SIPINA_W© : Une plate-forme logicielle pour l'Extraction de Connaissances à partir de Données

D.A. Zighed et R. Rakotomalala
Laboratoire ERIC
Université Lumière Lyon 2

Résumé

Nous présentons succintement une plate-forme logicielle d'Ingénierie de Connaissances qui reprend pas à pas la démarche générale de l'Extraction de Connaissances à partir de Données. Les différents modules sont brièvement explicités, nous nous attacherons surtout à décrire les différents cadres de leur utilisation.

1 Introduction

Un des principaux attraits de l'Extraction de Connaissances à partir de Données est la possibilité d'expérimenter différents scénarios d'exploitation des données sans que cela ne soit excessivement coûteux. Pour ce faire, il est nécessaire de disposer d'outils puissants qui permettent à la fois produire des connaissances, et de tester différentes stratégies afin de choisir celle qui semble le plus approprié. Notre équipe a essayé de répondre à ces deux motivations en proposant un système complet d'ingénierie des connaissances.

Le premier objectif est donc bien de servir de base de travail pour l'évaluation et la caractérisation des algorithmes d'induction. Quelques auteurs dans la littérature ont adopté la même démarche en proposant des systèmes complets de constructions d'inductions par graphes, les plus célèbres sont l'oeuvre d'auteurs illustres tels que [BC91] avec le système IND, ou encore [KJL+94] avec la bibliothèque MLC++. La plupart de ces outils sont conçus pour fonctionner sur des stations de travail, avec le système d'exploitation UNIX. Même si l'on dispose du code source, l'absence d'interface graphique les confine à une exploitation universitaire, idéale pour les chercheurs mais impraticable pour un utilisateur non averti.

Le second objectif est de créer un produit suffisamment fini pour que sa mise en oeuvre dans le milieu des entreprises (industriel, médical, administration...) soit possible sans aucune adaptation. Dans cette optique, le choix d'un progiciel fonctionnant sous le système Windows® n'est pas innocent. Malgré toutes les réserves que l'on peut émettre sur l'efficacité de ce système d'exploitation, il est certainement le plus répandu dans le monde des entreprises depuis l'explosion de la micro-informatique et la diffusion de plus en plus étendue des ordinateurs compatibles PC. Certes il existe de nombreux logiciels commerciaux fondés sur le paradigme des arbres de décision, nous nous en démarquons cependant car, plutôt que de proposer un outil confiné à la description ou à la structuration simple de données, nous avons utilisé explicitement le cadre général de l'extraction de connaissances à partir des données[1] lors de la spécification de l'architecture du logiciel. Nos repères sont principalement : l'acquisition et la sélection des données, le pré-traitement des observations pour l'extraction de connaissances proprement dite réalisée à l'aide des algorithmes d'induction par graphes, l'évaluation statistique et empirique des formes extraites, et la mise en forme des connaissances en vue de leur exploitation soit pour l'explication des formes découvertes, soit pour la généralisation dans la population étudiée.

1. Knowledge Discovery in Databases

Le logiciel SIPINA_W© est issu d'une longue évolution sous la direction et l'instigation du Pr. Zighed. La première version, écrite en Pascal sous DOS, prenait appui sur les résultats théoriques mis en exergue dans les travaux de [Zig85]. Le système à l'époque ne prenait en compte que les attributs catégoriels. Un premier saut qualitatif fût accompli avec la version livrée à l'intérieur de l'ouvrage "SIPINA : Méthode et logiciels" de [ZAD92]. Le logiciel pouvait traiter les attributs continus, que l'on rencontre souvent dans les problèmes réels, en introduisant le processus de discrétisation. A cette époque, compilé sous le système DOS, il fonctionnait en mode réel avec une limite de taille mémoire à 640 Ko, insuffisante pour traiter les grosses bases de données, courantes dans les applications d'ECD. Une seconde étape importante a été le passage sous le système Windows®. A partir d'une plate-forme générique d'induction de graphes comprenant un éditeur de données et une interface graphique, nous avons principalement travaillé sur trois axes : greffer les autres méthodes d'induction par graphes (C4.5 [Qui93], CART [BFOS84]...) ; mettre en place les dispositifs nécessaires aux fins de comparaisons, d'évaluation et de sélection de modèles (cross-validation, bootstrap, apprentissage répété....) ; intégration des derniers développements issus de la recherche, de notre propre équipe (extraction sélective des règles par validation [RC96], détection automatique de taille minimale des sommets [RZR96], discrétisation [ZRF97]...) ou issus de travaux d'autres chercheurs (agrégation de classifieurs [Bre96], simplification des règles [Qui93]...).

La diffusion du logiciel SIPINA_W© sur Internet nous a permis de l'améliorer constamment. Nous avons essayé de mettre à profit les critiques, remarques et suggestions en provenance d'utilisateurs du monde entier, pour la plupart d'origine anglo-saxonne, oeuvrant dans des domaines aussi divers que la recherche en apprentissage, la médecine, la géologie, ou encore l'assurance.

Dans ce qui suit, nous présentons l'architecture globale du logiciel SIPINA_W©. Nous mettrons en exergue les différents modules constitutifs qui s'inscrivent dans le cadre de l'extraction automatique de données par apprentissage supervisé. Bien entendu, rien n'est définitif, la plate-forme actuelle subit constamment des modifications dans le sens d'une amélioration des performances et de la souplesse de l'outil. Plutôt que de fournir une description approfondie des options associées aux nombreux menus du logiciel, disponible dans le guide de l'utilisateur [ZR96b], nous préférons présenter son esprit et les réalisations afférentes, passées ou parfois à venir.

2 Implémentation et élaboration de SIPINA_W©

Les exigences initiales à l'origine de la construction du logiciel d'induction par graphes sous l'environnement Windows® ont été les suivantes :

1. une interface graphique aux normes Windows® (figure 1) ;
2. une visualisation en temps réel de l'induction avec possibilité pour l'expert d'agir de manière interactive pour modifier la configuration du graphe, soit en choisissant les variables de segmentation, soit en choisissant une opération particulière (fusion, segmentation, limitation du nombre de noeuds). En ce sens, il est important de proposer de nombreux outils d'aide à la décision ;
3. des capacités de traitement élevées. Théoriquement, il est de $(2^{32}-1)$ observations et 16384 variables, dans la réalité la taille de la mémoire virtuelle[2] est la limite que l'on ne peut dépasser.

Le langage d'implémentation est le Pascal Objet qui prend appui sur l'interface de programmation visuelle DELPHI de Borland, de fait le code relatif à la gestion de l'interface est réduit au minimum. A l'heure actuelle, SIPINA_W© comporte près de 50000 lignes de codes, la mise à jour du logiciel par intégration de nouvelles fonctionnalités et/ou méthodes est relativement aisée. Le prochain enjeu est maintenant le passage de la version 16 bits à une version 32 bits, exploitant directement les nouvelles possibilités des systèmes d'exploitation 32 bits. Notons que la plate-forme actuelle fonctionne sur n'importe quel système Windows® (3.1, 3.11, 95, NT) pourvu qu'il tourne en mode étendu.

