Traitement du bruit en inférence exacte

Marc Sebban, Jean-Christophe Janodet

EURISE, Université Jean Monnet de Saint-Etienne, 23 rue Paul Michelon,
42023 Saint-Etienne, France.
{Marc.Sebban,janodet}@univ-st-etienne.fr

Résumé : La *tolérance au bruit* est une qualité que tout algorithme d'apprentissage automatique doit avoir pour traiter des problèmes réels. De nombreuses techniques ont été développées pour traiter des données numériques bruitées, mais peu de méthodes semblent exister lorsque les données d'apprentissage sont des séquences non bornées de symboles, comme c'est le cas en inférence grammaticale. Dans ce papier, nous proposons une approche statistique de traitement du bruit pendant l'inférence d'un automate avec RPNI. Notre technique est basée sur un test de comparaison de proportions qui relâche les contraintes d'acceptation d'une fusion de RPNI sans mettre statistiquement en danger ses performances en généralisation. Outre l'approche elle-même, nous étudions les propriétés théoriques du comportement de RPNI*, notre nouvelle version de RPNI, puis nous faisons une étude expérimentale comparative de ces deux algorithmes.

Mots-clés : Inférence grammaticale, traitement des données bruitées, RPNI.

1 Introduction

Grâce aux progrès récents en acquisition et en stockage d'information, les bases de données modernes sont souvent très grandes mais également fortement bruitées. Pour rester efficaces, les algorithmes d'apprentissage automatique nécessitent donc de plus en plus des traitements spécifiques aux données bruitées. En pratique, la présence, même minime, de données corrompues peut avoir un impact dramatique sur les performances d'un classifieur, en terme de complexité structurelle du modèle induit et de taux de succès en généralisation. De nombreux algorithmes de réduction de données ont été proposés avec pour but, soit la réduction de la dimension de représentation par sélection d'attributs (Aha, 1992; John *et al.*, 1994; Langley, 1994), soit la suppression d'exemples non pertinents par sélection de prototypes (Brodley & Friedl, 1996; Wilson & Martinez, 1997; Sebban & Nock, 2000; Sebban *et al.*, 2003). On peut remarquer que la majorité des travaux menés sur ces techniques de réduction sont capables de traiter des données numériques et non des données symboliques. Ces méthodes sont encore plus rares pour des séquences de symboles de longueurs non bornées (mots), données qui sont l'apanage de l'inférence grammaticale.

L'inférence grammaticale est un sous-domaine de l'apprentissage automatique dont le but est d'apprendre des modèles de langages (ensembles de mots). Parmi ces modèles,

les automates finis déterministes (AFD) sont des machines à états qui prennent des mots en entrée et qui les acceptent ou les rejettent. Les AFD sont donc des classifieurs qui séparent l'ensemble de tous les mots en deux classes, la classe dite positive des mots acceptés par l'AFD, et celle dite négative des mots rejetés par l'AFD. Une des raisons qui expliquent les efforts consentis pour apprendre de tels classifieurs tient au fait que de nombreux domaines d'application, comme la reconnaissance de la parole, la reconnaissance de formes ou le traitement des langues naturelles, tirent partie des propriétés structurelles et sémantiques des AFD (de la Higuera, 2002). Dans ce papier, nous nous concentrerons uniquement sur les algorithmes d'inférence grammaticale exacte qui utilisent un mécanisme de fusion d'états. Nous utiliserons en particulier RPNI (Oncina & García, 1992), même si nos résultats peuvent s'étendre facilement à d'autres algorithmes, variantes ou extensions de RPNI, comme EDSM (Lang *et al.*, 1998) ou ESMA (Coste, 1999).

RPNI fonctionne à partir de deux ensembles de données, E_+ et E_-, qui contiennent respectivement des mots acceptés et rejetés par l'AFD cible qu'on souhaite apprendre. Par exemple, l'AFD de gauche dans la Fig.1 a été obtenu avec RPNI pour $E_- = \{aa, ba\}$ et $E_+ = \{\varepsilon, b, ab, abb, bab\}$. RPNI est considéré comme un algorithme d'apprentissage

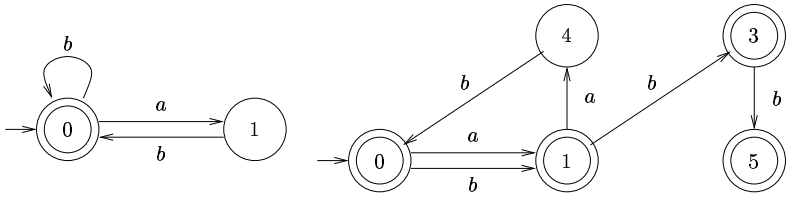

FIG. 1 – Deux AFD appris avec RPNI, en l'absence et en présence de données bruitées.

exact au sens où il retourne un AFD qui "colle" aux données : un exemple positif (de E_+) sera nécessairement reconnu par cet AFD, et un exemple négatif (de E_-) sera nécessairement rejeté. D'autre part, si les données d'apprentissage contiennent certains mots, dits *caractéristiques* de l'AFD cible, alors on peut montrer que RPNI trouve cette cible (Oncina & García, 1992). Ces propriétés ont un intérêt théorique indéniable. Mais dès qu'on tente d'utiliser RPNI sur un problème réel, où les données sont bruitées, ses propriétés sont en fait son principal handicap. Ainsi, l'AFD de droite dans la Fig. 1 est celui qu'on obtient avec RPNI en ajoutant le mot *bbbb* à E_-. Ce mot devrait appartenir à E_+, en l'absence de bruit, si l'AFD cible était celui de gauche dans la Fig. 1. Clairement, le fait que RPNI colle aux données induit un phénomène de sur-apprentissage ; cela se traduit, tout d'abord, par une augmentation du nombre d'états des AFD produits, et ensuite, par la chute du taux de succès en généralisation de ces AFD.

De nombreux types de bruit peuvent être considérés en apprentissage automatique (bruit sur les étiquettes, sur les lettres, données floues, incomplètes, etc). Nous ne considérerons ici que le bruit sur les étiquettes, transformant une donnée positive (dans E_+) en donnée négative (dans E_-), et réciproquement. Si on utilise la terminologie propre à la sélection d'attributs (John *et al.*, 1994), on peut envisager deux stratégies pour travailler avec de telles données. La première, dite *filter*, consiste à supprimer les

données non pertinentes en amont de la phase d'inférence, lors d'un prétraitement. Nous avons étudié une solution de ce type dans (Sebban *et al.*, 2002), fondée sur une règle de minimisation d'entropie dans un graphe de voisinage de mots. La principale difficulté de cette technique est la définition d'une distance judicieuse entre des mots, distance qui a un effet direct sur les prototypes sélectionnés. Pour éviter ce biais, la seconde stratégie, dite *wrapper*, consiste à détecter et à traiter les données bruitées pendant la phase d'inférence. C'est cette approche qui nous intéresse.

