Contributions à la reconnaissance de l'écriture arabe manuscrite

THÈSE

présentée et soutenue publiquement le ? ? juin 2008

pour obtenir le grade de

Docteur de l'Université Paris Descartes
Discipline : Informatique

par

Farès Menasri

Composition du jury

Rapporteurs	Mr L.Heutte	Professeur, Université de Rouen
	Mr C.Mokbel	Professeur, University of Balamand
Examinateurs	Mr G.Stamon	Professeur, Université Paris Descartes
	Mr V.Märgner	Professeur, Technical University Braunschweig
	Mr E.Augustin	Responsable R&D, A2iA
Directeur	Mme N. Vincent	Professeur, Université Paris Descartes

UNIVERSITÉ PARIS DESCARTES
UFR de Mathématiques et d'Informatique

Remerciements

merci.

Table des matières

Chapitre 1

Introduction

"Informatique" se dit "Computer Science" (littéralement "science du calculateur") en anglais, et "تكنولوجيا المعلومات" (littéralement "technologie de l'information") en arabe.

Le mot en français est intéressant : il est la contraction de **infor**mation et auto**matique**. L'informatique est se qui se rapporte au traitement automatique de l'information.

L'un des objectifs de la recherche en informatique est de repousser les limites de ce qui est automatisable. Les tâches répétitives, fastidieuses, portant sur de gros volumes de données, constituent de bons candidats.

Citons par exemple le traitement des chèques bancaires, le tri du courrier postal, l'indexation d'archives nationales (archives militaires, formulaires de recensements, fonds bibliothécaires, ...), indexation d'archives privées, traitement du courrier entrant des entreprises, etc ...

L'automatisation de tout ou partie de ces tâches nécessite de transmettre à la machine la capacité de lire l'écriture manuscrite. Or les types d'écriture (voir figure 1.1) et les styles peuvent varier de façon considérable selon les scripteurs.

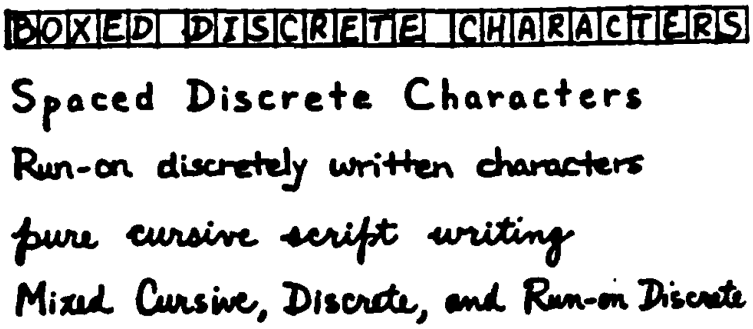


FIG. 1.1 – Types d'écriture (figure extraite de [177]).

Si le fait de lire est un acte relativement banal pour un humain, cette activité met néanmoins en oeuvre des processus complexes. Reconnaître correctement des symboles isolés ne suffit pas. Ce phénomène est illustré dans les figures 1.2 et 1.3. Ces deux images

FIG. 1.2 – "drogue" ou "olrogue" ? (cf figure 1.4)

FIG. 1.3 – "web" ou "met" ? (cf figures 1.5 et 1.6)

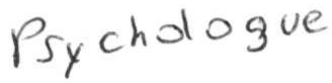


FIG. 1.4 – Image E114.champs16.tif de la base IRONOFF [187]. Les lettres "ologue" sont utilisées dans la figure 1.2

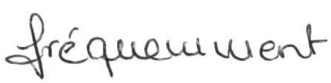


FIG. 1.5 – Image C88.champs45.tif de la base IRONOFF [187]. Le deuxième 'm' est utilisé dans la figure 1.3

FIG. 1.6 – Image E112.champs12.tif de la base IRONOFF [187]. Le dernier 't' est utilisé dans la figure 1.3

sont factices, et sont construites en partie à partir des images réelles E114.champs16.tif (figure 1.4), C88.champs45.tif (figure 1.5) et E112.champs12.tif (figure 1.6) de la base IRONOFF [187]. Certaines lettres, ou groupes de lettres, placées dans un contexte différent, peuvent prendre des significations différentes.

Durant le processus de lecture, un humain exploite implicitement des connaissances contextuelles et linguistiques, qui sont difficiles à transmettre et à faire reproduire à une machine. Faire lire une machine nécessite le recours à un arsenal mathématique conséquent, pour résoudre de manière automatique un problème qui semble pourtant à la porté d'un enfant sachant lire.

En cursif, la variabilité des styles d'écriture est telle que la connaissance du vocabulaire, et les connaissances a priori du domaine sont essentielles pour guider le déchiffrage de l'écriture, même pour un humain. Ainsi, la fâcheuse réputation qu'ont les médecins d'écrire de façon particulièrement illisible, tient sans doute davantage au fait que les patients n'ont pas la connaissance du vocabulaire des médicaments. Les pharmaciens, qui eux disposent de cette connaissance, s'en servent implicitement et de façon naturelle pour déchiffrer une ordonnance.

Une machine est dans la même situation. Faute de connaissance a priori, ses performances sont généralement médiocres sur l'écriture manuscrite, en particulier sur l'écriture cursive.

La plupart du temps, les systèmes de reconnaissance automatiques sont seulement capables de reconnaître des mots inclus dans un vocabulaire restreint, généralement compris entre 10 et 10000 mots. Ce vocabulaire réduit est souvent appelé «dictionnaire », «lexique », ou tout simplement «vocabulaire ». Il contient la liste des mots que le système de reconnaissance pourra être amené à reconnaitre. On utilise généralement la convention suivante :

– petits vocabulaires : quelques dizaines de mots.
– vocabulaires de tailles moyennes : quelques centaines de mots.
– vocabulaires de grandes tailles : quelques milliers de mots.
– vocabulaires de très grandes tailles : quelques dizaines de milliers de mots.
– vocabulaire ouvert : pas de lexique pour guider la reconnaissance.

Lorsque la taille du vocabulaire augmente, la tâche de reconnaissance devient de plus en plus complexe, car des mots ressemblants ont plus de chances d'être présents dans le vocabulaire. La complexité calculatoire est également liée à la taille du vocabulaire. L'article de A. L. Koerich et al. [90] présente cette problématique et brossent un panorama des techniques qui peuvent être mises en oeuvre dans le cas de très gros vocabulaires (réduction de la taille du lexique, organisation de l'espace de recherche, techniques de recherche, ...).

Dans le cas de vocabulaires ouverts (lorsque la taille du vocabulaire potentiel est telle que les calculs deviennent prohibitifs), on a souvent recours à des techniques statistiques comme les modèles de langages (des statistiques sur les successions de lettres sont utilisées pour améliorer le processus de reconnaissance).

Dans cette thèse, nous aborderons les deux approches sur deux problèmes différents :
– vocabulaire de taille moyenne sur une tâche de reconnaissance de noms de villes tunisiennes écrites en arabe, sur un vocabulaire de 937 villes (chapitre 4).
– vocabulaire ouvert sur une tâche de reconnaissance de noms de familles en français écrits en lettres capitales.

Cette thèse s'inscrit dans la lignée des thèses menées en partenariat avec l'entreprise A2iA (E. Augustin en 2001 [17], X. Dupré en 2003 [55], F. Carmagnac en 2005 [41]).

L'objectif de cette thèse était de proposer un système de reconnasisance de l'écriture manuscrite arabe hors-ligne. Ce système s'appuie sur celui développé par E. Augustin [17] pour la reconnaissance de l'écriture cursive latine (système hybride à base de Modèles de Markov Cachés et de Réseau de Neurones de type Perceptron Multi-Couches).

Arpès cette introduction (chapitre 1), dans le chapitre 2 nous présenterons rapidement les grandes étapes qui composent une chaîne de reconnaissance de l'écriture manuscrite.

Dans le chapitre 3, nous nous intéresserons plus particulièrement à l'écriture arabe et aux travaux effectués dans le domaine de la reconnaissance automatique de l'écriture arabe manuscrite.

Dans le chapitre 4, nous présenterons nos travaux sur la reconnaissance de l'écriture arabe manuscrite. L'arabe est une langue dans laquelle les signes diacritiques (points et autres signes secondaires) sont indispensables pour différentier certains groupes de lettres. L'originalité de notre approche consiste à séparer le traitement des corps de lettres et le traitement des signes diacritiques, avant de proposer une stratégie de combinaison des deux.

Pour cela, nous avons proposé un nouvel alphabet de corps de lettres, qui permet de mieux exploiter les redondances des formes des lettres (l'approche est présentée dans la section 4.2). Nous avons ainsi montré que la prise en compte des signes diacritiques n'est pas systématiquement nécessaire, même pour un système de reconnaissance de l'écriture arabe.

Nous avons également proposé un mécanisme qui permet de déterminer une liste de candidats de mots à partir de la reconnaissance d'une séquence de signes diacritiques (cette partie est présentée dans la section 4.4). Bien que la liste ainsi obtenue soit ambiguë, la stratégie de combinaison entre la reconnaissance sans diacritiques et la reconnaissance à partir des diacritiques permet d'améliorer significativement les performances du système.

Dans la section 4.5, nous abordons le problème de la reconnaissance des chiffres isolés farsis (persans), proches des chiffres indiens utilisés dans les pays arabes au Moyen Orient. En utilisant un réseau de neurones à convolutions, et des techniques de déformations élastiques pour accroître artificiellement la taille de base d'apprentissage à l'aide de données synthétiques, nous montrons que ces techniques déjà utilisées sur des chiffres arabes (utilisés principalement en Occident) s'appliquent également bien aux chiffres indiens (utilisés principalement au Moyen Orient).

Le chapitre 5 met en oeuvre un système qui s'appuie sur le formalisme des automates à états finis pondérés (WFSM pour Weighted Finiste State Machines), appliqués à une tâche de reconnaissance de noms de familles français écrits en lettres capitales. La reconnaissance est faite sans vocabulaire, mais dans la section 5.4, nous montrons que l'utilisation d'un modèle de langage peut se faire de manière simple et directe. Nous vérifions également qu'il est plus profitable d'optimiser le reconnaissance de manière globale au sein de la chaine de reconnaissance plutôt que de manière isolée.

La maîtrise des techniques mises en oeuvre dans le chapitre 5 ouvre la voie à la réalisation de l'idée présentée dans la section 4.4.2 pour la combinaison de la reconnaissance des corps de lettres et des signes diacritiques du système de reconnaissance de l'écriture arabe. La conclusion et les perspectives (chapitre 6) reviennent sur cette idée, qui fera l'objet de travaux ultérieurs.

Chapitre 2

Etat de l'art : Reconnaissance de l'écriture manuscrite

Dans cette partie nous présenterons les grandes étapes qui composent une chaîne de reconnaissance de l'écriture manuscrite. Nous n'aborderons qu'une partie de la problématique de l'extraction d'informations, et considérerons comme acquise l'étape qui consiste à localiser des zones d'intérêt dans le document avant de reconnaître leur contenu (par exemple, dans un courrier manuscrit, localiser les informations pertinentes comme un numéro de téléphone, un numéro de sécurité sociale, etc...). Nous considérerons que les champs à reconnaître sont déjà extraits et prêts à être reconnus.

Nous aborderons la question des prétraitements et normalisations, la segmentation, l'extraction de primitives, et nous passerons également en revue quelques classifieurs usuels. Enfin, nous terminerons ce survol des techniques utilisées en reconnaissance de l'écriture en présentant les post-traitements et les mesures de performance couramment utilisées.

Cette partie s'appuie en particulier sur des travaux effectués dans le cadre de la reconnaissance de l'écriture manuscrite latine. Un état de l'art plus spécifique de la reconnaissance de l'écriture manuscrite arabe sera donné dans le chapitre 3.

2.1 Prétraitements et Normalisations

Dans cette partie, nous abordons la question des prétraitements et normalisations couramment utilisés dans les systèmes de reconnaissance de l'écriture. De tels systèmes prennent en entrée des images numérisées, qui peuvent être binarisées ou en niveaux de gris. Lorsque les images peuvent être en niveaux de gris, les systèmes incluent généralement une étape de binarisation de l'image, qui consiste à séparer l'information utile (l'écriture) du fond. Nous présenterons rapidement les deux grandes approches des méthodes de binarisation. Nous verrons également l'extraction du squelette, qui est une manière de représenter l'information de la forme à reconnaître. Nous verrons également quelques techniques de normalisation, comme la correction de l'inclinaison des lignes (correction du skew), la correction de l'inclinaison des lettres au sein d'un mot (correction du slant), le lissage du contour, et la détection de la bande de base (qui permet d'effectuer une

normalisation en hauteur de l'écriture).

2.1.1 Binarisation

Pour une évaluation de différentes méthodes de binarisation, on pourra consulter les travaux de O. D. Trier et al. [183] [184]

On distingue en général deux approches : les approches à seuillage global, et les approches à seuillage adaptatif.

2.1.1.1 Seuillage global

Le seuillage global consiste à prendre un seuil ajustable, mais identique pour toute l'image. Chaque pixel de l'image est comparé à ce seuil et prend la valeur blanc ou noir selon qu'il est supérieur ou inférieur. Cette classification ne dépend alors que du niveau de gris du pixel considéré.

Cette méthode convient pour les documents simples et de bonne qualité. Néanmoins, elle n'est plus applicable lorsque la qualité d'impression du texte n'est pas constante dans toute la page, des caractères peuvent être partiellement perdus. Des problèmes surviendront également si le fond est bruité ou non homogène, dans ce cas des taches parasites peuvent apparaître.

Les histogrammes de la répartition des niveaux de gris des pixels de ces images contiennent deux pics nets : l'un pour la forme, l'autre pour le fond. Un seuil global peut aisément être sélectionné entre les deux pour classer les pixels selon qu'ils appartiennent aux formes ou au fond.

Pour l'évaluation du seuil optimal, on se référera aux travaux de N. Otsu [144].

2.1.1.2 Seuillage adaptatif

Dans les documents pour lesquels l'intensité du fond et l'intensité de la forme peuvent varier au sein du document, un seuillage global est inadapté. Il devient nécessaire de choisir le seuil de binarisation de manière locale. On calcule un seuil de binarisation pour chaque pixel de l'image, en fonction de son voisinage.

Par exemple, dans [28], J. Bernsen propose le calcul suivant pour déterminer le seuil de binarisation de chaque pixel :

$F_{min}(x,y) = \min_{(x_i,y_i)\in v_{(x,y)}} f(x_i,y_i)$

$F_{max}(x,y) = \max_{(x_i,y_i)\in v_{(x,y)}} f(x_i,y_i)$

$v_{(x,y)}$ est le voisinage du pixel (x,y)
$f(x,y)$ est la valeur de l'intensité du pixel (x,y) dans l'image
$F_{min}(x,y)$ est la valeur minimale de l'intensité dans le voisinage du pixel considéré
$F_{max}(x,y)$ est la valeur maximale de l'intensité dans le voisinage du pixel considéré

La moyenne des valeurs $F_{min}(x,y)$ et $F_{max}(x,y)$ est utilisée pour déterminer le seuil de binarisation du pixel (x,y) :

$g(x,y) = (F_{max}(x,y) + F_{min}(x,y))/2$

$$b(x,y) = \begin{cases} 1 & \text{si } f(x,y) < g(x,y) \\ 0 & \text{sinon} \end{cases}$$

Une alternative consiste à également prendre en compte la dynamique des valeurs dans le calcul du critère de seuillage :

$c(x,y) = F_{max}(x,y) - F_{min}(x,y)$

$$b(x,y) = \begin{cases} 1 & \text{si } (f(x,y) < g(x,y) \text{ et } c(x,y) > c^*) \text{ ou } (f(x,y) < f^* \text{ et } c(x,y) \leq c^*) \\ 0 & \text{sinon} \end{cases}$$

Cette méthode est suffisante pour les images dont le fond varie lentement (sans transitions abruptes dans le fond) de façon non uniforme.

La binarisation des documents bruités reste un domaine de recherche actif. Pour un survey des différentes techniques, on pourra consulter [166].

