Explicitation de connaissances tacites par évaluation assistée par ordinateur

Damien Follet, Nicolas Delestre, Nicolas Malandain, Laurent Vercouter

▶ **To cite this version:**

Damien Follet, Nicolas Delestre, Nicolas Malandain, Laurent Vercouter. Explicitation de connaissances tacites par évaluation assistée par ordinateur. Journée EIAH&IA 2013, May 2013, Toulouse, France. pp.1. hal-00824378

HAL Id: hal-00824378
https://hal.science/hal-00824378

Submitted on 21 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Explicitation de connaissances tacites par évaluation assistée par ordinateur

Damien Follet, Nicolas Delestre, Nicolas Malandain et Laurent Vercouter

LITIS, INSA Rouen, Avenue de l'Université
76800 Saint-Étienne-du-Rouvray, France
{damien.follet,nicolas.delestre,
nicolas.malandain,laurent.vercouter}@insa-rouen.fr

Résumé. Ces dernières années, l'enseignement adopte des formes variées et s'oriente vers l'enseignement massif (Massive Open Online Courses, premières années de médecine, ...) et l'évaluation diagnostique. De la masse de productions résultante naît un besoin de système d'évaluation assistée par ordinateur. Le besoin d'évaluation diagnostique détaillée popularise l'utilisation des grilles critériées. Nous envisageons dans cet article les apports de l'apprentissage automatique afin de réaliser un système d'évaluation assistée par ordinateur par grille critériée. Le système ainsi conçu serait indépendant du domaine d'application et permettrait l'explicitation des connaissances tacites. Nous commencerons par proposer une représentation générique des productions et des évaluations de l'apprenant, puis la modélisation de la fonction d'évaluation qui lie une évaluation à chaque production. Nous proposerons et évaluerons alors un algorithme d'apprentissage original permettant d'apprendre cette fonction à partir d'un ensemble d'apprentissage réduit.

Mots-clé : évaluation assistée par ordinateur, connaissances tacites, grille critériée, apprentissage automatique, problème d'induction, réduction de dimensionnalité, valeurs non-ordonnées discrètes, similarité

1 Introduction

Au cours des dernières années, l'enseignement adopte des formes variées afin d'arriver à transmettre des connaissances et s'oriente vers l'enseignement massif (MOOC, première année de médecine, ...) et l'évaluation diagnostique. De la masse de productions hétérogènes résultante naît un besoin croissant de système d'évaluation assistée par ordinateur, en anglais Computer-Assisted Assessment (CAA). Le besoin d'évaluation diagnostique détaillée, lui, entraîne une utilisation grandissante des grilles critériées. Pour rappel, une grille critériée ou grille de critères est une ensemble d'échelle de mesure. Chaque échelle de mesure définit plusieurs niveaux de connaissances ou de compétences de l'apprenant, ce qui permet donc de réaliser une évaluation formative ou diagnostique.

C'est à ce double besoin que nous cherchons à répondre avec un système d'évaluation par grilles critériées assistée par ordinateur. Il n'existe à notre connaissance aucun CAA capable d'évaluer par grille critériée. Il n'existe pas non

plus à notre connaissance de CAA capable de répondre à la masse de productions hétérogènes. En effet, il existe beaucoup de CAA capables d'évaluer de manière automatique les productions associées à un type spécifique d'exercices et de domaine d'application, comme [10] qui permet d'évaluer des essais dialectiques. Mais à cause de cette spécificité, il est très difficile et coûteux d'étendre ces techniques à d'autres types d'exercices ou de domaines d'application. Or de nouvelles façons d'évaluer ne cessent de voir le jour, comme le montre [4, 9] en remarquant qu'un exercice n'évalue pas l'apprenant de la même façon selon s'il est posé sous forme papier ou par ordinateur. De plus les domaines d'application potentiels se multiplient : en plus des traditionnels domaines académiques, des domaines liés aux nouvelles technologies (maitrise de nouveaux outils de communication ou de bureautique) apparaissent en suivant le rythme effrenné de l'évolution d'Internet. Dans ce contexte, il n'est pas envisageable de résoudre la problématique de l'enseignement massif sans concevoir un système qui soit le moins dépendant possible du domaine d'application.