Le reste de l'article est organisé de la manière suivante. Après quelques définitions usuelles, la Section 2 rappelle les principes de fonctionnement de RPNI. Dans la Section 3, nous présentons notre cadre statistique de traitement du bruit et la nouvelle règle de fusion d'états qui est utilisée par notre algorithme, RPNI*. Dans la Section 4, nous établissons plusieurs résultats théoriques sur le comportement des algorithmes inférant des AFD en présence de données bruitées, puis nous faisons une étude expérimentale et comparative de RPNI et de RPNI*, dans la Section 5. Nous donnons enfin quelques conclusions et perspectives dans la Section 6.

2 Quelques rappels sur RPNI

Notre objectif est ici de donner des intuitions sur le fonctionnement de RPNI en décrivant son exécution sur un exemple. Le lecteur intéressé par les détails techniques peut consulter (Dupont & Miclet, 1998; Cornuéjols & Miclet, 2002).

Commençons par rappeler quelques notions élémentaires. Un *alphabet* Σ est un ensemble fini non vide de *lettres* et on appelle *mot* sur Σ toute séquence finie $w = a_1 a_2 \dots a_n$ de lettres. Un *automate fini déterministe* (AFD) est un quadruplet $A = \langle Q, \delta, s_0, F \rangle$ tel que Q soit un ensemble fini d'états, $\delta : Q \times \Sigma \to Q$ soit une fonction de transition, $s_0 \in Q$ soit un état initial et $F \subseteq Q$ soit un ensemble d'états finaux. On dit qu'un état s *contient* un mot w ssi $\delta(s_0, w) = s$. On dit que w est *reconnu* par A si $\delta(s_0, w) \in F$ et qu'il est *rejeté* sinon. Ainsi, l'AFD de gauche de la Fig. 1 est défini par $Q = \{0, 1\}$, $s_0 = 0$, $F = \{0\}$, $\delta(0, a) = 1$ et $\delta(0, b) = \delta(1, b) = 0$. abb est reconnu car $\delta(0, abb) = \delta(1, bb) = \delta(0, b) = 0 \in F$. Par contre, ba et ab sont rejetés car $\delta(0, ba) = 1 \notin F$ et $\delta(0, ab)$ n'est pas défini.

Nous présentons maintenant le principe de fonctionnement de RPNI (Oncina & García, 1992). En entrée, RPNI prend un ensemble E_+ de mots reconnus par un AFD cible inconnu (exemples) et un ensemble E_- de mots que cet AFD rejette (contre-exemples). Ici, nous prendrons $E_+ = \{\varepsilon, b, ab, abb, bab\}$ et $E_- = \{aa, ba\}$ (ε dénote le mot vide, le mot sans lettre). Après la phase d'inférence que nous allons décrire ci-dessous, RPNI retourne un AFD qui généralise les données de E_+ tout en rejetant les données de E_-. Sous certaines conditions théoriques, RPNI trouve l'AFD cible qui a produit les données[1]. La première tâche entreprise par RPNI est la construction du PTA (*prefix tree acceptor*) des mots de E_+. Ce PTA est le plus grand AFD reconnaissant uniquement les mots de E_+. Ses états sont numérotés selon l'ordre hiérarchique sur les préfixes de E_+ (Oncina & García, 1992). Dans notre exemple, le PTA des mots de E_+ est l'AFD du

[1] RPNI identifie polynomialement par données fixées la classe des langages rationnels représentée par des AFD (Oncina & García, 1992; Dupont & Miclet, 1998).

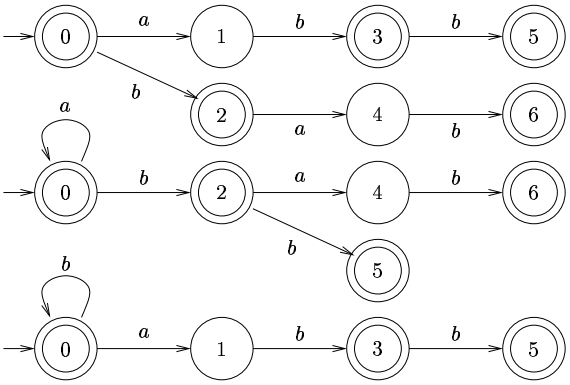

FIG. 2 – L'AFD du haut est le PTA de $E_+ = \{\varepsilon, b, ab, abb, bab\}$; celui du milieu résulte de la fusion des états 1 et 0 dans le PTA, et celui du bas, de la fusion des états 2 et 0.

haut dans la Fig. 2. Une fois le PTA construit, RPNI parcourt ses états dans l'ordre. Le traitement du $i$ème état consiste à essayer de le fusionner avec les états 0, 1, ..., $i-1$, dans l'ordre. Fusionner deux états consiste à les regrouper en un seul dont le numéro est le minimum entre les deux états fusionnés. Cet état est final si au moins un des deux états l'était. Les transitions sortantes des deux états sont elles-mêmes fusionnées deux à deux lorsqu'elles sont étiquetées par la même lettre, et dans ce cas, les deux états qu'elles pointent sont fusionnés, de manière récursive. Dans la Fig. 2, RPNI tente la fusion des états 1 et 0 du PTA. Ceci crée une boucle étiquetée par a sur l'état 0. Comme 1 et 0 ont tous deux des transitions sortantes d'étiquette b, les états 3 et 2 qu'elles pointent sont fusionnés, ce qui conduit à l'AFD du milieu dans la Fig. 2. Une fusion *réussit* si aucun exemple de E_- n'est accepté par l'AFD résultant de cette fusion ; elle *échoue* sinon. Dans notre exemple, la fusion de 1 et 0 échoue puisque $aa \in E_-$ est accepté. Donc RPNI abandonne cette fusion et tente la fusion des états 2 et 0, ce qui conduit à l'AFD du bas dans la Fig. 2. Celui-ci n'accepte aucun exemple de E_- donc la fusion réussit. RPNI considère donc ce nouvel AFD et tente, avec succès, la fusion des états 3 et 0. Celle-ci conduit à l'AFD de gauche de la Fig. 1, le résultat global de RPNI sur les données.

