

Les concepts de haut niveau dans le processus de database mining The high level concepts in the database mining process

Amrane Hocine*, Patrick Raffinat**

* Département d'Informatique
Faculté des Sciences et Techniques
Université de Pau
BP 1155 – 64013 Pau Cedex
E-mail : Amrane.Hocine@univ-pau.fr
Tél. 33 5 59 92 31 96
Fax : 33 5 59 80 83 74

** Département STID
IUT de Pau
Université de Pau
Avenue de l'Université - 64000 Pau
Email : Patrick.Raffinat@univ-pau.fr
Tél. 33 5 59 92 33 85
Fax : 33 5 59 92 33 99

Résumé :

Dans cet article, nous étudions un des aspects importants du processus de data mining : la sélection et l'extraction d'informations stockées dans une base de données, à partir desquelles seront découvertes de nouvelles connaissances. L'approche proposée est fondée sur l'intégration des connaissances spécifiques au domaine qui sont modélisées en termes de concepts de haut niveau (concepts hiérarchiques ou flous). La sélection est guidée par les besoins et le domaine d'intérêt de l'utilisateur. Elle s'exprime sous la forme d'une requête de type SQL utilisant des concepts de haut niveau ou élémentaires. La réponse à cette requête est exprimée dans les mêmes termes.

Notre approche est validée par un prototype réalisé sous Access et peut être implantée avec tout SGBD relationnel dont le langage SQL accepte des fonctions personnalisées.

Mots clés : base de données, concept flou, concept hiérarchique, découverte de connaissances, processus de data mining, requête sql.

Abstract :

In this paper, we study one of the important aspects of the data mining process : the selection and the extraction of information stored in a database. This information is used to discover new knowledge. The proposed approach is (i) based on the integration of domain knowledge which is modelised by the notion of high level concept (hierarchical or fuzzy concept), (ii) guided by the needs and the user's domain of interest. It is described by a query using high or elementary level concepts. The answer of this query is described using same terms as that query.

Our approach is validated by a prototype realized with Access DBMS. It can be implemented with any relational DBMS in which the SQL language accepts the personalized functions.

Key Words : database, data mining process, fuzzy concept, hierarchical concept, knowledge discovery, sql query.

1. Introduction

La découverte de connaissances dans les bases de données (KDD) désigne le processus global permettant de découvrir les connaissances (règles, modèles, concepts, régularités, tendances ...) utiles, intéressantes et intelligibles qui sont enfouies dans ces bases [Fay96a]. Le data mining est l'étape la plus connue et étudiée de ce processus (Fig. 1), mais elle n'en constitue qu'une petite partie (15 à 20% selon [Fay96b]). En revanche, la première étape du KDD (étape de sélection) semble relativement peu étudiée, même si elle a été identifiée comme fondamentale et complexe par de nombreux auteurs [Fay96a, Bra96, Imi96b, Han97, Ana97]. Deux raisons majeures justifient l'importance de cette étape : l'influence de la qualité des informations extraites sur la qualité des connaissances ensuite découvertes et le caractère itératif du processus de KDD, qui conduit l'utilisateur à multiplier les requêtes de sélection.

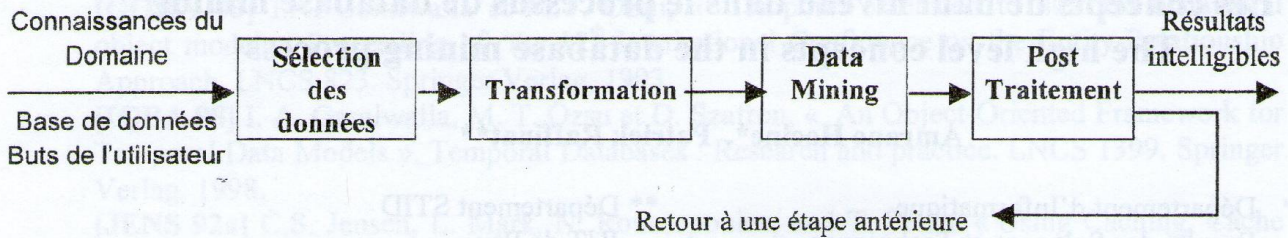


Figure 1 : une vue simplifiée du processus de KDD

La qualité des informations extraites peut être améliorée par l'introduction de connaissances du domaine, souvent exprimées sous la forme de hiérarchies de concepts. Malheureusement, les SGBD actuels ne permettent pas de formuler des requêtes de sélection en termes de concepts de haut niveau, c'est à dire de données non primitives de hiérarchies. C'est pourquoi nous cherchons à étendre aux concepts de haut niveau (éventuellement flous) un sous ensemble du langage SQL, ceci dans le cadre des SGBD relationnels afin de tirer partie des avantages qu'ils offrent. Les connaissances du domaine sont stockées dans une métabase, à partir desquelles des fonctions personnalisées sont engendrées. Ces fonctions permettent de transformer des requêtes de type SQL exprimées en termes de concepts de haut niveau en requêtes SQL valides dans un SGBD.