Citons aussi les travaux récents de B. Gatos et al. [62], appliqués à la binarisation de documents dégradés (taches, ombres). Cette méthode inclut une phase de filtrage, une première estimation grossière du découpage fond/forme, la phase de calcul des seuils locaux proprement dite, et enfin une phase de post-traitements spécifiques.

Et citons également les travaux de 2007 de Y. Xi et al. [192], qui combinent deux méthodes anciennes (Niblack [137] et Palumbo [145]). Le seuillage de Niblack conserve bien les caractères, mais a tendance à laisser beaucoup de bruits de fond dans l'image. Le seuillage de Palumbo filtre une grande partie des bruits de fond, mais a parfois tendance à également supprimer des parties de caractères. En combinant les deux méthodes, le système offre des résultats intéressants, pour une large gamme de situations dans lesquelles les images sont bruitées.

2.1.2 Squelette

La squelettisation est une opération qui permet de passer d'une image à sa représentation en "fil de fer". Le squelette a un pixel d'épaisseur. C'est une manière de représenter l'information indépendamment de l'épaisseur initiale de l'écriture.

Il permet d'extraire des caractéristiques importantes, comme les intersections et le nombre de tracés, leurs positions relatives. Il est également possible de renormaliser l'épaisseur de l'écriture à partir du squelette.

Il n'existe pas de définition unique du squelette. Le squelette doit seulement remplir trois conditions :

- Il doit être aussi fin que possible (typiquement, 1 pixel d'épaisseur)
- Il doit respecter la connexité
- Il doit être centré dans la forme qu'il représente

A chaque composante connexe de la forme, le squelette correspondant ne doit être composé que d'une seule composante connexe d'un pixel d'épaisseur incluse dans la première (voir figure 2.1).

Il existe de nombreuses méthodes de squelettisation. L'une des manières d'évaluer le squelette est de calculer l'axe médian de la forme, comme l'ensemble des centres des boules maximales de cette forme (voir figure 2.2).

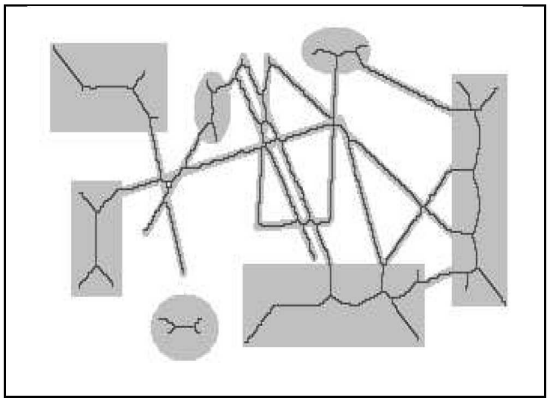


FIG. 2.1 – Squelettisation d'une forme quelconque (figure extraite de [55].

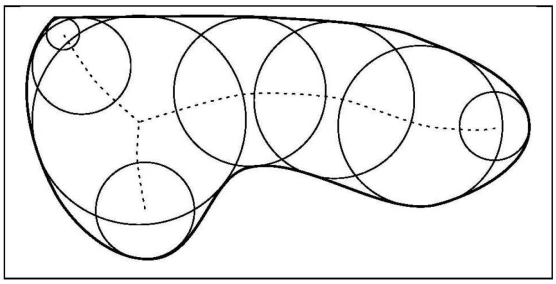


FIG. 2.2 – Axe médian d'une forme continue (trait pointillé) dont seul le contour est représenté (figure extraite de [179]) .

Un autre moyen d'obtenir le squelette est de procéder par affinages successifs : la forme est "épluchée" (peeling) de manière itérative, en maintenant valide le critère de connexité.

L'algorithme de Hilditch [71] exploite ce principe.

On définit le voisinage du pixel P_1 comme :

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

On définit également :
$A(P_1)$: nombre de transitions 0 vers 1 dans la séquence $P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$
$B(P_1)$: nombre de pixels noirs dans le voisinage de P_1 (P_1 étant exclu).

$$B(P_1) = \sum_{i=2}^{i=9} P_i, \text{ avec } \begin{cases} P_i = 1 & \text{si le pixel est noir} \\ P_i = 0 & \text{si le pixel est blanc} \end{cases}$$

L'image est parcourue de multiples fois, et un pixel noir est marqué comme effaçable s'il respecte les quatre conditions suivantes :

$$\begin{aligned} &2 \leq B(P_1) \leq 6 \\ &A(P_1) = 1 \\ &P_2 \cdot P_4 \cdot P_8 = 0 \text{ ou } A(P_2) \neq 1 \\ &P_2 \cdot P_4 \cdot P_6 = 0 \text{ ou } A(P_4) \neq 1 \end{aligned}$$

- $Condition1 : 2 \leq B(P_1) \leq 6$. La première inégalité s'assure que P_1 n'est ni un point isolé, ni une extrémité. La deuxième inégalité s'assure que P_1 est bien un point du contour de la forme.
- $Condition2 : A(P_1) = 1$. Cette règle maintient le critère de connectivité.
- $Condition3 : P_2 \cdot P_4 \cdot P_8 = 0$ ou $A(P_2) \neq 1$. Cette règle s'assure que les lignes verticales de 2 pixels de large ne sont pas totalement érodées.
- $Condition4 : P_2 \cdot P_4 \cdot P_6 = 0$ ou $A(P_4) \neq 1$. Cette règle s'assure que les lignes horizontales de 2 pixels de large ne sont pas totalement érodées.

Tous les points marqués comme effaçables sont ensuite effacés, et on réitère l'opération sur la nouvelle image, jusqu'à ce que plus aucun point ne soit effaçable.

De manière générale, les méthodes d'obtention du squelette par érosions successives fonctionnent selon le principe suivant :

1: **repeat**
2: Appliquer masques et règles pour déterminer les pixels noirs effaçables
3: Changer tous les pixels marqués comme effaçables de noir vers blanc
4: **until** Aucun nouveau pixel n'a été effacé

2.1.2.1 Barbules

Pour une analyse des différentes approches de squelettisation, on pourra se référer au travail de synthèse de X. Dupré [55].

Après la squelettisation, un grand nombre de petits segments du squelette peuvent s'avérer non pertinents, comme le montre la figure 2.3. Ces petits traits, appelés barbules, peuvent être nettoyés pour rendre le squelette plus lisible. Plusieurs critères ont été proposés pour supprimer ces traits superflus. Par exemple, on peut utiliser un simple

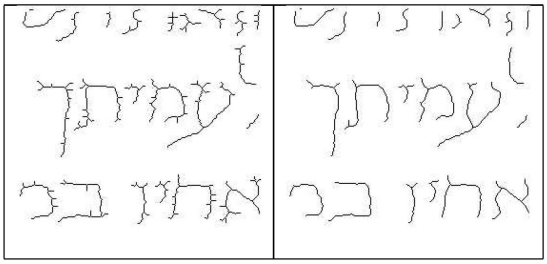


FIG. 2.3 – Barbules (figure extraite de [55]). Image de gauche : squelette brut. Image de droite : squelette nettoyé de ses barbules.

critère de taille : tous les arcs de squelette dont la longueur est inférieure à un seuil sont considérés comme du bruit et sont supprimés. Dans [76], B.K. Jang et R.T. Chin proposent un critère qui mesure le rapport entre la forme F à squelettiser et l'aire formée par l'ensemble G des boules maximales incluses dans F dont le centre appartient au squelette. Par définition $G \in F$, le critère proposé est égal à :

$$c_J = \frac{aire(G)}{aire(F} \leq 1$$

Le squelette est rogné à ses extrémités tant que le critère c_J est supérieur à un certain seul.

Un autre critère a été proposé par L. Huang dans [75]. Ce critère se base sur le rapport des longueurs du squelette et du contour. Par définition, le contour est au moins deux fois plus long que le squelette :

$$c_H = \frac{longueur(squelette)}{longueur(contour)} \leq 0.5$$

Là encore, le squeltte est rogné tant que le critère c_H est supérieur à un certain seuil. L. Huang et al. proposent un seuil de 0,4

2.1.2.2 Intersections

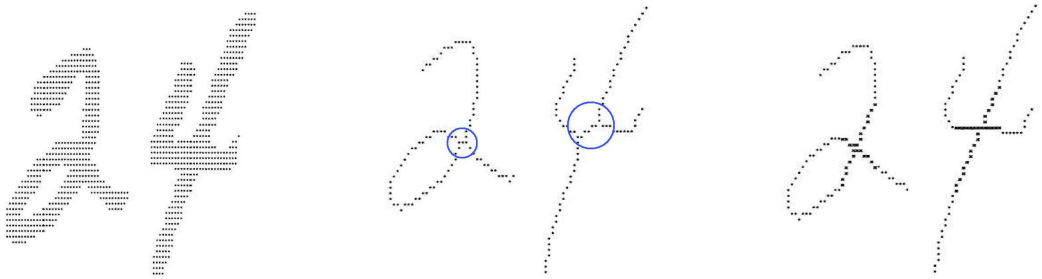
Dans [202], D.X. Zhong et H. Yan proposent une méthode pour améliorer le squelette. Les intersections entre deux segments de droites se retrouvent parfois scindés, comme le montre la figure 2.4-b. En parcourant l'image selon plusieurs directions, les auteurs proposent une méthode pour déterminer les zones d'embranchements. Ils font ainsi converger en un point unique tous les arcs de squelette qui sont connectés dans cette zone (figure 2.4-c).

2.1.3 Normalisations

Les normalisations ont pour objectif de rendre l'écriture la plus indépendante possible du scripteur.

Nous présenterons ici trois techniques de normalisation :

– La correction de l'inclinaison des lignes : skew



(a) (b) (c)

FIG. 2.4 – Correction du squelette afin de mieux représenter les intersections, figures extraites de [202]. (a) image initiale. (b) squelette initial ([201]). (c) squelette corrigé.

– La correction de l'inclinaison des lettres : slant
– La normalisation en hauteur

Cette phase de normalisation a pour objectif de faire en sorte que l'extraction de caractéristiques soit la moins perturbée possible par la variabilité des styles d'écriture.

2.1.3.1 Correction de l'inclinaison des lignes

La correction de l'inclinaison des lignes de texte (également appelée correction de "skew"), consiste à redresser horizontalement les lignes d'écriture obliques.

Plusieurs méthodes sont disponibles. Les deux plus populaires sont la transformée de Hough (appliquée sur les centre de gravité des composantes connexes), et les histogrammes de projection.

La méthode des histogramme consiste à parcourir l'image selon des directions d proches de l'horizontale, et à compter le nombre de pixels noirs selon ces direction pour chaque ligne (voir figure 2.5).

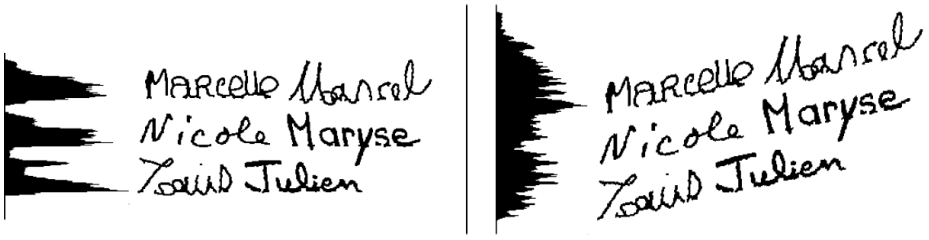


FIG. 2.5 – Correction de l'inclinaison des lignes à l'aide d'histogrammes de projection horizontale. Les pics les plus prononcés correspondent à la rotation correcte (figure extraite de [41])

La qualité de l'histogramme, ou sa pertinence, est estimée par son entropie. La direction la plus probable est celle qui maximise l'entropie [107]. L'histogramme d'entropie maximale est celui dont les extrema sont les plus marqués.

L'angle du document (ou de la ligne), θ, est celui qui correspond à l'histogramme d'entropie maximale.

Pour corriger cette inclinaison, il suffit d'appliquer une rotation de l'image d'angle θ :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2.1.3.2 Correction de l'inclinaison des lettres

Certains scripteurs écrivent leurs lettres de façon inclinée par rapport à l'axe vertical. Cette inclinaison de l'écriture est également appelée "slant". Les lettres peuvent être inclinées vers la droite ou vers la gauche. Pour la même raison que dans le paragraphe précédent, il convient de corriger cette inclinaison de l'écriture pour la rendre la plus indépendante possible des spécificités d'écriture du scripteur. Plusieurs approches ont été proposées :

Dans [33], R. M. Bozinovic et S. N. Srihari utilisent des portions d'écriture proches de la direction verticale (voir figure 2.6) pour évaluer l'inclinaison.

D'autres travaux utilisent [82] [188] des histogrammes de projection, de la même manière que pour la correction du skew, mais cette fois dans des directions proches de la verticale (voir figure 2.7). Là encore, l'histogramme d'entropie maximale est celui dont les extrema sont les plus marqués.

D'autres encore utilisent les contours [194] [36]. On parcourt le contour de l'image en comptant le nombre de fois qu'on se déplace dans trois directions privilégiées : n_1, n_2 et n_3 (voir figure 2.8).

L'angle d'inclinaison de l'écriture est obtenu en calculant :

$$\theta = tan^{-1}(\frac{n_1 - n_3}{n_1 + n_2 + n_3})$$

La correction de l'inclinaison se fait par translation des lignes :

$$\begin{aligned} x' &= x - y \cdot tan(\theta) \\ y' &= y \end{aligned}$$

Dans [176] [29], S. Uchida et al. proposent une méthode qui permet de corriger une inclinaison variable des lettres dans les mots (voir figure 2.9). A chaque colonne de l'image est associée une valeur de slant. La correction de slant non-uniforme est exprimée sous forme d'un problème d'optimisation, dont les corrections locales sont les variables à optimiser. Deux contraintes sont utilisées : les traits longs proches de la verticale fournissent une mesure locale précise de l'inclinaison ; et le fait que l'angle d'inclinaison varie de façon lisse et régulière. Les auteurs montrent que l'utilisation de cette technique permet d'obtenir une légère amélioration des performances.

2.1.3.3 Lissage du contour

Dans certains cas, la numérisation, binarisation ou les prétraitements peuvent introduire des bruits dans l'image, qui se traduisent en particulier par la présence d'irrégularités le long des contours des lettres. Ces bruits peuvent dégrader les performances de reconnaissance.

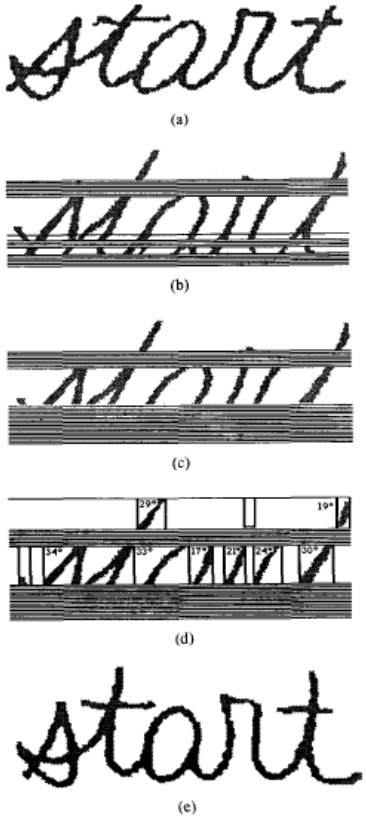


FIG. 2.6 – Correction du Slant par sélection de zones dont le tracé est proche de la verticale (figure extraite de [33])
(a) image originale.
(b) les portions horizontales suffisamment longues sont écartées.
(c) les petites portions horizontales de faible épaisseur sont également écartées.
(d) portions de l'image conservées pour l'évaluation de l'angle d'inclinaison de l'écriture.
(e) image corrigée.

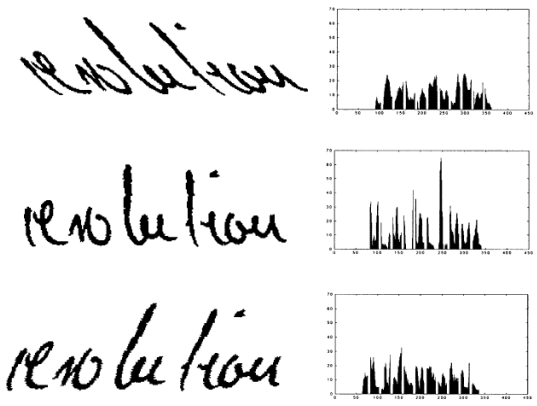


FIG. 2.7 – Inclinaison de l'écriture, et histogrammes de projection verticale correspondants.

