Techniques d'adaptation au scripteur pour la lecture de textes manuscrits dynamiques

Loïc Oudot, Lionel Prevost, Alvaro Moises

Techniques d'adaptation au scripteur pour la lecture de textes manuscrits dynamiques

Loïc Oudot – Lionel Prevost – Alvaro Moises

Laboratoire des Instruments et Systèmes d'Ile de France
4 Place Jussieu
75252 Paris Cedex 5

loic.oudot@lis.jussieu.fr

Résumé : *Nous avons récemment développé un système complet de reconnaissance de textes manuscrits saisis sur tablette à digitaliser. Ce système se base sur le modèle cognitif d'activation vérification et s'articule autour de trois experts dédiés respectivement à la segmentation du signal écrit en symboles, la classification des symboles et l'analyse lexicale des résultats de classification.*

Nous présentons dans cet article plusieurs stratégies d'adaptation non-supervisées au scripteur que nous comparons aux méthodes d'adaptation supervisées. Nous proposons une stratégie de modification dynamique non-supervisée des paramètres du classifieur permettant d'obtenir des résultats proches des méthodes supervisées. Les résultats obtenus sur une base de données de 90 textes (5 400 mots) écrits par 38 scripteurs différents sont très encourageants : une combinaison de plusieurs stratégies permet d'atteindre un taux de reconnaissance de 90 %.

Mots-clés : Adaptation non supervisée, lecture automatique, écriture dynamique scripte

1 Introduction

Nous disposons au sein de notre laboratoire d'un moteur de lecture de l'écriture scripte omni-scripteur performant mais encore insuffisant pour un usage régulier. L'adaptation de ce système à l'écriture d'un utilisateur particulier permet d'améliorer grandement ses performances. Il ne faut toutefois pas oublier que l'utilisation d'un stylo comme moyen d'interaction est de simplifier l'utilisation des appareils et qu'un apprentissage de la part du scripteur pour l'adaptation est à minimiser. Les méthodes classiques demandent une intervention humaine pour cet apprentissage. Nous proposons dans cet article plusieurs méthodes d'adaptation non supervisées au scripteur et les comparons aux techniques déjà existantes de type supervisé ou à enrolement.
Une revue des différentes techniques d'adaptation supervisée et non supervisée est présentée dans la seconde partie. Le système de lecture est présenté dans la troisième partie. Nous décrivons les techniques d'adaptation dans la quatrième partie et terminons par les conclusions et perspectives

2 Etat de l'art

Le principe de l'adaptation au scripteur à été mis en lumière par les travaux en psychologie perceptive. On peut remarquer que dans le cas d'une écriture difficile à lire, on lira plus facilement un mot d'une personne si on a auparavant lu un autre mot de cette même personne. Il s'agit d'un effet d'amorçage graphémique. Il est facile de constater ce phénomène dans la pratique lors de la lecture d'un texte très peu lisible. Quand un mot est illisible ou ambiguë, nous recherchons instinctivement dans ce texte, d'autres mots que nous pouvons lire pour comprendre ce premier mot. Nous apprenons donc les caractéristiques du scripteur à partir de mots que nous avons pu lire, pour ensuite utiliser ces nouvelles connaissances sur le reste des mots auparavant illisibles.
Dans la littérature on considère deux stratégies d'adaptation : les systèmes où l'adaptation a lieu une fois pour toute avant utilisation (hors-ligne ou *batch*) et les système à adaptation continue (en-ligne).

2.1 Adaptation hors-ligne

La plupart des systèmes utilisant une adaptation hors-ligne possèdent une base de données du scripteur étiquetée. Ces exemples permettent de réaliser un apprentissage supervisé(enrolement) du système, c'est-à-dire d'apprendre le style d'écriture du scripteur, avant son utilisation.
Dans [LI 02] les lettres sont décrites par plusieurs modèles neuronaux markoviens en parallèles, chacun ayant appris un certain allographe de lettre. Ces derniers sont déterminés par un algorithme de clustering sur la base d'apprentissage. Ce système s'adapte au scripteur en réalisant une modélisation plus fine des lettres en ré-estimant les paramètres des modèles sur les exemples étiquetés. Le système décrit par [SCH 93] utilise également un classifieur automatique de *strokes* par cartes de Kohonen pour identifier les primitives utilisées par le scripteur. Ainsi le classifieur de caractères n'utilise plus que les primitives utilisateurs au lieu de celles omni-scripteur. Dans [BRA 01] les HMM sont entraînés dans un premier temps sur une base omni-scripteur puis spécialisés à un scripteur particulier. Dans [CON 02], en plus de la ré-estimation des paramètres, de nouveaux modèles de caractères sont créer si nécessaire.