Nous explicitons maintenant le pseudo-code de RPNI (cf. Algo. 1). A chaque étape, RPNI maintient une partition déterministe π des états du PTA A de E_+. L'AFD courant, A/π, reconnaît tous les mots de E_+ et rejette tous les mots de E_-. Par une partition déterministe, nous entendons une partition qui est compatible avec la fonction de transitions de A : $\forall p_1, p_2 \in Q, \forall x \in \Sigma, \forall q_1, q_2 \in Q$, si $\pi(p_1) = \pi(p_2)$, $\delta(p_1, x) = q_1$ et $\delta(p_2, x) = q_2$, alors $\pi(q_1) = \pi(q_2)$. $\pi(q)$ désigne l'ensemble des états appartenant au même bloc que l'état q. Fusionner deux états i et j revient à calculer la plus petite partition déterministe π' telle que (1) $\forall s \in Q, \pi'(s) \supseteq \pi(s)$ et (2) $\pi'(i) = \pi'(j)$; elle est obtenue en calculant la fermeture de $\pi \setminus \{\pi(i), \pi(j)\} \cup \{\pi(i) \cup \pi(j)\}$ par la fonction de transition δ du PTA. Cette fusion réussit si l'AFD A/π' n'accepte aucun mot de E_-. Notons que, par construction, A/π' accepte nécessairement tous les mots de E_+,

Algorithm 1 Pseudo-code de RPNI

```
A ⟵ PTA(E+) ; π ⟵ {{q} | q ∈ Q}
for i = 1 to N − 1 do
  if i = min(π(i)) then
    j ⟵ 0 ; continuer ⟵ vrai
    while (j < i) and continuer do
      if j = min(π(j)) then
        π' ⟵ calcul_fusion(π, i, j, δ)
        if compatible(A/π', E−) then
          π ⟵ π' ; continuer ⟵ faux
        end if
      end if
      j ⟵ j + 1
    end while
  end if
end for
return(A/π)
```

ce qui préserve l'invariant de la boucle principale. Quant aux conditions techniques $i = \texttt{min}(\pi(i))$ et $j = \texttt{min}(\pi(j))$, elles permettent d'éviter de tenter des fusions qui ont déjà été testées.

3 Test de fusion statistique pour la gestion du bruit

Nous décrivons maintenant notre approche statistique de traitement des données bruitées. Rappelons que nous ne nous intéressons qu'au bruit sur les étiquettes des données, transformant des mots de E_+ en mots de E_-, et réciproquement.

3.1 De l'effet du bruit sur RPNI

Dans RPNI, deux états sont fusionnables si aucun exemple négatif n'est accepté par l'AFD résultant de leur fusion. De manière plus formelle, soit $B = A/\pi = \langle Q, \delta, s_0, F\rangle$ l'AFD résultant de la fusion. Considérons un état s de B et définissons les quantités n_1 et n_2 qui sont respectivement les nombres de mots de E_+ et de E_- contenus par s, *ie*, $n_1 = |\{w \in E_+ \mid \delta(s_0, w) = s\}|$ et $n_2 = |\{w \in E_- \mid \delta(s_0, w) = s\}|$. Dans RPNI, s est final dès que $n_1 > 0$. De plus, pour qu'une fusion réussisse, il faut nécessairement que $(n_1 > 0 \Longrightarrow n_2 = 0)$.

Mais en présence de bruit, il se peut très bien que s contienne plusieurs mots de E_+ et de E_-, que $n_1 > 0$ et $n_2 > 0$, sans pour autant qu'une erreur d'apprentissage soit en train d'être commise. En fait, le bruit sur les étiquettes a deux effets :

1. Lorsqu'un mot w appartient à E_+ du fait du bruit (il serait donc dans E_- en l'absence de bruit), l'état s qui le contiendrait après fusion acquerrait le statut d'état final. Donc tout autre mot non bruité de E_- qui serait contenu par s jouerait

le rôle de contre-exemple à la fusion. Cette fusion serait donc rejetée par RPNI alors qu'elle aurait été acceptée en l'absence de bruit.

2. Lorsqu'un mot w appartient à E_- du fait du bruit (il serait donc dans E_+ en l'absence de bruit), l'état s qui le contiendrait après fusion aurait malgré tout de fortes chances de contenir d'autres exemples de E_+ (non bruités, eux), donc d'être un état final. Le mot w jouerait alors le rôle de contre-exemple à la fusion. De nouveau, cette fusion serait rejetée par RPNI alors qu'elle aurait été acceptée en l'absence de bruit.

De l'analyse précédente, nous avons tiré deux conclusions. D'une part, nous devons relâcher la règle d'attribution du statut d'état final à un état, et nous utiliserons donc désormais la règle de majorité suivante :

Définition 1
On dira qu'un état est positif (ou final) s'il contient strictement plus de mots positifs (de E_+) que de mots négatifs (de E_-) : $s \in F$ ssi $n_1 > n_2$. On dira qu'il est négatif sinon.

D'autre part, nous devons relâcher la contrainte d'acceptation d'une fusion en permettant à un certain nombre de mots de E_+ et de E_- d'être mal classés par l'AFD résultant de la fusion :

Définition 2
1. *On dira qu'un exemple négatif (resp. positif) est mal classé s'il est contenu par un état positif (resp. négatif).*
2. *On dira qu'une fusion est statistiquement acceptable si la proportion p_2 d'exemples mal classés dans tout l'*AFD *après fusion n'est pas significativement plus grande que la proportion p_1 d'exemples mal classés avant fusion.*

Cette manière de faire évite les phénomènes de sur-apprentissage qui sont dus à la présence de bruit et qui conduisent à la construction d'AFD avec un grand nombre d'états et un faible pouvoir de généralisation. Nous acceptons de réduire, par fusion d'états, la taille des AFD appris si elle n'induit pas d'augmentation significative de l'erreur. Comme les fluctuations d'échantillonnage peuvent avoir une influence lors d'une décision d'acceptation ou de rejet d'une fusion, une simple comparaison entre p_1 et p_2 ne serait pas très pertinente. Une règle statistique plus judicieuse consiste à utiliser un test de comparaison de proportions avec des intervalles de confiance.