Il existe ainsi certains Environnements informatiques pour l'apprentissage humain (EIAH) comme [8] qui permettent de soumettre l'apprenant à différents types d'exercices, puis d'évaluer automatiquement les productions associées grâce à un module CAA à qui on aura fourni l'ensemble des bonnes et mauvaises réponses. Or cet ensemble représente des connaissances tacites, c'est-à-dire quelque chose que l'on sait mais que l'on trouve très difficile sinon impossible à énoncer[1]. Des travaux [11, 12] ont montré que tout transfert de connaissances utiles implique la transmission de connaissances explicites ainsi que de connaissances tacites. Dans notre cas, l'enseignant cherche à transmettre à l'ordinateur l'ensemble des bonnes et mauvaises réponses, qui correspond effectivement à des connaissances utiles car il s'agit des informations nécessaires pour évaluer les productions d'un ensemble d'apprenants. L'explicitation des bonnes et mauvaises réponses est donc très difficile pour l'enseignant.

Nous envisageons dans cet article la possibilité d'expliciter ces connaissances tacites non pas par l'humain, mais par le CAA en utilisant des techniques de l'apprentissage automatique. Nous présenterons donc une modélisation de l'apprenant selon deux niveaux d'abstraction : les productions et les évaluations de l'apprenant, puis une modélisation implicite du domaine d'application : la fonction d'évaluation qui associe à chaque production une évaluation. Nous présenterons ensuite comment fonctionne l'algorithme d'apprentissage qui permettra d'apprendre la fonction d'évaluation, permettant ainsi d'expliciter les connaissances tacites de l'évaluation et sur lequel nous baserons notre ébauche de CAA. Enfin nous présenterons les résultats des tests préliminaires afin de jauger la fiabilité et la rapidité de notre algorithme.

2 Modélisation du Problème

Nous allons commencer par modéliser le problème de façon à être le plus indépendant possible du domaine d'application : nous modélisons l'apprenant à deux niveaux d'abstraction différents, la production et l'évaluation. Nous

passerons du premier au second grâce à la fonction d'évaluation que l'on définira par apprentissage automatique à partir d'un ensemble d'exemples TS[1]. Cette fonction d'évaluation permettra donc de modéliser implicitement les informations liées au domaine.

2.1 Modélisation de l'Apprenant : Productions et Evaluations.

Définition 1. *La production d'un apprenant sera modélisée par un ensemble de couples (attribut observable, valeur) où la valeur appartient à un domaine Non-Ordonné Discret (NOD).*

Exemple 1. Voici un domaine non-ordonné discret :
{chien; chat; poisson; marteau; lumiere}

Nous noterons l'ensemble des productions d'apprenant PS[2], l'ensemble des attributs observables OS[3] et la cardinalité du plus grand domaine des attributs observables D_o.

Définition 2. *L'évaluation sera modélisée par une interprétation de grille critériée, c'est-à-dire un ensemble de couples (attribut inféré, valeur) où la valeur appartient à un domaine ordonné discret.*

Nous noterons l'ensemble des évaluations ES[4], l'ensemble des attributs inférés IS[5] et la cardinalité du plus grand domaine des attributs inférés D_i.

Définition 3. *Un exemple sera modélisé par un couple (production,évaluation), où "évaluation" est associée manuellement à "production" par l'enseignant.*

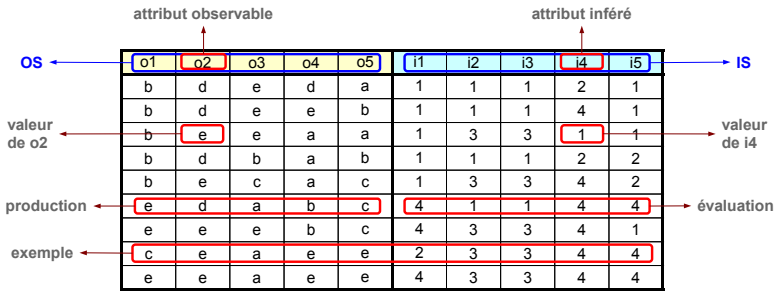


Fig. 1. Illustration du vocabulaire employé

[1] pour Training Set, i.e. ensemble d'apprentissage
[2] Production Set
[3] Observable attribute Set
[4] Evaluation Set
[5] Inferred attribute Set

2.2 Modélisation Implicite du Domaine : Fonction d'Evaluation.

Nous cherchons ici à modéliser l'évaluation de l'enseignant. Comme décrit dans [7], l'évaluation humaine est soumise à un certain nombre de biais : effet de tendance centrale, hypothèse socio-arithmétique, effet de fatigue, de halo, de séquence, d'inertie, de contamination, de flou et de sévérité systématique. Elle ne peut donc pas être considérée comme strictement déterministe.

