la racine et une feuille. Tous les ensembles sont triés ce qui simplifie la recherche. Le procédé d'existence est une simple récurrence sur la longueur de l'ensemble.

Principe :

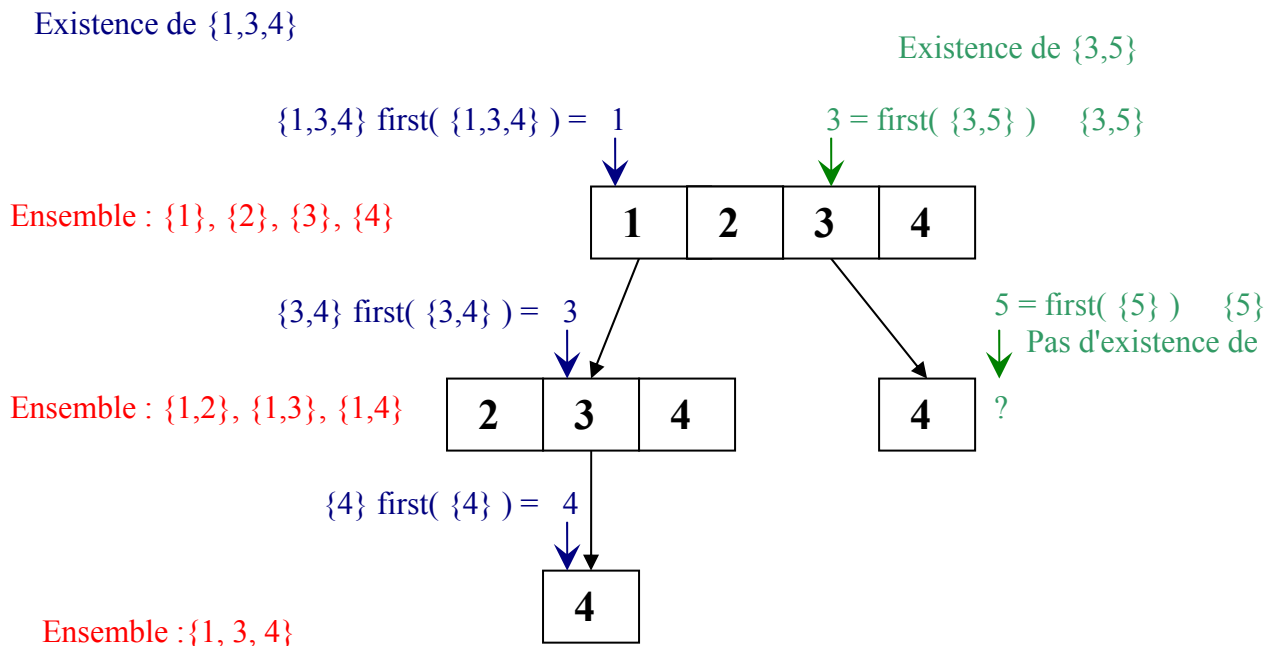
Soit A l'arbre des sous-ensembles
 Soit E l'ensemble recherché
 Soit k le niveau courant dans l'arbre

```
Existence( A, E, k )
  Test = existence( A, first(E), k )
  Si test = Vrai alors
    Si taille( E ) = 1
      Alors Vrai
      Sinon
        Si suivant(A, first(E), k) = vrai
          Alors existence (A, E-first(E), k+1)
          Sinon Faux
        Fsi
      Fsi
    Sinon
      Faux
  Fsi
```

First : renvoie le premier élément E
 Si il appartient au nœud courant
 Si c'est le dernier élément à tester dans l'ensemble alors l'ensemble est trouvé.

Sinon on doit poursuivre la recherche au nœud suivant si ce dernier existe

L'élément n'appartient pas au nœud courant alors l'ensemble n'existe pas dans l'arbre.