3.2 Test de comparaison de proportions

Comme nous l'avons vu, nous choisissons d'accepter une fusion si la proportion p_2 d'exemples mal classés par l'AFD après fusion n'augmente pas significativement par rapport la proportion p_1 d'exemples mal classés avant fusion. Nous devons donc tester l'hypothèse nulle $\mathcal{H}_0 : p_1 = p_2$ contre l'hypothèse alternative $\mathcal{H}_a : p_2 > p_1$. Ce test est unilatéral puisque seule une valeur suffisamment grande de notre statistique, $p_2 - p_1$, doit conduire au rejet de l'hypothèse testée. A l'inverse, une petite valeur de $p_2 - p_1$, et plus encore une valeur négative, ne remet pas en cause la qualité d'une fusion. Les quantités p_1 et p_2 sont inconnues : elle correspondent respectivement aux erreurs

théoriques de l'AFD avant et après la fusion. On les estime par les erreurs empiriques $\hat{p_1} = \frac{N_1}{N}$ et $\hat{p_2} = \frac{N_2}{N}$ calculées sur l'ensemble d'apprentissage, où N_1 (resp. N_2) est le nombre d'exemples d'apprentissage mal classés avant (resp. après) la fusion, et N est le nombre total d'exemples (*ie*, le nombre de mots de $E_+ \cup E_-$). $\hat{p_1}$ et $\hat{p_2}$ sont des variables aléatoires indépendantes, et des estimateurs non biaisés de p_1 et p_2.

Dans notre test, nous nous intéressons à la différence $\hat{p_2} - \hat{p_1}$, qui, sous l'hypothèse nulle $\mathcal{H}_0$, a pour espérance et pour variance,

$$E(\hat{p_2} - \hat{p_1}) = p_2 - p_1 = 0$$
$$Var(\hat{p_2} - \hat{p_1}) = \frac{p_2(1-p_2)}{N} + \frac{p_1(1-p_1)}{N} = \frac{2pq}{N}$$

avec $p = p_1 = p_2$ et $q = 1 - p$. On estime habituellement p par la moyenne des deux proportions d'exemples mal classés avant et après fusion : $\hat{p} = (\hat{p_1} + \hat{p_2})/2$. Si les contraintes $Np > 5$ et $Nq > 5$ sont vérifiées [2], les conditions d'approximation par une distribution normale sont satisfaites, donc la variable $T = \hat{p_2} - \hat{p_1}$ suit une loi normale $\mathcal{N}(p_2 - p_1, \sqrt{\frac{2\hat{p}\hat{q}}{N}})$. Nous devons déterminer le seuil Z_α, appelé *valeur critique au risque* α, qui définit la borne du rejet de $\mathcal{H}_0$, et qui correspond au $(1-\alpha)$-percentile de la loi $\mathcal{N}(p_2 - p_1, \sqrt{\frac{2\hat{p}\hat{q}}{N}})$.

$$P(T > Z_\alpha) = P(\frac{T - (p_2 - p_1)}{\sqrt{\frac{2\hat{p}\hat{q}}{N}}} > \frac{Z_\alpha - (p_2 - p_1)}{\sqrt{\frac{2\hat{p}\hat{q}}{N}}}) = P(T^{cr} > \frac{Z_\alpha}{\sqrt{\frac{2\hat{p}\hat{q}}{N}}})$$

$$\text{et donc, } P(T > Z_\alpha) = \alpha \text{ ssi } Z_\alpha = U_\alpha \sqrt{\frac{2\hat{p}\hat{q}}{N}}$$

où T^{cr} est la variable centrée réduite et U_α est le $(1-\alpha)$-percentile de la loi normale $\mathcal{N}(0,1)$. Si $T > Z_\alpha$, on rejette l'hypothèse $\mathcal{H}_0$, donc la fusion, avec un risque $\alpha\%$. Inversement, si $T <= Z_\alpha$, la fusion est statistiquement validée, donc acceptée.

3.3 RPNI*, une extension de RPNI

Nous sommes maintenant en mesure de présenter les modifications que nous apportons à RPNI de sorte qu'il tolère les données bruitées. Nous donnons le pseudo-code de notre nouvel algorithme, RPNI*, dans l'Algo. 2. RPNI* fonctionne comme RPNI, à ceci près qu'il attribue le statut d'état final aux états contenant une majorité d'exemples de E_+, et qu'il utilise le test statistique de comparaison de proportions précédent pour décider de la réussite ou de l'échec d'une fusion. Du point de vue de RPNI*, le risque α est un paramètre de généralisation qu'on peut faire varier entre 0 et 0.5. Une bonne utilisation de ce paramètre consiste donc à tester RPNI* pour différentes valeurs de α, puis à garder le meilleur AFD produit, en terme de taux de succès en généralisation et de nombre d'états.

Notons que plus α est proche de 0, plus Z_α est grand, plus il faut un nombre important d'exemples mal classés pour rejeter une fusion, et donc plus souple est la règle de

[2] On pourra toujours utiliser un test exact de Fisher dans le cas contraire.

Algorithm 2 Pseudo-code de RPNI*

```
A ⟵ PTA(E+) ; π ⟵ {{q} | q ∈ Q}
for i = 1 to N − 1 do
  if i = min(π(i)) then
    j ⟵ 0 ; continuer ⟵ vrai
    while (j < i) and continuer do
      if j = min(π(j)) then
        π' ⟵ calcul_fusion(π, i, j, δ)
        B ⟵ attribue_etat_finaux(A/π', E+, E−)
        if statistiquement_compatible(B, E+, E−, α) then
          π ⟵ π' ; continuer ⟵ faux
        end if
      end if
      j ⟵ j + 1
    end while
  end if
end for
B ⟵ attribue_etat_finaux(A/π, E+, E−)
return(B)
```

fusion. A la limite, quand $\alpha = 0$, RPNI* retourne l'automate universel. A l'inverse, plus α est proche de 0.5, plus Z_α est proche de 0, plus il faut un nombre faible d'exemples mal classés pour rejeter une fusion, et donc plus rigide est la règle de fusion. Quand $\alpha = 0.5$, une fusion est rejetée dès qu'un nouvel exemple est mal classé. Cela signifie qu'en l'absence de bruit, RPNI* se comporte exactement comme RPNI : aucun exemple n'est mal classé par le PTA, et une fusion échoue dès qu'un exemple est mal classé à la suite d'une fusion. RPNI* est donc bien une généralisation de RPNI au cas des données bruitées. Enfin, quand $\alpha > 0.5$, Z_α est négatif, ce qui signifie que même les fusions qui classent correctement tous les mots de l'ensemble d'apprentissage sont rejetées ; RPNI* retourne alors le PTA.

4 Résultats théoriques en présence de bruit

Comme nous l'avons vu précédemment, nous avons obtenu RPNI* à partir de RPNI en changeant la règle d'attribution du statut d'état final d'une part, et en relâchant la contrainte d'acceptation des fusions d'autre part. C'est sur les conséquences de la première règle que nous nous proposons de travailler. On considère donc un AFD A retourné par RPNI* (ou par tout autre algorithme) à partir des ensembles E_+ et E_- de données bruitées. Rappelons qu'alors, un état s de A est positif (final) s'il contient strictement plus d'exemples de E_+ que d'exemples de E_-, et qu'il est négatif sinon.