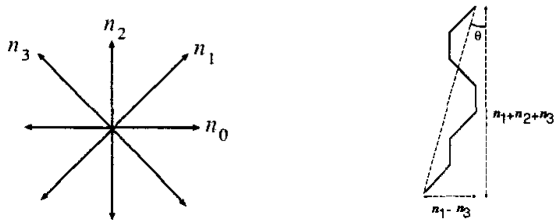


FIG. 2.8 – Inclinaison moyenne de l'écriture évaluée sur le contour de l'image (figure extraite de [194]). A gauche : n_0, n_1, n_2 et n_3 sont compteurs associés à 4 directions privilégiées. A droite : l'angle d'inclinaison se déduit en parcourant le contour.

FIG. 2.9 – L'inclinaison des lettres peut être variable sur une ligne complète, et même au sein d'un même mot (figure extraite de [29])

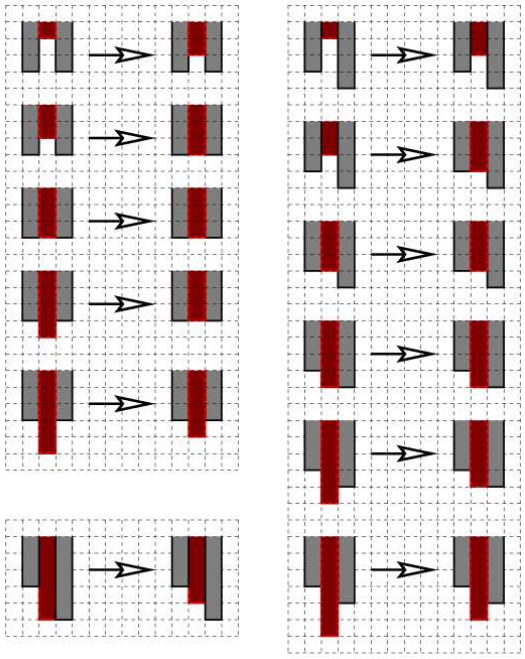


FIG. 2.10 – Masques qui permettent de lisser le contour des formes. Les masques complémentaires s'obtiennent par rotation (figure extraite de [171])

Des techniques simples et rapides à mettre en oeuvre permettent de réduire l'influence de ces bruits.

=	=	=
=	p	=
x	x	x

x	=	=
x	p	=
x	=	=

x	x	x
=	p	=
=	=	=

=	=	x
=	p	x
=	=	x

TAB. 2.1 – Exemples de masques de filtrage qui permettent de lisser des images. p est le pixel courant. Les pixels notés 'x' ne sont pas pris en compte. Les pixels notés '=' sont tous égaux entre eux. [49]

Dans [49], M. Cheriet et al. proposent quatre masques simples (table 2.1). Lorsqu'une de ces configurations est rencontrée, le pixel central 'p' prend la valeur des pixels notés '='. Ces règles sont appliquées indistinctement sur les pixels de forme ou les pixels de fond.

Dans [171], P. Slavik et V. Govindaraju proposent un autre jeu de masques, présentés figure 2.10. Les masques complémentaires s'obtiennent par rotation.

2.1.3.4 Lignes d'appui

Les lignes d'appui encadrent la bande des minuscules et délimitent les zones contenant les ascendants et descendants. Ces lignes sont importantes en reconnaissance de l'écriture. Cette information est utilisée pour détecter ascendants et descendants, et

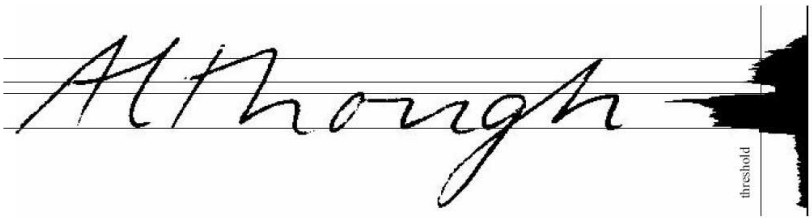


FIG. 2.11 – Histogramme de projection horizontale et bande de base.

également pour normaliser les primitives, les rendant ainsi moins dépendantes de la hauteur de l'écriture.

Plusieurs méthodes permettent de détecter ces lignes virtuelles, mais toutes nécessitent la reconnaissance de quelques lettres afin de retourner un résultat fiable. Il est en effet impossible de distinguer certaines lettres (un 'o' minuscule d'un 'O' majuscule, un 'p' minuscule d'un 'P' majuscule, un 'e' minuscule d'un 'l' minuscule, un 'c' minuscule d'un 'C' majuscule, etc...) si on ne dispose d'aucun contexte.

Dans [189], J. Whang et al. proposent une méthode basée sur les extrema locaux du contour extérieur de l'image du mot. Ces lignes sont estimées sur des mots complets (et parfois sur plusieurs mots consécutifs dans le cas de mots courts), à l'aide de transformées de Hough. Le résultat est contraint de telle sorte que les deux lignes extraites soient proches et parallèles.

Dans [109], S. Madhvanath et V. Govindaraju suggèrent que l'estimation globale de la position de ces lignes induit trop souvent un résultat de mauvaise qualité, surtout si les lettres ne sont pas disposées sur une droite ou si les caractères en minuscule sont de tailles différentes. La méthode proposée dans cet article s'appuie sur les extrema du contour de l'image d'un mot, puis regroupe sous forme de petits segments les points localement proches et presque alignés. L'ensemble de ces petits segments définit des lignes d'appui variables tout au long du mot.

Une autre méthode couramment utilisée consiste à s'appuyer sur les histogrammes de projection horizontale (voir figure 2.11). Le maximum se situe dans la bande de base. Un seuil haut et un seuil bas permettent de déterminer une approximation de la bande de base. Dans [135], P. Nagabhushan et al déterminent la ligne haute (resp. basse) comme étant la première ligne en partant du haut (resp. bas) à avoir une valeur de projection horizontale supérieure à la moyenne. Un certain nombre de règles ad hoc permettent de valider ou de rejeter ce résultat. Bien qu'approximatives, ces solutions restent néanmoins satisfaisantes dans bon nombre de cas : les méthodes basées sur des histogrammes sont en général robustes. Une approximation et un décalage de quelques pixels par rapport à la bande de base idéale n'altèrent généralement pas significativement les résultats de la reconnaissance : les modèles de reconnaissance utilisés en aval permettent d'absorber cette variabilité.

A. Hennig et al. [69, 70] proposent néanmoins une approche intéressante basée sur des splines [155]. Les zones (ascendants, zone médiane et descendants) peuvent être évaluées sur une ligne complète, ce qui permet de lever les ambiguïtés que rencontrent les tech-

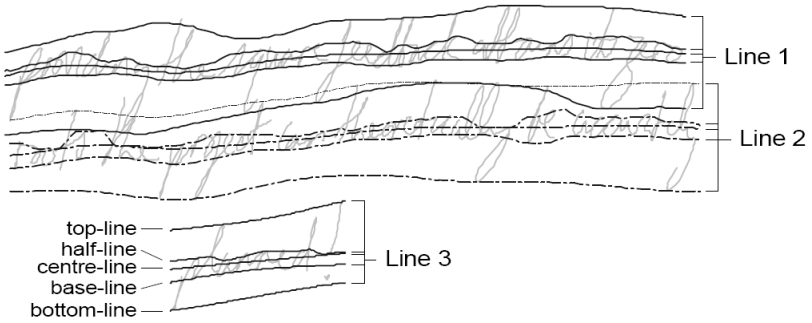


FIG. 2.12 – Evaluation des zones de l'écriture (ascendants, zone médiane, descendants) à l'aide de 4 splines (figure extraite de [69]).

niques à base d'histogrammes sur les mots courts ou qui ne possèdent pas d'ascendants ni de descendants (voir figure 2.12), tout en restant robuste aux variations de l'inclinaison et de la hauteur que peuvent pendre les mots au sein d'une même ligne.

N. Sherkat évalue l'apport de cette technique en terme de reconnaissance, à l'aide d'un reconnaisseur holistique [167]. Il constate ainsi une amélioration significative du taux de reconnaissance (+12% en première position, +5% dans le top-5).

2.2 Segmentation

2.2.1 Approche Holistique / Approche Analytique

L'approche holistique [39] [110] consiste à extraire des primitives sur le mot complet : boucles, ascendants, descendants, profils haut/bas, vallées, longueur, points terminaux, points de croisements, ainsi que d'autres primitives locales et globales. Cette approche est plus simple que l'approche analytique, car elle ne nécessite pas de segmentation.

En revanche, le défaut de cette approche est qu'elle n'est applicable que sur de petits vocabulaires. Au delà de quelques dizaines de classes, la capacité discriminante des primitives extraites globalement diminue. Les confusions possibles entre mots du vocabulaire augmentent, ce qui dégrade les performances. En pratique, pour une approche holistique, on se limite généralement à quelques dizaines de classes de mots au maximum.

L'approche analytique décompose le mot en une séquence de caractères intermédiaires, qui font partie d'un alphabet prédéfini. La reconnaissance du mot complet sera obtenue par la combinaison des reconnaissances de ces caractères intermédiaires. Il est donc nécessaire de découper le mot à reconnaître en une séquence de symboles, ce qui n'est pas toujours trivial (voir les sections suivantes). Ces symboles intermédiaires seront éventuellement recomposés pour former des caractères. Cette suite de caractères forme le mot à reconnaître.

L'avantage de cette méthode est qu'elle permet de travailler sur des vocabulaires ouverts ou de grande taille : on n'apprend non plus des classes de mots, mais des classes de caractères, ce qui permet d'exploiter pour une même classe de caractères des informations qui proviennent de classes de mots différentes.

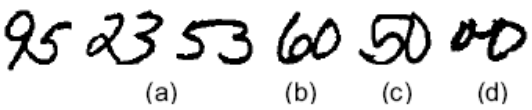


FIG. 2.13 – Types de contacts (figure extraite de [175]) : (a) Un seul point de contact. (b) Une portion de trait en commun. (c) Doublement connectés. (d) Une ligature.

Un tour d'horizon des techniques de segmentation est donné par R. G. Casey et E. Lecolinet [42] ou par Y. Lu et M. Shridhar [105].

Dans la suite de cette section, nous aborderons deux problèmes de segmentation. Tout d'abord, le problème de la segmentation en caractères sur une tâche de reconnaissance d'une séquence numérique. Puis nous aborderons le problème plus complexe de la segmentation de l'écriture cursive.

On définit les termes suivants :

- sur-segmentation : une lettre est découpée en plusieurs sous-parties. Les sur-segmentations peuvent être problématiques ou non, selon l'utilisation qui en sera faite. Dans le cas d'une segmentation graphèmes (voir section 2.2.3), ce cas de figure correspond au comportement normal.
- sous-segmentation : certains objets de segmentation sont à cheval sur deux lettres consécutives. Ce cas de figure pose un problème, car certaines lettres qui composent le mot ne pourront pas être correctement reconnues.

2.2.2 Segmentation de séquences de chiffres

La reconnaissance de séquences de chiffres est une tâche importante dans le domaine du traitement automatique de documents. Il peut s'agir de la reconnaissance de numéros de téléphone, d'identifiants numériques (numéros de sécurité sociale, codes clients, etc...), de code postal, etc... De nombreuses applications sont concernées : tri postal, traitement de chèques, traitement de formulaires, courrier entrant, etc... Dans ces documents, les chiffres peuvent être isolés, déformés, liés par un ou plusieurs points de contact (comme l'illustre la figure 2.13, partiellement superposés, ou au contraire en plusieurs morceaux. Le but de la segmentation est de proposer soit le découpage correct en chiffres, soit une liste d'options de segmentation, dont on espère qu'elle contiendra la segmentation correcte.

De nombreux algorithmes ont été proposés pour estimer la segmentation de couples de chiffres liés. Certains travaux sont basés sur les pixels de la forme [173], d'autres s'appuient sur les pixels du fond [48, 106]. D'autres encore combinent les deux approches, en exploitant à la fois les squelettes du fond et de la forme, ce qui leur permet d'obtenir de meilleures performances [44, 45, 156]. Ces algorithmes permettent dans certains cas de supprimer les ligatures superflues, qui pourraient perturber la reconnaissance (voir figure 2.14).

D'autres s'appuient sur le nombre, la taille, le centre de gravité et la position des réservoirs d'eau [185, 146] pour déduire des points de coupure potentiels, en incluant un grand nombre d'heuristiques.

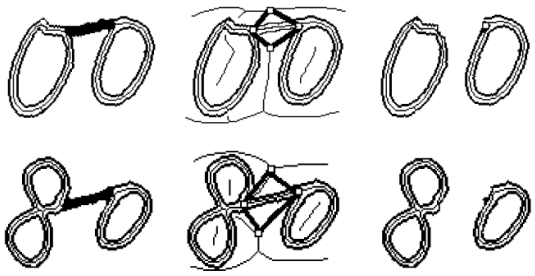


FIG. 2.14 – Segmentation à l'aide des squelettes de la forme et du fond. Suppression des ligatures superflues pour améliorer la qualité des symboles segmentés (figure extraite de [44]).

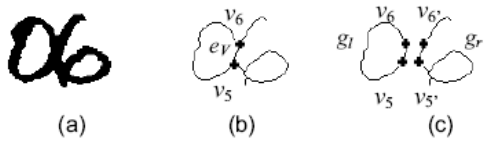


FIG. 2.15 – (a) Image originale. (b) Graphe correspondant. (c) Segmentation par duplication de l'arc e_v (figure extraite de [175]).

Plus récemment, certaines approches s'appuient sur des graphes. Segmenter une paire de chiffres liés revient à créer deux graphes disjoints à partir du premier graphe. Un ensemble de contraintes (tailles des composantes, superposition horizontale et verticale, etc...) permettent de restreindre le nombre de candidats.

Dans [175], M. Suwa propose une technique qui permet de segmenter un couple de chiffres de manière satisfaisante même lorsque ces derniers partagent des portions de traits en commun, comme le montre la figure 2.15.

Il propose également une mesure de pertinence basée sur le rapport des hauteurs et des largeurs des composantes obtenues :

$$Sc = \left(1.0 - \frac{min(w_t, w_r)}{max(w_t, w_r)}\right) \times \left(1.0 - \frac{min(h_t, h_r)}{max(h_t, h_r}\right)$$

Où (w_t, w_r) et (h_t, h_r) sont les largeurs et les hauteurs des composantes gauche et droite respectivement.

Dans [50], C. Renaudin et al. couplent leur système avec un reconnaisseur. Le meilleur candidat de segmentation est celui qui correspond au meilleur produit des probabilités. Cette proposition est intéressante, car elle permet de résoudre des problèmes complexes comme l'imbrication partielle des caractères, comme le montre la figure 2.16. Cette approche reste toutefois très lente, en raison des nombreuses alternatives de reconnaissance à évaluer. Par ailleurs, dans cette approche les données sont générées de façon pseudo-aléatoire à partir d'images de chiffres isolés de la base IRON/OFF [187]. Ces exemples ne prennent donc pas en compte les ligatures (traits supplémentaires provoqués par l'absence de levée de stylo).

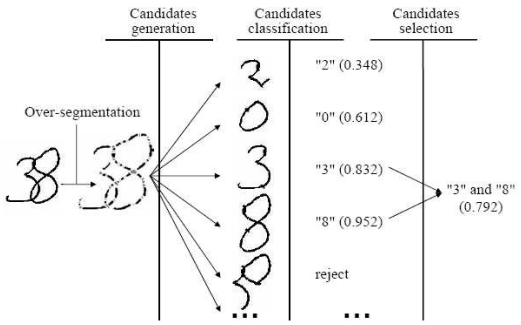


FIG. 2.16 – La meilleure segmentation est celle pour laquelle le produit des probabilités des symboles obtenus est maximal (figure extraite de [50]).