Notre travail suit une approche de type « database mining » [Imi96a, Imi96b, Est98], qui s'appuie sur la technologie des bases de données et offre une interactivité effective avec l'utilisateur. Cette approche est adoptée par plusieurs systèmes de data mining, qui étendent le langage SQL par des primitives ou des opérations de data mining. Parmi les plus connus de ces langages pseudo-sql, nous pouvons citer DMQL (Data Mining Query Language) dans le système DBMiner [Han96a, Han 97, Han98], M-SQL dans le système Data Mine [Imi96a] et Mine Rule[Meo96].

La section deux de cet article est consacrée aux hiérarchies de concepts, qui sont le principal (et le seul ?) type de connaissances du domaine ayant un profond impact sur le data mining. Nous y présentons diverses sortes de hiérarchies, dont les hiérarchies floues qui étendent la notion de hiérarchie classique. La section 3 décrit la modélisation de telles hiérarchies dans une métabase et montre comment sont engendrées les fonctions personnalisées à partir du contenu de la métabase. La section 4 explique comment sont interprétées et satisfaites les requêtes de type SQL formulées par l'utilisateur.

Notre travail a été validé par un prototype réalisé sous Access [Bin97, Caz98].

2. Hiérarchies de concepts

2.1. Présentation générale

A notre connaissance, il n'existe pas de définition standard d'une hiérarchie de concepts. De surcroît, les diverses descriptions proposées sont souvent intuitives, semi-formelles et exprimées dans des formalismes différents. Afin de recouvrir diverses approches, dont celle relative aux hiérarchies floues, nous allons cependant tenter de donner quelques « définitions générales ».

Définition 1 :

Un concept est un sous-ensemble de l'ensemble des « entités du monde réel » (noté Ω) pouvant être décrit à la fois en extension (description exhaustive) et en intention (description plus synthétique). Chaque concept porte un nom qui lui est propre et qui est appelé identifiant du concept.

En pratique, une entité peut être assimilée à un n-uplet d'une relation (ou d'une vue) d'une base de données. La description en intention, quant à elle, s'exprime sous la forme d'une formule logique formée d'attributs, d'opérateurs usuels ($=$, $>$, $\text{and} \dots$), d'éléments du domaine des attributs, voire de prédicats logiques [Dha93].

Exemple :

Soit filière un attribut de domaine $\text{Dom}(\text{filière}) = \{\text{scientifique}, \text{économique}, \text{littéraire}\}$

Soit note un attribut de domaine $\text{Dom}(\text{note}) = [0, 20]$ (Fig. 2)

Voici quelques descriptions en intention du concept « moyen- » envisageables :

- $Note \geq 8$ et $Note < 10$ (noté ultérieurement $Note = [8, 10[$)
- ($Note = [9, 11[$ et filière=économique) ou ($Note = [8, 10[$ et filière=scientifique)

Définition 2 :

Une hiérarchie de concepts est un arbre dont les nœuds sont des concepts. Un nœud est une généralisation de ses fils, c'est à dire l'union ensembliste de ses nœuds fils.

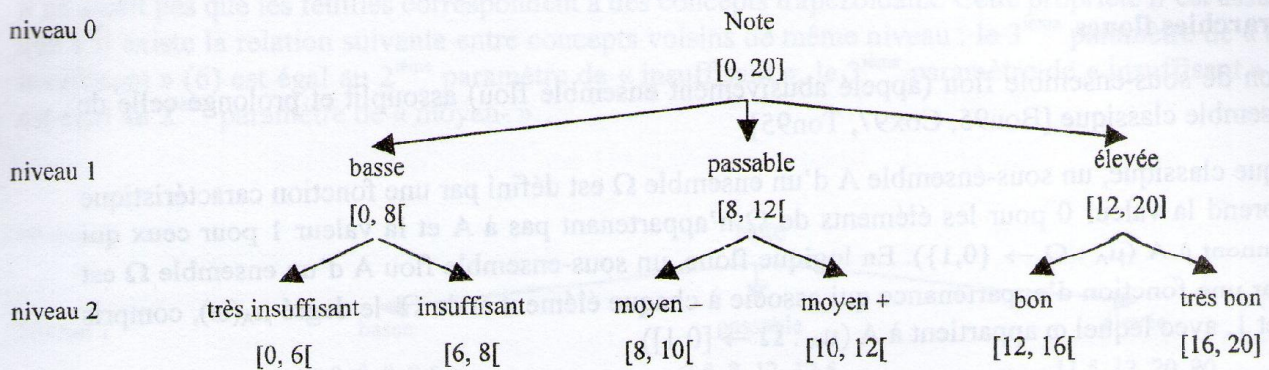


Figure 2 : hiérarchie numérique associée à l'attribut Note

Les hiérarchies très majoritairement utilisées en data mining sont mono-attribut (Fig. 2, Fig. 3), ce qui signifie que les formules définissant en intention leurs concepts ne contiennent qu'un attribut et que cet attribut est commun. Leurs concepts peuvent être assimilés à des valeurs (de haut niveau) de l'attribut : nous noterons donc parfois un concept sous la forme « attribut=nom du concept » (par exemple, « note=moyen- » au lieu de « moyen- »). Il s'agit d'un abus de langage car les noms de concepts de haut niveau ne font pas partie du domaine de l'attribut.