2. Elle dépend à la fois de la mémoire RAM et de l'espace disque disponible.

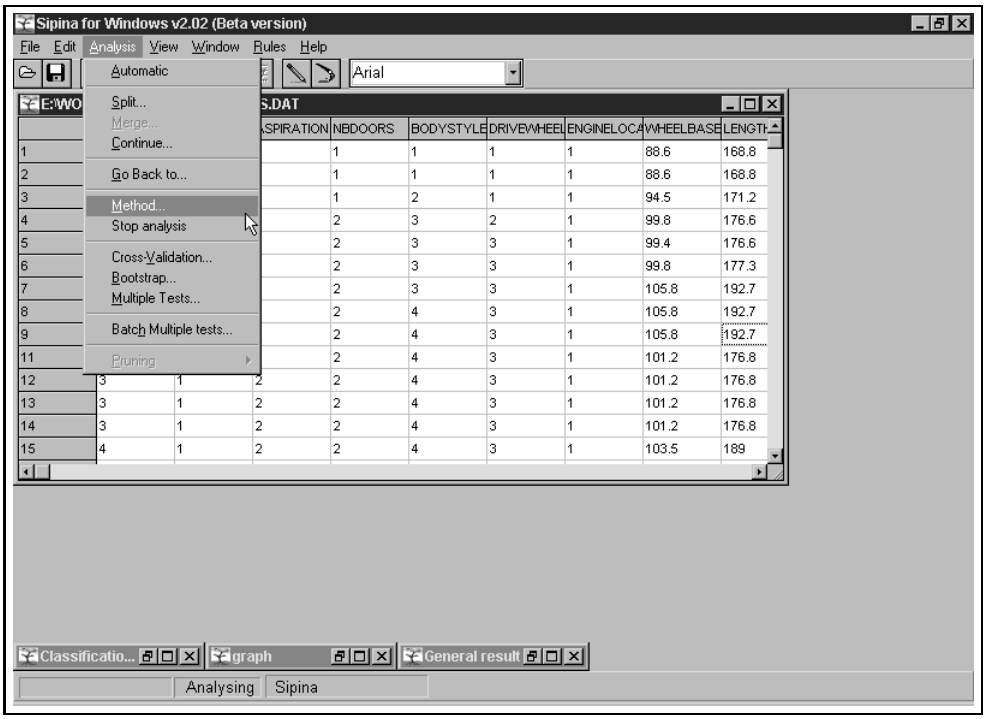

FIG. 1: *Interface de base du logiciel SIPINA-W avec son menu principal*

3 Architecture globale de SIPINA_W© dans le cadre de l'ECD

Notre propos est donc de développer un système complet incluant les étapes de l'extraction automatique de connaissances à partir de données qui permettront à l'utilisateur final : d'une part, d'analyser les causalités mises à jour; d'autre part, de proposer un diagnostic aussi fiable que possible, ou du moins dont on connaît la propension à l'erreur. Les modules composant le logiciel sont résumés dans le graphique 2. Dans les sections suivantes, nous les aborderons tour à tour.

3.1 Mise en forme des données

L'acquisition des données comporte une première phase qui consiste à circonscrire au mieux le domaine d'étude afin que les individus étudiés soient aussi homogènes que possible, ce travail revient essentiellement à l'expert. L'étape suivante est l'extraction d'une collection d'observations : dans le cadre de l'ECD elle a souvent pour origine une requête sur un serveur de données, dans le cas général il peut provenir de n'importe quelle source. En tous les cas, nous devons disposer d'une matrice de données avec en ligne les observations, en colonne les attributs qui les décrivent, y compris la variable à prédire.

Le format de fichier SIPINA_W© est un format propriétaire hérité des précédentes versions du logiciel, afin d'en augmenter la souplesse, nous avons intégré des convertisseurs qui permettent d'importer d'autres formats de fichiers (figure 3) dont les principaux sont : tableurs (Lotus), base de données (Paradox, dBase), éditeurs de texte (fichiers texte avec séparateurs), ou encore fichiers au format C4.5 que l'on peut trouver sur certains serveurs de données [MA95]. Nous sommes en train actuellement de travailler sur les opportunités de produire des données tabulaires directement à partir de vues produites par des requêtes sur des serveurs de base de données. Compte tenu de la puissance de l'environnement de programmation Delphi, il est à prévoir que l'intégration de cette option sera relativement aisée. Bien entendu, le pendant de l'importation, l'exportation, a été également implémenté.

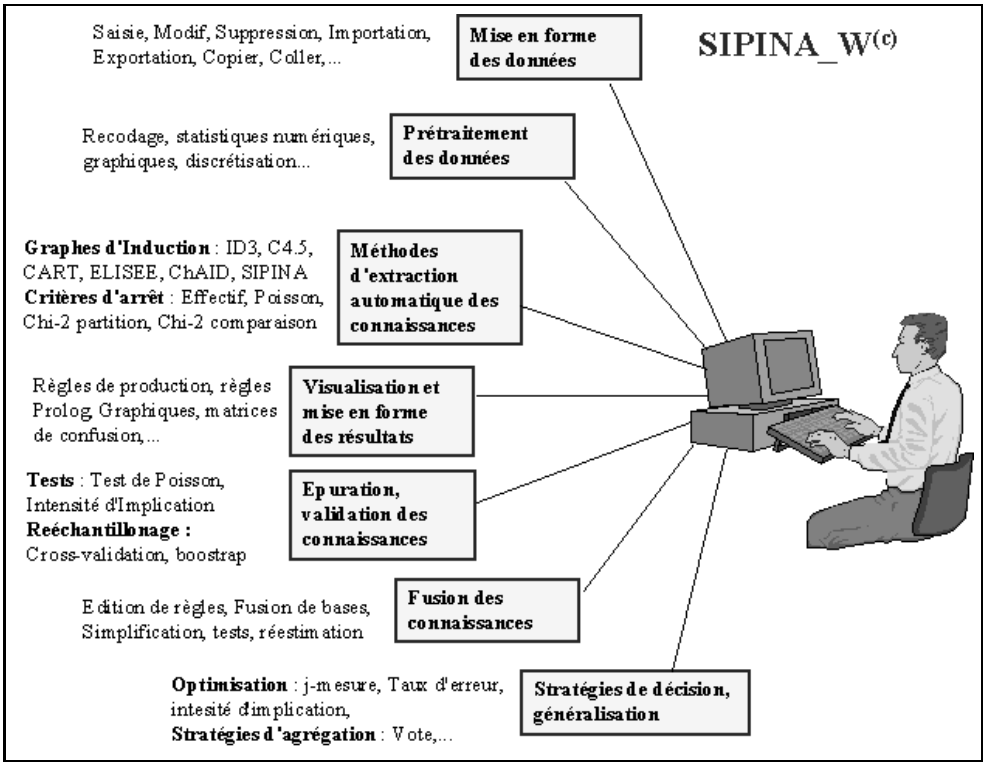

FIG. 2: *Architecture du logiciel SIPINA-W(c)*

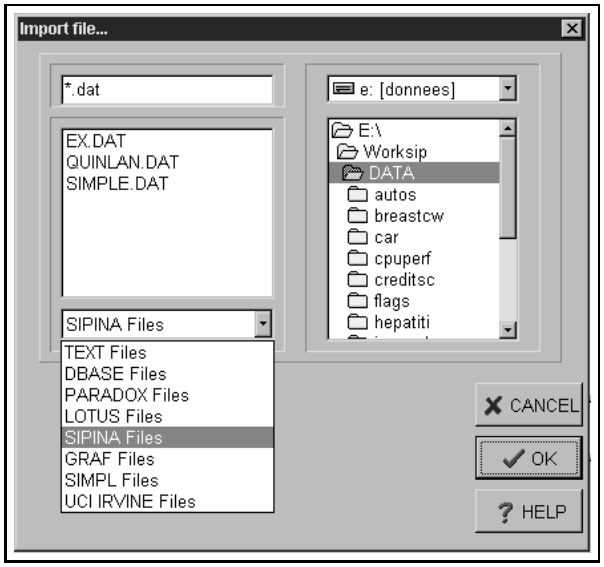

FIG. 3: *Boîte de dialogue d'importation de données dans DataManager*

3.2 Pré-traitement des données

C'est une étape fondamentale de l'ECD. Souvent les données sont entâchées d'imperfections (données manquantes, attributs non-pertinents...). De plus, certaines méthodes d'extraction de connaissances exigent qu'elles soient présentées d'une manière particulière (données de type continu et distribution normale pour l'analyse discriminante, données de type catégoriel pour les algorithmes symboliques...). Il importe alors de développer des stratégies qui nous mènent vers la meilleure adéquation entre l'algorithme d'apprentissage et les observations en entrée tout en contrôlant les approximations introduites par ce pré-traitement.

Certaines des fonctions présentées dans cette sous-section sont en cours de développement, nous nous contenterons donc de décrire les principes qui sous-tendent notre travail dans les différents cas de figure.

3.2.1 Sélection des individus

Avec l'augmentation de la capacité de stockage des machines, la disponibilité des observations n'est plus un problème, c'est la situation inverse qui semble maintenant prédominer : il y a trop de données disponibles. Dans ce contexte, les réactions sont diverses :

- développer des algorithmes spécifiques extrêmement rapides [Cat91];
- développer des approches adaptées aux machines parallèles [HKS96];
- développer enfin des techniques d'échantillonnage (ou de fenêtrage) qui permettent d'extraire le maximum d'informations avec le minimum de données [Qui93].

Pour ou contre l'échantillonnage, les avis sont partagés. Il est vrai que dans certains cas, les phénomènes sont suffisamment rares pour que son appréhension ne soit possible qu'à travers des bases de grande taille. Si nous considérons le cas des fraudes bancaires [FMdB96], il y a à peu près une transaction frauduleuse pour 10^5 opérations. Si l'on procède à un échantillonage aléatoire simple de 1000 individus, il y a de grandes chances que la classe "Fraude" ne soit tout simplement pas représentée.