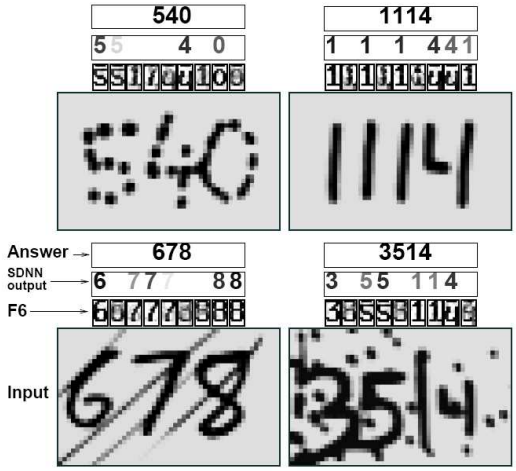


FIG. 2.17 – Système de reeconnaissance à fenêtre glissante de type SDNN (figure extraite de [94]).

D'autres approches ont également été proposées, basées sur la segmentation implicite par fenêtre glissante [100, 94].

Le système présenté par Y. LeCun et al. [94] semble faire preuve de capacités de robustesse impressionnante aux bruits, recouvrements partiels, et chiffres en plusieurs morceaux (voir figure 2.17). Cette approche constitue un axe de recherche intéressant. Pourtant, il semble que pour l'instant peu de travaux aient été menés dans cette direction.

Dans le cas de l'écriture cursive, le problème est encore plus complexe. Dans la communauté de la reconnaissance de l'écriture manuscrite, il est admis qu'il est impossible de segmenter directement un mot cursif en lettres. Pour segmenter un caractère, il faut qu'il ait préalablement été identifié ; et pour reconnaître un caractère, il faut qu'il soit préalablement correctement segmenté. C'est le paradoxe de Sayre [162].

Pour sortir de ce dilemme, il est nécessaire de découper le mot en sous-parties de lettres.

Pour la segmentation de l'écriture cursive, on peut se référer au travail de synthèse de X. Dupré [55]. On distinguera deux approches :

- la segmentation en graphèmes [92], appelée également segmentation explicite : le mot est segmenté en sous-parties qui sont presque des lettres.
- l'analyse par fenêtres glissantes, appelée également segmentation implicite : le mot est découpé en bandes verticales.

2.2.3 Segmentation de l'écriture cursive

Les graphèmes sont des images extraites de l'image à segmenter. Passer d'une seule image à une séquence de graphèmes pose le problème de la taille de ces éléments. Ils ne doivent pas être trop petits afin d'être statistiquement significatifs, et pas trop gros afin de ne pas dépasser la taille d'une lettre. Il est en effet important qu'un graphème donné soit une sous-partie d'une seule lettre : cette condition est nécessaire pour construire un modèle de mot comme étant la concaténation de modèles de lettres.

2.2.3.1 Segmentation à partir du squelette

A partir du squelette (voir section 2.1.2), on cherche à repérer certains motifs, pour en déduire les candidats de points de coupures (voir figure 2.18). La détection de ces motifs introduit des calculs de courbures et d'angles, qui sont comparés à des seuils ajustés de manière à obtenir le résultat désiré.

X. Dupré [55] souligne que cette approche est erronée dans environ 10% des cas. Les configurations difficiles à segmenter sont celles pour lesquelles les lettres sont souvent enchevêtrées, comme les "tt", ou les lettres à liaison haute ('b', 'o', 'v', 'w') avec leur successeur.

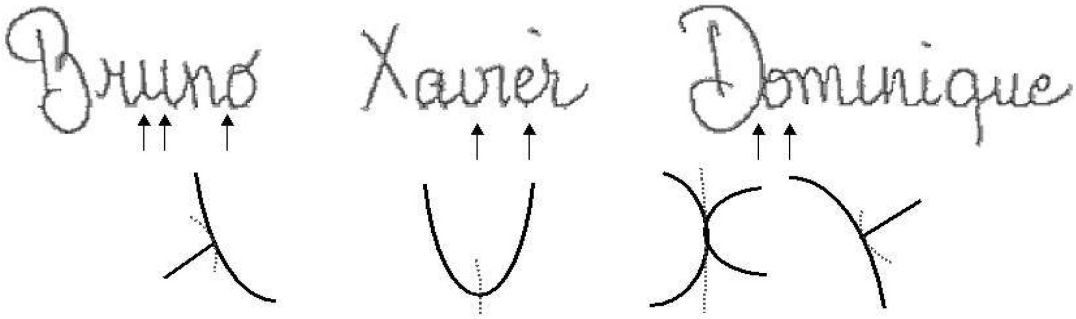


FIG. 2.18 – Segmentation à base de squelette : basée sur des motifs (figure extraite de [55]).

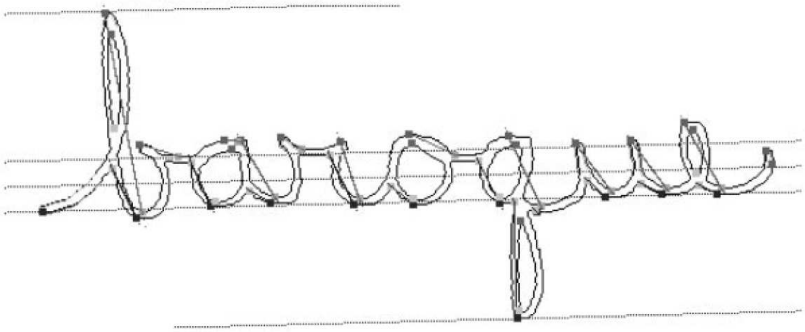


FIG. 2.19 – Extrema du contour supérieur et inférieur sont associés, et reliés par une corde (figure extraite de [111]).

2.2.3.2 Segmentation à partir du contour

S. Madhvanath et al. proposent une méthode de segmentation appliquée aux contours [111]. Ils déterminent les meilleurs points candidats de coupure entre graphèmes, en s'appuyant sur les extrema locaux du contour, qui sont associés selon un critère de proximité (voir figure 2.19).

Comme le précise X. Dupré [55], la segmentation en graphèmes à partir du contour nécessite de nombreux ajustements avant de trouver les critères de décision. Cette mise au point par tâtonnements est le point commun de nombreux traitements d'images liés à la reconnaissance de l'écriture manuscrite. Faciles à ajuster lorsque la qualité de l'écriture est bonne, ces prétraitements peuvent avoir des comportements tout à fait erratiques lorsque l'écriture est de mauvaise qualité.

2.2.3.3 Segmentation à partir des histogrammes

Cette méthode simple est proposée par B. Yanikoglu et P. Sandon [197]. Elle consiste à calculer des histogrammes de projection dans plusieurs directions proches de la verticale. Les droites choisies sont celles qui interceptent le moins de pixels noirs, avec une contrainte d'espacement régulier dans l'image (voir figure 2.20).

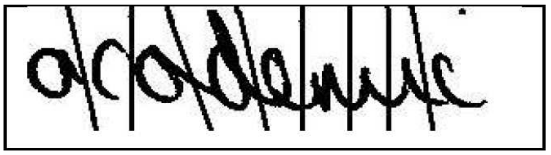


FIG. 2.20 – Segmentation à partir d'histogrammes de projection selon plusieurs directions proches de la verticale.

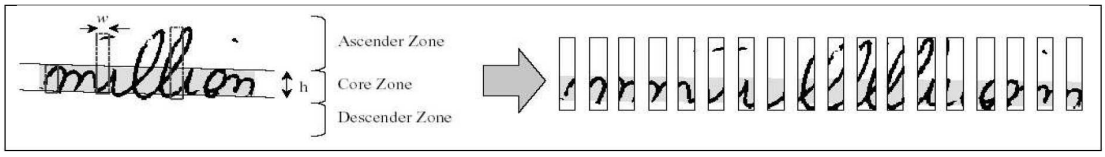


FIG. 2.21 – Segmentation à base de fenêtre glissante : découpage du mot en bandes verticales.

Cette méthode montre néanmoins ses limites lorsque les lettres sont très proches ou enchevêtrées.

2.2.3.4 Segmentation basée sur des réservoirs

X. Dupré étend à l'écriture cursive la technique à base de réservoirs [55] initialement appliquée à la segmentation de chiffres liés (voir section 2.2.2). X. Dupré souligne que les règles de décision sont plus difficiles à mettre en place dans le cas des lettres, car ces dernières sont de tailles variables.

2.2.3.5 Fenêtres glissantes

L'utilisation d'une fenêtre glissante [178] revient à découper l'image en bandes verticales. Ce découpage peut être régulier ou non, éventuellement avec recouvrement partiel des bandes successives.

Cette technique présente l'avantage d'être simple, robuste au bruit, et est indépendante de la connexité. Le défaut de cette méthode est que la séquence générée contient beaucoup de bruit (recouvrement de deux lettres successives). C'est également vrai dans le cas des lettres superposées verticalement, mais qui ne se touchent pas nécessairement : une barre de 't' avec la lettre suivante, ou les descendants comme 'ر' ou 'و' en arabe.

2.3 Extraction de primitives

L'extraction de primitives consiste à transformer une image (caractère, graphème, bande verticale, ...) en un vecteur de primitives de taille fixe.

Cette transformation revient à changer l'espace de représentation des données, du plan de l'image vers un espace à N dimensions ($\Re^N$)

Le choix des primitives est critique, et influence nettement le résultat. Ces primitives doivent avoir deux propriétés :

– être discriminantes : permettre une bonne différentiation entre les classes de symboles à reconnaître.
– maintenir un nombre de dimensions limité, afin d'éviter le phénomène de malédiction de la dimensionnalité (curse of dimensionality).

Des techniques comme l'analyse en composantes principales (ACP), appelée également transformation de Karhunen-Loève, permettent de réduire le nombre de dimensions, en projetant les données sur un sous-ensemble d'axes qui maximisent la dispersion des échantillons disponibles.

Ces axes principaux sont obtenus en calculant les valeurs propres et les vecteurs propres de la matrice de variance-covariance des données. Les axes principaux sont donnés par les vecteurs propres, par ordre décroissant de leurs valeurs propres associées. L'ACP permet donc de conserver un maximum d'informations en un minimum de dimensions.

L'ACP peut également être utilisée pour visualiser des données de grandes dimensions en les projetant dans un espace de dimension réduite (dans le plan).

Dans la suite de cette section, nous présenterons quelques grandes familles de primitives couramment utilisées :
– Les moments invariants
– Les descripteurs de Fourier
– Les primitives extraites à partir des profils et des contours

2.3.1 Moments invariants

Les moments invariants ont été introduits par Hu [73] puis étendus par Li [102].
La description d'une forme est invariante par rotation et translation.
L'image est définie par une fonction f telle que

$$f(x,y) = \begin{cases} 1 & \text{si le pixel (x,y) appartient à la forme} \\ 0 & \text{sinon} \end{cases}$$

Les moments μ_{pq} sont définis par :
$\overline{x} = \int_x \int_y x f(x,y) dxdy$
$\overline{y} = \int_x \int_y y f(x,y) dxdy$
$\mu_{pq} = \int_x \int_y (x-\overline{x})^p (y-\overline{y})^p f(x,y) dxdy$
Une première liste de moments invariants est présentée dans [74] :

$I_{20} = (\mu_{20} - \mu_{02}) - 2i\mu_{11}$	$I_{30} = \overline{I_{03}}$
$I_{11} = \mu_{20} + \mu_{02}$	$I_{40} = (\mu_4 0 - 6\mu_{22} + \mu_{04}) - 4i(\mu_{31} - \mu_{13})$
$I_{02} = \overline{I_{20}}$	$I_{31} = (\mu_{40} - \mu 04) - 2i(\mu_{31} + \mu_{13})$
$I_{30} = (\mu_{30} - 3\mu_{12} - i(3\mu_{21} + \mu_{30}$	$I_{22} = \mu_{40} + 2\mu_{22} + \mu_{04}$
$I_{21} = (\mu_{30} + \mu_{12}) - i(\mu_{21} + \mu_{03})$	$I_{13} = \overline{I_{13}}$
$I_{12} = \overline{I_{21}}$	$I_{04} = \overline{I_{04}}$

TAB. 2.2 – Moments invariants [74]

Dans [191] Wong et al. définissent d'autres moments invariants en rotation à partir de combinaisons non linéaires des moments définis dans le tableau 2.2. Une liste non

exhaustive est donnée dans le tableau 2.3. Chaque moment est renormalisé par un facteur $\frac{1}{alpha_i}$ où $alpha_i$ est la norme du moment i

I_{11}	$I_{20}I_{02}$	
$I_{21}I_{12}$	$I_{30}I_{03}$	$I_{21}^2I_{02}$
$I_{30}I_{12}I_{20}$	$I_{30}I_{03}^3$	$I_{30}I_{21}I_{02}^2$
$I_{30}^2I_{02}^3$	$I_{30}I_{12}^5I_{02}$	
I_{22}	$I_{31}I_{13}$	$I_{40}I_{04}$
$I_{31}I_{02}$	$I_{40}I_{13}^2$	$I_{40}I_{02}^2$
$I_{31}I_{12}^2$	$I_{40}I_{13}I_{02}$	$I_{40}I_{03}I_{12}$
$I_{31}I_{03}I_{21}$	$I_{40}I_{31}I_{03}^2$	$I_{40}I_{13}I_{12}^2$
$I_{40}I_{13}^3I_{20}$	$I_{40}I_{03}^2I_{20}$	...

TAB. 2.3 – Moments invariants [191]

Les moments invariants calculés par la méthode présentée dans [191] sont complexes. Pour chacun d'entre eux, les parties réelles et imaginaires peuvent être considérées comme des caractéristiques.

2.3.2 Descripteurs de Fourier elliptiques

Les descripteurs de Fourier elliptiques sont introduits par F. P. Kuhl et C. R. Giardina [91] pour décrire les formes des contours fermés.

Une démonstration visuelle de leur fonctionnement est disponible sur [34].

Soit un contour de T pixel, dont chaque pixel à pour coordonnées $(x(t), y(t))$, avec $t \in \{0, 1, \ldots, T-1\}$.

$x : t \to x(t)$ et $y : t \to y(t)$ sont périodiques de période T (le temps nécessaire pour tracer l'ensemble du contour à vitesse constante).

Ces signaux périodiques sont décomposables en séries de Fourier :

$$x(t) = A_0 + \sum_{n=1}^{\infty} \left[a_n cos\frac{2n\pi t}{T} + b_n sin\frac{2n\pi t}{T} \right]$$

$$y(t) = C_0 + \sum_{n=1}^{\infty} \left[c_n cos\frac{2n\pi t}{T} + d_n sin\frac{2n\pi t}{T} \right]$$

avec,

$$A_0 = \frac{1}{T}\int_0^T x(t)dt$$

$$C_0 = \frac{1}{T}\int_0^T y(t)dt$$

$$a_n = \frac{2}{T}\int_0^T x(t)cos\frac{2n\pi t}{T}dt$$

$$b_n = \frac{2}{T}\int_0^T x(t)sin\frac{2n\pi t}{T}dt$$

$$c_n = \frac{2}{T}\int_0^T y(t)cos\frac{2n\pi t}{T}dt$$

$$d_n = \frac{2}{T}\int_0^T y(t)sin\frac{2n\pi t}{T}dt$$

Différents niveaux d'approximation peuvent être obtenus en faisant varier le nombre d'harmoniques. L'approximation de rang N est obtenue comme :

$$x_N(t) = A_0 + \sum_{n=1}^{N}\left[a_n cos\frac{2n\pi t}{T} + b_n sin\frac{2n\pi t}{T}\right]$$

$$y_N(t) = C_0 + \sum_{n=1}^{N}\left[c_n cos\frac{2n\pi t}{T} + d_n sin\frac{2n\pi t}{T}\right]$$

Les coefficients a_n, b_n, c_n et d_n peuvent être utilisés comme primitives.