2.2 Hiérarchies « classiques » (non floues)

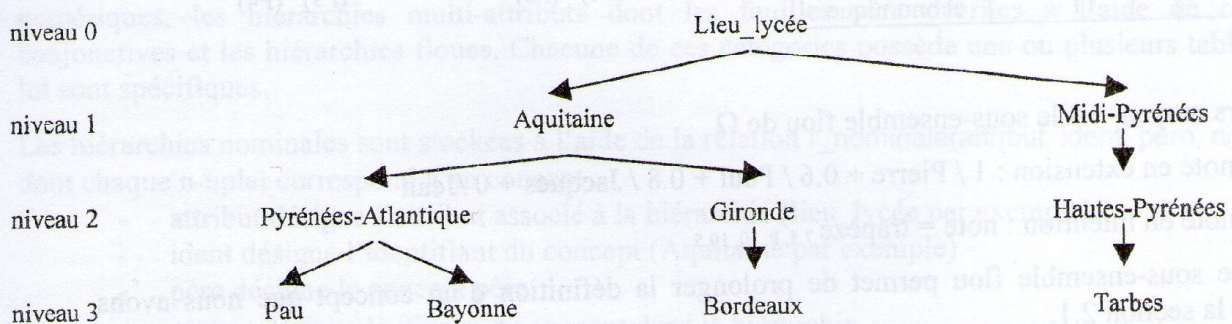


Figure 3 : (partie de la) hiérarchie nominale associée à l'attribut lieu_lycée

Une hiérarchie mono-attribut peut être nominale ou numérique, suivant la nature de l'attribut associé. La hiérarchie nominale lieu_lycée (Fig. 3) décrit le lieu où est situé un lycée. Les données brutes sont des communes (feuilles de l'arbre), qui peuvent être généralisées en départements ou en régions. Un indice de niveau est utilisé pour différencier les divers niveaux de généralisation de l'attribut.

La hiérarchie numérique note (Fig. 2) peut être utilisée pour classer des lycéens en fonction de leur moyenne générale en terminale, ainsi que nous le supposerons par la suite. Les données brutes (les notes) sont regroupées ici en intervalles, comme pour la plupart des hiérarchies numériques utilisées en pratique. Les intervalles de telles hiérarchies peuvent soit être fixées par un expert du domaine, soit être obtenues par un algorithme de discrétisation dans le cas où leurs bornes ne sont pas « évidentes ».

Les hiérarchies multi-attributs [Dha93, Lu98] sont des hiérarchies dont les concepts sont définis en intention à l'aide de plusieurs attributs. Dans la modélisation proposée par Dhar, chaque concept est

associé à un prédicat défini par l'utilisateur, c'est à dire une disjonction de clauses conjonctives dont chaque formule atomique est soit une condition sur les attributs, soit un autre prédicat.

Les hiérarchies multi-attributs sont relativement peu utilisées en data mining, mais peuvent néanmoins présenter un intérêt dans certaines situations. Ainsi, un enseignant chargé du recrutement de lycéens interprète généralement les notes en fonction du type de bac. Par exemple, le concept « moyen- » peut être caractérisé par les clauses conjonctives suivantes :

- note ≥ 8 et note < 10 et filière='scientifique'
- note ≥ 9 et note < 11 et filière='économique'

2.3 Hiérarchies floues

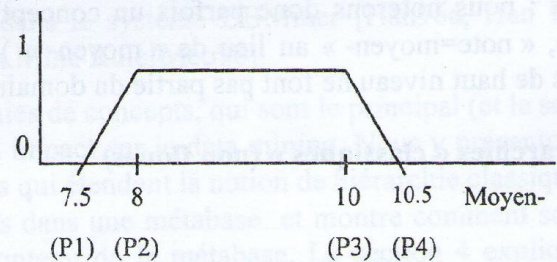
La notion de sous-ensemble flou (appelé abusivement ensemble flou) assouplit et prolonge celle de sous-ensemble classique [Bou95, Cox97, Ton95].

En logique classique, un sous-ensemble A d'un ensemble Ω est défini par une fonction caractéristique μ_A qui prend la valeur 0 pour les éléments de Ω n'appartenant pas à A et la valeur 1 pour ceux qui appartiennent à A ($\mu_A : \Omega \rightarrow \{0,1\}$). En logique floue, un sous-ensemble flou A d'un ensemble Ω est défini par une fonction d'appartenance qui associe à chaque élément ω de Ω le degré $\mu_A(\omega)$, compris entre 0 et 1, avec lequel ω appartient à A ($\mu_A : \Omega \rightarrow [0,1]$).