Notre hypothèse de travail est que dans certaines conditions : évaluation par grille critériée d'un petit nombre de productions anonymisées et standardisées, il est possible de limiter la portée de ces biais et approximer l'évaluation humaine comme étant déterministe.

Cela nous permettra à terme de contruire un système capable d'inférer l'évaluation associée à une nouvelle production à partir d'un petit ensemble d'exemples évalués manuellement par l'enseignant dans les conditions évoquées précédemment. Dans un premier temps le système analyserait les informations contenues dans les exemples afin de reconstruire une fonction d'évaluation très proche de l'évaluation humaine. Dans un second temps, il utiliserait cette fonction d'évaluation afin d'évaluer de nouvelles productions.

Avant de détailler ce système, nous allons préciser la notion de fonction d'évaluation. Nous modélisons donc l'évaluation humaine par une fonction dite d'évaluation qui vérifie certaines propriétés.

Définition 4. *Parmi toutes les fonctions qui associent des productions et des évaluations, une fonction sera dite fonction d'évaluation si elle est déterministe, complètement nécessaire et bien informée. Elle sera alors notée* F^{s}_{eval} .

Le premier qualificatif "déterministe" signifie que toute production sera toujours associée à la même évaluation. Cela permet de s'assurer que les prédictions de notre système auront un sens. Le second qualificatif "complètement nécessaire" signifie que pour tout attribut inféré i chaque valeur de i est nécessaire, c'est-à-dire qu'il existe au moins une production de PS qui y sera associée. Le dernier qualificatif "bien informée" signifie que pour tout attribut inféré i, tous les critères permettant de choisir la valeur de i font partie de l'ensemble des attributs observables OS.

Ce dernier qualificatif implique que les attributs inférés sont indépendants les uns des autres ou à défaut redondants. Ainsi si l'on note $\#E$ le cardinal d'un ensemble E quelconque, cela nous permet de diviser le problème d'apprentissage simultané de $\#IS$ attributs inférés en autant de sous-problèmes d'apprentissage d'un seul attribut inféré à la fois.

Ces propriétés ne sont pas adaptées pour exprimer simplement l'ensemble de toutes les fonctions d'évaluations potentielles, c'est pourquoi nous proposons une propriété équivalente : la propriété de partitionnement.

Définition 5. *Soit* $Dom(v)$ *le domaine de la variable* v*, soit* $proj_i(V)$ *la fonction de projection qui retourne la valeur de la composante* i *extraite du vecteur* V*. Nous appellerons* $antecedents_F(i \leftarrow v_i) = \{p \in PS / proj_i(F(p)) = v_i\}$ *l'ensemble des antécédents de* $v_i \in Dom(i)$ *par* F*.*

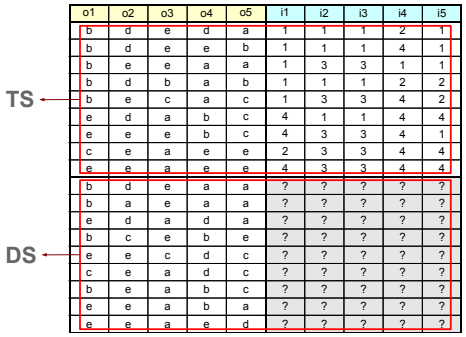
Définition 6. *Nous appellerons* $Antecedents_F(i) = \{antecedents_F(i \leftarrow v_i) / v_i \in Dom(i)\}$ *l'ensemble des ensembles d'antécédents des valeurs du domaine de l'attribut inféré* i *par* F.

Définition 7. *Une fonction* F *est partitionable si et seulement si pour chaque attribut inféré* i, $Antecedents_F(i)$ *forme une partition discernable*[6] *de* PS.