On peut intuitivement penser que la probabilité que s soit mal étiqueté croît avec le niveau de bruit. C'est le premier résultat théorique que nous établirons. En conséquence, même un algorithme *idéal* qui réussirait à identifier la structure d'un AFD cible à partir

de données bruitées finirait par se tromper sur le signe de ses états, quand le niveau de bruit augmente. D'autre part, nous avons constaté expérimentalement (cf. Section 5) que l'accroissement du niveau de bruit a une conséquence directe sur le taux d'erreur en généralisation des AFD produits par RPNI*. Notre approche nous permettra de justifier théoriquement cette divergence et d'en fournir une estimation, à l'aide de la théorie des marges.

4.1 Probabilité qu'un état soit mal étiqueté

Soit s un état contenant n_1 exemples positifs (dans E_+) et n_2 exemples négatifs (dans E_-). Supposons, sans perte de généralité, que s soit un état négatif à cause du bruit, et qu'il serait positif en l'absence de bruit. Comme s est négatif, on a $n_1 \leq n_2$.

D'autre part, si s est mal étiqueté du fait du bruit, cela signifie que parmi les n_1 exemples positifs, il en existe un certain nombre, n_+, qui sont positifs à cause du bruit, et qui seraient négatifs en l'absence de bruit. De même, parmi les n_2 exemples négatifs, il en existe un certain nombre, n_-, qui sont négatifs à cause du bruit, et qui seraient positifs en l'absence de bruit.

Aussi, en l'absence de bruit, il y aurait $n_1 - n_+ + n_-$ exemples positifs dans s, et $n_2 - n_- + n_+$ exemples négatifs dans s. De plus, s serait un état positif, donc $n_1 - n_+ + n_- > n_2 - n_- + n_+$. Si on suppose enfin que le bruit est uniformément distribué sur l'ensemble d'apprentissage, on en déduit :

Théorème 1
La probabilité $P_s(\gamma, n_1, n_2)$ que l'état s soit mal étiqueté en présence d'un taux γ de bruit est égale à :

$$P_s(\gamma, n_1, n_2) = \sum_{n_+=0}^{n_1} \sum_{n_- > n_+ + \frac{n_2 - n_1}{2}}^{n_2} P(X_1 = n_+) P(X_2 = n_-) \text{ avec}$$

$P(X_1 = n_+) = \binom{n_1}{n_+} \gamma^{n_+} (1-\gamma)^{n_1 - n_+}$, et $P(X_2 = n_-) = \binom{n_2}{n_-} \gamma^{n_-} (1-\gamma)^{n_2 - n_-}$.

Démonstration. Soit X_1 la variable aléatoire du nombre d'exemples mal classés parmi les n_1 exemples positifs. X_1 est le nombre de succès parmi n_1 tirages indépendants avec probabilité γ. Donc X_1 suit une loi binomiale $\mathcal{B}(n_1, \gamma)$ dont la fonction de répartition est $P(X_1 = n_+) = \binom{n_1}{n_+} \gamma^{n_+} (1-\gamma)^{n_1 - n_+}$. De même, soit X_2 la variable du nombre d'exemples mal classés parmi les n_2 exemples négatifs. On a $P(X_2 = n_-) = \binom{n_2}{n_-} \gamma^{n_-} (1-\gamma)^{n_2 - n_-}$. Enfin, les quantités n_+ et n_- sont liées par une contrainte, puisqu'il faut que $n_1 - n_+ + n_- > n_2 - n_- + n_+$, c'est-à-dire, $n_- > n_+ + \frac{n_2 - n_1}{2}$. D'où la formule finale. □

4.2 Divergence théorique de $P_s(\gamma, n_1, n_2)$

Nous montrons maintenant que la probabilité qu'un état s soit mal étiqueté augmente avec le bruit. En conséquence, même si un algorithme idéal réussissait systématiquement à identifier la structure d'un AFD cible, il n'arriverait plus, à partir d'un certain niveau de bruit, à déterminer si les états sont finaux ou non.

Théorème 2
$P_s(\gamma, n_1, n_2)$ est une fonction croissante de γ.

Démonstration. On déduit du Théorème 1 que

$$P_s(\gamma, n_1, n_2) = \sum_{n_+=0}^{n_1} P(X_1 = n_+) P(X_2 > n_+ + \frac{n_2 - n_1}{2})$$

$$= P(X_2 - X_1 > \frac{n_2 - n_1}{2})$$

Comme les fonctions de répartition des lois binomiales sont difficiles à manipuler, nous utilisons ici leurs propriétés de convergence vers les lois normales. Ainsi, X_1 et X_2 suivent asymptotiquement des distributions normales $X_1 \approx \mathcal{N}(n_1\gamma, \sqrt{n_1\gamma(1-\gamma)})$ et $X_2 \approx \mathcal{N}(n_2\gamma, \sqrt{n_2\gamma(1-\gamma)})$. Comme X_1 et X_2 sont indépendantes, la différence $X_2 - X_1$ suit elle aussi une loi normale de paramètres $E(X_2 - X_1) = E(X_2) - E(X_1) = (n_2 - n_1)\gamma$ et $V(X_2 - X_1) = V(X_2) + V(X_1) = (n_1 + n_2)\gamma(1-\gamma)$. Nous en déduisons que :

$$P_s(\gamma, n_1, n_2) = P(X_2 - X_1 > \frac{n_2 - n_1}{2}) = P(\mathcal{N}(0,1) > \frac{\frac{n_2 - n_1}{2} - E(X_2 - X_1)}{\sqrt{V(X_2 - X_1)}})$$

$$= P(\mathcal{N}(0,1) > u) \text{ avec } u = \frac{n_2 - n_1}{2\sqrt{n_1 + n_2}} \frac{1 - 2\gamma}{\sqrt{\gamma(1-\gamma)}}$$

Il est facile de montrer que $\frac{\partial u}{\partial \gamma} = \frac{n_2 - n_1}{4\sqrt{n_1 + n_2}} \frac{-1}{(\gamma(1-\gamma))^{\frac{3}{2}}}$.