2.2 Adaptation en-ligne

Contrairement aux systèmes décrit plus haut qui sont figés une fois l'adaptation effectuée, les systèmes suivants évoluent en permanence au cours de l'utilisation.
Le système de lecture en-ligne avec adaptation au scripteur

de [PLA 97] utilise une méthode d'adaptation incrémentale supervisée. Le système de base est omni-scripteur et utilise un unique réseau de neurones de type MLP (RN) possédant 72 classes de sortie (62 lettres et 10 signes de ponctuations). Un module d'adaptation disposé en sortie du RN permet de modifier son vecteur de sortie. Ce module d'adaptation est constitué par un réseau de type RBF (*Radial Basis Function*). A chaque erreur de classification, l'utilisateur prévient le système de l'erreur ce qui entraîne l'apprentissage des RBF (ré-estimation des centres existants ou ajout d'un nouveau centre) à partir de l'étiquette entrée par l'utilisateur.
Deux autres systèmes assez proches utilisent un TDNN (*Time Delay Neural Network*) comme classifieur à la place du RN. Ce TDNN est appris sur une base omni-scripteur et la couche de sortie de ce réseau est remplacée soit par un classifieur de type k-pvv [GUY 92] soit par un classifieur discriminant [MAT 93].
Le système de [VUO 02] est très proche du notre mais se limite à la reconnaissance de caractères alphanumériques isolés. Le classifieur de caractères est de type k-ppv et utilise comme métrique une distance élastique. L'adaptation consiste à ajouter les caractères non reconnus dans la base de prototypes. Les prototypes non utilisés peuvent être supprimés de la base pour éviter une croissance excessive de celle-ci.

3 Système de lecture

Pour les expérimentations, nous avons collecté une grande base de textes écrits par 38 scripteurs différents. Les scripteurs ayant écrit en moyenne 150 mots, cette base représente 5400 mots et 26000 lettres. Tous les textes ont été étiquetés par un expert humain. Les techniques d'adaptation présentées ici sont itératives : les performances du système s'améliorent pendant l'utilisation. Nous présenterons donc l'évolution du taux de reconnaissance sur trois plages correspondant respectivement à 50, 100 et 150 mots utilisés pour l'adaptation.
Le classifieur de caractère de type 1-ppv utilise une base de prototypes de *strokes* omni-scripteur construite à l'aide d'un algorithme de clustering [PRE 00] à partir des exemples de la base UNIPEN sur le corpus Train-R01/V07. Cette base de caractères contient environs 3000 prototypes pour les 62 classes (26 majuscules, 26 minuscules et 10 chiffres).
Enfin, nous utilisons pour l'analyse lexicale un dictionnaire contenant les 8 000 mots les plus fréquents de la langue française pour obtenir une vitesse de traitement de 6 mots par seconde (P4 1,8GHz Matlab) et un encombrement mémoire réduit de 500Ko (en incluant le système, le lexique et la base de prototypes).
La structure générale de notre moteur de lecture est schématisée figure 1. Le modèle cognitif d'activation vérification de Paap [PAA 82] a servi de schéma de principe. Le système se présente sous la forme d'une série d'experts d'encodage [OUD 01] qui permettent d'extraire les informations de type géométrique et morphologique du texte d'entrée. Ces experts fournissent des informations probabilistes au niveau *stroke*. Toutes ces informations, aussi bien au niveau local qu'au niveau global permettent d'activer une liste de mots du lexique. La cohérence de chaque mot hypothèse de la liste est ensuite évaluée par un moteur probabiliste qui valide la meilleure hypothèse pour la retranscription.

4 Adaptation par modification des paramètres de classification

Lors de l'étude du moteur de lecture [OUD 04], nous avons pu constater que la base de prototypes du classifieur, aussi riche soit elle, n'est pas encore assez représentative, et l'utilisation des caractères issus de la base de textes d'apprentissage pour une utilisation en mode multi-scripteur, permet d'améliorer les taux de reconnaissance. Il y a deux sources majeures d'erreurs de classification :
– Le graphème est absent de la base omni-scripteur, il faut donc enrichir cette dernière : le graphème est stocké dans la base de prototypes (Ajout).
– Le graphème est confusif : pour un scripteur donné, les modèles de la base omni-scripteur doivent être supprimés afin d'éviter toute confusion (Suppression).
Les classifieurs à prototypes (1-ppv) peuvent s'adapter aisément en ajoutant un nouvel exemple de caractère dans la base de prototypes utilisateur et en désactivant les prototypes existants.