Informellement, cela signifie que pour tout attribut inféré i, toute production de PS est associée à exactement une valeur de i et toute valeur $v_i \in Dom(i)$ est associée à au moins une production de PS. Toute fonction partitionable est donc une fonction d'évaluation potentielle. On peut donc à présent se représenter l'espace des fonctions d'évaluation potentielles par le produit cartésien de $\#IS$ partitions de PS composées d'exactement D_i sous-ensembles discernables. Soit $\left\{ {n \atop p} \right\}$ le nombre de Stirling de seconde espèce qui représente le nombre de partitions possibles d'un ensemble à n éléments en p parties non vides non discernables, il existe donc $(\left\{ {D_o^{\#OS} \atop D_i} \right\} * D_i!)^{\#IS}$ fonctions d'évaluation potentielles différentes.

3 Algorithme d'Apprentissage

Nous rappellons que notre objectif est de trouver la fonction d'évaluation $F^{\mathrm{s}}_{\mathrm{eval}}$ qui se rapproche le plus possible de l'évaluation de l'enseignant. Pour cela nous disposons d'un ensemble d'exemples TS comme illustré sur la figure 2. La fonction d'évaluation devra bien entendu être consistante avec TS : informellement cela signifie que les prédictions de la fonction d'évaluation ne sont pas contradictoires avec les exemples de TS.



o1	o2	o3	o4	o5	i1	i2	i3	i4	i5
b	d	e	d	a	1	1	1	2	1
b	d	e	e	b	1	1	1	4	1
b	e	e	a	a	1	3	3	1	1
b	d	b	a	b	1	1	1	2	2
b	e	c	a	c	1	3	3	4	2
e	d	a	b	c	4	1	1	4	4
e	e	e	b	c	4	3	3	4	1
c	e	a	e	e	2	3	3	4	4
e	e	a	e	e	4	3	3	4	4
b	d	e	a	a	?	?	?	?	?
b	a	e	a	a	?	?	?	?	?
e	d	a	d	a	?	?	?	?	?
b	c	e	b	e	?	?	?	?	?
e	e	c	d	c	?	?	?	?	?
c	e	a	d	c	?	?	?	?	?
b	e	a	b	c	?	?	?	?	?
e	e	a	b	a	?	?	?	?	?
e	e	a	e	d	?	?	?	?	?

Fig. 2. Ensemble d'apprentissage TS et ensemble de décision DS sur un exemple

[6] "discernable" signifie que chaque élément de la partition est identifiée par une valeur $v_i \in Dom(i)$. Il y a donc $D_i!$ fois plus de partitions discernables que de partitions non discernables.

Nous présenterons dans cette section le problème que pose la dimensionalité élevée de l'espace de recherche ainsi que notre approche pour le résoudre. Nous résumerons ensuite l'algorithme d'apprentissage en trois étapes, puis nous exposerons le protocole des tests préliminaires et leurs résultats.

3.1 Dimensionalité de l'Espace de Recherche

La définition de fonction bien informée nous permet de décomposer le problème d'apprentissage de $\#IS$ attributs inférés en $\#IS$ sous-problèmes à un attribut inféré chacun, comme le montre la figure 3 pour i_1. Cela permet donc de restreindre l'espace de recherche : éliminer une valeur $v_i \in Dom(i)$ revient à éliminer plusieurs fonctions d'évaluation potentielles en même temps. Cet espace R est donc plus petit que l'espace des fonctions d'évaluations potentielles : on se le représentera par la juxtaposition de $\#IS$ partitions de PS composées d'exactement D_i sous-ensembles discernables.

o1	o2	o3	o4	o5	i1
b	d	e	a	a	?
b	a	e	a	a	?
e	d	a	d	a	?
b	c	e	b	e	?
e	e	c	d	c	?
c	e	a	d	c	?
b	e	a	b	c	?
e	e	a	b	a	?
e	e	a	e	d	?