Comme $n_1 \leq n_2$, on en déduit que $\frac{\partial u}{\partial \gamma} \leq 0$ pour tout $\gamma \in]0, 1[$. Donc u est une fonction décroissante de γ, et par conséquent $P_s(\gamma, n_1, n_2)$ est une fonction croissante de γ. □

4.3 Expression de la marge à l'aide de $P_s(\gamma, n_1, n_2)$

D'après la section précédente, la probabilité $P_s(\gamma, n_1, n_2)$ qu'un état s soit mal étiqueté augmente avec γ. Comme chaque état est une sorte de sous-classifieur de l'AFD global, nous sommes maintenant en mesure d'évaluer l'impact du bruit sur les performances de ce dernier. Selon la théorie des marges utilisée par les *support vector machines* et le *boosting* (Schapire *et al.*, 1998), la marge de classification pour un exemple w est définie comme la différence entre le poids associé par un classifieur à l'étiquette correcte de w et le poids maximal assigné parmi les étiquettes incorrectes de w. (Schapire *et al.*, 1998) a montré qu'une optimisation de cette mesure de confiance sur l'ensemble d'apprentissage garantissait une amélioration de la borne sur l'erreur de généralisation.

Dans notre cas, on peut utiliser $P_s(\gamma, n_1, n_2)$ pour définir la marge d'un exemple w. En effet, quand l'analyse d'un exemple w termine dans un état s, w hérite de l'étiquette positive ou négative de s. Comme $P_s(\gamma, n_1, n_2)$ est la probabilité que s soit mal étiqueté, w est correctement classé avec une probabilité $1 - P_s(\gamma, n_1, n_2)$, et

mal classé avec une probabilité $P_s(\gamma, n_1, n_2)$. Supposons qu'il y ait n_{max} exemples de la classe majoritaire et n_{min} exemples de la classe minoritaire dans s. Pour les n_{max} exemples, le poids assigné à l'étiquette correcte (resp. incorrecte) vaut donc $1 - P_s(\gamma, n_1, n_2)$ (resp. $P_s(\gamma, n_1, n_2)$), et la marge vaut $m(w) = 1 - 2P_s(\gamma, n_1, n_2)$. De même, pour les n_{min} exemples, le poids assigné à l'étiquette correcte (resp. incorrecte) vaut $P_s(\gamma, n_1, n_2)$ (resp. $1 - P_s(\gamma, n_1, n_2)$), donc la marge est $m(w) = 2P_s(\gamma, n_1, n_2) - 1$.

Notons qu'ainsi définie, la marge est un réel compris entre -1 et $+1$, et qu'un exemple est classé correctement ssi sa marge est positive. De plus, la confiance en l'étiquette d'un exemple est d'autant plus grande que sa marge est proche de $+1$. Pour les n_{max} exemples, la marge vaut $+1$ quand $P_s(\gamma, n_1, n_2) = 0$, c'est-à-dire quand le bruit γ vaut 0. Quand le bruit croît, nous avons montré dans la section précédente que $P_s(\gamma, n_1, n_2)$ augmentait elle aussi, ce qui se traduit, compte tenu des formules précédentes, en une diminution de la marge pour les n_{max} exemples et une augmentation pour les n_{min} exemples. Puisque $n_{max} \geq n_{min}$, la moyenne des marges décroît pour chaque état, ce qui se traduit donc en une augmentation de l'erreur en généralisation sur tout l'AFD.

En exprimant la marge avec $P_s(\gamma, n_1, n_2)$, nous pouvons finalement faire les remarques suivantes. Tout d'abord, comme nous allons le voir dans la section expérimentale, de petites valeurs de γ induisent souvent une valeur nulle de $P_s(\gamma, n_1, n_2)$, donc une marge de $+1$. Cela signifie que malgré la présence d'exemples mal classés pendant la phase d'apprentissage, on peut théoriquement inférer un AFD ne commentant aucune erreur en généralisation. Ensuite, plus γ augmente, plus $P_s(\gamma, n_1, n_2)$ diffère de 0, ce qui justifie, même pour un algorithme qui serait optimal, une divergence de l'erreur en généralisation des AFD qu'il produit.

5 Résultats expérimentaux

Dans cette section, nous évaluons l'efficacité de RPNI* en fonction de deux mesures de performances, à savoir le pouvoir de généralisation et le nombre d'états des AFD inférés. Pour ce faire, nous avons mené trois types de tests. Le premier vise à étudier le comportement de RPNI* sur une base de données spécifique en fonction de différents niveaux de bruit. Dans la seconde série d'expérimentations, nous comparons RPNI* et RPNI sur deux types de bases de données, pour un certain niveau de bruit fixé : (i) huit bases synthétiques générées artificiellement à l'aide du simulateur Gowachin[3], et (ii) trois bases de l'UCI[4] ainsi qu'une base de prénoms français. Pour finir, nous étudions la tolérance au bruit de RPNI* sur les bases précédentes en faisant varier le bruit de 0 à 20%. Toutes nos expérimentations ont été menées à l'aide d'une procédure de cross-validation (sur 10 folders) pour estimer l'erreur de généralisation. Pour vérifier si l'AFD cible était correctement inféré, nous avons seulement bruité les ensembles d'apprentissage (soit l'union de neuf folders sur dix) à chaque étape sans toucher à l'ensemble de test (le dixième folder) utilisé en validation.

[3] http ://www.irisa.fr/Gowachin

[4] http ://www.ics.uci.edu/~mlearn/MLRepository.html

5.1 Validation des propriétés théoriques

Notre objectif est ici de vérifier expérimentalement les remarques faites dans la section précédente. Pour cela, nous utilisons une base non bruitée, BASE1, simulée avec Gowachin. L'utilisation de RPNI sur cette base permet d'inférer l'AFD cible, qui a 9 états et qui ne commet aucune erreur en généralisation. Puis nous faisons varier le taux de bruit γ de 0 à 20% et comparons RPNI et RPNI* en terme de taux de généralisation et de nombre d'états des AFD inférés.

Dans la Fig. 3, on note que jusqu'à 6% de bruit, RPNI* est capable de "digérer" complètement les données bruitées : sur les dix folders de la procédure de cross-validation, RPNI* infère systématiquement la cible, garantissant ainsi un taux de succès de 100%. Cela confirme donc que les marges sont égales à $+1$ pour de petites valeurs du bruit, malgré la présence de certains mots mal classés dans les états. De 6% à 10%, RPNI* retourne des AFD qui ne font pas d'erreurs en généralisation malgré des nombres d'états légèrement supérieurs à 9. Au delà de 10%, le niveau de bruit est trop important pour être totalement digéré, ce qui induit une légère divergence des performances en terme de taux de succès et de nombre d'états. Cela confirme donc nos résultats théoriques, même si cette dégradation est relative, et que globalement, le taux de succès est très acceptable (environ 93% et 11 états en moyenne pour 20% de bruit). Concernant RPNI, les conséquences du bruit dans les données sont évidentes. Dès qu'un peu de bruit apparaît, la dégradation est perceptible. Non seulement le taux de succès chute régulièrement, mais la taille des AFD inférés croît, ce qui confirme le phénomène de sur-apprentissage.