Malgré tout, il nous semble vain d'essayer de traiter entièrement les grosses bases de données. Quelle que soit l'augmentation de la puissance des machines, il est peu probable qu'elle suive la croissance de la masse de données, il est sans aucun doute plus profitable de mettre en oeuvre d'autres schémas de tirage tout en adaptant les algorithmes d'apprentissage en conséquence. Fort heureusement, les graphes sont équipés pour répondre de manière adéquate à ce type de problème [BFOS84], notre rôle consistera alors à implémenter les différentes méthodes d'échantillonnage évoluées (stratifiés, méthode des quotas...) qui répondent exactement à la problématique de l'utilisateur.

3.2.2 Sélection des variables

Les graphes d'induction sont réputés pour sélectionner uniquement les variables pertinentes en induction grâce aux mesures d'évaluation des segmentations. Ceci est moins vrai lorsqu'il s'agit de stratégies d'induction par arbres qui utilisent le principe de l'élagage. La construction est dite "hurdling", une variable non-pertinente est introduite même si elle n'apporte pas significativement d'informations; l'objectif sous-jacent est bien sûr, comme on a pu le montrer dans cette thèse, l'appréhension des concepts pour lesquels certaines variables inter-agissent.

Dans cette optique, la sélection des variables devient cruciale car elle évite l'insertion d'attributs non-pertinents dans le classifieur. Ces dernières années de nombreux auteurs se sont attentivement penchés sur ce problème. Deux écoles s'affrontent : ceux qui préconisent la sélection des variables hors algorithme d'apprentissage [AD91] [RSG93], et ceux qui préconisent d'utiliser explicitement l'algorithme d'apprentissage [KJ97] [CS96].

La seconde méthode est peut-être plus efficace, mais elle est très coûteuse en temps de calcul. En effet relancer plusieurs fois l'algorithme d'apprentissage pour sélectionner le bon ensemble de variables entraîne

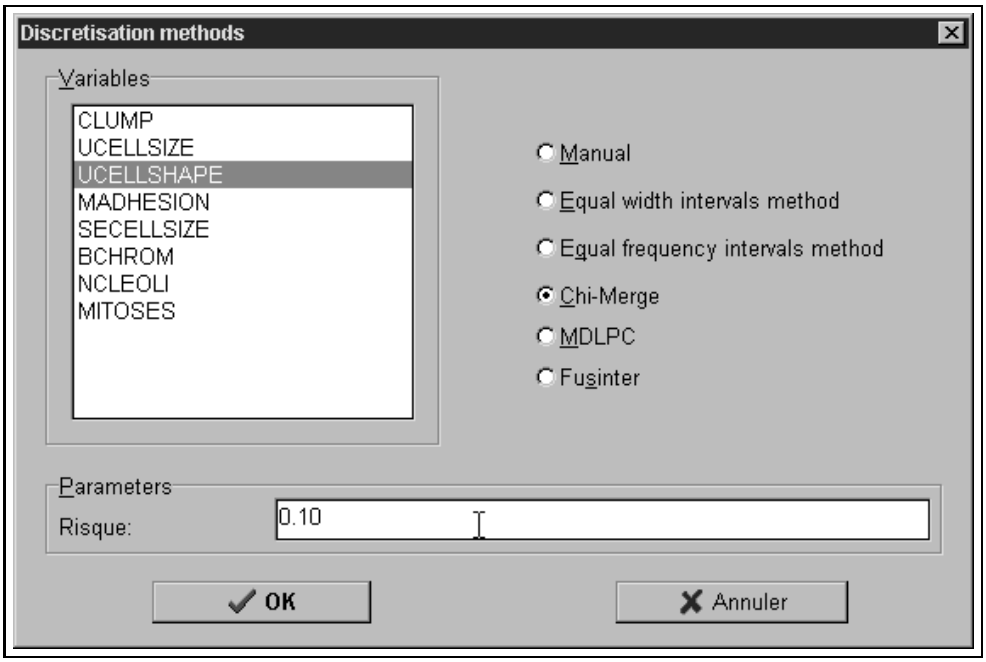

FIG. 4: *Boîte de dialogue de sélection des méthodes de discrétisation dans une phase de recodage des attributs continus*

une multiplication des coûts qui peut s'avérer préjudiciable. De plus, cette méthode ne s'inscrit pas dans le cadre de notre propos ici.

La première méthode, une sélection ad-hoc des variables avant l'application de l'algorithme, paraît cependant discutable en reconnaissance de formes. Rien ne laisse présager qu'une variable sélectionnée d'une manière quelconque puisse être pertinent par la suite dans un algorithme d'apprentissage s'appuyant sur un système de représentation des connaissances particulier [Ima94]. Il reste néanmoins dans la pratique que la réduction de la dimension est nécessaire tant les graphes d'induction peuvent être sensibles aux attributs non-pertinents dans certains cas, notamment lorsqu'il y a interaction entre plusieurs variables [LS94].

3.2.3 Recodage et transformation des données

Ces transformations sont parfois considérées comme des méthodes d'appauvrissement ou d'enrichissement des données, dans ce dernier cas l'opération est rendue possible par l'adjonction d'hypothèses supplémentaires. Les principaux objectifs sont l'homogénéisation des données et l'adjonction de nouvelles propriétés de manière à pouvoir appliquer directement les différents algorithmes d'apprentissage.

Selon la nature des données, nous distinguons différentes transformations que nous avons implémentées dans SIPINA_W© (figure 4):

1. attributs catégoriels: une technique fréquemment utilisée est le codage disjonctif complet, en fait nous retrouvons ici les méthodes introduites dans la partie sur le regroupement des valeurs des attributs.

2. attributs continus: dans le cadre de l'induction par graphes, une transformation qui nous tient à coeur est la discrétisation des attributs, dans la phase préparatoire seul le découpage en intervalles multiples (soit fixé par l'utilisateur, soit déterminé par la méthode) est véritablement justifié.

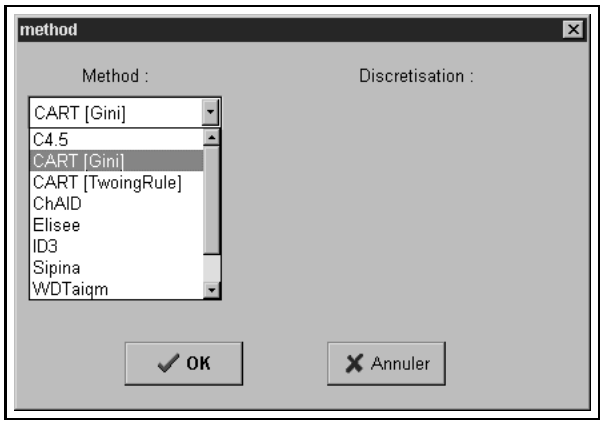

FIG. 5: *Sélection des méthodes dans SIPINA-W(c)*

3.2.4 Traitement des données manquantes

Le problème des données manquantes peut être considéré sous différents angles dans le classement par graphes. Le premier, que nous n'aborderons pas ici, se situe dans la phase de généralisation [Qui87] [Bru92]. Le second survient en apprentissage, un ou plusieurs individus possèdent des attributs non référencés.

La solution la plus simple consiste à supprimer les observations correspondantes. Il est clair qu'une telle politique n'est pas très appropriée dans le traitement sur bases réelles. L'exclusion d'individus occasionne des pertes d'informations, d'autant plus regrettables que parfois les observations manquantes portent sur des variables non discriminantes[3] en apprentissage. Les autres solutions, autrement plus efficaces, sont regroupées en deux familles distinctes :

- soit nous traitons directement l'absence de données dans l'induction, [Qui89] recense les différentes méthodes utilisées en ce sens, l'objectif étant de construire le classifieur le plus performant en généralisation;
- soit nous cherchons à reconstruire les données originelles par différentes méthodes mono ou multivariées. Pour les données continues nous pouvons envisager l'utilisation de la moyenne, la médiane, ou encore la régression; pour les données catégorielles, nous pouvons nous servir de la valeur la plus fréquente, de l'analyse bivariée sur un tableau de contingence, ou encore de la méthode des plus proches voisins. L'objectif dans ce cas est de retrouver la bonne valeur relative à l'individu concerné.

Nous penchons pour le deuxième type d'approche, elle est générale à tout type de processus d'apprentissage. Nous travaillons actuellement à implémenter les stratégies appropriées afin de restituer les valeurs originelles des cases vides dans les matrices de données.