Exemple

7. Outils et environnement de programmation :

Le choix du langage de programmation fût très rapidement effectué. En effet, le langage Java a été choisi pour de nombreuses raisons :

- nous avons déjà une bonne connaissance de ce langage
- portabilité des applications sur toutes les plate-formes (Windows, Unix, Linux, Macintosh, ...).
- contrairement aux idées reçues, Java est souvent plus rapide que d'autres langages de programmation.
- programmation orientée objet qui permet un meilleur découpage de l'application.
- API Swing permettant la création d'interfaces graphiques très rapidement.
- réutilisation de package déjà développé dans d'autres applications

De nombreux outils de développement sont disponibles pour Java, et ils ont l'avantage d'être très souvent en open source et/ou gratuit. Nous avons principalement utilisé deux environnements suivant nos préférences :

- Borland JBuilder 9.0 : <http://www.borland.fr/jbuilder/>
- NetBeans 3.5 : <http://www.netbeans.org/>

8. Formats des fichiers :

Les fichiers contenant les bases de données peuvent être dans deux formats de représentation différents :

Format ARFF :

Ce format est utilisé par le logiciel WEKA « Waikato Environment for Knowledge Analysis » qui permet également l'extraction des règles d'association.

Exemple de fichier .arff :

```
@relation contact-lenses

@attribute age           {young, pre-presbyopic, presbyopic}
@attribute spectacle-prescrip {myope, hypermetrope}
@attribute astigmatism    {no, yes}
@attribute tear-prod-rate {reduced, normal}
@attribute contact-lenses {soft, hard, none}

@data
%
% 24 instances
%
young,myope,no,reduced,none
young,myope,no,normal,soft
young,myope,yes,reduced,none
young,myope,yes,normal,hard
young,hypermetrope,no,reduced,none
young,hypermetrope,no,normal,soft
young,hypermetrope,yes,reduced,none
young,hypermetrope,yes,normal,hard
pre-presbyopic,myope,no,reduced,none
pre-presbyopic,myope,no,normal,soft
pre-presbyopic,myope,yes,reduced,none
pre-presbyopic,myope,yes,normal,hard
pre-presbyopic,hypermetrope,no,reduced,none
pre-presbyopic,hypermetrope,no,normal,soft
pre-presbyopic,hypermetrope,yes,reduced,none
pre-presbyopic,hypermetrope,yes,normal,none
presbyopic,myope,no,reduced,none
presbyopic,myope,no,normal,none
presbyopic,myope,yes,reduced,none
presbyopic,myope,yes,normal,hard
presbyopic,hypermetrope,no,reduced,none
presbyopic,hypermetrope,no,normal,soft
presbyopic,hypermetrope,yes,reduced,none
presbyopic,hypermetrope,yes,normal,none
```

Format CSV :

Ce format est très simple puisqu'il contient juste une transaction par ligne et pour chaque transaction les items sont séparés par des « ; ». Le nombre d'items par ligne peut être variable et le premier item correspond au numéro de la transaction.

Les fichiers Excel, ainsi que de nombreuses bases de données contiennent des fonctions permettant l'exportation au format CSV.

Exemple de fichier .csv :

```
TRANSACTIONID;ITEMS
5;rainy;cool;normal;FALSE;yes
6;rainy;cool;normal;TRUE;no
7;overcast;cool;normal;TRUE;yes
8;sunny;mild;high;FALSE;no
9;sunny;cool;normal;FALSE;yes
10;rainy;mild;normal;FALSE;yes
11;sunny;mild;normal;TRUE;yes
12;overcast;mild;high;TRUE;yes
13;overcast;hot;normal;FALSE;yes
14;rainy;mild;high;TRUE;no
1;sunny;hot;high;FALSE;no
2;sunny;hot;high;TRUE;no
3;overcast;hot;high;FALSE;yes
4;rainy;mild;high;FALSE;yes
```

9. Manuel utilisateur :

L'interface de Merja utilise une représentation sous la forme de quatre onglets ; Fichier/BDD, Données, Itemsets fréquents, Règles d'association.