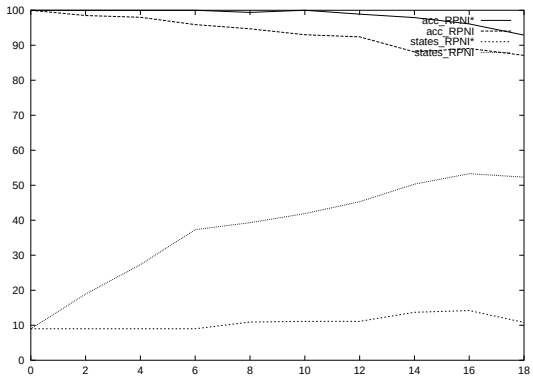

FIG. 3 – Résultats obtenus par 10-cross-validation sur BASE1 : acc_RPNI* et acc_RPNI représentent le taux en généralisation de RPNI* et RPNI ; states_RPNI* et states_RPNI représentent le nombre d'états moyen des AFD inférés avec RPNI* et RPNI.

5.2 Comportements moyens de RPNI* et de RPNI

Nous avons réalisé une étude comparative plus importante entre RPNI* et RPNI, sur douze bases, pour un niveau de bruit fixé à $\gamma = 4\%$. Notre but est de montrer des résultats significatifs, présentés dans la Table 1, de l'efficacité de RPNI* sur un large panel de données. Notons que pour les huit bases simulées BASE1, ..., BASE8, nous connaissons *a priori* les AFD cibles car RPNI est capable de les inférer en l'absence de bruit. Ce n'est pas le cas pour les quatre autres bases, AGARICUS, BADGES, PROMOTERS et FIRSTNAME, pour lesquelles nous ne connaissons pas d'AFD cible, et qui sont déjà probablement bruitées. Chaque valeur de la Table 1 est une moyenne calculée par cross-validation.

TAB. 1 – Résultats de RPNI* et de RPNI sur douze bases. Les moyennes et les écarts-types sont calculés par 10-cross-validation. Le nombre d'états de l'AFD cible est donné entre parenthèse quand il est connu.

	RPNI*		RPNI	
BASES	TAUX DE SUCCÈS	NB D'ÉTATS	TAUX DE SUCCÈS	NB D'ÉTATS
BASE1 (9)	100 ± 0	9 ± 0	98.0 ± 1.15	27.3 ± 5.3
BASE2 (9)	94.8 ± 8.0	14.1 ± 2.2	86.3 ± 4.5	39.8 ± 2.6
BASE3 (21)	94.1 ± 4.0	29.2 ± 2.8	86.5 ± 3.2	192.7 ± 5.5
BASE4 (38)	99.7 ± 0.4	59.5 ± 4.5	92.2 ± 1.1	252.5 ± 10.7
BASE5 (38)	89.5 ± 5.8	52.2 ± 4.9	85.2 ± 2.5	284.6 ± 9.5
BASE6 (10)	100 ± 0	10 ± 0	99.8 ± 0.2	17.4 ± 5.5
BASE7 (12)	100 ± 0	16.5 ± 0.5	95.8 ± 1.1	131.4 ± 10.7
BASE8 (15)	100 ± 0	15 ± 0	98.9 ± 2.5	77 ± 9.5
AGARICUS	88.9 ± 1.7	82.7 ± 3.3	88.9 ± 1.7	82.7 ± 3.3
BADGES	72.1 ± 7.6	2.3 ± 0.4	55.4 ± 10.5	8.7 ± 0.9
PROMOTERS	51.9 ± 15.7	5.3 ± 0.5	48.8 ± 21.7	23.1 ± 0.9
FIRSTNAME	66.5 ± 8.8	3.5 ± 0.5	59.5 ± 9.1	10.0 ± 0.6
MOYENNE	88.1	25.6	82.9	95.6

Plusieurs remarques intéressantes peuvent être faites. Tout d'abord, les différences entre RPNI* et RPNI sont significatives pour dix bases sur douze (ce qu'on a vérifié à l'aide d'un test de Student apparié sur les taux de succès). Seules AGARICUS et BASE6 ne satisfont pas cette contrainte statistique. Mais on peut de toute façon constater que RPNI ne fait jamais mieux que RPNI*, en particulier lorsqu'on s'intéresse à la taille des AFD inférés. Cette différence est encore plus significative quand on fait la moyenne générale des taux de succès (88.1% contre 82.9%) et des nombres d'états. Sur les douze bases, on constate que le nombre d'états reste toujours relativement faible avec RPNI* (25.6 états en moyenne) alors que la moyenne avec RPNI (95.6) souligne le phénomène de sur-apprentissage liés aux données bruitées. Pour finir, notons qu'en présence de 4% de bruit, RPNI* est capable d'identifier l'AFD cible pour trois bases (BASE1, BASE6 et BASE8), et qu'il fait 100% en généralisation sur une quatrième (BASE7).

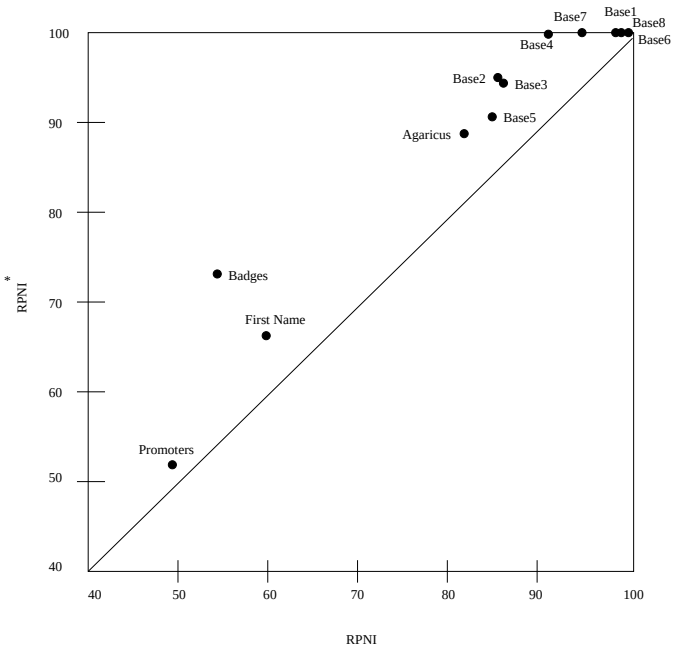

FIG. 4 – Scatterplot sur les douze bases.

Nous proposons une autre représentation de nos résultats dans la Fig. 4. Chaque base est représenté par un point dont l'abscisse, x, est le taux de succès de RPNI, et y, le taux de succès de RPNI*. Remarquons que tous les points sont aux dessus de la droite d'équation $y = x$, ce qui montre que RPNI* est toujours meilleur que RPNI.