Fig. 3. sous-problème d'apprentissage de i_1

Soit le nombre de productions $D_o^{\#OS}$, le nombre d'évaluations $D_i^{\#IS}$ et le nombre de Stirling $\left\{ {n \atop p} \right\}$[7], il est alors possible d'exprimer le nombre d'éléments de l'espace de recherche R du problème :

$$\#R = \#IS * \left\{ {D_o^{\#OS} \atop D_i} \right\} * D_i! \ . \tag{1}$$

Cette formulation n'est pas très pratique car le nombre de Stirling est difficile à décomposer : $\left\{ {n \atop p} \right\} = \frac{1}{p!} \sum_{k=0}^{p} (-1)^{p-k} \binom{p}{k} k^n$. Mais il est facile d'exprimer sa borne inférieure $L(n,p) \leq \left\{ {n \atop p} \right\}$ en fonction de n et de p :

$$L(n,p) = \frac{1}{2}(p^2 + p + 2) * p^{n-p-1} - 1 = O(p^{n-p+1}) \ . \tag{2}$$

Nous pouvons en déduire une borne inférieure au nombre d'éléments de l'espace de recherche R :

[7] on rappelle que le nombre de Stirling représente le nombre de partitions possibles d'un ensemble à n éléments en p parties non vides non discernables

$$\#R \geq O(\#IS * D_i^{D_o^{\#OS}-D_i+1} * D_i!) \ . \tag{3}$$

Exemple 2. Cela signifie que par exemple, pour $\#OS = 5$, $\#IS = 5$, $D_o = 5$ et $D_i = 4$, il existe 3125 productions différentes, 625 évaluations et l'espace de recherche contient plus de $\#R \geq 30{*}4^{3123} \simeq 5{*}10^{1881}$ éléments. En comparaison, on estime à 10^{80} le nombre d'atomes dans l'univers observable et à $52! \simeq 8 * 10^{67}$ le nombre d'ordonnancements possibles d'un jeu de 52 cartes.

Il est donc nécessaire de réduire encore l'espace de recherche. Nous devrons ainsi utiliser un algorithme d'apprentissage permettant la réduction de dimensionalité.

La plupart des algorithmes de réduction de dimensionalité actuels sont conçus pour traiter des données numériques et non NOD[5, 6]. En effet, des méthodes comme Principal Component Analysis (PCA), Factor Analysis, Independent Component Analysis (ICA), Non-linear PCA, Random projections, Kernel PCA reposent sur l'algèbre linéaire matriciel[5, 6], lui-même basé sur le concept d'applications linéaires. Or si multiplier un réel par un autre a du sens parce que $\mathbb{R}$ est stable pour la multiplication par un réel, multiplier une valeur NOD par un réel ou additionner deux valeurs NOD n'en a pas vraiment car le résultat ne correspond ni à un élément de $\mathbb{R}$ ni à un élément du domaine NOD. Que signifierait $5 * \text{chien} + 3 * \text{marteau}$ ou $\text{lumiere} + \text{marteau} + \text{chat}$?

Les méthodes basées sur l'algèbre linéaire matriciel ne sont donc pas directement applicables à notre problème, pas plus que celles qui sont directement basées sur les applications linéaires comme Kohonen Self-Organizing Maps, Non-linear ICA, Local Linear Embedding, Locally Linear Coordination, Manifold Charting, Local Tangent Space Analysis ou MultiLayer autoencoders[5, 6]. En conséquence, les méthodes basées sur la moyenne ou la variance comme Projection Pursuit, Principal Curves, Neural Network, Vector Quantization[5, 6] ne sont pas non plus directement envisageables.

D'autres méthodes comme MultiDimensional Scaling et Maximum Variance Unfolding calculent la distance euclidienne entre chaque paire d'exemple[5, 6]. Appliquer ces méthodes sur notre problème reviendrait donc à soustraire une valeur NOD à une autre puis multiplier le résultat par lui-même ; pour la même raison que précédemment, nous ne pouvons pas appliquer directement ces méthodes.

Après avoir considéré toutes ces méthodes, nous avons conclu qu'à notre connaissance les méthodes actuelles de réduction de dimensionalité ne sont pas adaptées à notre problème. Nous avons donc développé un algorithme original réduisant l'espace de recherche et permettant d'apprendre des données NOD.