Une barre de boutons est présente à gauche de l'interface permettant d'effectuer plus rapidement certaines actions. Enfin un log en bas de l'interface tient au courant l'utilisateur sur les traitements en cours ainsi que sur les éventuelles erreurs.

Bugs :

Lorsque le nombre d'itemsets ou le nombre de règles est très important, de l'ordre de plus de 30 000 éléments. L'affichage de ces éléments dans les zones de textes peut provoquer des exceptions (out of memory). Cela est dû à une limitation du composant Java `JTextArea` qui ne peut afficher qu'un nombre limité de caractères.

Pour résoudre ce problème il faut soit arrêter le traitement à l'aide des boutons « stop », soit limité le nombre d'éléments en augmentant le support ou la confiance minimum.

Barre de boutons :



Ouverture d'un fichier contenant une base de données au format ARFF ou CSV.



Lance la recherche des itemsets fréquents de la base courante.



Lance la recherche des règles d'association de la base courante.



Ferme la base de données courante et efface tous les champs.



Stoppe le traitement de l'onglet courant.



Affiche une boîte de dialogue avec des informations sur Merja.

Log :

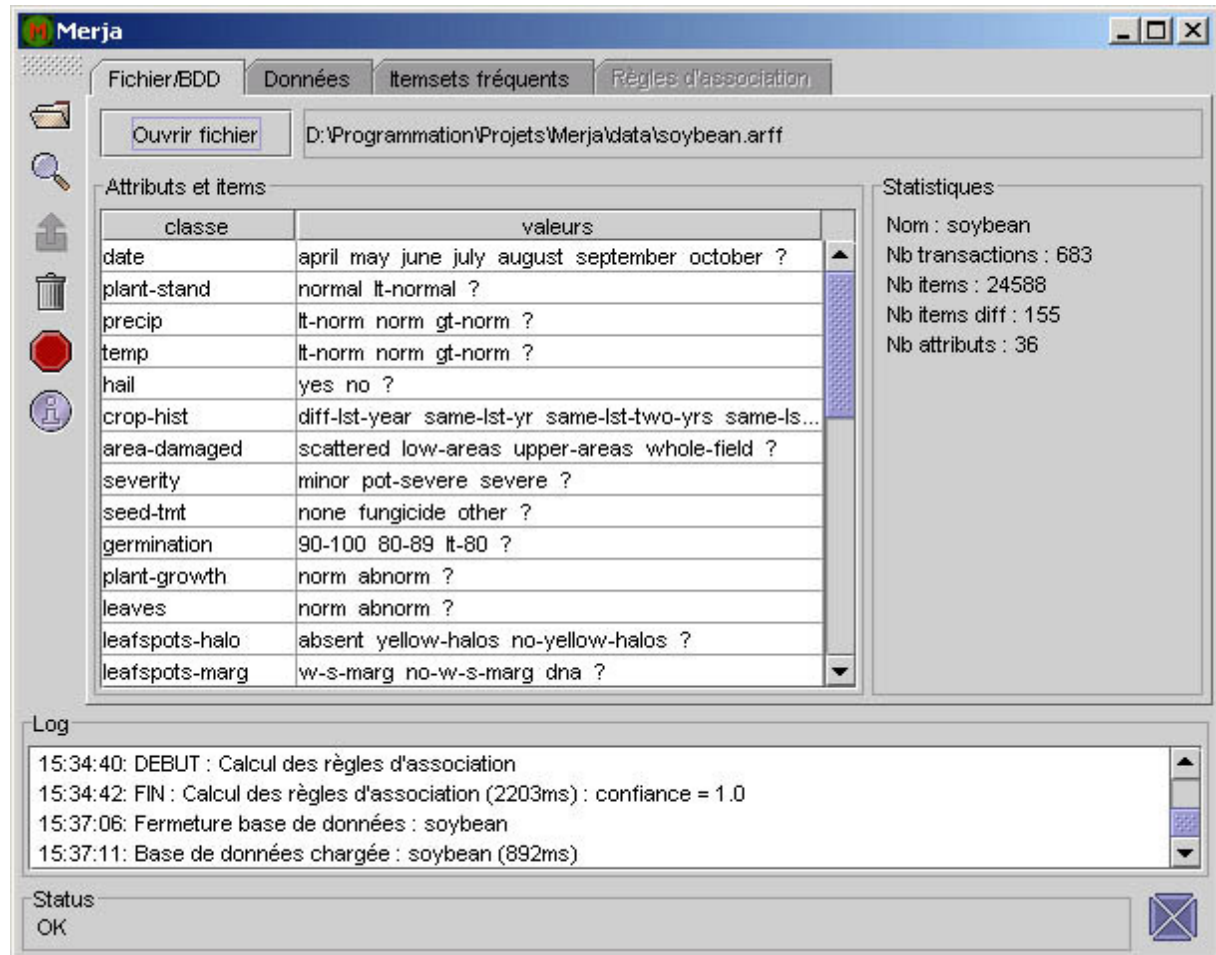
Le log permet d'afficher des informations sur le début ou la fin d'un traitement et l'ouverture ou la fermeture d'une base de données. Un champ de texte « Status » informe si la dernière action a été un succès ou un échec.