3.3 Extraction de connaissances

C'est le point d'orgue du logiciel. Nous nous sommes attachés à comprendre et à tester les différentes méthodes rapportées dans la littérature. Finalement, les algorithmes d'induction par graphes diffèrent peu dans leur principe, à partir d'une base simple intégrant la segmentation et le regroupement de noeuds, nous avons pu couvrir l'ensemble des stratégies. Les véritables points de différence sont la recherche de la taille optimale qui peut être réalisée grâce à l'adoption d'une règle d'arrêt (les différents critères peuvent être mis en oeuvre sur n'importe quel algorithme) ou grâce à l'élagage (par optimisation ou par estimation du taux d'erreur).

Dans la plate-forme SIPINA_W©, ces algorithmes sont accessibles via une boîte de dialogue où l'on peut choisir la méthode appropriée (figure 5). Pour la méthode d'induction de graphes Sipina [ZR96a],

3. cet argument milite encore en faveur de la sélection des variables comme pré-traitement

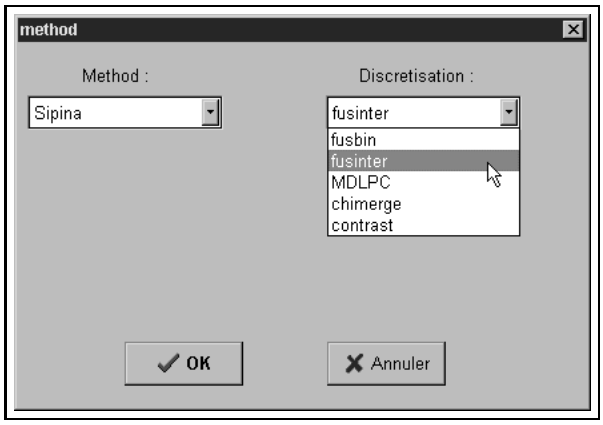

FIG. 6: *Sélection de la discrétisation locale dans SIPINA-W(c)*

nous disposons d'une pléiade de stratégies de discrétisation binaire ou n-aire (figure 6). Avec ces différentes combinaisons, nous pouvons tester plusieurs possibilités pour choisir la méthode la plus appropriée sur un domaine donné, en utilisant les outils de comparaisons que nous détaillerons plus loin.

Aux fins de vérifications et de comparaisons, nous avons beaucoup travaillé sur une implémentation fidèle des algorithmes d'induction par graphes disponibles dans la littérature. Nous détaillons dans cette section les stratégies que nous avons implémentées, nous insisterons surtout sur les références bibliographiques utilisées.

3.3.1 Les méthodes d'induction par graphes implémentées dans SIPINA_W©

Ce sont les méthodes de base, comportant des heuristiques propriétaires. Certaines sont des références incontournables, nous les avons testées sur les bases exemples utilisées dans cette thèse (voir Annexes) :

- **C4.5** : on peut trouver sur le site WEB de l'auteur les codes sources des dernières améliorations de cette méthode qui figure parmi les références reconnues de la littérature. Nous nous en sommes tenus aux procédures décrites dans l'ouvrage de [Qui93]. Notons qu'il existe actuellement une version commerciale C5.0 intégrant les derniers développements de l'auteur sus-cité.

- **CART** : deuxième incontournable de la littérature, nous avons programmé les deux versions de cette approche [BFOS84]. Notons que notre implémentation de la méthode *Twoing* ne prend en compte que les variables continues, celle de la *Gini* est une extension n-aire.

- **ChAID** : c'est une transcription de la méthode de [Kas80], on peut la retrouver dans des logiciels commerciaux tels que Knowledge Seeker© d'Angoss Software.

- **Elisee** : très proche de la méthode Twoing de CART, elle utilise une distance du χ^2 et produit des arbres binaires. Cette méthode, telle que nous l'avons programmée, ne fonctionne correctement que sur des attributs prédictifs continus. Le papier de référence est celui de [BT70].

- **ID3** : c'est la méthode certainement la plus citée dans la littérature en intelligence artificielle, nous nous sommes servis de la stratégie de base [Qui79], couplée avec la règle d'arrêt basée sur un test d'indépendance nous retrouvons l'algorithme décrit dans [Qui86].

- **SIPINA** : c'était notre point de départ, la stratégie pour laquelle nous avons développé la plateforme. Avec le recul, nous nous rendons compte que ce fût un choix heureux car étant une généralisation des arbres elle englobe ainsi quasiment toutes les méthodes d'induction par graphes. Les documents de référence sont [Zig85], [ZAD92] et [ZR96a]. Notons que nous pouvons démultiplier les

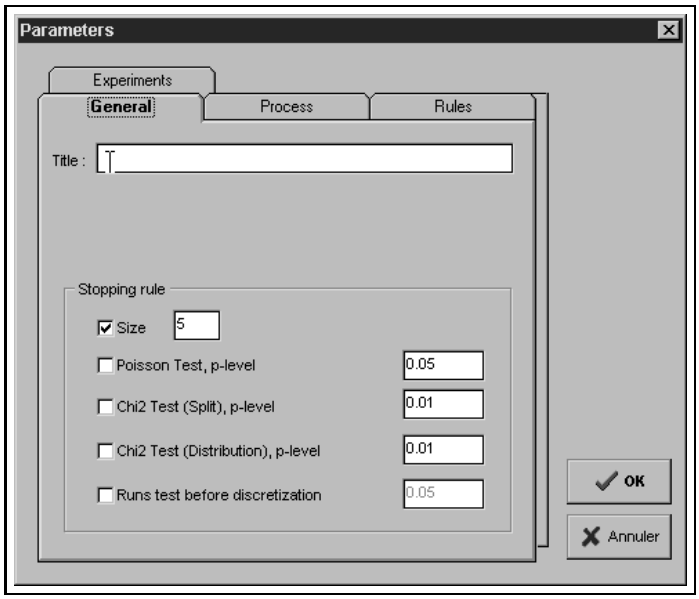

FIG. 7: *Options de règles d'arrêt dans la plate-forme SIPINA-W(c)*

variantes en choisissant des procédures de discrétisation différentes : FUSINTER [4] [ZRR96], MDLPC [FI93], CHI-MERGE [Ker92] et CONTRAST [dM93]. A notre connaissance, SIPINA_W© est la seule plate-forme diffusée, avec le package IND de [BC91] qui implémente les graphes de [OW91], qui propose la généralisation des arbres de décision aux graphes d'induction.

- **WDTaiqm** : c'est une transcription fidèle du papier de [Weh93] qui, depuis un certain nombre d'années s'intéresse de près à l'analyse de la stabilité transitoire des réseaux électriques [WP91].
- **QR_MDL** : elle s'appuie sur la théorie de la description minimale décrite dans [QR89], nous avons utilisé ici l'option arbre n-aire.
- **PRETree** : cette stratégie induit des arbres de décision en utilisant l'interprétation en terme de régression de l'induction par graphes [RZ97].

3.3.2 Les règles d'arrêt d'expansion du graphe

Lors de la phase d'expansion du graphe, diverses règles d'arrêt peuvent limiter la taille du classifieur. Certaines sont classiques, que l'on retrouve dans des algorithmes standards, d'autres en revanche sont ad-hoc et correspondent à des tentatives propres à notre travail de recherche. Voici la liste de règles d'arrêt disponibles dans la plate-forme SIPINA_W© (figure 7) :

- **taille d'un sommet** : comme condition d'acceptabilité elle permet de contrôler les sommets produits dans le graphe i.e si un au moins des sommets enfants produits par un éclatement comporte un effectif inférieur à cette limite, la partition est refusée. De fait, il ne peut y avoir de noeuds de taille insuffisante [BFOS84] [ZAD92].
- **le test de Poisson** : c'est un test fondé sur l'intensité d'implication de [LGR81], on arrête le développement sur un noeud dès qu'il est possible d'extraire une classe conclusion au seuil fixé par l'utilisateur.

4. FUSBIN est une variante binaire de FUSINTER

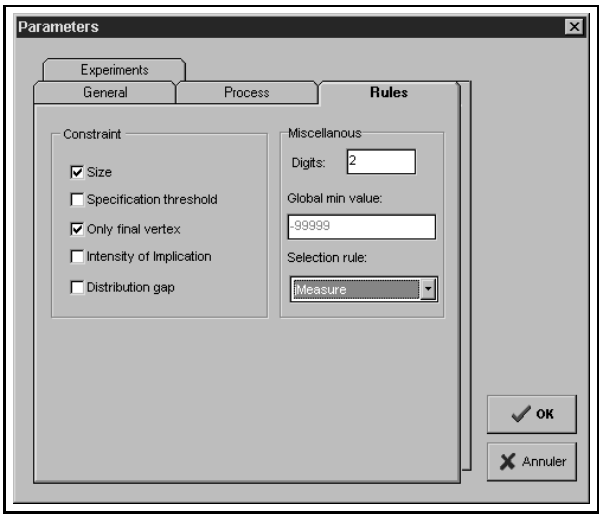

FIG. 8: *Modes de génération des règles dans SIPINA-W(c)*

- **le test d'indépendance du χ^2** : on l'utilise comme condition d'acceptabilité d'un éclatement. Si l'indépendance entre l'attribut prédictif et la variable à prédire n'est pas démentie au seuil fixé, nous refuserons, à l'instar de [Kas80] [Qui86], la segmentation.