2.3.3 Profils et contours

Dans [139], J. J. Oliveira et al. présentent un certain nombre de primitives couramment utilisées en reconnaissance de l'écriture. Ils distinguent trois catégories :

- les primitives perceptuelles
- les primitives directionnelles
- les primitives topologiques

Les auteurs indiquent que les primitives perceptuelles sont celles qui permettent d'obtenir les meilleurs résultats, et leur combinaison avec les autres est utile pour améliorer la reconnaissance.

La bande de base (voir section 2.1.3.4) est utile pour extraire et pour normaliser ces primitives.

2.3.3.1 les primitives perceptuelles

Dans la liste des primitives perceptuelles usuelles, on retrouve : la position des ascendants et descendants, leur hauteur, la position et la taille des boucles, des primitives extraites sur l'enveloppe convexe du mot, le nombre de transitions noir/blanc dans la zone centrale, ...

2.3.3.2 les primitives directionnelles

Ces primitives sont généralement basées sur le squelette, ces types de primitives donnent la direction générale du tracé, longueurs et angles.

2.3.3.3 les primitives topologiques

Ce type de primitives est basé sur des densités de pixels. On peut par exemple projeter des images de tailles différentes (les graphèmes) dans une matrice de taille fixe. Les caractéristiques extraites sont les valeurs des cellules de cette matrice.

Dans ce type de primitives, on compte également les profils et histogrammes. Pour maintenir un vecteur de taille fixe, on divise l'image en un nombre fixe de bandes horizontales et verticales. Les caractéristiques sont les moyennes des valeurs sur ces bandes.

2.4 Reconnaissance

Dans la suite de cette section, nous présenterons un ensemble de reconnaisseurs couramment utilisés en reconnaissance de l'écriture, et nous discuterons de leurs forces et de leurs faiblesses. Nous passerons en revue les nuées dynamiques, les réseaux de neurones (des perceptrons multicouches aux réseaux profonds), les modèles de Markov cachés, et les machines à vecteur support. Nous présenterons également le formalisme des réseaux transformateurs de graphes.

2.4.1 Nuées dynamiques

L'algorithme des nuées dynamiques (ou k-means) est un algorithme de classification non supervisée. C'est un algorithme de clustering, ou de partition de l'espace.

Etant donné un ensemble de N points dans un espace de dimension d, et un entier K, le problème est de déterminer un ensemble de K points de R^d, appelés centres, tels que ces K points minimisent une fonction de coût : généralement la distance de chaque point à son centre le plus proche.

Un bon partitionnement de l'espace consiste à :

- minimiser la variance intra-classe, afin d'obtenir des grappes les plus homogènes possibles
- maximiser la variance inter-classe afin d'obtenir des sous-ensembles bien différenciés

Le but de l'algorithme des nuées dynamiques est d'obtenir la meilleure adéquation possible entre une partition P et sa représentation par K noyaux au sens d'un critère W défini par l'utilisateur.

W est de la forme :

$$W(P, N) = \sum_{i=1}^{K} D(P_i, N_i)$$

$D(P_i, N_i)$ représente une mesure de dissemblance entre la partie P_i et le noyau N_i.

Lorsque N_i est représentatif de P_i, la mesure de dissemblance D est faible.

Une mesure souvent utilisée est la somme du carré des distances euclidiennes entre les noyaux et les éléments de la partie qu'ils représentent. On peut également utiliser une distance pondérée sur les dimensions de l'espace d'entrée, ou une distance de Mahalanobis.

La distance de Mahalanobis est une distance euclidienne. Elle est définie de la manière suivante.

Soient X et Y deux vecteurs de R^n, la distance de Mahalanobis entre ces deux vecteurs issus d'une même distribution avec une matrice de covariance $\sum$, est définie par :

$$d_M(X,Y) = \sqrt{(X-Y)^T \sum^{-1} (X-Y)}$$

2.4.1.1 Initialisation et minimum local

L'algorithme des k-means est itératif et dépend fortement de l'initialisation. Une initialisation aléatoire peut faire converger l'algorithme vers un minimum local.

L'autre principale difficulté de cet algorithme porte sur le choix de K, le nombre d'éléments de la partitions à effectuer

2.4.1.2 Algorithme

1: Initialisation : choisir aléatoirement K vecteurs de caractéristiques comme positions initiales des centres des K clusters.
2: **repeat**
3: E : Pour chacun des vecteurs de caractéristiques, déterminer le centre le plus proche (au sens de la métrique définie). Chaque vecteur est associé à un seul centre.
4: M : Mise à jour des positions des centres des clusters de telle sorte que la nouvelle position d'un centre de cluster soit égale à la moyenne des vecteurs qui composent ce cluster.
5: **until** Plus aucun vecteur ne change de cluster d'une itération à l'autre

Cet algorithme est assuré de converger.

2.4.1.3 k-means comme initialisation d'un modèle plus complexe

L'algorithme des k-means peut servir de première phase d'initialisation pour un modèle plus complexe. Le résultat d'un k-means sera par exemple utilisé comme annotation pour l'apprentissage d'un premier réseau de neurones, en vue de l'initialisation d'un système hybride MMC/PMC. Cette partie sera développée en annexe B.

2.4.2 Réseaux de neurones

Bien que leur fonctionnement en "boite-noire" ne permette pas d'interpréter les valeurs de leurs paramètres internes, leur facilité de mise en oeuvre et leurs bonnes performances ont fait des réseaux de neurones l'un des outils les plus utilisés en apprentissage automatique au cours des 20 dernières années.

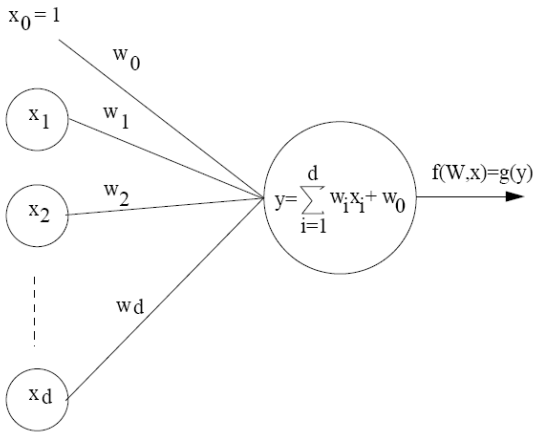


FIG. 2.22 – Neurone artificiel.

Le neurone formel s'inspire du neurone biologique (voir figure 2.22). Le neurone reçoit en entrée la description d'une observation x, décrite par un vecteur de d attributs $(x_1, \ldots, x_d)$ numériques. Chacune de ces composantes x_i est pondérée par un poids synaptique w_i. On ajoute également un biais w_0, et la sortie z du neurone est obtenue par l'application d'une fonction de transfert (appelée également fonction d'activation, ou fonction de seuil) du neurone.

Fonction linéaire	$\sum_i w_i.x_i + w_0$
Fonction sigmoïde	$\frac{1}{1+e^{-\lambda x}}$
Fonction tanh	$\frac{e^{2x}-1}{e^{2x}+1}$
Fonction softmax	$\frac{e^x}{\sum_i e^{x_i}}$
Fonction à base radiale de centre x_c	$exp\left(-\frac{\|x-x_c\|^2}{2\sigma^2}\right)$

TAB. 2.4 – Quelques fonctions de transfert usuelles. x est le vecteur d'entrées.

En pratique, les neurones sont disposés en couches (voir figure 2.23), et tous les neurones d'une couche donnée ont la même fonction d'activation (les fonctions d'activation les plus courantes sont données dans le tableau 2.4).

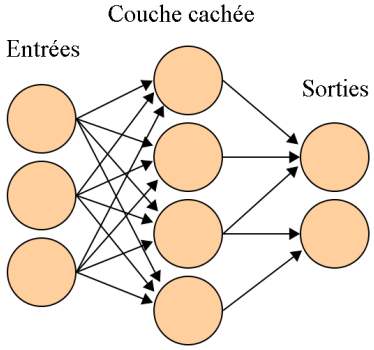


FIG. 2.23 – Réseau de neurones artificiels à une couche cachée.
$w_1, \ldots, w_n$ sont les poids synaptiques
w_0 est le biais
$y = \sum_i w_i.x_i + w_0$ est le potentiel
$f(W, x) = g(\sum_i w_i.x_i + w_0)$ est la sortie du neurone, avec g la fonction d'activation

Le plus souvent, l'entraînement d'un réseau de neurones se fait de façon itérative, par rétropropagation du gradient d'erreur. Les fonctions d'activation des neurones doivent être dérivables par rapport aux entrées (les entrées d'une couche étant les sorties de la couche précédente) de manière à pouvoir rétropropager le gradient d'erreur de couche en couche.

De plus, si la fonction de transfert est également dérivable par rapport aux paramètres du système (les w_i, qui sont les poids synaptiques), il est possible d'optimiser ces poids de manière itérative, par descente de gradient.

La réestimation itérative d'un coefficient $w_i \in W$ par l'algorithme de descente de gradient s'écrit :

$$w_i(t+1) = w_i(t) - \lambda \frac{\partial Err}{\partial w_i(W)}$$

λ est le coefficient d'apprentissage. Ce poids est important : s'il est trop grand, il risque de faire diverger l'apprentissage, s'il est trop faible l'erreur risque de converger vers un minimum local. En pratique, ce paramètre décroît au fur et à mesure de l'apprentissage, de manière analogue à la température dans une procédure de recuit simulé.

2.4.2.1 Perceptrons multicouches

Les perceptrons multicouches sont les réseaux de neurones les plus populaires, du fait de leurs bonnes performances et de leur facilité de mise en oeuvre.

Ces sont des réseaux de neurones à propagation directe sans cycles, avec au moins une couche cachée. La fonction de transfert des neurones est généralement de type sigmoïde ou tanh(voir tableau 2.4), à valeurs dans [0 ;1] ou dans [-1 ;1].

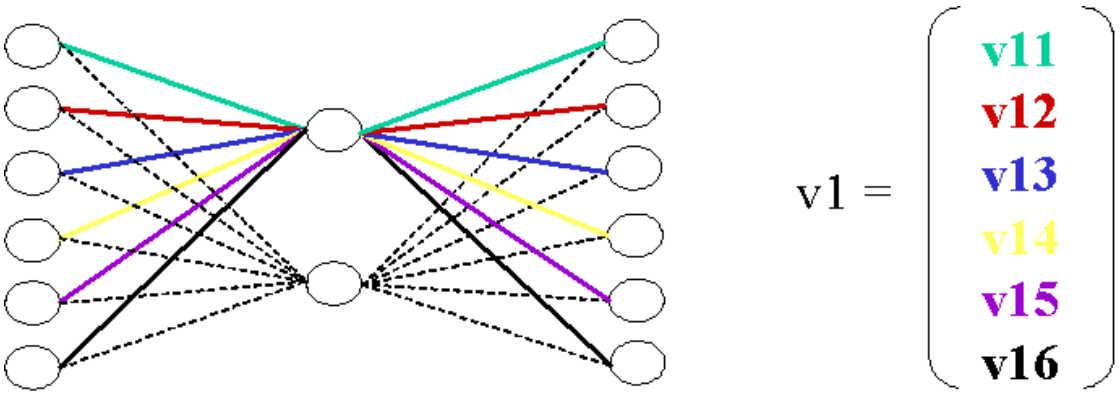


FIG. 2.24 – ACP à l'aide d'un réseau diabolo. 6 entrées, 6 sorties, et deux neurones sur la couche cachée. Les axes correspondent aux poids synaptiques des neurones de la couche cachée.

Les perceptrons multicouches définissent une classe de fonctions qui permet d'approximer n'importe quelle fonction continue à support compact.

La fonction Softmax associée à une fonction de coût J d'entropie relative (distance de Kullbak-Leibler) a été proposée par Bridle, pour évaluer des probabilités a posteriori des classes en sortie ($P(y_i|x_i)$). La fonction Softmax force les sorties à vérifier les propriétés des probabilités : les sorties sont comprises entre 0 et 1 et leur somme est égale à 1.

2.4.2.2 Fonctions à bases radiales

Les réseaux de neurones de type RBF (Radial Basis Function) ont généralement une couche cachée. Les neurones de cette couche cachée sont de types gaussiens (voir tableau 2.4). Les neurones de sorties sont une combinaison linéaire des neurones gaussiens de la couche cachée, parfois suivis d'une fonction de transfert sigmoïde ou softmax.

Les caractéristiques des neurones gaussiens (centres et variances) sont généralement estimées au début de l'apprentissage. Ces paramètres sont ensuite figés. L'entraînement du réseau de neurones consiste à apprendre la couche de combinaison linéaire de sortie, par descente de gradient.

2.4.2.3 Réseaux Diabolo

Un réseau de type diabolo est un réseau auto-associatif (entraîné de telle sorte que les sorties répliquent les entrées) dont la couche centrale constitue un goulet d'étranglement. Les poids synaptiques sont liés de part et d'autre de la couche centrale.

Ces réseaux peuvent être utilisés pour la compression des données, ou pour calculer une analyse en composantes principales (ACP).

Le réseau diabolo le plus simple est un réseau de neurones linéaires à une couche cachée (voir figure 2.24). Les poids synaptiques d'un tel réseau sont contraints de telle sorte que les coefficients de ses deux couches restent symétriques tout au long de l'apprentissage.

Lors de la phase d'apprentissage du réseau, on présente les exemples en entrée, et on adapte les paramètres du réseau de telle sorte que les sorties répliquent les entrées (les deux couches linéaires doivent rester symétriques tout au long de l'apprentissage). Les sorties désirées sont les entrées. On adapte les paramètres du réseau de neurones à l'aide de l'algorithme de rétropropagation du gradient.

Lorsque le réseau est appris, les poids synaptiques des n neurones de la couche cachée engendrent le même espace vectoriel que les n vecteurs propres de la matrice XX^T associés à ses n valeurs propres les plus importantes (voir figure 2.24). En appliquant une orthonormalisation de Schmidt sur cette liste de vecteurs, on construit ainsi une base orthonormée $e_1, \ldots, e_n$ du sous-espace vectoriel engendré par les vecteurs propres $v_1, \ldots, v_n$.

Le procédé d'orthonormalisation de Schmidt, ou procédé de Gram-Schmidt, est une méthode qui permet d'orthonormaliser une famille libre de vecteurs d'un espace vectoriel muni d'un produit scalaire.

A partir d'une famille libre $(v_1, \ldots, v_n)$, on construit une famille orthonormale $(e1, \ldots, e_n)$ qui engendre les mêmes espaces vectoriels successifs :
$\forall j \leq n, F_j = Vect(e_1, \ldots, e_j) = Vect(v_1, \ldots, v_j)$

Le principe de l'algorithme consiste à soustraire au vecteur v_{j+1} sa projection orthogonale sur l'espace F_j. On s'appuie sur la famille orthonormale déjà construite pour le calcul de la projection.

Soit l'opérateur de projection sur une droite vectorielle : $proj_u v = \frac{\langle u,v \rangle}{\langle u,u \rangle} u$

Le procédé de Gram-Schmidt est alors :

$$\begin{array}{lll} u_1 = v_1 & , & e_1 = \frac{u_1}{\|u_1\|} \\ u_2 = v_2 - proj_{u_1} v_2 & , & e_2 = \frac{u_2}{\|u_2\|} \\ u_3 = v_3 - proj_{u_1} v_3 - proj_{u_2} v_3 & , & e_3 = \frac{u_3}{\|u_3\|} \\ \vdots & & \vdots \\ u_k = v_k - \sum_{i=1}^{k-1} proj_{u_i} v_k & , & e_k = \frac{u_k}{\|u_k\|} \end{array}$$

E. Van Der Werf et R. Duin [190] ont également proposé une topologie de réseaux diabolo non-linéaires (voir figure 2.25). Ce type de réseau permet de calculer une Analyse en Composantes Principales non-linéaire.

R. Belaroussi et al. ont également utilisé ce type de réseaux de neurones dans une application de détection de visages [21].

2.4.2.4 Réseaux à délais

Cette architecture de réseau de neurones estime une probabilité a posteriori non seulement en fonction d'une observation, mais également en fonction de son contexte (ses voisins temporels).

La figure 2.26 illustre le fonctionnement d'un réseau TDNN à une couche cachée.