Cette image permet de savoir si un traitement est en cours, en s'animant.

Onglet 1 : Fichier/BDD

Permet l'ouverture d'un fichier au format ARFF ou CSV et affiche les principales caractéristiques.



Le bouton « ouvrir un fichier » affiche une boîte de dialogue permettant de choisir un fichier à traiter. Le champ à droite donne le chemin d'accès de ce fichier.

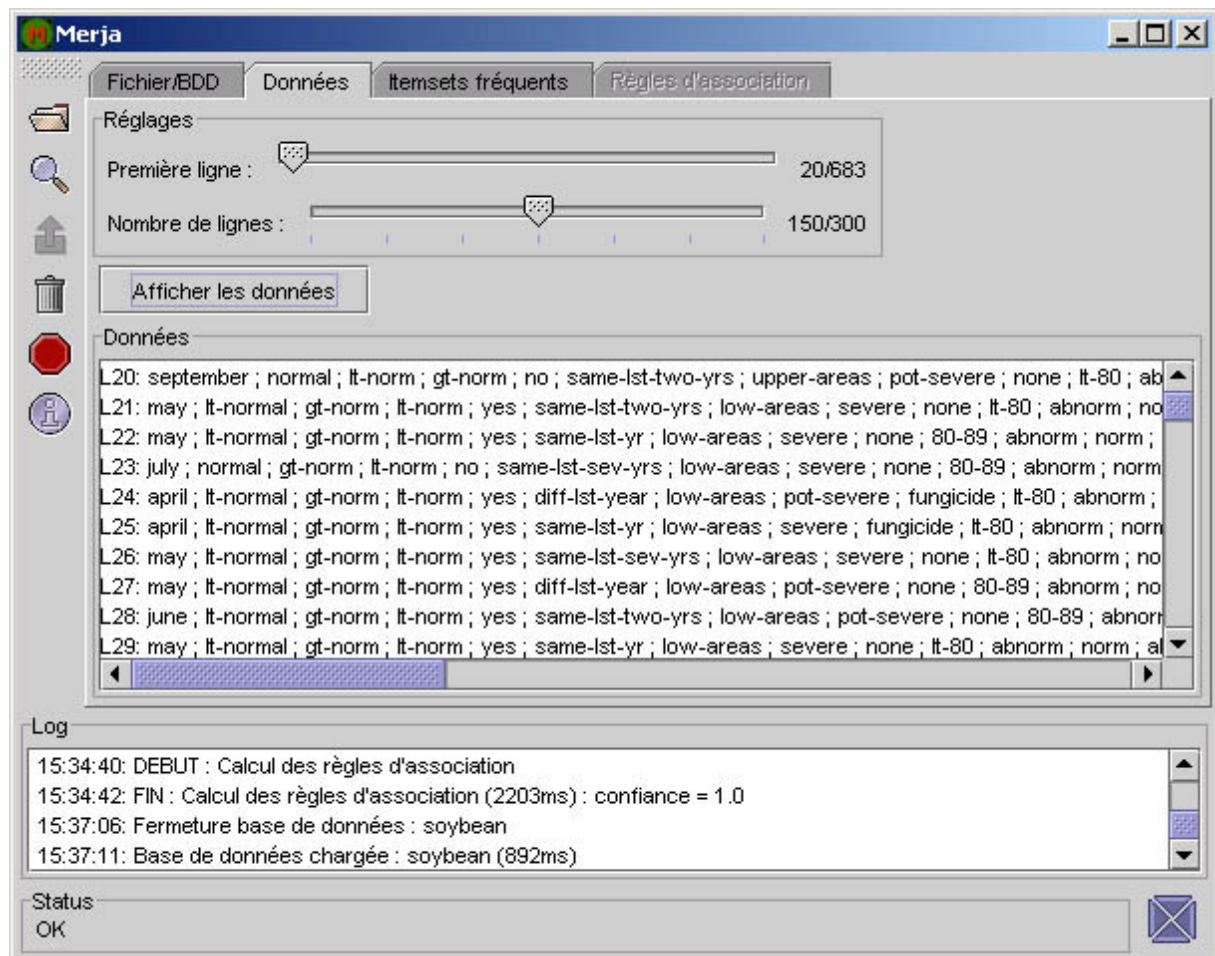
Pour les fichiers ARFF la zone « Attribut et items » donne tous les attributs avec leurs valeurs possibles.