- **le test d'équivalence distributionnelle du χ^2** : ce test peut être vu comme une généralisation du test de Poisson ci-dessus. Un noeud est déclaré terminal dès que l'on observe une divergence significative de sa distribution des classes avec celle du sommet initial. On procède de manière analogue dans le premier algorithme CN2 [CN89] pour stopper la spécialisation des règles en induction.

- **test des séquences, préalable à la discrétisation** : pour que l'on discrétise un attribut continu, il faudrait s'assurer de la séparabilité des individus. Nous avons beaucoup travaillé sur l'opportunité de l'introduction de ce test avant le découpage [RRS95] [Rak95], il est apparu que sa puissance est du même niveau qu'une discrétisation classique où l'on refuse de segmenter lorsqu'elle aboutit à la construction d'un seul intervalle. Il reste néanmoins qu'en excluant les variables peu pertinentes, ce test se révèle être un moyen très efficace de réduction des temps de calcul.

3.3.3 Extraction et traitement des règles

Le passage du graphe aux règles est une étape importante de notre système d'apprentissage : d'une part, nous mettons la connaissance sous une forme exploitable, notamment par les systèmes à base de connaissances fonctionnant en Prolog ou utilisant des formats de règles natifs tels que le logiciel SelfMind (Système Expert en Logique Floue et Multi-valuée pour l'Imitation Naturelle des Décisions) développé au sein de notre laboratoire; d'autre part, ce changement de représentation permet l'application de nouveaux opérateurs en vue d'améliorer les performances du classifieur (e.g. simplification, fusion avec d'autres bases de règles...).

Dans cette sous-section, nous détaillerons deux facettes du traitement des règles dans notre plate-forme d'ingénierie des connaissances :

1. stratégies et contraintes d'extraction des règles : les règles extraites du graphe ne sont certainement pas de la même qualité, certaines sont meilleures parce qu'à la fois plus précises et plus générales que d'autres. Sur la base des travaux de [GS88], nous avons mené une étude approfondie de l'influence de l'indicateur de qualité des règles sur les performances du classifieur induit [RZF97]. Les résultats obtenus confirment la viabilité et l'avantage que l'on peut tirer d'un "bon" indicateur dans les caractéristiques de la base de règles construite. Sachant que chaque noeud non-initial du graphe

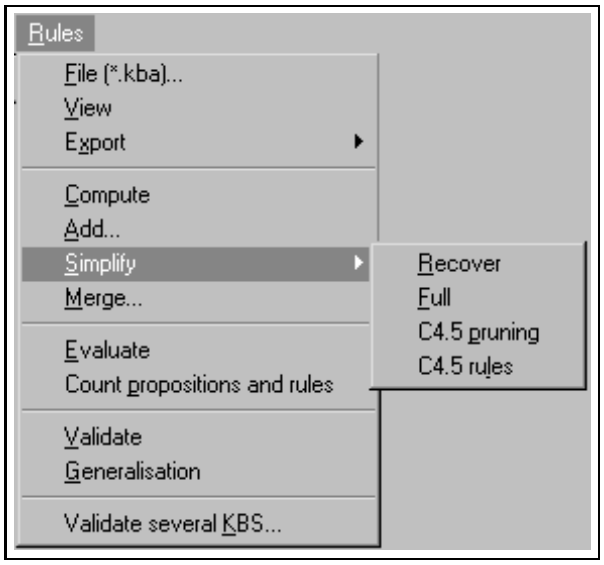

FIG. 9: *Menu "Traitement des règles" dans Sipina-W(c)*

constitue une règle potentielle, il apparaît plus opportun de les valider individuellement quitte à les mettre en concurrence pour définir le meilleur ensemble de règles extrait du graphe [RZR96] [RC96]. L'exclusion des règles redondantes peut alors être facilement résolue par un algorithme de simplification [RRZ96]. Les options de génération de règles disponibles (figure 8) dans notre logiciel sont :

- effectif minimal couvert par la règle : en-deçà d'un certain seuil, on suppose que la règle couvre trop peu d'individus pour qu'elle soit fiable. On ne peut pas dire par exemple que les malgaches sont myopes sous prétexte que l'on en a vu un exemplaire.
- seuil de spécialisation : si le nombre de contre-exemples à la règle est trop élevé, il est clair qu'elle est peu crédible. Le seuil de spécialisation indique la valeur limite du nombre de bon classements en apprentissage à partir duquel nous accordons notre confiance à la règle. Nous remarquerons qu'associés à la contrainte précédente, nous retrouvons réunis les critères de précision (peu de contre-exemples) et de généralité (beaucoup d'exemples couverts). Ce tandem a été utilisé par [Zig85] pour valider les règles dans les graphes bien avant les travaux de [GS88], ces derniers ont avant tout eu le mérite de proposer un indicateur synthétique en se basant sur l'explicitation de ces deux contraintes.
- extraction des règles sur les sommets terminaux : selon que cette option est cochée ou non, les règles seront extraites classiquement des feuilles ou de tous les noeuds non-initiaux [RZ96]. Pour que cette dernière stratégie soit efficace, elle doit être couplée avec l'option de validation et la simplification afin d'éliminer les règles peu fiables et/ou redondantes.
- validation des règles à l'aide de l'intensité d'implication : une règle pourra être validée à l'aide du test de causalité de [LGR81]. On note que cette possibilité rend d'immenses services dans des domaines où, plus que le classement à tout prix, il est préférable de ne pas conclure dès que la décision n'atteint pas un niveau de fiabilité suffisant. C'est le cas par exemple en médecine où les décisions à tort sont d'un coût très élevé.
- écart de distribution : c'est une extension fondée sur le test du χ^2 du test de causalité précédent, on veut s'assurer que les distributions de classes s'écartent suffisamment de la distribution initiale.

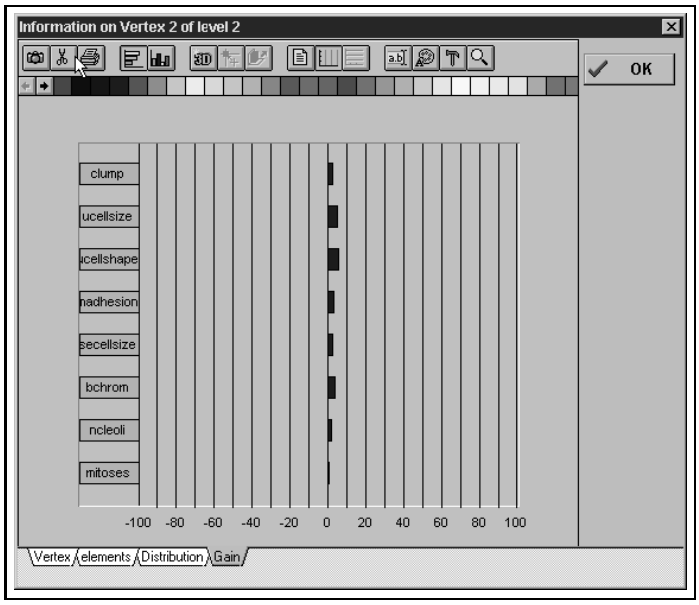

FIG. 10: *"Ucellshape" est la meilleure variable en segmentation, mais on se rend compte que la variable "UcellSize" aurait tout aussi bien induit une partition quasiment de même qualité (au regard du gain d'incertitude)*

2. stratégies de simplification des règles : le traitement des règles tient une place prépondérante dans le logiciel SIPINA_W© (figure 9). Elle peuvent être modifiées, supprimées, ou encore réévaluées sur d'autres fichiers de données... Parmi les différentes opérations possibles, la simplification est assez particulière. Elle répond à deux impératifs : une diminution de la complexité du classifieur (estimée par le nombre moyen de propositions dans les prémisses et/ou le nombre de règles dans la base) et une augmentation concomittante des performances en généralisation. Si le bien fondé de cette dernière propriété fait l'objet d'âpres discussions, tous les chercheurs s'accordent à reconnaître qu'une réduction de la complexité entraîne une meilleure compréhensibilité. Dans notre travail nous nous sommes intéressés à plusieurs types d'algorithmes que nous avons évalués dans le chapitre sur le traitement des règles dans les graphes. Les différentes stratégies de simplifications sont réunies dans le sous-menu "Simplification" visible dans la figure 9.