4.1 Adaptation supervisée

Pour évaluer la pertinence de l'adaptation au scripteur, nous avons utilisé les étiquettes des textes pour effectuer une adaptation supervisée en segmentation connue. Les formes du texte sont classifiées l'une après l'autre. Les hypothèses de classification (qui correspondent aux meilleurs réponses, top_1, du classifieur de caractères) sont comparées aux étiquettes. Si elles ne correspondent pas, la forme en question est ajoutée à la base utilisateur (figure 2). Deux approches ont été envisagées : l'approche *texte* où les formes sont ajoutées en fin de texte et l'approche *ligne* où les formes sont ajoutées en fin de chaque ligne. Les résultats (tableau 1) montrent les gains d'adaptation de notre système de lecture lorsque la segmentation du texte en mots et en lettres est connue.

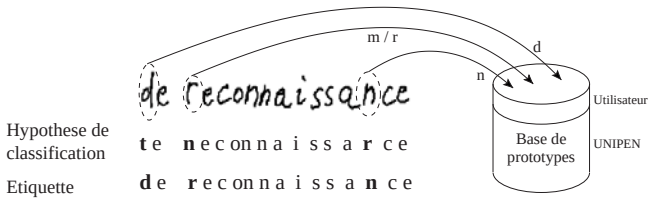

FIG. 2 – Ajout de prototypes dans la base utilisateur (comparaison étiquette hypothèse de classification).

L'approche *ligne* permet une amélioration plus rapide du taux de reconnaissance tout en ajoutant moins de prototypes par scripteur que l'approche *texte*. En effet, en ajoutant les prototypes à la fin de l'analyse complète d'un texte, on peut ajouter plusieurs prototypes semblables d'un même caractère (d'où l'augmentation du nombre moyen de prototypes). L'approche ligne, par contre, ajoute le premier prototype du caractère mal classifié. Les caractères suivants étant reconnus grâce à cet ajout, ils ne viennent pas modifier la base de

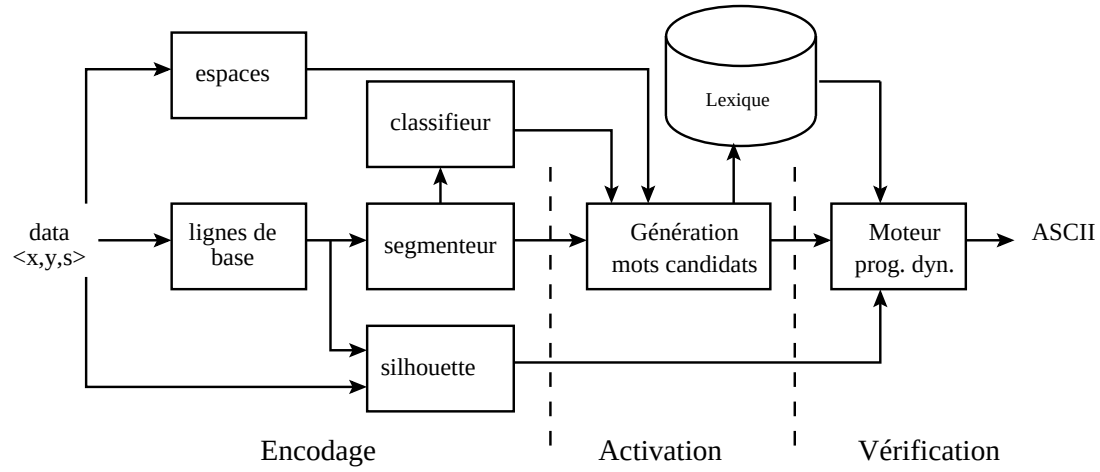

FIG. 1 – Structure générale du système de lecture.