La logique floue permet donc d'introduire une nouvelle catégorie d'éléments, à savoir ceux appartenant partiellement à un ensemble. C'est le cas de Paul et Jacques dans l'exemple suivant :

Soient Ω l'ensemble des lycéens et $\mu_{\text{moyen-}}$ la fonction d'appartenance au sous-ensemble flou $A = \text{« moyen- »}$:

nom	note	filiale
Pierre	9	scientifique
Paul	7,8	scientifique
Jacques	7,9	économique
Jean	11	économique



Alors moyen- est le sous-ensemble flou de Ω

- noté en extension : $1 / \text{Pierre} + 0.6 / \text{Paul} + 0.8 / \text{Jacques} + 0 / \text{Jean}$
- noté en intention : note = trapèze $7.5, 8, 10, 10.5$

La notion de sous-ensemble flou permet de prolonger la définition d'un concept que nous avons donnée dans la section 2.1.

Définition 3 :

Un concept flou est un sous-ensemble flou de l'ensemble des « entités du monde réel » (noté Ω) pouvant être décrit à la fois en extension (description exhaustive) et en intention (description plus synthétique). Chaque concept porte un nom qui lui est propre et qui est appelé identifiant du concept.

En pratique, nous nous limiterons à des concepts flous dont la fonction d'appartenance est trapézoïdale. La forme générale de la description en intention d'un concept flou sera donc « attribut = trapèze p_1, p_2, p_3, p_4 (par exemple, note = trapèze $7.5, 8, 10, 10.5$ pour le concept flou « moyen- »).

Pour définir une hiérarchie floue, il reste à donner la définition de l'union de sous ensembles flous. La voici, avec celle de l'intersection :

- L'union de deux sous ensembles flous de Ω est le sous ensemble flou noté $A \cup B$ tel que, pour chaque élément ω de Ω , on ait $\mu_{A \cup B}(\omega) = \max(\mu_A(\omega), \mu_B(\omega))$.
- L'intersection de deux sous ensembles flous de Ω est le sous ensemble flou noté $A \cap B$ tel

que, pour chaque élément ω de Ω , on ait $\mu_{A \cap B}(\omega) = \min(\mu_A(\omega), \mu_B(\omega))$.

Définition 4 :

Une hiérarchie floue de concepts est un arbre dont les nœuds sont des concepts flous. Un nœud est une généralisation de ses fils, c'est à dire l'union ensembliste de ses nœuds fils.

La figure 4 montre une hiérarchie floue, associée à l'attribut note, qui étend la hiérarchie numérique représentée sur la figure 2. Pour obtenir une hiérarchie floue dont tous les concepts sont trapézoïdaux, il ne suffit pas que les feuilles correspondent à des concepts trapézoïdaux. Cette propriété n'est assurée que s'il existe la relation suivante entre concepts voisins de même niveau : le 3^{ème} paramètre de « très insuffisant » (6) est égal au 2^{ème} paramètre de « insuffisant », le 3^{ème} paramètre de « insuffisant » (8) est égal au 2^{ème} paramètre de « moyen - »...

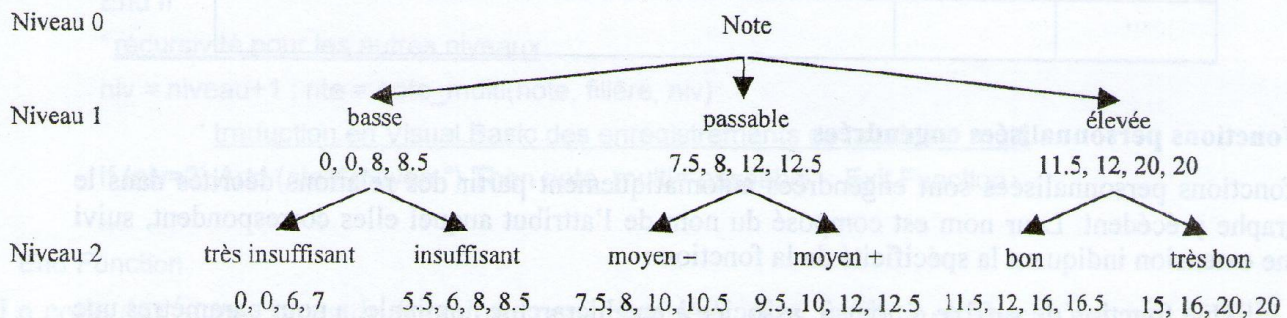


Figure 4 : hiérarchie floue associée à l'attribut note

3 Métabase des connaissances du domaine

3.1 Représentation des hiérarchies

Les connaissances du domaine sont stockées dans des tables relationnelles, ce type de modélisation étant naturel dans une approche orientée « database mining ». Quatre catégories de hiérarchies sont à prendre en compte : les hiérarchies mono-attribut nominales, les hiérarchies mono-attribut numériques, les hiérarchies multi-attributs dont les feuilles sont décrites à l'aide de clauses conjonctives et les hiérarchies floues. Chacune de ces catégories possède une ou plusieurs tables qui lui sont spécifiques.