3.3.4 Construction interactive des graphes

L'élaboration automatique des graphes est un challenge très important, elle constitue un champ de recherche très fertile en Intelligence Artificielle. Mais il est des problèmes où la connaissance du domaine est primordiale, leur intégration dans la construction du classifieur permet de trouver parmi l'ensemble des solutions performantes celles qui sont les meilleures du point de vue de l'expert : cela peut se traduire par un graphe qui explique mieux le phénomène étudié i.e correspondant à un scénario que l'expert peut comprendre et justifier; cela peut également correspondre à un choix de variables discriminantes moins coûteuses ou plus faciles d'accès.

Afin d'offrir à l'expert un outil suffisamment performant, il nous a semblé important de lui proposer un ensemble d'instruments d'aide à la décision. Cette idée n'est pas nouvelle, rappelons que l'acronyme SIPINA [Zig85], méthode originelle à la base de nos développements, veut dire "Système Interactif Pour l'Induction Non-Arborescente". Le fait de travailler sous une interface en mode graphique, le bureau Windows® , permet de matérialiser ces instruments à travers des séries de graphiques, en sus des indicateurs numériques, qui résument de manière synthétique les différentes opportunités au niveau du choix de la variable de segmentation.

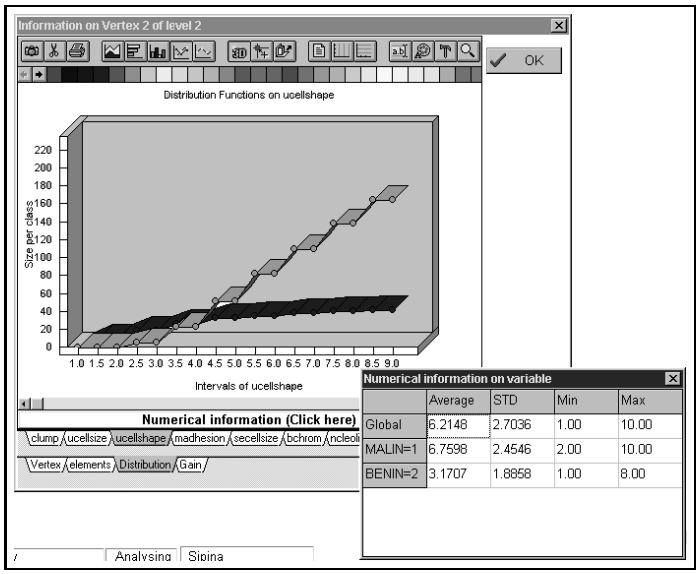

FIG. 11: *Fonction de répartition de l'attribut "UcellShape" conditionnellement aux classes (Malin, Bénin)*

Au sein de la plate-forme SIPINA_W©, nous avons multiplié ces instruments. Il faut avouer que cela fût, entre autres[5], à l'origine du succès de sa diffusion sur Internet en donnant aux néophytes un logiciel avec une interface agréable, presque ludique. Avec le problème du cancer du sein (Breast Cancer Wisconsin [MA95]), nous illustrons dans les graphiques 10 (resp. 11) les segmentations concurrentes (resp. les distributions conditionnellement aux classes de la variable à prédire et les indicateurs statistiques afférents) sur un sommet du graphe.

L'interactivité se manifeste également dans la possibilité qui est offerte aux utilisateurs de choisir l'opération qui leur convient (fusion, segmentation). De plus, ils peuvent à tout moment restructurer le graphe par un mécanisme de retour en arrière où il est possible de transformer un sommet quelconque en une feuille par un clic de souris.

A moins d'un courage et d'une patience hors pair, on voit mal l'application d'une procédure de cross-validation ou de bootstrap dans ce contexte. La validation empirique des modèles d'experts passe nécessairement par un second fichier dit de validation. A cet effet, il est possible de juger la pertinence du modèle, et même individuellement des règles, en l'appliquant sur ce second fichier. Des indicateurs de qualité de la prédiction, notamment une estimation du taux de bon classement en généralisation, sont facilement accessibles.

3.4 Visualisation et mise en forme des résultats

Un des objectifs majeurs de l'apprentissage est l'explication. Ceci éclaire en grande partie le succès des méthodes d'induction de règles face à des classifieurs "boîte noire" plus puissants telles que les réseaux de neurones ou les méthodes des plus proches voisins. Par rapport aux méthodes symboliques classiques issus de l'algorithme AQ [Mic69], les graphes d'induction possèdent l'incommensurable avantage de pouvoir s'exprimer à travers deux systèmes de représentation : la première, très visuelle, est le graphe de décision où l'on peut suivre le processus de prise de décision en traçant les questions successives menant à la conclusion; la seconde, la base de règles, est plus fonctionnelle tout en restant quand même très compréhensible à l'être humain tant que le nombre de règles est raisonnable.

Dans la plate-forme SIPINA_W©, ces deux modes d'expressions du classifieur sont disponibles. Le graphe (figure 12), tout comme la base de règles associée (figure 13), peut être modifié manuellement par

5. Nous espérons que ce n'en est pas l'unique raison quand même

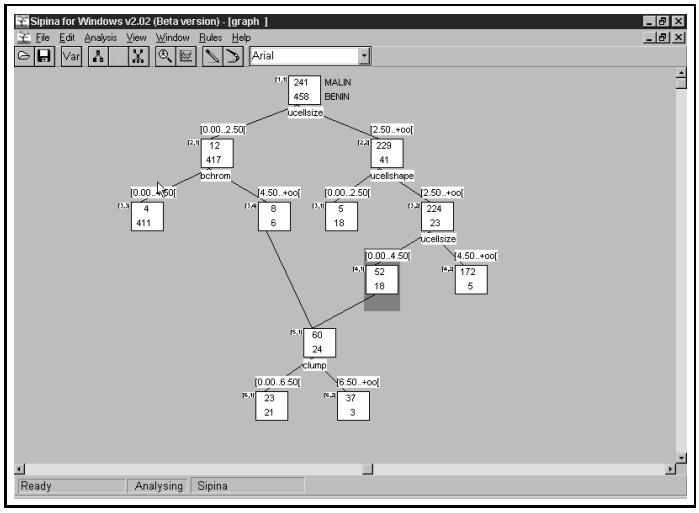

FIG. 12: *Un graphe d'induction construit dans le logiciel Sipina-W(c) sur le fichier des Cancer du Sein*

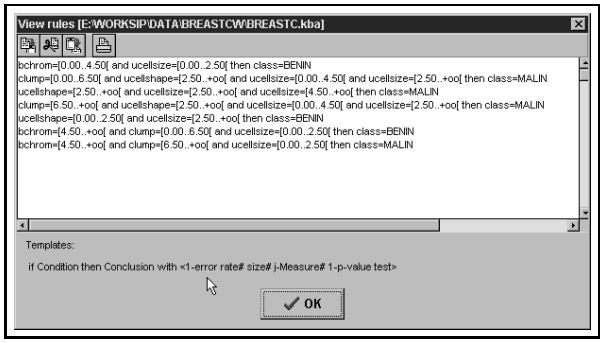

FIG. 13: *Une base de règle construit dans le logiciel Sipina-W(c) sur le fichier des cancer du sein*

l'utilisateur. Nous avons produit un effort important en faveur de la lisibilité des résultats. Nous avons été bien aidé, il est vrai, par l'environnement de programmation Delphi de Borland.

3.5 Epuration et validation des connaissances

L'apprentissage est éminemment empirique. A partir de régularités décelées dans un échantillon extrait selon une procédure définie par l'expert, nous définissons un classifieur (une partition) sur la population totale. Certains auteurs [Sch94] affirment haut et fort que la connaissance seule de l'échantillon ne permet en aucune manière d'extrapoler sur la population totale, il y a toujours des a priori derrière une démarche d'apprentissage. Nous ne contestons pas cette affirmation, dans les graphes on suppose que les concepts sont des hyper-rectangles. De l'adéquation de cet a priori avec la réalité dépend alors les performances en généralisation du classifieur. Comment peut-on en juger ?

Nous disposons d'un arsenal d'indicateurs qui nous permet d'évaluer le comportement du classifieur en généralisation, un test statistique permet de vérifier sa pertinence face aux modèles de référence que constituent l'affectation aléatoire de la conclusion au prorata de la distribution des classes dans l'échantillon d'apprentissage, ou encore l'affectation systématique à la classe la plus fréquente.