Pour les fichiers CSV la zone affiche tous les items différents que contient la base de transactions.

La partie « Statistiques affiche » diverses informations sur la base de données chargées.

Onglet 2 : Données

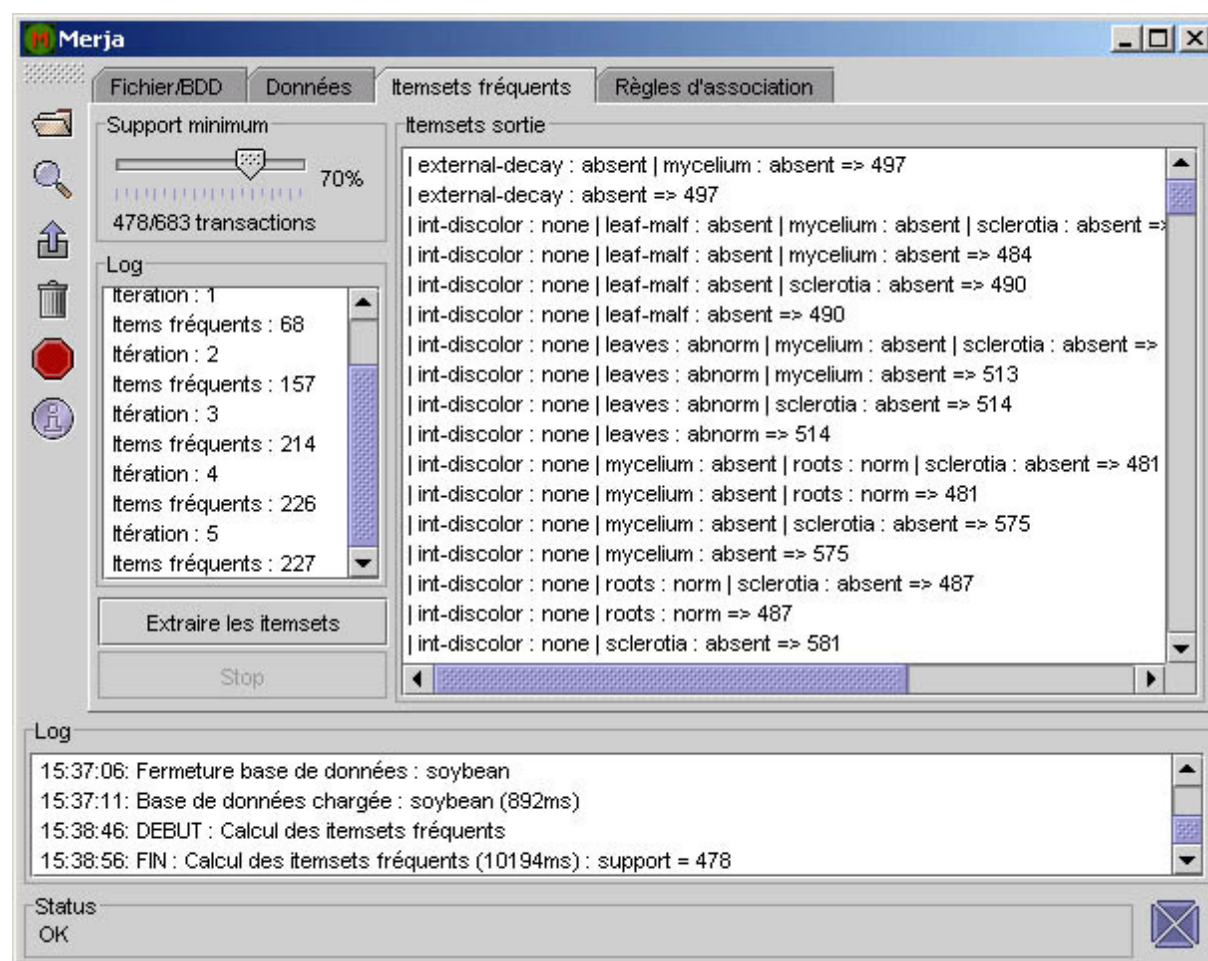
Permet d'afficher les données du fichier précédemment chargé.