Les hiérarchies nominales sont stockées à l'aide de la relation $r_nominale(attribut, ident, père, niveau)$, dont chaque n-uplet correspond à un concept :

- attribut désigne l'attribut associé à la hiérarchie (lieu_lycée par exemple)
- ident désigne l'identifiant du concept (Aquitaine par exemple)
- père désigne le concept père
- niveau désigne le niveau du concept dans la hiérarchie

Exemple :

attribut	ident	père	niveau
Lieu_lycée	Pau	Pyrénées_Atlantiques	3
Lieu_lycée	Bayonne	Pyrénées_Atlantiques	3
Lieu_lycée	Gironde	Aquitaine	2
Lieu_lycée	Midi_Pyrénées	Lieu_lycée	1
...			

Les autres types de hiérarchies sont stockées dans des tables similaires ($r_numérique$, r_multi et r_flou), les hiérarchies multi-attributs requérant en outre une relation supplémentaire (r_clause). Les

hiérarchies floues sont stockées à l'aide de la relation $r_floue(attribut, ident, père, niveau, P1, P2, P3, P4)$, dont les attributs P1 à P4 correspondent aux abscisses des sommets d'un trapèze. Les clauses associées aux feuilles d'une hiérarchie multi-attributs, quant à elles, sont enregistrées dans la relation $r_clause(attribut, ident, clause)$: attribut désigne un des attributs de la hiérarchie multiple, appelé attribut dominant (note par exemple), qui est privilégié afin de pouvoir conserver une interprétation attribut-valeur comme dans le cas mono-attribut.

Exemple :

attribut	ident	clause
note	Moyen-	Note $\geq$ 8 and Note $<$ 10 and filière= "scientifique"
note	Moyen-	Note $\geq$ 9 and Note $<$ 11 and filière= "économique"
...		

3.2. Fonctions personnalisées engendrées

Des fonctions personnalisées sont engendrées automatiquement partir des relations décrites dans le paragraphe précédent. Leur nom est composé du nom de l'attribut auquel elles correspondent, suivi par une extension indiquant la spécificité de la fonction :

- Une fonction de suffixe « nomi », associée à une hiérarchie nominale, a pour paramètres une valeur brute (une feuille de la hiérarchie) et un niveau dans la hiérarchie. Elle renvoie le concept de l'arbre correspondant : lieu_lycée_nomi("Pau", 2) vaut "Pyrénées_Atlantiques", lieu_lycée_nomi("Bordeaux", 1) vaut "Aquitaine"...
- Une fonction de suffixe multi, associée à une hiérarchie multi-attributs, a pour paramètres les divers attributs et un niveau de la hiérarchie. Elle renvoie le concept de l'arbre correspondant : par exemple, note_multi(8.5, "scientifique", 2) vaut "moyen-".
- Une fonction de suffixe numé, associée à une hiérarchie numérique, a pour paramètres une valeur brute de l'attribut et un niveau de la hiérarchie. Elle renvoie le concept de l'arbre correspondant : par exemple, note_numé(8.5, 1) vaut "passable".
- Une fonction de suffixe flou, associée à une hiérarchie floue, a pour paramètres une valeur brute de l'attribut et un niveau de la hiérarchie. Elle renvoie le concept du niveau mentionné pour lequel le niveau d'appartenance est maximal : par exemple, note_flou(8.5, 1) vaut "passable".

Le code d'une fonction de suffixe « nomi » se déduit de la relation $r_nominale$. Le code Visual Basic de la fonction lieu_lycée_nomi, donné à titre d'exemple, fait clairement apparaître le lien puisqu'à chaque enregistrement correspond une instruction conditionnelle :

Function lieu_lycée_nomi(lieu_lycée, niveau)

 ' feuilles

 If (niveau = 3) Then lieu_lycée_nomi=lieu_lycée : Exit Function

 ' récursivité pour les autres niveaux

 niv = niveau+1 : lyc = lieu_lycée_nomi(lieu_lycée, niv)

 ' traduction en Visual Basic des enregistrements de la table $r_nominale$

 If (niv=3) And (lyc="Pau") Then lieu_lycée_nomi="Pyrénées_Atlantiques" : Exit Function

 If (niv=3) And (lyc="Bayonne") Then lieu_lycée_nomi="Pyrénées_Atlantiques" : Exit Function

 If (niv=2) And (lyc="Gironde") Then lieu_lycée_nomi="Aquitaine" : Exit Function

End Function

Le code des fonctions de suffixe « multi » peut être obtenu avec une technique similaire, la différence se situant au niveau des feuilles de l'arbre. Le code Visual Basic de la fonction `note_multi`, donné à titre d'exemple, fait clairement apparaître le lien entre les feuilles de l'arbre et la table `r_clause` puisqu'à chaque enregistrement correspond une instruction conditionnelle :

Function `note_multi`(note, filière, niveau)

 ' feuilles

 If (niveau = 2) Then

 ' traduction en Visual Basic des enregistrements de la table r_clause

 If note >= 9 And note < 11 and filière = "économique" Then `note_multi` = "moyen-" : Exit Function

 If note >= 8 And note < 10 and filière = "scientifique" Then `note_multi` = "moyen-" : Exit Function

 ...