Il repose sur le principe suivant. A cause de cette dimensionalité élevée, il n'est pas possible d'explorer directement l'espace de recherche du problème pour déterminer les fonctions d'évaluation potentielles qui sont consistantes avec TS. Mais il est possible de l'explorer de manière indirecte : comme la valeur d'un attribut inféré i dépend toujours des valeurs du même sous-ensemble d'attributs

observables, nous allons tout d'abord chercher à identifier ce sous-ensemble, puis nous déterminerons les valeurs associées à chaque valeur de i présentes dans les exemples de TS. Ce sous-ensemble sera par la suite appelé ensemble des dépendances $Deps(i)$ de l'attribut inféré i.

Notre espace de recherche s'en trouve fortement réduit :

Exemple 3. En reprenant l'exemple précédent, au lieu de chercher dans $\#R \geq 5{*}10^{1881}$ fonctions d'évaluations nous explorerons $2^{\#OS} = 32$ dépendances possibles pour chaque attribut inféré, ce qui fait un total de $(2^{\#OS}){*}\#IS = 160$ éléments à parcourir.

La section suivante présente le fonctionnement de cet algorithme.

3.2 Résumé de l'Algorithme

Notre algorithme prend en entrée un ensemble d'exemples TS et retourne une fonction d'évaluation F^{s}_{eval} .

Soit o_1 et o_2 deux attributs observables, i_1 un attribut inféré, $Dom(o_1) = \{a; b; c; d; e\}$, $Dom(o_2) = \{a; b; c; d; e\}$, $Dom(i_1) = \{1; 2; 3; 4\}$ et $TS = \{(ac, 1); (bc, 1); (cd, 1); (dc, 2)\}$ tel que $(ab, 1)$ signifie : "la production $(o_1 \leftarrow a; o_2 \leftarrow b)$ est évaluée à $(i_1 \leftarrow 1)$". Une représentation graphique de TS est présenté dans la figure 4.

o1	o2	i1
a	c	1
b	c	1
c	d	1
d	c	2

Fig. 4. Représentation graphique de TS

L'algorithme fonctionne en trois étapes :

Premièrement, pour chaque attribut inféré i nous réduisons l'espace de recherche des dépendances en calculant les parties de l'ensemble des attributs observables OS qui ne sont pas consistantes avec TS, et en déduisons les parties qui le sont. Ces parties seront par la suite appelées candidats.

Une partie P de OS sera dite inconsistante avec TS pour l'attribut inféré $i \in IS$ si et seulement si une valuation de P est associée à deux valeurs différentes de i. Elle sera dite consistante dans le cas contraire.

Exemple 4.

1. $o_1 \leftarrow \{a; b; c\}$ pour $i_1 = 1$
   $o_1 \leftarrow \{d\}$ pour $i_1 = 2$
   $\Rightarrow \{a; b; c\} \cap \{d\} = \emptyset$, o_1 consistant avec i_1
2. $o_2 \leftarrow \{c; d\}$ pour $i_1 = 1$
   $o_2 \leftarrow \{c\}$ pour $i_1 = 2$
   $\Rightarrow \{c; d\} \cap \{c; d\} = \{c\}$, o_2 inconsistant avec i_1
3. $\{o_1; o_2\} \leftarrow \{ac; bc; cd\}$ pour $i_1 = 1$
   $\{o_1; o_2\} \leftarrow \{dc\}$ pour $i_1 = 2$
   $\Rightarrow \{ac; bc; cd\} \cap \{dc\} = \emptyset$, $\{o_1; o_2\}$ consistant avec i_1

Cette étape se rapproche donc de la phase de réduction de dimensionalité de l'espace de version et de la Programmation Logique Inductive.

Deuxièmement nous utilisons une heuristique de type rasoir d'Ockham : nous sélectionnons parmi les candidats celui qui apparait le moins complexe, c'est-à-dire celui qui minimise le score de complexité. Ce candidat sera alors assimilé à l'ensemble des dépendances $Deps(i)$ de i.

Le score de complexité s d'un candidat sera défini comme la somme du score de dépendance et du score d'alternatives. Le score de dépendance s_{dep} est défini par $\#P$ le nombre d'attributs observables qui définissent le candidat. Le score d'alternatives $s_{\text{alt}}(c, i)$ est défini par le nombre d'alternatives que le candidat c induit, c'est-à-dire le nombre de valuations différentes des attributs observables pour une même valeur de l'attribut inféré i.