5.3 Tolérance au bruit de RPNI*

Nous étudions enfin la tolérance de RPNI* pour différents niveaux de bruit. Dans ces dernières expérimentations, nous faisons varier le bruit γ de 0 à 20%. Les résultats sont présentés dans la Fig. 5. Notons que chaque point des courbes est une moyenne sur les douze bases précédentes. Globalement, la différence entre RPNI et RPNI* croît avec le niveau de bruit, et toujours en faveur du second. Cela signifie que RPNI est toujours moins performant que RPNI*, et qu'en plus, il se dégrade plus rapidement. Concernant le taux en généralisation, nous remarquons que les performances des deux algorithmes se dégradent logiquement avec le niveau de bruit, ce qui confirme de nouveau les propriétés de divergence que nous avons établies dans la section précédente. Enfin, tandis que RPNI* semble en mesure de contrôler effectivement la taille des AFD (de 18 à 30 états en moyenne), RPNI souffre largement de l'accroissement du bruit, retournant des AFD ayant jusqu'à 160 états en moyenne, pour 20% de bruit.

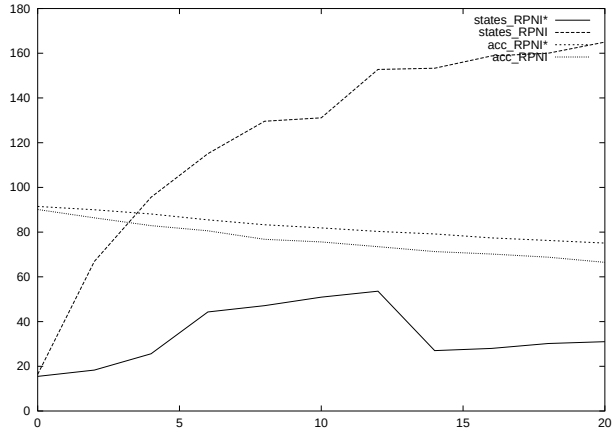

FIG. 5 – Taux de succès et nombres d'états en fonction de différents niveaux de bruit.

6 Conclusion et perpectives

Dans ce papier, nous avons proposé une approche statistique de traitement des données bruitées en inférence grammaticale. Nous l'avons appliqué à RPNI, dont la règle de fusion est relativement simple. D'autres algorithmes, comme EDSM (Lang *et al.*, 1998), utilisent des règles de fusion plus sophistiquées. Il nous paraît pertinent d'adapter notre approche à ce type d'algorithmes, et d'exploiter plus finement les caractéristiques de notre test statistique (par exemple, les erreurs de première et de seconde espèce).

D'autre part, nous avons vu que RPNI* présentait une bonne tolérance au bruit jusqu'à un certain niveau (environ 10% pour les bases artificiellement générées). Nous pensons qu'il est possible d'améliorer ce seuil en étudiant la *nature* des exemples mal classés avant de faire une fusion. En fait, les erreurs de classement sur l'ensemble d'apprentissage sont dues à deux phénomènes. Le premier est directement lié à la présence de bruit : certains exemples sont mal classés par l'AFD parce qu'ils sont bruités alors qu'ils seraient bien classés en l'absence de bruit. Le second phénomène est lié au fait qu'on a relâché les contraintes d'une fusion : certains exemples mal classés sont de vrais contre-exemples qui devraient remettre en cause une fusion ; c'est un rôle qu'ils ne jouent plus puisque notre règle de fusion est plus lâche.

Pour finir, nous avons supposé dans ce papier que le bruit n'affectait que les étiquettes, positive ou négative, d'un mot. Si cette hypothèse n'est pas trop forte sur un large panel de problèmes du monde réel, nos expérimentations ont montré que notre méthode n'était pas adaptée au cas où ce sont les lettres des mots qui sont bruitées. Nous pensons néanmoins qu'il est possible de l'étendre pour traiter ce type de bruit.

Remerciements

Nous remercions tous nos relecteurs de CAP'03 : par la précision et la qualité de leurs suggestions, ils nous ont donné de nouvelles idées pour poursuivre ce travail.

Références

AHA D. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *Int. Journal Man-Machine Studies*, p. 267–287.

BRODLEY C. & FRIEDL M. (1996). Identifying and eliminating mislabeled training instances. In *13th Nat. Conf. on Artif. Intell.*, p. 799–805.

CORNUÉJOLS A. & MICLET L. (2002). *Apprentissage artificiel - Concepts et algorithmes*. Eyrolles.

COSTE F. (1999). *State Merging Inference of Finite State Classifiers*. Rapport interne, Publication interne n° 1250.

DE LA HIGUERA C. (2002). *A Bibliography of Grammatical Inference*. Rapport interne, Rapport de Recherche EURISE RR 0301.

DUPONT P. & MICLET L. (1998). *Inférence grammaticale régulière : fondement théoriques et principaux algorithmes*. Rapport interne, Rapport de Recherche INRIA RR 3449.

JOHN G., KOHAVI R. & PFLEGER K. (1994). Irrelevant features & the subset selection problem. In *11th Int. Conf. on Mach. Learn.*, p. 121–129.

LANG K., PEARLMUTTER B. & PRICE R. (1998). Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *4th Int. Coll. on Grammatical Inference*, p. 1–12.

LANGLEY P. (1994). Selection of relevant features in machine learning. In *AAAI Fall Symp. on Relevance*.

ONCINA J. & GARCÍA P. (1992). *Inferring Regular Languages in Polynomial Update Time*, In *Pattern Recognition and Image Analysis*, volume 1 of *Machine Perception and Artificial Intelligence*, p. 49–61. World Scientific.

SCHAPIRE R., FREUND Y., BARTLETT P. & LEE W. (1998). Boosting the margin : a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, **26**, 1651–1686.

SEBBAN M., JANODET J.-C. & YAHIAOUI A. (2002). Effets de la suppression des mots bruités en inférence grammaticale. In *7ème Conf. Nat. de Classification*, p. 311–314.

SEBBAN M. & NOCK R. (2000). Instance pruning as an information preserving problem. In *17th Int. Conf. on Machine Learning*, p. 855–862.

SEBBAN M., NOCK R. & LALLICH S. (2003). Stopping criterion for boosting-based data reduction techniques : from binary to multiclass problems. *Int. Journal of Machine Learning Research*, **3**, 863–885.

WILSON D. & MARTINEZ T. (1997). Instance pruning techniques. In *14th Int. Conf. on Machine Learning*, p. 404–411.