	Taux d'erreur			Base utilisateur
Mots	50	100	150	
Sans adaptation	15 %			100 %
App. texte : min	0 %	0 %	0 %	+3 %
moy	1.3 %	1.1 %	0.6 %	+6 %
max	10 %	5.1 %	4.5 %	+9 %
App. ligne : min	0 %	0 %	0 %	+2 %
moy	**1.1 %**	**0.7 %**	**0.4 %**	**+4 %**
max	6.2 %	5.2 %	3.7 %	+8 %

TAB. 1 – Evolution du taux d'erreur mots (segmentation connue, adaptation supervisée). La dernière colonne correspond au taux de croissance de la base de prototypes utilisateur.

prototypes. Le nombre de prototypes ajoutés est plus faible. D'un point de vue perceptif, cet ajout de prototypes mime – au niveau lettre – l'effet d'amorçage par répétition constaté au niveau mot : la présentation antérieure d'un mot réduit la quantité d'information nécessaire à son identification ultérieure et rend cette dernière plus rapide.

4.2 Adaptation non-supervisée

L'adaptation non-supervisée se fait de façon complètement transparent pour le scripteur qui n'est plus sollicité. Les informations du classifieur de caractères et du correcteur lexical sont comparées pour déterminer les prototypes à ajouter (figure 2).

4.2.1 Ajout systématique

Dans la stratégie d'adaptation par ajout systématique, on considère que le taux d'erreur du correcteur lexical est nul. Donc, à chaque erreur (différence entre l'hypothèse de classification et l'hypothèse lexicale) le caractère est ajouté à la base de prototypes utilisateur. Les erreurs de correction lexicale cumulées à celles de segmentation, conduisent à l'ajout de prototypes dans de mauvaises classes. Ces erreurs d'ajouts entraînent de nombreuses erreurs de classifications par la suite. Les performances du système de lecture s'en ressentent et diminuent drastiquement (tableau 2) mais restent tout de même meilleur qu'en l'absence d'adaptation.

4.2.2 Ajout conditionnel

La procédure d'ajout systématique étant peu performante, il semble nécessaire d'étudier le comportement de l'analyseur lexical pour n'ajouter que des prototypes utiles. Deux conclusions s'imposent lors de cette étude :

– La fiabilité de l'analyseur lexical augmente avec le nombre de lettres composant le mot.
– La fiabilité de l'analyseur lexical diminue lorsque le nombre d'erreur (lettres du mot mal classées) de classification augmente.

Pour un mot donné, on définit un *critère de fiabilité lexicale* pour estimer la probabilité d'erreur du correcteur lexical. Ce critère est basé sur deux paramètres : le nombre de lettres du mot ($nbTot$) et le nombre d'erreurs de classification (défini comme le nombre total de différence entre les hypothèses du classifieur et du lexique : $nbDif$). La stratégie d'adaptation par ajout conditionnel est définie comme suit. Si, pour un mot donné, la relation 1 est vérifié, alors les lettres différentes entre le mot du lexique et le mot reconnu sont ajoutées à la base utilisateur.

$$nbDif \leq \mathrm{E}(nbTot/K) \quad K \in \mathbb{N}^{+} \tag{1}$$

Le paramètre K est déterminé par recherche exhaustive. La valeur $K = 4$ montre que plus le mot est long meilleur est la correction lexicale.

	Taux d'erreur			Base utilisateur
Mots	50	100	150	
Sans adap.	28 %			100 %
Systématique : min	0 %	1.9 %	2 %	+2 %
moy	25 %	23 %	23 %	+6 %
max	53 %	73 %	51 %	+14 %
Conditionnel : min	0 %	0 %	2 %	+1 %
moy	**22 %**	**20 %**	**17 %**	**+2 %**
max	71 %	58 %	43 %	+3 %

TAB. 2 – Ajout conditionnel : évolution du taux d'erreur mots et de la base utilisateur

L'ajout conditionnel permet de réduire considérablement les mauvais ajouts de prototypes de l'ajout systématique. On voit très nettement l'amélioration du taux de reconnaissance avec un gain d'adaptation de 35 %. Cette méthode n'est toutefois pas encore complètement satisfaisante, car les prototypes de caractères des mots de moins de quatre lettres ne sont jamais traités du fait du critère de fiabilité. Or, ces derniers représentent 45 % des mots.