Exemple 5.

1. $s_{\text{dep}}(\{o_1\}) = 1$
   $s_{\text{alt}}(\{o_1\}, i_1) = 3 + 1 = 4$
   car $o_1 \leftarrow \{a; b; c\}$ pour $i_1 = 1$ et $\{d\}$ pour $i_1 = 2$.
   $\Rightarrow s(\{o_1\}, i_1) = 1 + 4 = 5$.
2. $s_{\text{dep}}(\{o_1; o_2\}) = 2$
   $s_{\text{alt}}(\{o_1; o_2\}, i_1) = 3 + 1 = 4$
   car $o_2 \leftarrow \{ac; bc; cd\}$ pour $i_1 = 1$ et $\{dc\}$ pour $i_1 = 2$.
   $\Rightarrow s(\{o_1; o_2\, i_1) = 2 + 4 = 6$.

D'où $Deps(i_1) = \{o_1\}$.

Troisièmement, nous croisons cette information avec TS afin d'obtenir les couples (ensemble de valeurs d'attribut observable, valeur d'attribut inféré) pour chaque attribut inféré de IS pour former $F^{\text{s}}_{\text{eval}}$, explicitant ainsi les connaissances tacites de l'évaluation.

Exemple 6. On rappelle que $Deps(i_1) = \{o_1\}$ (déduit de l'étape précédente) et que $TS = \{(ac, 1); (bc, 1); (cd, 1); (dc, 2)\}$.

1. $(ac, 1) \Rightarrow (\{o_1 \leftarrow a\}, i_1 \leftarrow 1)$
2. $(bc, 1) \Rightarrow (\{o_1 \leftarrow b\}, i_1 \leftarrow 1)$
3. $(cd, 1) \Rightarrow (\{o_1 \leftarrow c\}, i_1 \leftarrow 1)$
4. $(dc, 2) \Rightarrow (\{o_1 \leftarrow d\}, i_1 \leftarrow 2)$

F^{s}_{eval} modélise l'évaluation de l'enseignant et est formée de ces 4 derniers couples. La figure 5 en propose une représentation graphique :

o1	o2	i1
a	*	1
b	*	1
c	*	1
d	*	2

Fig. 5. Fonction d'évaluation F^{s}_{eval}

Nous pourrons ensuite utiliser cette fonction F^{s}_{eval} dans la phase de décision de l'algorithme d'apprentissage afin d'inférer la valeur de i_1 à partir des seules valeurs de o_1, comme le montre la figure 6 :

o1	o2	i1
a	b	1
b	d	1
c	e	1
d	a	2

Fig. 6. Quatre nouvelles productions évaluées en utilisant F^{s}_{eval}

3.3 Tests Préliminaires

Les méthodes numériques étant peu adaptées à notre problème, nous avons envisagé de comparer l'intérêt de notre algorithme avec des techniques d'apprentissage symbolique qui peuvent traiter des données NOD. Nous avons conduit des tests de performance préliminaires avec ID3[8] pour l'apprentissage d'arbres de décision [3] et WkNN[9] pour l'apprentissage basé sur la distance

[8] Iterative Dichotomiser 3
[9] Weighted k-Nearest Neightbours

MVDM[10] couplée à une classification k-NN pondéré (k=1 selon les recommandations de [2]).

Pour ce faire, nous avons fixé arbitrairement la valeur des paramètres de l'espace des productions et de l'espace des évaluations : 5 attributs observables à 5 valeurs NOD chacun, évalués sur 5 attributs inférés à 4 valeurs chacune. Nous en avons généré aléatoirement à 4 reprises deux ensembles disjoints : l'ensemble d'apprentissage TS et l'ensemble de validation DS[11]. Ils sont formés respectivement de 12 et 10 éléments. Pour chaque couple (TS, DS), nous avons généré aléatoirement 180 fonctions d'évaluations différentes qui ont servi à étiqueter TS puis à valider les prédictions de l'algorithme d'apprentissage.

Les résultats sont présentés dans le tableau récapitulatif 1. Ils montrent que c'est notre algorithme qui est le plus fiable avec seulement 17% d'erreurs, contre deux fois plus pour ID3 (34%) et trois fois plus pour WkNN (50%).