Cette page permet d'afficher une partie des lignes de la base de données. Il suffit de choisir la première la lignes ainsi que le nombre de lignes à afficher à partir de cette première ligne.

Onglet 3 : Itemsets fréquents

Recherche et affichage des itemsets fréquents.



La spécification du support minimum s'effectue en choisissant un certain pourcentage, la correspondance entre ce pourcentage et le nombre de transactions est également affiché.

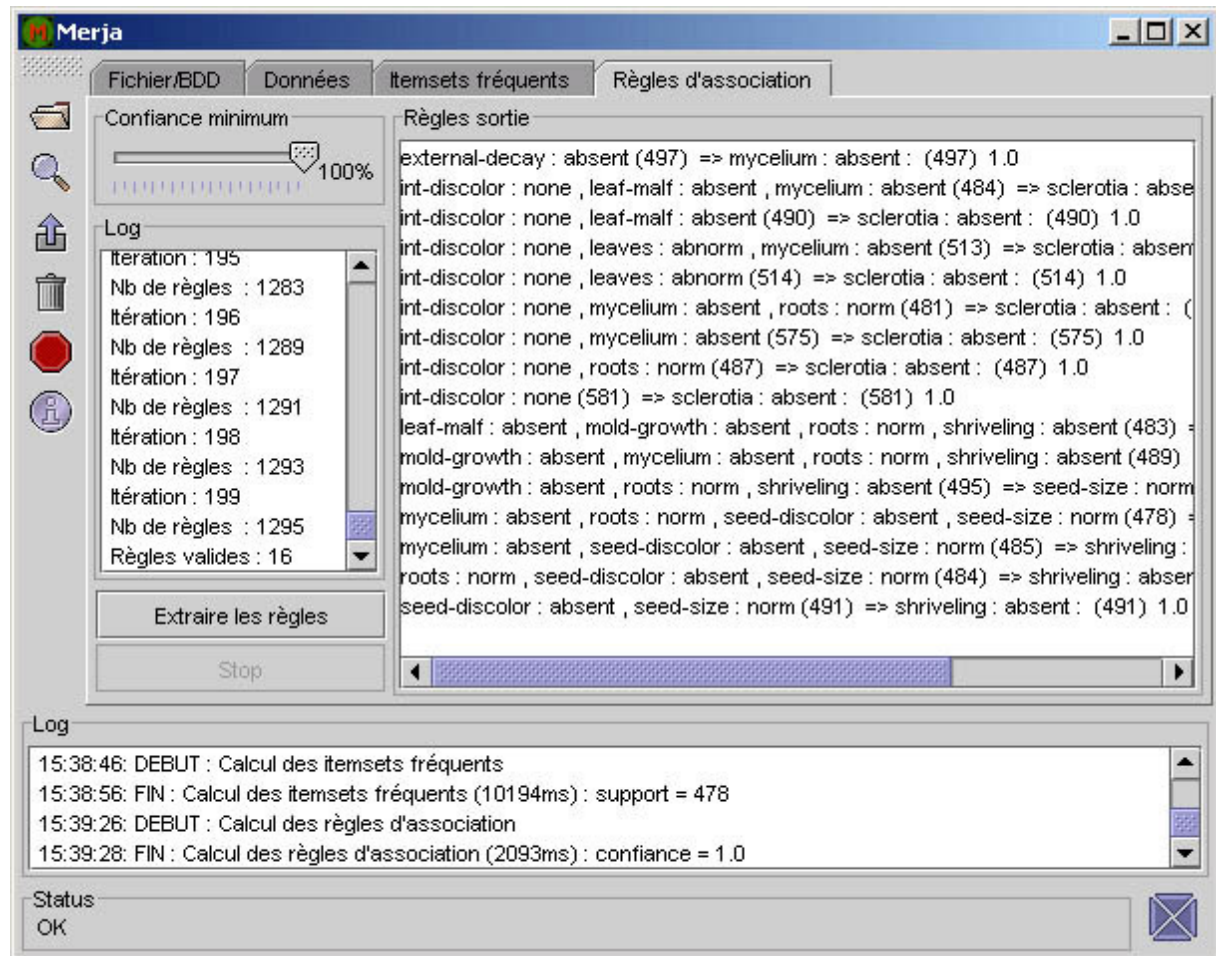
Lors de l'extraction des itemsets fréquents la zone « log » affiche les itérations de l'algorithme ainsi que le nombre d'itemsets pour chaque itération.

Enfin l'affichage des itemsets fréquents se fait de la façon suivante :

```
| item | item | ... | item => support
```

Onglet 4 : Règles d'association

Recherche et affichage des règles d'association.



La spécification de la confiance minimum s'effectue en choisissant un pourcentage à l'aide d'une réglette.

Lors de la recherche des règles la zone « log » affiche les itérations ainsi que le nombre de règles trouvées par l'algorithme. A la fin du traitement le nombre de règles valides (conformément à la confiance minimum) est affiché.

Enfin l'affichage des règles d'association se fait de la façon suivante :

item , ... , item (support) => item , ... (support) confiance

10. Conclusion :

La réalisation du projet a été amenée à terme, le but d'implémenter l'algorithme APriori a été réalisé et le logiciel est fonctionnel. De plus, les différents tests donne des résultats tout à fait convaincants.

En comparaison au logiciel Weka, le logiciel Merja semble plus lent mais n'extrait pas les règles de la même manière que Weka, pourtant ces logiciels utilisent l'algorithme APriori pour extraire les règles d'associations.

Les améliorations possibles de ce logiciel seront d'implanter d'autres algorithmes comme AprioriHybrid et AprioriTid et de tester les performances. De plus, il serait intéressant de tester ce logiciel sur des bases données plus cohérentes que celles utilisés lors de la phase de tests, mais cela demande un travail de récupération de données depuis une base ainsi que la classification de données.

Darnet Camille : camsss@fr.st
Vouriot Thierry : veri@fr.st
<http://www.yeri.fr.st>