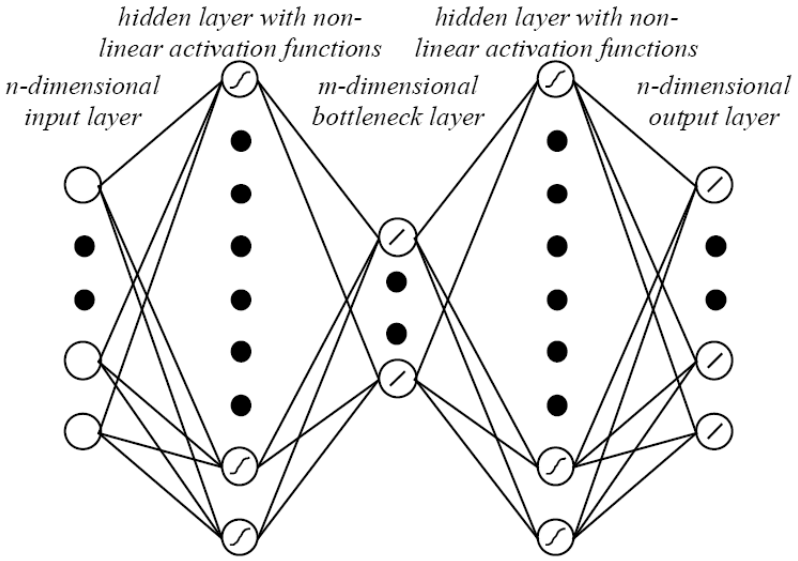


FIG. 2.25 – Réseau de neurones de type diabolo non-linéaire (figure extraite de [190]).

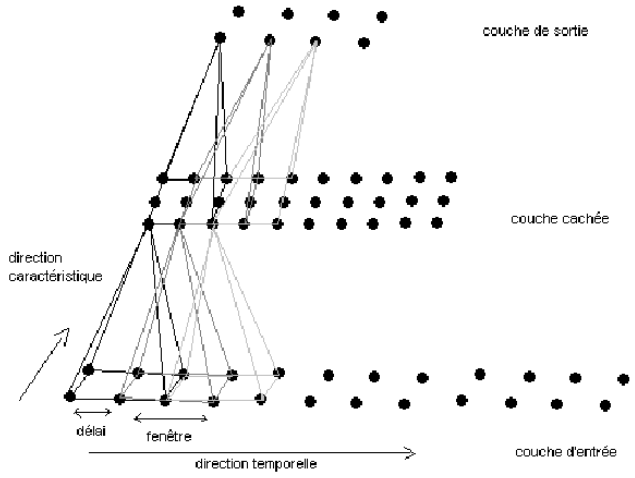


FIG. 2.26 – TDNN : le réseau prend en compte plusieurs vecteurs d'entrée en contexte pour déterminer la sortie.

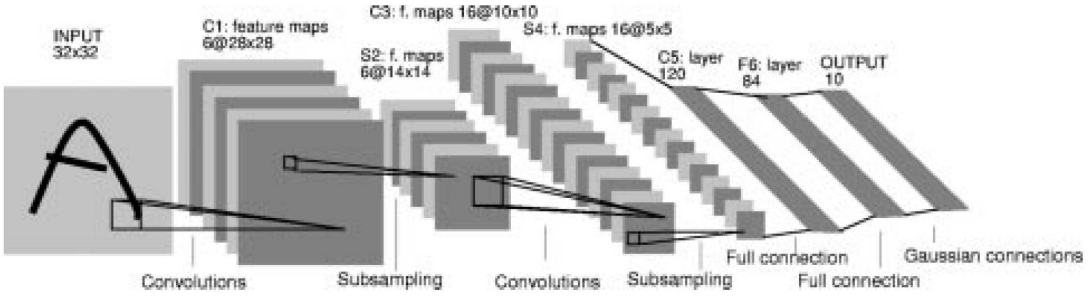


FIG. 2.27 – Architecture d'un réseau de neurones de type LeNet-5. Figure extraite de [94]

Les poids du réseau de neurones sont partagés, de telle sorte que la même fonction de transfert soit appliquée à chaque vecteur du contexte. Chaque neurone est connecté à un voisinage local temporel dans la couche précédente. Ce type de topologie permet d'augmenter la taille du contexte tout en conservant un nombre de coefficients restreint.

2.4.2.5 Réseaux à convolutions

Ce type de réseaux de neurones tire profit du partage des poids qui limite le nombre de coefficients à apprendre, et donc la complexité du modèle. Il peut donc être appliqué directement sur l'image, ce qui permet de s'affranchir de l'étape d'extraction de primitives. C'est un atout intéressant, compte tenu du fait que le choix du jeu de primitives est souvent l'un des facteurs limitants d'un système de reconnaissance : le réseau de neurones apprend automatiquement sa propre extraction de caractéristiques en fonction des données d'apprentissage.

Sa topologie (voir figure 2.27) alterne couches de convolution et couches de sous-échantillonnage et lui permet d'extraire progressivement des primitives de plus en plus haut niveau.

Appliqués à un champ complet, les réseaux de neurones à convolution sont répliqué dans l'espace, à la manière d'un TDNN. On parle alors de SDNN (Space Displacement Neural Network), qui sont une sorte de généralisation des TDNN à des données spatiales, à la différence près qu'il n'y a pas de notion de mémoire dans un SDNN. La réponse du réseau pour une position donnée dans l'image ne dépend pas de ses positions précédentes.

Le système LeNet 5, dont l'illustration est donnée plus haut, a été développé par LeCun et al. [94]. Il a notamment été appliqué à la reconnaissance automatique de chiffres.

Ce système est à la fois robuste aux translations, rotations, facteurs d'échelle, étirements / écrasements, et épaisseur des traits. Il propose également des capacités de robustesse exceptionnelles aux bruits : hachures, ratures, bruit poivre et sel [93].

Ce type de réseau de neurones en particulier donne d'excellents résultats sur la base de chiffres MNIST. La reconnaissance de chiffres isolés ou en contexte est une application pour laquelle ce type de réseaux est particulièrement bien adapté. Ils ont également été appliqués avec succès à des tâches de détection de visages [143, 142, 46], de plaques d'immatriculations [46], de la détection d'objets sous différents angles et différentes conditions

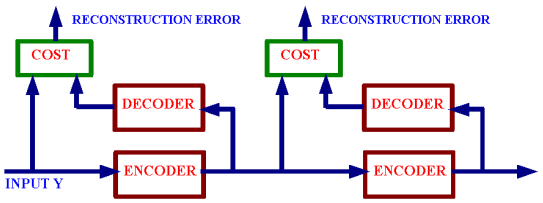


FIG. 2.28 – Les fonctions de transfert de chaque couple encodeur/décodeur sont liées. Chaque couple encodeur/décodeur est entraîné pour répliquer ses entrées. Les couches sont apprises successivement, les entrées d'une couche sont les sorties de la couche précédente. Figure extraite de [98]

d'éclairage [96], ou encore pour la commande automatique du déplacement d'un robot d'exploration [97]. A chaque fois, le système de reconnaissance tire avantage de la capacité du réseau de neurones à apprendre sa propre extraction de caractéristiques en fonction des données d'apprentissage.

Nous indiquerons les résultats obtenus avec un tel réseau de neurones sur une tâche de reconnaissance de chiffres farsis isolés (voir section 4.5).

2.4.2.6 Réseaux profonds (DBN)

G. E. Hinton et al. ont récemment ravivé l'intérêt pour les réseaux profonds (DBN pour Deep Belief Networks) en proposant une nouvelle procédure d'apprentissage [72]. Il s'agit d'un sujet de recherche actif [198].

Y. Lecun et al. [95] indiquent que les architectures peu profondes comme les Perceptrons Multi Couches ou les Machines à Vecteurs de Supports sont mal adaptées pour la résolution de problèmes complexes comme la vision artificielle. Le problème des architectures peu profondes (au plus deux couches de paramètres non-linéaires) est qu'elles peuvent nécessiter un nombre exponentiel d'éléments sur la couche cachée.

Les réseaux profonds sont composés de plusieurs couches de modules non-linéaires. L'apprentissage du réseau de neurones complet par rétropropagation du gradient fait converger le réseau vers un minimum local. La technique proposée par G. E. Hinton et al. [72] consiste à apprendre les couches séquentiellement et de façon non-supervisée.

La première couche W_1 est apprise avec sa transposée à la manière d'un diabolo (voir section 2.4.2.3), mais sans la contrainte qui impose que la couche cachée contienne moins de neurones que la couche d'entrée. Les paramètres sont entraînés de telle sorte que ce module ainsi constitué de la première couche et de sa transposée réplique ses entrées. Une fois l'adaptation terminée, les paramètres sont figés, et la transposée de la première couche est retirée.

On réitère la même opération avec la deuxième couche W_2. Cette dernière va être entraînée (avec sa transposée) à répliquer ses entrées (les entrées de W_2 sont les sorties de W_1). Une fois les paramètres de W_2 adaptés, on retire sa transposée.

On réitère à nouveau la même opération avec la troisième couche W_3. Les entrées de W_3 sont les sorties du signal propagé à travers W_1 puis W_2. Une illustration de ce processus est donnée figure 2.28.

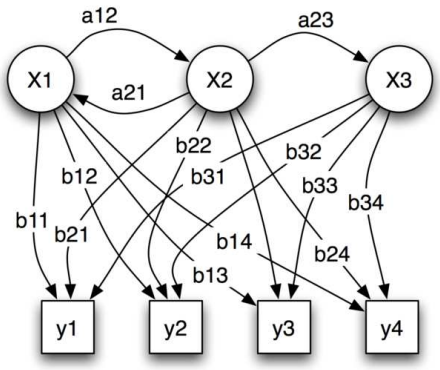


FIG. 2.29 – Modèle de Markov Caché. x_1, x_2, x_3 : états. y_1, y_2, y_3, y_4 : classes d'observations. a_{ij} : transition de l'état i vers l'état j. b_{xy} : probabilité d'émettre un symbole de la classe y depuis l'état x.

L'apprentissage en cascade des n couches $W_1, \ldots, W_n$ fournit l'extraction de caractéristiques d'un réseau de neurones de type MLP qui sera rajouté en cascade. Une fois ce réseau de neurones de décision appris, il est possible de relâcher la contrainte sur les poids pour optimiser globalement l'ensemble du réseau par rétropropagation du gradient.

La même idée est reprise par M. A. Ranzato et Y. LeCun pour l'apprentissage d'un réseau de neurones à convolution [154].

2.4.3 Modèles de Markov Cachés

2.4.3.1 HMM 1D

Les Modèles de Markov Cachés gauche-droite [152] [30] sont des modèles statistiques bien adaptés à la modélisation des données séquentielles. Ils sont de ce fait très utilisés en reconnaissance automatique de la parole et en reconnaissance automatique de l'écriture.

Ils ont notamment 3 avantages en reconnaissance de l'écriture :

- Ce sont des modèles stochastiques qui permettent de prendre en compte la variabilité des formes et du bruit qui perturbent la reconnaissance de l'écriture.
- Ce sont des modèles qui permettent de prendre en compte des séquences de longueurs variables. Ce point est particulièrement important en reconnaissance de l'écriture manuscrite, où la longueur des mots peut varier considérablement en fonction des styles d'écriture et des spécificités des scripteurs. Deux séquences de la même classe ne sont pas forcément de longueurs identiques.
- C'est une approche qui s'appuie sur un ensemble d'algorithmes standard et éprouvés. L'implémentation de ces algorithmes s'appuie sur des techniques bien maîtrisées de programmation dynamique. Un certain nombre de librairies et de modules sont également publiquement accessibles pour l'apprentissage et le décodage de Modèles de Markov Cachés [199] [51] [54] [99] ...

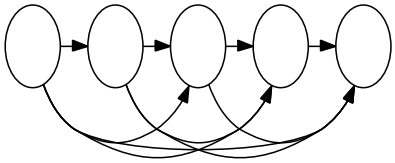


FIG. 2.30 – Topologie classique d'un HMM gauche/droite. On rajoute également parfois des transitions qui permettent de faire reboucler les états sur eux-mêmes.

Les HMM sont des automates probabilistes doublement stochastiques. La chaîne des états est "cachée" car non directement observable. Chaque état S émet un symbole observable O, selon sa propre distribution de probabilités $p(O_i|s_j)$. C'est cette séquence $(O_1, \ldots, O_T)$ qui constitue la séquence des symboles observés.

Cette séquence d'observations résulte donc bien d'un processus doublement stochastique : les transitions entre états et les émissions des observations.

En reconnaissance de l'écriture, on utilise en général des modèles gauche/droite (voir figure 2.30). Ces modèles permettent en effet de modéliser l'aspect séquentiel de l'écriture. Les observations correspondent aux symboles (vecteurs de primitives) parcourus dans l'ordre de lecture.

Les Modèles de Markov Cachés sont définis par le jeu de paramètres suivant :

$$L = \{\prod, A, B, T\}$$

$\prod = (\pi_i) = (P(q_1 = s_i))$: Probabilité de commencer dans l'état i
$A = (a_{ij}) = (P(q_{t+1} = s_j | q_t = s_i))$: Probabilité de transition de l'état i vers l'état j
$B = (b_i(O_t)) = P(o_t | q_t = s_i)$: Probabilité d'émettre o_t lorsqu'on est dans l'état i
$T = (\tau_i) = (P\,(q_T = s_i))$: Probabilité de terminer dans l'état i

Leur apprentissage est présenté en annexe A.

2.4.3.2 HMM Planaires

Dans [101], E. Levin et R. Pieraccini introduisent le DPW (Dynamic Plane Warping) comme une généralisation à deux dimensions du DTW (Dynamic Time Warping), un algorithme largement utilisé en reconnaissance automatique de la parole qui permet de faire un alignement entre deux séquences de longueurs différentes. Dans le même article, les auteurs présentent une interprétation statistique et une généralisation de l'approche DPW : les HMM planaires (ou PHMM pour Planar-HMM).

Les HMM planaires sont une extension des HMM pour résoudre des problèmes à deux dimensions. En pratique, les probabilités d'émission du HMM principal sont remplacés par la vraisemblance d'un deuxième HMM. En général, le HMM principal décrit l'image verticalement ligne par ligne, tandis que les HMM secondaires décrivent une ligne site par site. Ces sites sont un voisinage de pixels qui doivent être à la fois suffisamment grands pour permettre des mesures, et suffisamment petits pour s'attacher à extraire une caractéristique locale.

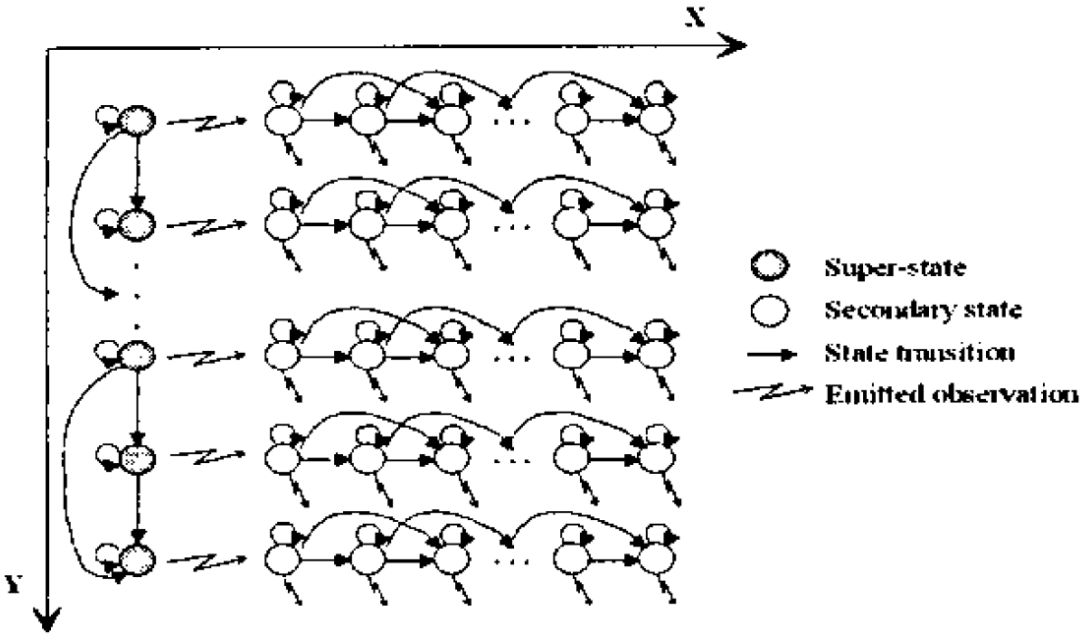


FIG. 2.31 – Topologie classique de PHMM (figure extraite de [64]).