 End If

 ' récurtivité pour les autres niveaux

 niv = niveau + 1 : nte = `note_multi`(note, filière, niv)

 ' traduction en Visual Basic des enregistrements de la table r_multi

 If (niv = 2) And (nte = "moyen-") Then `note_multi` = "passable" : Exit Function

 ...

End Function

Le code des autres types de fonctions personnalisées, plus facile à obtenir, n'est pas détaillé ici.

4 Sélection des données

La sélection des données à fournir à un algorithme de découverte peut mobiliser beaucoup de temps et d'énergie, d'autant plus qu'on doit généralement répéter cette opération. C'est pourquoi il faut offrir à l'utilisateur la possibilité d'exprimer ses requêtes à l'aide de concepts de haut niveau. Dans ce paragraphe, nous nous intéressons à une famille de requêtes de type SQL pouvant être transformées en requêtes SQL valides dans un SGBD relationnel. Ces requêtes, sans emboîtement de « select », ont la forme suivante :

Select A1[!niveau], A2[!niveau] ...

From R1, R2 ...

Where condition_de_jointure and condition_conceptuelle

[AvecSeuilFlou ...]

Une condition conceptuelle est une expression conditionnelle définie uniquement à partir de certains opérateurs logiques (and, or, not), de concepts bruts (`note=10`, `note>10`...) et de haut niveau (`note=moyen`...). Chaque concept de haut niveau d'une condition conceptuelle est à transformer en une expression SQL valide, suivant une technique dépendant de la nature de ce concept :

- Un concept nominal est transformé en une expression faisant apparaître toutes les feuilles du sous arbre dont il est la racine [Han96b] : par exemple, `lieu_lycée = 'Aquitaine'` est transformé en `lieu_lycée in ('Pau', 'Bayonne', 'Bordeaux')`.
- Un concept numérique est transformé en une expression faisant intervenir ses deux bornes : par exemple, `note = 'moyen'` est transformé en `(note >= 8 and note < 10)`.
- Un concept flou est transformé en une expression dépendant d'un seuil flou plancher [Bos95] : par exemple, `note = 'moyen'` est transformé en `(note >= 7.9 and note <= 10.1)` pour un seuil de 0.8.
- Un concept multi-attributs est transformé en une expression faisant apparaître la fonction de suffixe « multi » qui lui est associée et son niveau dans l'arbre : par exemple, `note = 'moyen'` est transformé en `note_multi(note, filière, 2) = "moyen"`.

Souvent, l'utilisateur ne souhaite pas récupérer des valeurs brutes, mais des valeurs de haut niveau. C'est pourquoi il a la possibilité de fixer un niveau de généralisation pour chaque attribut sélectionné, comme l'illustre la requête suivante :

```
Select filière, note!2, lieu_lycée!1
From Lycéen
Where (filière='scientifique' and note='moyen+') or (note='élevée')
AvecSeuilFlou 0.8
```

Les attributs sont alors transformés en une expression faisant intervenir les fonctions personnalisées introduites dans la section précédente : lieu_lycée!1 est transformé en lieu_lycée_nomi(lieu_lycée, 1) et note!2 est transformé en note_flou(note, 2). La requête SQL valide finalement obtenue est alors la suivante :

```
Select filière, note_flou(note, 2), lieu_lycée_nomi(lieu_lycée, 1)
From Lycéen
Where (filière='scientifique' and note>=9.9 and note<=12.1)
or (note>=11.9 and note<=20)
```

5 Conclusion et perspectives

Notre travail facilite l'extraction des informations à fournir aux algorithmes de data mining (langage de requête plus naturel grâce aux concepts de haut niveau) et améliore la qualité de ces informations. Il ajoute une « dose de flou » à l'approche hiérarchique classique. La méthode de sélection proposée peut aussi être considérée comme une interface intelligente pour l'interrogation de bases de données relationnelles, la seule contrainte étant de disposer d'un langage SQL acceptant des fonctions personnalisées (norme SQL3). Elle a été validée par un prototype implanté avec le SGBD Access.