Le passage aux règles introduit une dimension supplémentaire à la validation du classifieur. Maintenant, nous pouvons évaluer une à unes ses composantes de manière à exclure les règles les moins intéressantes, celles qui dégradent les performances en généralisation. Nous pouvons une fois de plus adopter une démarche empirique en appliquant le classifieur sur un échantillon test, mais nous avons également la possibilité d'utiliser des indicateurs calculés sur l'échantillon d'apprentissage : le taux de bon classement, le nombre d'individus couverts, la j-mesure de [GS88] et l'intensité d'implication de [LGR81]. Cette dernière notamment peut être utilisée pour valider ou invalider une règle.

3.6 Fusion de connaissances

Travailler sur un système de représentation aussi universel que les bases de règles nous autorise une opération supplémentaire : la fusion de connaissances. Les graphes constituent une manière particulière d'extraire des informations des données, ils privilégient certaines formes d'exploration qui peut-être favorisent certaines formes de concepts. Si l'on utilise l'algorithme initial ID3 [Qui79], nous avantagerons les prédicteurs introduisant une partition très fragmentaire de l'espace de représentation. En utilisant d'autres méthodes d'induction de règles, il est envisageable de trouver d'autres classifieurs, qui peuvent être au moins aussi intéressants. Nous pensons entre autres à une méthode d'acquisition des connaissances très répandue qu'est l'interview d'expert, ce dernier se soucie peu d'optimisation d'indicateurs numériques, il essaie avant tout de traduire son savoir. Rajouter de telles règles dans la base ne peut qu'améliorer ses performances. Les opérations d'adjonction manuelle et de fusion de bases de règles sont disponibles via le menu de manipulation des règles (figure 13).

3.7 Stratégies de décision et généralisation

Un graphe d'induction induit une partition non recouvrante de l'espace de représentation. Cependant, avec les opérations d'adjonction et/ou de fusions de bases de connaissances, il arrive souvent qu'en généralisation (i.e l'application du classifieur dans la population mère) plusieurs règles peuvent être déclenchées lors du classement d'un individu, menant à des conclusions contradictoires. Il nous faut alors définir des stratégies de décision.

A priori, la méthode la plus simple consiste à choisir la règle qui se trompe le moins dans la population mère. Cette information étant indisponible, nous pouvons choisir l'un des indicateurs de qualité des règles ci-dessus, sous couvert qu'il soit corrélé avec le taux de succès en généralisation dans le problème étudié. Le choix de cet indicateur est accessible via la boîte de dialogue de spécification des règles (figure 14).

A la lumière des travaux de [Bre96] sur l'agrégation des classifieurs, nous pouvons également adopter une autre démarche en faisant voter les règles. Cela bien sûr pose le problème de la pondération : Faut-il

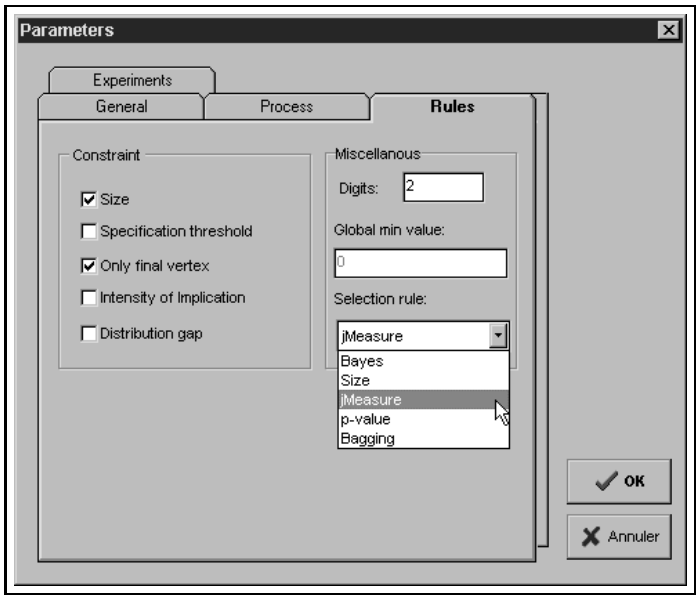

FIG. 14: *Sélection des règles en généralisation*

voter à la majorité absolue? Faut-il accorder aux règles un crédit proportionnel aux valeurs des indicateurs de qualité associés? Nous ne disposons pas de réponse tranchée à l'heure actuelle, nous constatons empiriquement en tous les cas que la stratégie de vote est nettement supérieure à la sélection de la règle la plus crédible (au vu des indicateurs tels que la j-Mesure ou l'intensité d'implication) dans les bases de connaissances construites par agrégation de prédicteurs issus de bootstrap.

4 SIPINA_W©, outil de comparaison des stratégies d'apprentissage

Elaborer un outil pour l'ECD était effectivement un des objectifs de l'élaboration du logiciel SIPINA_W©, la seconde était de construire une plate-forme de mise en oeuvre et d'évaluation pour les travaux de recherche. Si l'on recense toutes les possibilités qu'offre la version actuelle, on se rend compte que le nombre de choix possibles est très grand, en effet nous pouvons combiner :

- les méthodes d'apprentissage,
- les règles d'arrêt,
- les méthodes de discrétisation, locales ou globales,
- les contraintes d'extraction de règles.

Les comparaisons peuvent se faire par domaine d'application en chargeant classiquement un fichier afin de comparer deux ou plusieurs techniques d'apprentissage. La comparaison porte sur des résultats issus de validation croisée ou par répétition du couple apprentissage-validation. Pour les évaluations à grande échelle, nous avons imaginé une fiche de commande qui permet de spécifier différents traitements sur plusieurs fichiers successifs, les résultats sont consignés dans un fichier texte désigné par l'utilisateur (figure 15).

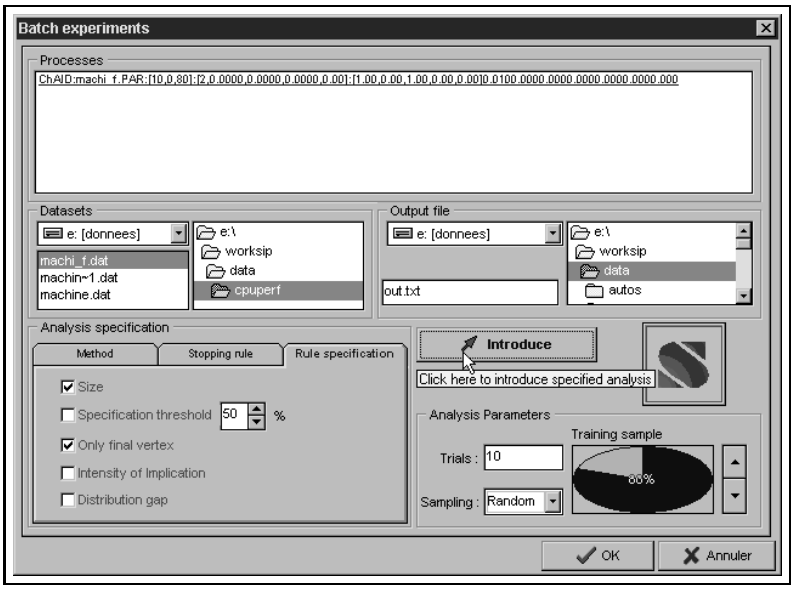

FIG. 15: *Fiche de commande pour le traitement par lots dans Sipina-W(c)*

5 Conclusion

En tant que féru de programmation, SIPINA_W© a été un des projets les plus enthousiasmants auquel nous avons eu à participer ces dernières années. Au-delà de l'expérience acquise dans la gestion d'un programme d'une telle importance (au moins en taille de code programme), la diffusion du logiciel sur Internet nous a également permis de prendre contact avec de nombreux chercheurs à travers le monde. Si l'on se fie uniquement aux connexions sur notre serveur, il y a près d'un millier de personnes qui ont téléchargé au moins une fois la plate-forme. Ce chiffre est sans aucun doute une borne inférieure puisqu'il ne prend pas en compte les individus qui ont téléchargé à partir des sites miroirs SIMTEL qui étaient notre deuxième mode de distribution.