Les PHMM modélisent les déformations élastiques sur l'ensemble des réalisations d'un même type de document : chaque état secondaire représente un ou plusieurs sites de l'image. L'ensemble des états peuvent être vus comme une matrice déformable qui permet de modéliser la variabilité d'un type de document.

Durant l'apprentissage, sur chaque site on extrait un vecteur de primitives, et on dispose également d'un label. La phase d'apprentissage consiste à adapter la matrice d'états pour qu'elle maximise la vraisemblance sur l'ensemble des données d'apprentissage.

Les PHMM ont ainsi été appliqués à la reconnaissance de chiffres manuscrits [101] [63].

Dans [138], O. E. Agazzi et al utilisent des PHMM de lettres dans une tâche de reconnaissance de mots imprimés très bruités. Dans cette architecture, les super-états sont horizontaux, les états sont donc organisés en bandes verticales (voir figure 2.32).

Dans [31], R. Bippus utilise des PHMM pour reconnaître des montants littéraux allemands. Il teste son système avec un alphabet de 5 PHMM pour un lexique de 47000 entrées. Sur cette application, les PHMM donnent de meilleurs résultats que les HMM-1D. Dans [64], J. Golenzer et al utilisent les PHMM pour localiser des régions d'intérêt dans un document numérisé. Ils testent leur système sur une tâche de localisation du montant littéral dans des images de chèques.

Les systèmes à base de HMM planaires ont également été appliqués à la reconnaissance de l'écriture arabe. Ces travaux seront présentés dans la section 3.2.2.4.5.

Les PHMM ont aussi été utilisés dans le traitement d'images naturelles, comme par exemple sur des tâches de reconnaissance de visages [56] ou le suivi de personnes [35].

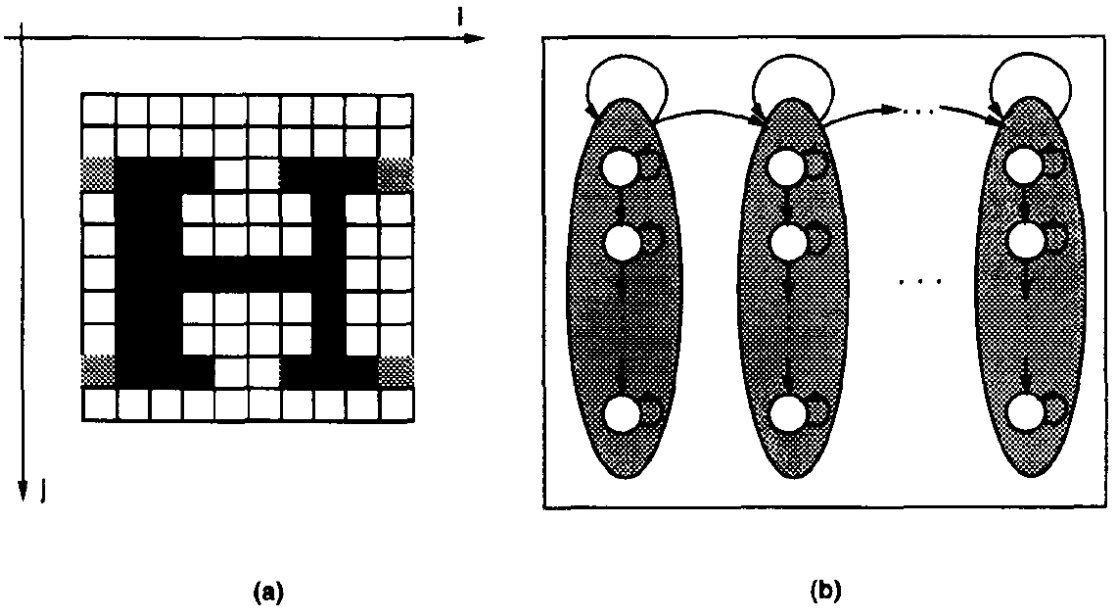


FIG. 2.32 – Modèle PHMM de lettre (figure extraite de [138]). (a) la case située à l'abscisse i et à l'ordonnée j représente l'état j associé au super-état i. Le niveau de gris associé à la case correspond à la probabilité d'activation d'un pixel noir. (b) Transitions entre états et super-états associés à la matrice (a).

2.4.4 Machines à Vecteurs de Support

Les méthodes à noyaux sont inspirées de la théorie de l'apprentissage statistique que Vladimir Vapnik. Les Machines à Vecteurs de Supports (SVM) [186] [37] [52] sont une famille des méthodes à noyaux, qui permettent de résoudre efficacement des problèmes de classification ou de régression.

Cette technique a connu un grand succès au cours des dernières années. Des outils comme la librairie LIBSVN [43] permettent de tester facilement un système à base de SVM, et ont contribué à son essor.

Les SVM sont des classifieurs à 2 classes, qui cherchent à maximiser la marge entre les deux classes. Le principe des SVM consiste à utiliser une transformation non linéaire pour effectuer un plongement des données dans un espace de plus grande dimension. Des données non linéairement séparables dans l'espace initial seront plus simples à classer dans l'espace de grande dimension (idéalement, ces données deviendront linéairement séparables, il sera donc possible de déterminer un hyperplan de séparation). L'idée est alors de trouver la frontière de séparation linéaire qui maximise la marge dans l'espace de grande dimension (voir figure 2.33). Cet hyperplan de séparation linéaire dans l'espace de grande dimension, revient d'une certaine manière à trouver une frontière de séparation non linéaire dans l'espace de départ.

Les données qui servent d'appui pour le calcul de la frontière de séparation sont les vecteurs de support. Seul un sous-ensemble des données d'entraînement jouent le rôle

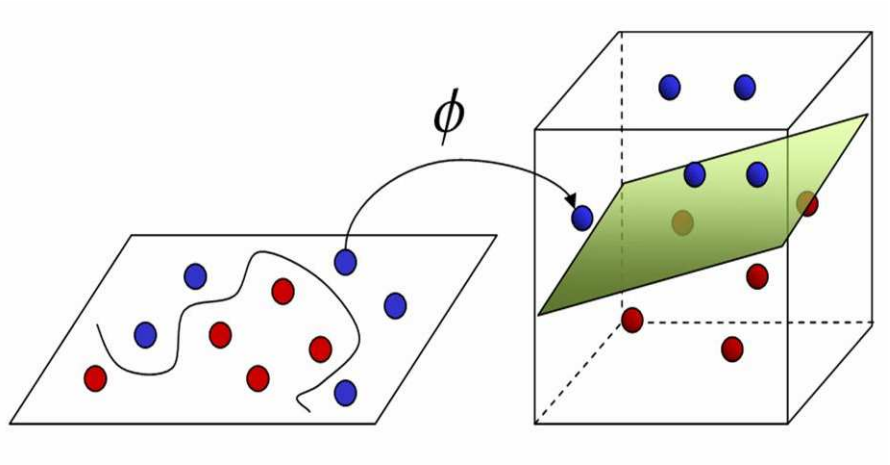


FIG. 2.33 – SVM : trouver une frontière de séparation linéaire dans un espace de grande dimension revient à trouver une frontière non-linéaire dans l'espace de départ.

de vecteurs de supports. C'est sur ce sous-ensemble qu'on s'appuie pour effectuer la classification.

Soit l'ensemble d'entraînement de l échantillons : $\{(x_i, y_i)\}$, avec $i = 1, \ldots, l$, et $y_i \in \{+1; -1\}$

La décision d'obtient en évaluant le signe de :

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i \cdot k(x, x_i) + b$$

Les vecteurs de support sont les vecteurs x_i pour lesquels les α_i sont non-nuls, et :

$$k(x, x_i) = \langle \Phi(x), \Phi(x_i) \rangle$$

$k(x, x_i)$ est une fonction noyau [164]. L'utilisation d'une fonction noyau appliquée aux données dans l'espace initial à la place du calcul du produit scalaire dans l'espace de grande dimension est appelé "kernel trick". Cette astuce permet de remplacer un produit scalaire dans un espace de grande dimension par une fonction noyau, plus rapide à calculer. De cette manière, un classifieur linéaire peut facilement être transformé en un classifieur non linéaire. Un autre avantage des fonctions noyau est qu'il n'est pas nécessaire d'expliciter la transformation Φ, seul le résultat du produit scalaire compte.

Les noyaux le plus souvent utilisés sont :

– Le noyau sigmoïde : $k(x, x_i) = tanh(K \cdot (x \cdot x_i) + \theta)$
– Le noyau polynomial : $k(x, x_i) = (K \cdot (x \cdot x_i) + 1)^p$
– Le noyau RBF : $k(x, x_i) = exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$

K, θ, p, σ^2 sont des paramètres des noyaux, dont on détermine l'optimum sur un problème donné à l'aide d'une grille.

Les SVM offrent généralement un assez bon comportement sur des tâches de classification [117].

Les SVM sont intrinsèquement un système de classification à deux classes. J. Milgram [125] étudie les différentes stratégies de combinaisons pour traiter le cas multi-classe. Il propose également une approche pour évaluer des probabilités de classification à l'aide des SVM, ce qui permettrait de les intégrer dans un environnement probabiliste.

2.4.5 Réseaux Transformateurs de Graphes

L'utilisation des automates à états finis pondérés est assez répandue dans le domaine de la reconnaissance de la parole [126].

Y. LeCun et al. proposent d'étendre ce formalisme à la reconnaissance de l'écriture [94]. Cette idée est également reprise par X. Dupré [55].

Les Réseaux Transformateurs de Graphes, ou GTN pour Graph Transformer Network, sont un cadre qui permet d'unifier les différents modules qui composent un système de reconnaissance, en les regroupant au sein d'un formalisme unique.

2.4.5.1 Définitions

2.4.5.1.1 Transducteur Un transducteur est un 7-uplet $(Q, I, F, \sum, \Delta, \delta, \sigma)$
- Q l'ensemble des états,
- $I \subseteq Q$ l'ensemble des états initiaux,
- $F \subseteq Q$ l'ensemble des états finaux,
- $\sum$ l'ensemble des symboles qui constituent l'alphabet d'entrée,
- Δ l'ensemble des symboles qui constituent l'alphabet de sortie,
- δ l'ensemble des transitions, qui va de $Q \times \sum$ vers Q,
- σ la fonction de sortie, qui va de $Q \times \sum$ vers Δ.

2.4.5.1.2 Accepteur Un accepteur est un transducteur qui ne possède qu'un seul alphabet.

En pratique, nous représenterons ces automates sous forme de graphes. En particulier, les résultats intermédiaires (graphe de segmentation, graphe de reconnaissance) sont des graphes accepteurs, tandis que les grammaires sont des graphes transducteurs.

De façon générale, chaque élément de la chaîne de reconnaissance (segmenteur, extracteur de primitives, reconnaisseur, et même modèle de langage) peut être vu comme une fonction de transfert d'un graphe accepteur vers un autre graphe accepteur.

2.4.5.1.3 Composition On introduit la composition $S \otimes T$ d'un accepteur S ayant en entrée des suites de symboles dans $\sum$, avec un transducteur T ayant en entrée des symboles dans $\sum$ et en sortie des symboles dans Δ :

$$(S \otimes T)(t) = \sum_{s \in \sum} (S(s) \times T(s,t))$$

La composition $S \otimes T$ assigne une probabilité w à l'accepteur d'une suite t s'il existe une suite s telle que S accepte s avec une probabilité u, T transduit s vers t avec une probabilité v, et $w = u \cdot v$

La composition d'un graphe accepteur dans $\sum$ et d'un graphe transducteur $\sum$ vers Δ donne un graphe accepteur dans Δ. Une telle composition permet d'extraire tous les chemins qui sont compatibles aux deux graphes, et assigne à chaque arc du graphe résultat une probabilité égale au produit des probabilités de ses deux arcs parents.

2.4.5.2 Utilisation en reconnaissance

Les éléments du système de reconnaissance acceptent des entrées sous forme de graphes, et fournissent des sorties sous forme de graphes également.

La fonction de transfert du module en question s'exprime à travers une composition entre le graphe qui représente ce module, et le graphe d'entrée.

Les graphes sont un formalisme intéressant pour représenter l'information. Les algorithmes qui manipulent ces structures sont nombreux et éprouvés (pruning, calcul de meilleurs chemins, etc ...).

Le recours à ce formalisme permet d'envisager l'optimisation globale de la chaîne complète, plutôt que l'optimisation de chaque module indépendamment les uns des autres.

Cette approche sera expérimentée dans le chapitre 5 sur une tâche de reconnaissance de noms de familles écrits en français en lettres capitales.

2.5 Post-Traitements

2.5.1 Distance de Levenshtein

En reconnaissance de l'écriture, la distance de Levenshtein est la méthode la plus couramment utilisée pour le calcul de distance d'édition entre deux chaînes. On appelle distance de Levenshtein entre deux mots S et T le coût minimal pour aller de S à T en effectuant les opérations élémentaires suivantes :

- substitution d'un caractère de S en un caractère de T.
- ajout dans S d'un caractère de T.
- suppression d'un caractère de T.

On définit un coût de substitution pour tous les couples de symboles, ainsi qu'un coût d'insertion et de suppression pour chacun de ces symboles. Dans le cas le plus simple, le coût de substitution vaut 0 lorsque les symboles sont identiques et 1 s'ils sont différents, les coûts d'insertion et de suppression sont également égaux à 1.

Mais ces différents coût peuvent être adaptés en fonction d'une application particulière. Les coûts de substitution, d'insertion et de suppression peuvent être définis manuellement de façon heuristique, et ces coefficients peuvent également être adaptés sur un ensemble de données.

Soit s la longueur de la chaîne S et t la longueur de la chaîne T. Lalgorithme qui permet de calculer la distance d'édition entre les chaînes S et T est donné par :

1: Construire une matrice M de $s+1$ lignes et $t+1$ colonnes
2: Initialiser la première ligne à : $[0, 1, \ldots, t]$
3: Initialiser la première colonne à : $[0, 1, \ldots, s]$
4: **for** $i = 1$ to t **do**

		T	C	T	A	G	A	C	T
	0	1	2	3	4	5	6	7	8
T	1	**0**	1	2	3	4	5	6	7
G	2	1	**1**	2	3	3	4	5	6
T	3	2	2	**1**	2	3	4	5	5
A	4	3	3	2	**1**	**2**	3	4	5
A	5	4	4	3	2	2	**2**	3	4
C	6	5	4	4	3	3	3	**2**	3
T	7	6	5	4	4	4	4	3	**2**

TAB. 2.5 – Calcul de distance d'édition entre les mots TCTAGACT et TGTAACT. La distance entre ces deux chaînes est : 2.

5: **for** $j = 1$ to s **do**
6: $M_{i,j} = min\left(M_{i-1,j} + C_{ins}(S_j), M_{i,j-1} + C_{suppr}(T_i), M_{i-1,j-1} + C_{subst}(S_j, T_i)\right)$
7: **end for**
8: **end for**
9: Resultat = $M_{s,t}$

Avec :
$M_{i,j}$: l'élement de la matrice M situé ligne i et colonne j
S_j : la j-ème lettre du mot S
T_i : la i-ème lettre du mot T

Un exemple de calcul de distance d'édition entre les mots S = "TCTAGACT" et T = "TGTAACT" est donné dans le tableau 2.5.

2.5.2 N-gram

Les N-gram [174] sont une modélisation statistique du langage. L'idée est d'observer la fréquence des enchaînements de lettres à l'intérieur des mots, ou la fréquence des enchaînements des mots à l'intérieur des phrases.

Les N-gram permettent donc de stocker des connaissances a priori sur la langue. Ces connaissances pourront être utilisées pour pondérer les résultats d'une reconnaissance automatique sans dictionnaire.