Ce prototype a ensuite été perfectionné afin de faciliter la recherche de règles d'association inter-dimension et multi-niveaux [Sri95]. Ces règles sont des règles d'association dont les items sont des concepts provenant de plusieurs attributs d'une relation [Zhu98]. Par exemple, des règles possibles pour la relation Lycéen(filière, note, bac) sont :

(R1) note=moyen+ → Bac=oui

(R2) note=passable et filière=scientifique → Bac=oui

L'indice de confiance d'une règle [Ber97], c'est à dire la fréquence de la conclusion conditionnellement à la prémisse, est la mesure majoritairement utilisée pour rechercher des règles d'association inter-dimension intéressantes. Cet indice présente l'inconvénient de ne pas tenir compte du nombre d'enregistrements, alors que le bon sens nous incite à accorder une confiance plus grande à une règle vérifiée 300 fois sur 400 qu'à une règle vérifiée 3 fois sur 4 (ces règles ont un même indice de confiance valant 0.75). C'est pourquoi nous utilisons aussi dans notre prototype un indicateur allant dans ce sens. Cet indicateur, qui repose sur un modèle statistique de « l'implication non stricte » [Gra96], est appelé « intensité d'implication de x sur y » et mesure la confiance qu'il faut accorder à la règle « Si X=x Alors Y=y » :

$$\varphi(x, \bar{y}) = \frac{1}{\sqrt{2\pi}} \int_0^{\frac{t^2}{2}} e^{-t^2/2} dt \quad \text{où} \quad q(x, \bar{y}) = \frac{(\text{card}_x - \text{card}_{xy}) - \frac{\text{card}_x \times (\text{card} - \text{card}_y)}{\text{card}}}{\sqrt{\frac{\text{card}_x \times (\text{card} - \text{card}_y)}{\text{card}}}}$$

Les divers cardinaux permettant de calculer l'indice de non implication $q(x, \bar{y})$ sont le cardinal de la prémisse (card_x), le cardinal de la conclusion (card_y), le nombre total d'enregistrements (card) et le cardinal de l'intersection de la prémisse avec la conclusion (card_{xy}).

L'intensité d'implication est intégrée dans une méthode interactive et itérative de découverte de règles exploitant les fonctionnalités offertes par les SGBD. Cette méthode, à perfectionner, est fondée sur l'utilisation du langage SQL qui intervient dans plusieurs étapes du processus de data mining : dans la

sélection des données, bien évidemment, mais aussi dans l'algorithme de découverte et l'exploitation des résultats. Les données de l'algorithme sont :

- une table R contenant les données à analyser ; cette table peut être obtenue avec le langage de requêtes présenté dans la section 4.
- les attributs de la table R sur lesquels portent la recherche de règles ; l'utilisateur doit spécifier les attributs conditionnels (notés $X_1, X_2 \dots X_n$) et l'attribut décisionnel (noté Y) ; il doit également fixer un niveau de discours pour chaque attribut associé à une hiérarchie.

Les règles découvertes sont ajoutées à la relation Règles(if, then, confiance, phi...), qui stocke toutes les règles dans une unique relation indépendante des attributs, de leur nombre et de leur niveau de généralisation. L'intérêt majeur d'une table unique est de faciliter le développement d'un formulaire de consultation des diverses règles obtenues.

Les principales étapes de l'algorithme sont les suivantes :

1. Calcul de l'effectif (card_y) de chaque modalité de l'attribut décisionnel à l'aide d'une requête SQL.
2. Calcul de l'effectif total card.
3. Création et exécution d'une requête pour chaque modalité(y) de l'attribut décisionnel

Insert into Règles(if, then, confiance, phi...)

```
Select 'X1=' & X1 & ' and ' & ... & ' and ' & 'Xn=' & Xn,      /* la prémisse « if » */
       'Y=y',          /* la conclusion « then » */
       Sum(Egal01(Y,"y")) / count(*),      /* l'indice de confiance */
       Phi ( sum(Egal01(Y,"y")), count(*), card_y, card ) /* l'indice d'implication */
...

```

From R

Group by X1 ... Xn

Les requêtes d'insertion utilisent deux fonctions personnalisées : Egal01 et Phi. La fonction Egal01 compare ses deux paramètres et renvoie la valeur 0 (paramètres distincts) ou 1 (paramètres égaux) : elle permet de calculer card_xy. La fonction Phi(card_xy, card_x, card_y, card), quant à elle, calcule le degré d'implication. Il est à noter que la syntaxe utilisée ici est adaptée à Access (opérateur de concaténation & et utilisation de guillemets dans la fonction Egal01).

L'algorithme présente l'avantage majeur d'être facile à mettre en œuvre dans un SGBD relationnel. Son principal défaut, dans sa version actuelle, est de devoir être répété trop souvent pour répondre à certaines interrogations de l'utilisateur. Supposons par exemple que l'utilisateur étudie la réussite au bac en fonction de la moyenne générale obtenue au lycée et en fonction de la filière. L'algorithme doit alors être appliqué trois fois : une première avec filière et note comme prémisses, une deuxième avec filière comme prémisse et une troisième avec note comme prémisse. Le nombre de répétitions est encore plus grand si l'utilisateur change de niveau de discours d'un des attributs. Or, il est clair que ces différentes exécutions de l'algorithme ne devraient pas être effectuées indépendamment les unes des autres. Un autre point à étudier est l'apport effectif de l'indice d'implication sur la pertinence des règles obtenues.