Ils montrent également que durant la phase d'apprentissage, ID3 est l'algorithme le plus rapide (1,3 ms) ; notre algorithme est environ trois fois plus lent mais reste dans le même ordre de durée (4,4 ms), et WkNN se montre environ vingt fois plus lent (24,8 ms). Durant la phase de décision, ID3 et notre algorithme sont quasiment aussi rapides (de l'ordre de 105 μs), loin devant WkNN (7,8 ms).

Tableau 1. Résultats du test de performance en fiabilité et en rapidité

	délai moyen (μs)				pourcentage d'erreur (%)	
algorithmes	apprentissage	$\sigma_{\text{apprentissage}}$	décision	σ_{decision}	moyenne	$\sigma_{\%\text{erreurs}}$
notre algorithme	4 411,9	6 376,3	111,4	115,7	16,8	6,6
ID3	1 274,1	2 102,1	98,2	77,1	33,8	10,7
WkNN	24 820,2	13 122,5	7 796,9	3 745,7	50,3	10,5

4 Conclusion

Les résultats des tests montrent qu'en plus de permettre d'expliciter les connaissances tacites de l'évaluation, cet algorithme est compétitif en terme de fiabilité et de rapidité.

Bien sûr nous devrons effectuer des tests de validation croisée afin de confirmer ces résultats, ainsi que comparer notre algorithme à d'autres algorithmes utilisés pour résoudre le problème d'induction symbolique comme par exemple les réseaux bayésiens. Il restera également à implémenter l'Interface Homme-Machine du CAA, puis à effectuer des tests sur des données réelles, ce qui pourrait éventuellement nous confronter à des problèmes de mise à l'échelle. Il serait

[10] Modified Value Difference Metric
[11] Decision Set.

alors utile d'envisager certaines optimisations de l'algorithme, comme améliorer encore la réduction de l'espace de recherche en appliquant une technique similaire à l'élagage alpha-beta sur le score de complexité.

Enfin, la modélisation actuelle étant basée sur des valeurs NOD, il sera nécessaire d'étendre l'algorithme actuel pour prendre en compte une similarité entre valeurs d'attributs fournie par l'utilisateur afin d'améliorer la sélection par le score d'alternatives. On pourrait également étendre l'algorithme pour gérer des valeurs de type graphe.

5 Bibliographie

References

1. Rust, C., Price, M., & O'Donovan, B. (2003). Improving students' learning by developing their understanding of assessment criteria and processes. Assessment & Evaluation in Higher Education, 28(2), 147-164.
2. Cost, S., & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. Machine learning, 10(1), 57-78.
3. Peng, W., Chen, J., & Zhou, H. (2009). An Implementation of ID3—Decision Tree Learning Algorithm. From web. arch. usyd. edu. au/wpeng/DecisionTree2. pdf Retrieved date : May, 13.
4. Conole, G., & Warburton, B. (2005). A review of computer-assisted assessment. ALT-J, 13(1), 17-31.
5. Fodor, I. K. (2002). A survey of dimension reduction techniques.
6. Van der Maaten, L. J. P., Postma, E. O., & Van den Herik, H. J. (2009). Dimensionality reduction : A comparative review. Journal of Machine Learning Research, 10, 1-41.
7. Demeuse, M. (2004). Introduction aux théories et aux méthodes de la mesure en sciences psychologiques et en sciences de l'éducation.
8. Beevers, C. E., & Paterson, J. S. (2003). Automatic assessment of problem-solving skills in Mathematics. Active Learning in Higher Education, 4(2), 127-144.
9. Fiddes, D., Korabinski, A., McGuire, G., Youngson, M., & McMillan, D. (2002). Are Examination Results affected by the mode of delivery. Alt-J, 10, 60-69.
10. Christie, J. R. (1999, June). Automated essay marking-for both style and content. In Proceedings of the Third Annual Computer Assisted Assessment Conference, Loughborough University, Loughborough, UK.
11. POLANYI, M. (1998) The tacit dimension, in : L. PRUSAK (Ed.) Knowledge in Organization (Boston, MA, Butterworth Heineman).
12. Tsoukas, H. (1996). The firm as a distributed knowledge system : a constructionist approach. Strategic management journal, 17(WINTER), 11-25.