Ne nous leurrons cependant pas, SIPINA_W© est loin d'être parfait. Certes, il a été exploité avec succès dans différentes études; nous l'avons également mis à contribution dans toutes nos analyses empiriques. Il reste que la base actuelle est irrémédiablement 16 bits, aucune connexion avec les serveurs de bases de données n'est faite. Le passage à une version 32 bits est une occasion unique d'introduire un nouveau saut qualitatif dans l'architecture de la plate-forme :

- une exploitation plus pointue du multi-tâche avec une programmation par threads qui permettra de tirer profit des machines multi-processeurs tournant sous Windows NT®;
- l'introduction des modes enrichis de description des données telles que les descripteurs flous [Ram94] ou symboliques [Did95];
- une connexion directe avec les serveurs de bases de données de manière à ce que les produits de requêtes SQL deviennent automatiquement des données en entrée pour l'apprentissage. Apparemment, avec les outils de développement récents, la définition d'architecture client-serveur est sérieusement envisageable avec un coût relativement faible.

Cette liste n'est évidemment pas exhaustive, elle montre en tous les cas qu'il nous reste beaucoup de travail et que la version actuelle, si elle est d'un excellent niveau, reste largement perfectible.

Références

[AD91] H. Almuallim and T. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552, 1991.

[BC91] W. Buntine and R. Caruana. Introduction to ind and recursive partitionning. Technical Report FIA-91-28, RIACS and NASA Ames Research Center, Moffett Field, CA, 1991.

[BFOS84] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. California : Wadsworth International, 1984.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[Bru92] T. Brunet. Le probleme de la resistance aux valeurs inconnues dans l'induction: une foret de branches, 1992.

[BT70] J.P. Bouroche and M. Tenenhaus. Quelques méthodes de segmentation. *RAIRO, 42*, pages 29–42, 1970.

[Cat91] J. Catlett. *Megainduction : machine learning on very large databases*. PhD thesis, University of Sydney, 1991.

[CN89] P. Clark and T. Niblett. The CN2 induction algorithm. 3:261–283, 1989.

[CS96] K.J. Cherkauer and J.W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[Did95] E. Diday. Probabilistic objects for a symbolic data analysis. *DIMACS*, 19, 1995.

[dM93] T. Van de Merckt. Decision trees in numerical attribute spaces. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1016–1021, 1993.

[FI93] U.M. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of The 13th Int. Joint Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufmann, 1993.

[FMdB96] E. Fortune, P. Makris, and J.P. Asselin de Beauville. Pertinence des indicateurs de bon classement dans le cas d'evenements rares. In *Actes Des Quatriemes Journees de La Societe Francophone de Classification*, 1996.

[GS88] R.M. Goodman and P. Smyth. Information-theoretic rule induction. In *Proceedings of European Conference on Artificial Intelligence*, 1988.

[HKS96] M. Holsheimer, M. Kersten, and A. Siebes. Data surveyor: searching the nuggets in parallel. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[Ima94] I.F. Imam. An empirical study on the incompetence of attribute selection criteria. Technical report, Machine Learning and Inference Laboratory - George Mason University, 1994.

[Kas80] G.V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

[Ker92] R. Kerber. Discretization of numeric attributes. In MIT Press, editor, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 123–128, 1992.

[KJ97] R. Kohavi and G. John. Wrappers for feature subset selection. *Journal of Artificial Intelligence, Special issue on Relevance*, 1997.

[KJL+94] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. Mlc++: A machine learning library in c++. *Tools with Artificial Intelligence*, pages 740–743, 1994.

[LGR81] I.C. Lerman, R. Gras, and H. Rostam. Elaboration et evaluation d'un indice d'implication pour donnees binaires. *Mathematiques et Sciences Humaines*, 74:5–35, 1981.

[LS94] P. Langley and S. Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94, Workshop on Case-Based Reasoning*, 1994.

[MA95] P.M. Murphy and D.W. Aha. Uci repository of machine learning databases, 1995. Available at http:/www.ics.uci.edu/ mlearn/MLRepository.html.

[Mic69] R.S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the 5th International Symposium on Information Processing*, pages 125–128, Bled Yugoslavia, 1969.

[OW91] J.J. Oliver and C.S. Wallace. Inferring decision graphs. In *Proceedings of Workshop 8 - Evaluating and Changing Representation in Machine Learning*, 1991.

[QR89] J.R. Quinlan and R.L. Rivest. Inferring decision trees using the minimum description length. *Information and Computation*, 80:227–248, 1989.

[Qui79] J. R. Quinlan. Discovering rules by induction from large collections of examples. In D. Michie, editor, *Expert Systems in the Microelectronic Age*, pages 168–201, Edinburgh, 1979. Edinburgh University Press.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui87] J.R. Quinlan. Decision trees as a probabilistic classifiers. In *Proceedings of the 4th International Workshop on Machine Learning*, 1987.

[Qui89] J.R. Quinlan. Unknow attribute values in induction. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 164–168, 1989.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Rak95] R. Rakotomalala. La distribution theorique des sequences. Technical report, Laboratoire ERIC, Universite Lumiere Lyon 2, 1995.

[Ram94] M. Ramdani. *Systeme d'induction formelle a base de connaissances imprecises*. PhD thesis, Institut Blaise Pascal, Universite Paris VI, 1994.

[RC96] R. Rakotomalala and E. Chettouh. Extraction de regles par validation dans les graphes d'induction. In *Actes des 4emes journees de la Societe Francophone de Classification*, 1996.

[RRS95] S. Rabaseda, R. Rakotomalala, and M. Sebban. Discretization of continuous attributes : a survey of methods. In *Proceedings of the 2nd Annual Joint Conference on Information Sciences*, pages 164–166, 1995.

[RRZ96] S. Rabaseda, R. Rakotomalala, and D.A. Zighed. Rules extracted automatically by induction. In *Proceedings of the 6th Conference on Information Processing and Management of Uncertainty*, pages 551–556, 1996.

[RSG93] T.W. Rauber and A.S. Steiger-Garcao. Feature selection of categorical attributes based on contingency table analysis. In *Proceedings of the 5th Portuguese Conference on Pattern Recognition*, 1993.

[RZ96] R. Rakotomalala and D.A. Zighed. Evaluation et validation des sommets dans les graphes d'induction : une alternative a l'elagage. In *Actes du XXVeme Colloque International l'Association Rhodanienne pour l'Avancement de l'Econometrie - ARAE'96*, 1996.

[RZ97] R. Rakotomalala and D.A. Zighed. Mesures d'association dans les graphes d'induction : une appoche statistique de l'arbitrage generalite-precision. In *Proceedings of the 7th Conference of International Association for the Development of Interdisciplinary Research*, pages 131–134, 1997.

[RZF97] R. Rakotomalala, D.A. Zighed, and F. Feschet. Rule characterization in induction process. In *Indo-French Workshop on Symbolic Data Analysis and its Applications*, pages 190–199, 1997.

[RZR96] R. Rakotomalala, D.A. Zighed, and S. Rabaseda. Validation of rules issued from induction graphs. In *Proceedings of the 6th Conference on Information Processing and Management of Uncertainty*, pages 1259–1264, 1996.

[Sch94] C. Schaffer. A conservation law for generalization performance. In *Proceedings of the 11th International Conference on Machine Learning*, pages 259–265, 1994.

[Weh93] L. Wehenkel. Decision tree pruning using an additive information quality measure. In B. Bouchon-Meunier, L. Valverde, and R. Yager, editors, *Uncertainty in Intelligent Systems*, pages 397–411. Elsevier, North Holland, 1993.

[WP91] L. Wehenkel and M. Pavella. Decision trees and transient stability of electric power system. *Automatica*, 27(1):115–134, 1991.

[ZAD92] D.A. Zighed, J.P. Auray, and G. Duru. *Sipina : Methode et logiciel*. Lacassagne, 1992.

[Zig85] D.A. Zighed. *Methodes et outils pour les processus d'interrogation non arborescents*. PhD thesis, Universite Claude Bernard - Lyon 1, 1985.

[ZR96a] D.A. Zighed and R. Rakotomalala. A method for non arborescent induction graphs. Technical report, Laboratory ERIC, University of Lyon 2, 1996.

[ZR96b] D.A. Zighed and R. Rakotomalala. $SIPINA_W(c) for Windows : User's Guide. Laboratory ERIC - University of Lyon 2$, 1996.

[ZRF97] D.A. Zighed, R. Rakotomalala, and F. Feschet. Optimal multiple intervals discretization of continuous attributes for supervised learning. In *Proceedings of the 3rd International Conference in Knowledge Discovery in Databases*, 1997.

[ZRR96] D.A. Zighed, R. Rakotomalala, and S. Rabaseda. A discretization method of continuous attributes in induction graphs. In *Proceedings of the 13th European Meetings on Cybernetics and System Research*, pages 997–1002, 1996.