4.2.3 Gestion dynamique des prototypes

Cette stratégie a deux buts. La suppression des prototypes erronés dus aux erreurs lors de l'ajout conditionnel (erreur du correcteur lexical) et la réduction de la taille de la base utilisateur en supprimant les prototypes inutiles [VUO 02] afin d'accélérer la vitesse de classification. On attribue à chaque prototype (de la base omni-scripteur comme de la base utilisateur) une adéquation initiale ($Q(0) = 1000$). Cette adéquation sera modifiée à chaque occurrence des prototypes suivant l'utilité de ces derniers dans le processus de classification en comparant l'hypothèse de classification à l'hypothèse lexicale (équation 2). Considérons le prototype i de la classe j, trois paramètres sont nécessaires au déroulement de ce processus :

- **C** : Récompense (+) du prototype i quand l'hypothèse de classification est correcte.
- **I** : Pénalisation (-) du prototype i quand l'hypothèse de classification est incorrecte.
- **N** : Pénalisation (-) pour les prototypes inutilisés de la classe j.

$$Q_j^i(n+1) = Q_j^i(n) + [C(n) - I(n) - N(n)]/F_j \quad (2)$$

Avec F_j la fréquence de la classe j dans la langue française. Les trois paramètres sont mutuellement exclusifs *i.e.* à chaque occurrence, seul un paramètre est activé. Lorsque $Q_j^i = 0$, le prototype est éliminé.
Après recherche exhaustive des paramètres (C, I, N) optimaux pour une combinaison avec la méthode d'ajout conditionnel, les meilleurs résultats sont obtenus avec la configuration (30, 200, 8). La base subit une grande réduction de sa taille tout en conservant le taux de reconnaissance de l'adaptation par ajout conditionnel.

	Taux d'erreur	Base utilisateur
Sans adap.	28 %	100 %
Gestion dyn.	**17 %**	**-40 %**

TAB. 3 – Gestion dynamique des prototypes : évolution du taux d'erreur et de la base utilisateur

Nous allons étudier l'évolution de l'adéquation de certains prototypes (figure 3). Pour certain scripteur, les prototypes omni-scripteur sont suffisants. Pour la classe 'a', 2 prototypes sont utilisés et donc l'adéquation des 45 autres décroît. Pour la classe 's', 4 prototypes sont utiles (le scripteur a probablement une écriture instable, voir figure 4) et les 36 autres sont inactifs. Pour les autres scripteurs (classe 's' et 'e'), des prototypes utilisateurs (en gras) sont nécessaires. Au départ, un prototype omni-scripteur est utilisé et après quelques occurrences, un prototype utilisateur est ajouté (l'utilisateur s'est

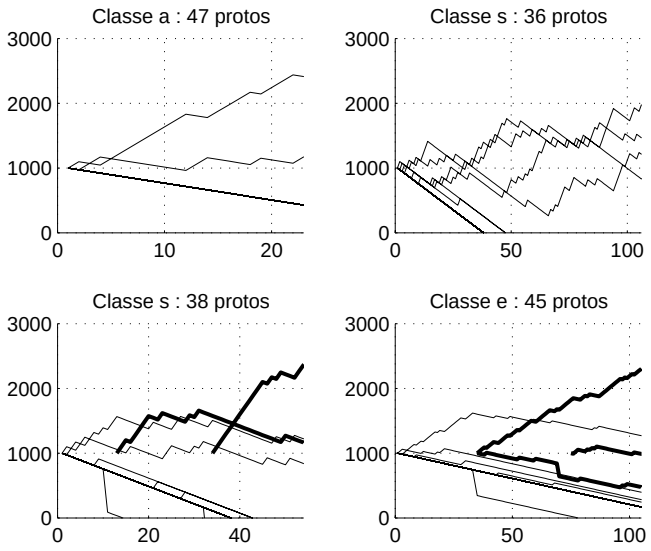

FIG. 3 – Evolution de l'adéquation des prototypes fonction du nombre d'occurence.

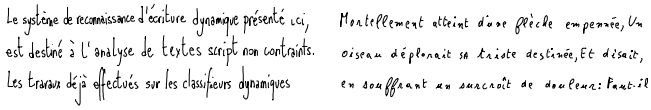

FIG. 4 – Scripteurs les mieux et moins bien reconnus (respectivement 99 % et 70 % de taux de reconnaissance).

habitué à la tablette). Après 150 mots, la taille de la base utilisateur a été réduite de 40 %. Une régression linéaire peut raisonnablement amener cette réduction à plus de 90 % de la talle initiale.