Références

- [Ana97] S.S. Anand, B. W. Scotney, M. G. Tan, S. I. McClean, D. A. Bell, J.G. Hughes, "Designing a Kernel for Data Mining", *IEEE Expert*, Vol.12, No 2, March-April.1997, pp.65-74.
- [Ber97] M.J.A. Berry, G. Linoff, "Data Mining : Techniques appliquées au marketing, à la vente et au service client", Ed. InterEdition, 1997.
- [Bin97] R. Bina, L. Licéaga. "SQL flou et bases de données relationnelles", *Dess ingénierie mathématique et outils informatiques*, Université de Pau, France, 1997.

- [Bos95] P. Bosc, O. Pivert, "SQLf : A Relational Database Language for Fuzzy Querying", *IEEE Trans. On Fuzzy systems*, Vol. 3, No 1, Feb.1996, pp.1-17
- [Bou95] B. Bouchon-Meunier, " *La logique floue et ses applications* ", Ed. Addison Wesley, 1995.
- [Bra96] R. Brachman, T. Anand, "The Process of Knowledge Discovery in Databases", in *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996, pp.37-57.
- [Caz98] S. Cazaubon, N. Gieudes. "Contribution à la réalisation d'un environnement de data mining adapté au domaine d'intérêt de l'utilisateur", *Dess ingénierie mathématique et outils informatiques*, Université de Pau, France, 1998.
- [Cox97] E. Cox, " *La logique floue pour les affaires et l'industrie* ", Inter. Thomson Publisher, Ed., 1997.
- [Dha93] V. Dhar, A. Tuzhilin, "Abstract-Driven Pattern Discovery in Databases", *IEEE Trans. Knowledge and Data Engineering*, Vol. 5, No 6, Dec.1995, pp.926-938.
- [Est98] M. Ester, R. Wittmann. "Incremental Generalization for Mining in a Data Warehousing Environment", *Proc. Of Int. Conf. On Extending Database Technology*, Valencia, Spain, 1998.
- [Fay96a] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "Knowledge Discovery and Data Mining : Towards a Unifying Framework", in *Proc. KDD-96: Second International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA: AAAI Press, 1996, pp. 82-88.
- [Fay96b] U. Fayyad, "Data Mining and Knowledge Discovery: Making Sense Out of Data", *IEEE Expert*, Vol.11, No 5, Oct.1996, pp.20-25.
- [Gra96] R. Gras, " *L'implication statistique* ", Ed. La Pensée Sauvage, 1996.
- [Han93] J. Han, Y. Cai, N. Cercone. "Data-Driven Discovery of Quantitative Rules in Relational Databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No 1, Feb.1993, pages 29-40.
- [Han96a] J. Han, Y. Fu, K. Kopersky, W. Wang, O. Zaiane. "DMQL : A Data Mining Query Language for Relational Databases", *Proc. of 1996 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, June 1996.
- [Han96b] J. Han, Y. Fu, "Attribute-Oriented Induction in Data Mining", *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996, pp.399-421.
- [Han97] J. Han, S. Chee, J.Y. Chiang. "DBMiner: A system for Data Mining in Relational Databases and Data Ware Houses", *Proc. CASCON'97: Meeting of Minds*, Toronto, Canada, November, 1997.
- [Han98] J. Han, S. Chee, J.Y. Chiang. "Issues for On-Line Analytical Mining of Data Warehouses", *Proc. of 1996 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Seattle, Washington, June 1998.
- [Imi96a] T. Imielinski, "DataMine : Application Programming Interface and Query Language for Database Mining", in *Proc. KDD-96 : Second International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA : AAAI Press, 1996, pp.256-260.
- [Imi96b] T. Imielinski, H. Mannila, "A database perspective on knowledge discovery", *Communication of ACM*, Vol. 39, No. 11, Nov. 1996, pp. 58-64.
- [Kod96] Y. Kodratoff, "L'extraction de connaissances à partir des données : un nouveau sujet pour la recherche scientifique", in *Actes du Congrès INFORSID '96*, Bordeaux, Juin 1996, pp. 3-22.
- [Lu98] Y. Lu. "Concept hierarchies in Data Mining : Specification, Generation and Application", *M. Sc. Thesis*, Simon Fraser University, Canada, Jan 98.
- [Meo96] R. Meo, G. Psaila, S. Ceri, "A New SQL-like Operator for Mining Association Rules", in *Proceedings of the 22nd VLDB Conference Mumbai(Bombay), India, 1996*
- [Sri95] R. Srikant, R. Agrawal. "Mining generalized association rules", *Proc. 1995 Int. Conf. VLDB*, pages 407-419, Zurich, Sept 1995.
- [Ton95] J.R. Tong, " *La logique floue* ", Ed.Hermes, 1995.
- [Zhu98] H. Zhu. "On-Line Analytical Mining of Association Rules", *M. Sc. Thesis*, Simon Fraser University, Canada, Dec 98.