4.3 Combinaison supervisée / non-supervisée

Les performances du système de lecture réel (segmentation inconnue) en utilisant la base de prototype utilisateur déterminé auparavant) atteint 12 % d'erreur au lieu de 28 % du système seul sans adaptation.
L'adaptation non supervisée atteint presque les performances du système après enrolement (17 % contre 12 %) sans aucune intervention de l'utilisateur. Solliciter l'utilisateur pour l'écriture de 150 mots est beaucoup trop contraignant, en revanche lui faire écrire quelques dizaines de mots est acceptable surtout si le taux de reconnaissance est grandement amélioré. Cette combinaison consiste à réaliser une adaptation supervisée du système sur quelques mots puis à utiliser la procédure d'adaptation non-supervisée par gestion dynamique.
Demander à l'utilisateur d'écrire une phrase de 30 mots permet d'améliorer le taux de reconnaissance à plus de 90 %.

Conclusions et perspectives

Nous avons présenté plusieurs stratégie d'adaptation non-supervisée. Le classifieur à base de prototypes originellement entraîné à reconnaître l'écriture dans le cadre omni-scripteur peut très facilement, grâce à sa structure, apprendre de nouvelles écritures. Le processus d'adaptation combinant les stratégies non-supervisée et supervisée permet de dimi-

Mot enrolement	Taux de reconnaissance	
	Après enrolement	100 mots
0	70 %	76 %
10	74 %	83 %
20	74 %	88 %
30	**74 %**	**90 %**
50	75 %	91 %

TAB. 4 – Taux de reconnaissance, adaptation semi-supervisée

nuer le taux d'erreur (de 28 % à 10 %) et d'accroître la vitesse de classification (près du double). Et donc de recentrer une base de prototypes omni-scripteur en une base mono-scripteur de très grande qualité et de faible encombrement mémoire de manière automatique.
Il serait intéressant d'évaluer une stratégie semi-supervisée où l'utilisateur n'est sollicité que dans les cas ambigus. Il s'agit maintenant d'adapter l'étape de segmentation qui génère le plus d'erreur : une ré-estimation des paramètres des experts d'encodage devrait être bénéfique.

Références

[BRA 01] BRAKENSIEK A., KOSMALA A., RIGOLL G., Comparing adaptation techniques for on-line handwriting recognition, *ICDAR*, vol. 1, 2001.

[CON 02] CONNEL S. D., JAIN A. K., Writer Adaptation of Online Handwriting Models, *IEEE Transaction PAMI*, vol. 24, n° 3, 2002, pp. 329–346.

[GUY 92] GUYON I., HENDERSON D., ALBRECHT P., CUN Y. L., DENKER J., *Writer independent and writer adaptative neural network for on-line character recognition*, S. Impedovo, From Pixels to Featuress III, Elsevier, 1992.

[LI 02] LI H., Traitement de la variabilité et développement de systèmes robustes pour la reconnaissance de l'écriture manuscrite en-ligne, PhD thesis, UPMC Paris 6, 2002.

[MAT 93] MATIĆ N., GUYON I., DENKER J., VAPNIK V., Writer-Adaptation for on-line Handwritten character recognition, *ICDAR*, vol. 1, 1993.

[OUD 01] OUDOT L., PREVOST L., MILGRAM M., Dynamic recognition in the omni-writer frame : application to hand printed text recogntion, *ICDAR*, vol. 1, 2001.

[OUD 04] OUDOT L., PREVOST L., MOISES A., Un modèle d'activation vérification pour la lecture de textes manuscrits dynamiques, *CIFED*, vol. Soumis, 2004.

[PAA 82] PAAP K., NEWSOME S. L., MCDONALD J. E., SCHVANEVELDT R. W., An activation-verification model for letter and word recognition : The word superiority effect, *Psychological Review*, vol. 89, 1982, pp. 573–594.

[PLA 97] PLATT J. C., MATIĆ N. P., A Constructive RBF Network for Writer Adaptation, *Advances in Neural Information Processing Systems, 9*, vol. 1, 1997, pp. 765–771.

[PRE 00] PREVOST L., MILGRAM M., Modelizing character allographs in omni-scriptor frame : a new non-supervised algorithm, *Pattern Recognition Letters*, vol. 21, n° 4, 2000, pp. 295–302.

[SCH 93] SCHOMAKER L., HELSPER E. H., TEULINGS H.-L., ABBINK G. H., Adaptive recongnition of online, cursive handwriting, *6th ICOHD*, vol. 1, 1993.

[VUO 02] VUORI V., LAAKSONEN J., KANGAS J., Influence of Erroneous Learning Samples on Adaptation in On-line Handwriting Recognition, *Pattern Recogntion*, vol. 35, n° 4, 2002, pp. 915-926.