Le paramètre N correspond à la taille du contexte considéré. Plus le contexte est grand et plus il est porteur d'informations, mais plus la taille du corpus devra être importante.

Pour pallier le manque de données, F. Perraud et al. [151] et H. Yamamoto et al. [195] proposent de regrouper certains symboles en classes, et de ne s'intéresser qu'aux transitions entre classes.

En pratique, on se limite généralement à des bi-gram (on prend en compte la lettre précédente) ou des tri-gram (on prend en compte les deux lettres précédentes).

Les N-gram peuvent être représentés sous forme de graphes.

La figure 2.34 donne un exemple de modèle de bi-gram représenté sous forme de graphe. Dans cet exemple, l'alphabet se compose de 3 symboles : $\{A, B, C\}$. Les mots composés d'un seul caractère sont exclus.

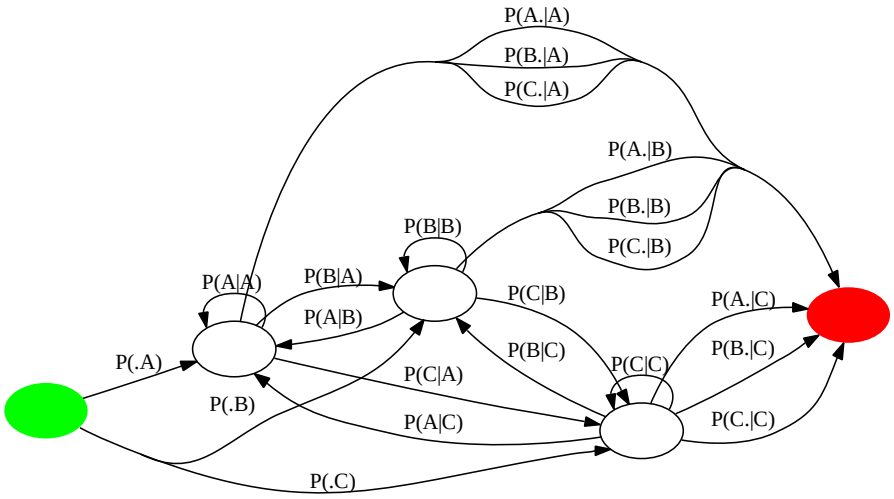


FIG. 2.34 – Exemple de modèle de Bigrammes. Vert : état initial. Rouge : état final.

$P(.A)$ symbolise la probabilité que le mot commence par le symbole A
$P(A.|x)$ symbolise la probabilité que le mot se termine par le symbole A, sachant que la lettre précédente est x.

2.6 Mesures de performances

2.6.1 Reconnaissance/Substitution

Le taux de reconnaissance correspond au rapport :

$$T_{reco} = \frac{N_{corr}}{N_{tot}}$$

T_{reco} : taux de reconnaissance
N_{corr} : nombre de documents correctement reconnus
N_{tot} : nombre total de documents

Lorsque le système de reconnaissance renvoie une mesure de confiance, il est possible d'établir un seuil de rejet, et donc d'évaluer le taux de reconnaissance en fonction de ce

seuil :

$$T_{reco(seuil)} = \frac{N_{corr(seuil)}}{N_{tot}}$$

$T_{reco(seuil)}$: taux de reconnaissance pour un seuil donné
$N_{corr(seuil)}$: nombre de documents correctement reconnus dont le score de confiance est supérieur à un seuil donné
N_{tot} : nombre total de documents

De manière analogue, on peut également définir le taux de substitution de la manière suivante :

$$T_{sub(seuil)} = \frac{N_{err(seuil)}}{N_{tot}}$$

$T_{sub(seuil)}$: taux de substitution pour un seuil donné
$N_{err(seuil)}$: nombre de documents mal reconnus dont le score de confiance est supérieur à un seuil donné
N_{tot} : nombre total de documents

$T_{reco(seuil)}$	14.6	31.5	43.6	50.0	55.3	62.5	69.0	76.2	79.2	79.6	79.6	79.6	79.6	80.0
$T_{sub(seuil)}$	0.00	0.90	1.45	2.50	4.07	8.06	13.6	17.8	19.9	20.2	20.4	20.4	20.4	20.4
Seuil	990	950	900	850	800	700	600	500	400	300	200	100	50	0

Tab. 2.6 – Tableau reconnaissance / substitution : courbe 1.

$T_{reco(seuil)}$	55.2	63.0	65.0	69.5	70.5	75.0	76.4	78.4	79.5	79.9	80.0	80.0	80.0	80.0
$T_{sub(seuil)}$	0.34	1.14	1.64	1.98	4.76	8.18	13.4	17.5	19.4	19.9	20.0	20.0	20.0	20.0
Seuil	990	950	900	850	800	700	600	500	400	300	200	100	50	0

Tab. 2.7 – Tableau reconnaissance / substitution : courbe 2.

La figure 2.35 montre deux courbes de reconnaissance / substitution. Les deux systèmes de reconnaissance ont le même taux de reconnaissance brute : 80% de reconnaissance et 20% d'erreur. Mais les mesures des taux de reconnaissance et de substitution montrent que le deuxième système a un meilleur comportement que le premier : si l'on choisit un point de fonctionnement aux alentours de 2% de substitution (on autorise le système à faire 2% d'erreur dans les réponses qu'il renvoie) :

– Le système 1 traitera correctement environ 46% des données, en ne rendant une réponse uniquement lorsque le score de fiabilité sera supérieur à 875. Les documents pour lesquels le score est inférieur à ce seuil seront rejetés.

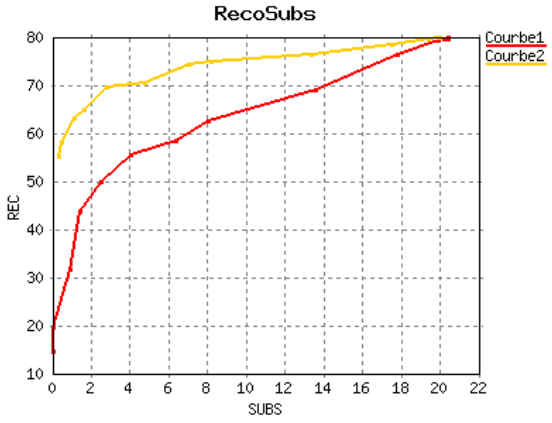


FIG. 2.35 – Courbes de Reconnaissance/Substitution.

– Le système 2 traitera correctement environ 70% des données, en ne rendant une réponse uniquement lorsque le score de fiabilité sera supérieur à 850. Les documents pour lesquels le score est inférieur à ce seuil seront rejetés.

En pratique, dans une application industrielle, un système de traitement automatique des documents vise rarement les 100% de reconnaissance sur l'ensemble du flux de documents.

Un système de reconnaissance performant visera à automatiser le maximum du flux de documents, tout en maintenant le taux de substitution proche de celui qu'introduirait un opérateur humain.

Dans le cas du traitement automatique de chèques par exemple, le taux de substitution d'un opérateur humain se situe aux alentours de 1%. Le point de fonctionnement (le seuil de confiance en deçà duquel le système rejette le document) sera donc choisi de telle sorte que le système de traitement automatique de chèques tolère une substitution similaire. Les documents rejetés sont traités manuellement.

2.6.2 Rappel/Précision

Les mesures de Rappel / Précision sont utilisées en extraction d'information.
Le taux de rappel est défini comme suit :

$$TR = \frac{N_{occ}}{N_{tot}}$$

TR : taux de rappel
N_{occ} : nombre d'occurrences correctes extraites
N_{tot} : nombre d'occurrences totales à extraire

On définit également le taux de précision, qui est une mesure de la quantité de bruit dans les résultats renvoyés :

$$TP = \frac{N_{occ}}{N_{extr}}$$

TP : taux de précision
N_{occ} : nombre d'occurrences correctes extraites
N_{extr} : nombre d'occurrences totales extraites

On définit également la F-mesure comme critère de performance :

$$F = \frac{2 \cdot (Precision \cdot Rappel)}{Precision + Rappel}$$

On l'appelle également F_1 : il s'agit du cas particulier pour lequel Rappel et Précision sont pondérés de la même manière. Dans le cas plus général, on aura :

$$F_\alpha = \frac{(1 + \alpha) \cdot (Precision \cdot Rappel)}{\alpha \cdot Precision + Rappel}$$

Outre F_1, les deux valeurs souvent utilisées sont F_2 (pour laquelle le rappel compte deux fois plus que la précision) et $F_{0.5}$ (pour laquelle la précision compte deux fois plus que le rappel).

Dans [129], H. Mouchere et E. Anquetil définissent deux natures de rejet.
- le rejet en confusion : correspond aux données qui sont proches des frontières entre les classes.
- le rejet en distance, qui permet d'écarter les données parasites, qui n'appartiennent à aucune classe.

Ils décrivent également une méthode générique pour sélectionner ces seuils de rejet.

Chapitre 3

Reconnaissance de l'écriture arabe manuscrite

3.1 Problématique

3.1.1 Présentation de la langue arabe

L'arabe littéral, arabe moderne unifié ou encore classique est le nom que l'on donne à une variante de la langue arabe, utilisée comme langue officielle dans tous les pays arabes, et comme langue commune entre pays arabes. Elle est également employée dans la plupart des écrits et, à l'oral, dans les situations officielles ou formelles (discours religieux, politiques, journaux télévisés).

L'arabe littéral se distingue ainsi de l'arabe dialectal, qui est la langue vernaculaire parlée au quotidien et ce depuis l'expansion de l'islam. Cette variété de la langue recouvre plusieurs dialectes locaux pouvant varier assez fortement d'un pays à l'autre. Dans tous les pays arabes, un dialecte national composé par plusieurs dialectes locaux est parlé. Aucun d'entre ces dialectes n'est identique complètement à l'arabe classique ou littéraire.

Le monde arabe se compose de vingt-deux pays s'étendant de la Mauritanie à l'ouest, au sultanat d'Oman à l'est. Sa population est estimée à 355 millions de personnes. Il se divise en Machreq à l'est, et en Maghreb à l'ouest (la frontière entre le Machreq et le Maghreb est le Nil). L'intercompéhension est possible, voire facile, entre les locuteurs des variantes d'arabe dialectal du Machreq, de même l'est-elle entre les locuteurs des variantes d'arabe dialectal d'Afrique du nord, mais elle est plus difficile entre les locuteurs du Machreq et du Maghreb.

Dans la réalité des échanges linguistiques, il n'y a pas de séparation étanche entre arabe littéral et arabe dialectal, mais plutôt un continuum où dominent les formes mixtes. Néanmoins, les spécificités géographiques et historiques influencent les différentes versions de l'arabe dialectal. Citons par exemple :

- l'influence du berbère pour les dialectes maghrébins
- l'influence de l'égyptien ancien pour l'égyptien
- l'influence du phénicien pour le syro-libano-palestinien, le tunisien, l'algérien et le maltais

- l'influence de l'italique pour le maltais
- les emprunts français, italiens et castillans pour les dialectes maghrébins et le libanais
- les emprunts turcs pour le syro-libano-palestinien.

En ce qui concerne les styles d'écriture, les différences sont également importantes. Ainsi, la plupart des systèmes présentés lors de la compétition ICDAR 2007 [112], entraînés sur des données tunisiennes, accusent une baisse de performances de 10 à 15 points entre les résultats obtenus sur un ensemble de test en provenance de Tunisie, et un autre ensemble de test en provenance des Emirats-Arabes-Unis. La proportion de ligatures verticales est plus élevée chez les scripteurs du Moyen-Orient, et la variabilité des styles d'écriture est également plus importante. Des adaptations de la chaîne et des réentraînements semblent nécessaires. Notons également que le farsi (persan), utilisé principalement en Iran et en Afghanistan, partage un grand nombre de points communs avec l'écriture arabe. Cet aspect est hors du cadre de cette thèse : nous nous restreignons ici à l'étude de l'écriture arabe manuscrite sur des données en provenance de scripteurs tunisiens. Sauf mention contraire, les données utilisées proviennent de la base IFN/ENIT [147], une base de noms de villes tunisiennes.

3.1.2 Difficultés inhérentes à la reconnaissance de l'écriture arabe

3.1.2.1 Alphabet

L'alphabet arabe comporte 28 lettres (voir le tableau 3.1) La forme des lettres dépend de leur position dans le mot. Certaines lettres prennent jusqu'à 4 formes différentes : par exemple le (ع ـعـ ـع) ou le (ه ـه ـه ه).

Mais pour la plupart des lettres, les formes début/milieu et fin/isolé sont identiques à la ligature près. La présence d'une ligature avec la lettre précédente ou avec la lettre suivante ne modifie pas la forme de la lettre de manière significative (pas plus que dans l'écriture manuscrite cursive latine).

En arabe, les ligatures se situent toujours au niveau de la ligne d'écriture, c'est à dire qu'il n'existe pas de lettre à liaison haute comme le 'o' ou le 'v' en alphabet latin.

En écriture arabe, il existe toutefois des ligatures verticales, que nous étudierons plus particulièrement dans la section 3.1.2.5.

3.1.2.2 Signes diacritiques

Le terme 'signe diacritique' peut porter à confusion : dans certains travaux, seules les voyelles arabes sont appelées diacritiques. Dans d'autres travaux, en revanche, tous les signes secondaires sont appelés diacritiques, qu'il s'agisse des voyelles, des points ou des autres signes (chadda, madda, hamza, ...).

C'est cette deuxième terminologie que nous employons ici : un signe diacritique est une composante secondaire d'une lettre, qui vient la compléter ou en modifier le sens. Dans la suite de cette thèse, les "signes diacritiques" désigneront à la fois points, voyelles et autres signes secondaires.

Isolé	Fin	Milieu	Debut	Code ArabTeX
ا ou ى	ا ou ى	ا	ا	a ou Y
ب	ب	بـ	بـ	b
ت ou ة	ت ou ة	تـ	تـ	t ou T
ث	ث	ثـ	ثـ	_t
ج	ج	جـ	جـ	^ g
ح	ح	حـ	حـ	.h
خ	خ	خـ	خـ	_h
د	د	د	د	d
ذ	ذ	ذ	ذ	_d
ر	ر	ر	ر	r
ز	ز	ز	ز	z
س	س	سـ	سـ	s
ش	ش	شـ	شـ	^ s
ص	ص	صـ	صـ	.s
ض	ض	ضـ	ضـ	.d
ط	ط	طـ	طـ	.t
ظ	ظ	ظـ	ظـ	.z
ع	ع	عـ	عـ	‘
غ	غ	غـ	غـ	.g
ف	ف	فـ	فـ	f
ق	ق	قـ	قـ	q
ك	ك	كـ	كـ	k
ل	ل	لـ	لـ	l
م	م	مـ	مـ	m
ن	ن	نـ	نـ	n
ه	ه	هـ	هـ	h
و	و	و	و	w
ي	ي	يـ	يـ	y

Tab. 3.1 – Alphabet arabe.

FIG. 3.1 – Points en arabe : un, deux ou trois points.

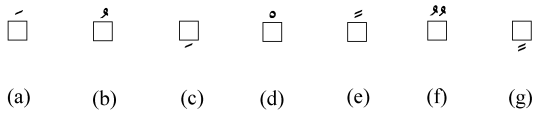


FIG. 3.2 – Voyelles en arabe : (a) A , (b) OU , (c) I , (d) - , (e) AN , (f) OUN, (g) IN.

3.1.2.2.1 Des points nécessaires pour différencier les lettres Dans l'alphabet arabe, 15 lettres parmi les 28 possèdent un ou plusieurs points. Ces signes diacritiques sont situés soit au-dessus, soit en dessous de la forme à laquelle ils sont associés, mais jamais les deux à la fois.

La figure 3.1 illustre la variabilité des styles d'écriture des points ou groupes de points en écriture manuscrite arabe. Un groupe de deux points peut ainsi s'écrire sous forme d'une seule, ou de deux composantes connexes. On remarque la très forte similarité entre deux points reliés par un trait, et une voyelle de type 'A' ou 'I' dont les exemples sont donnés figure 3.2.

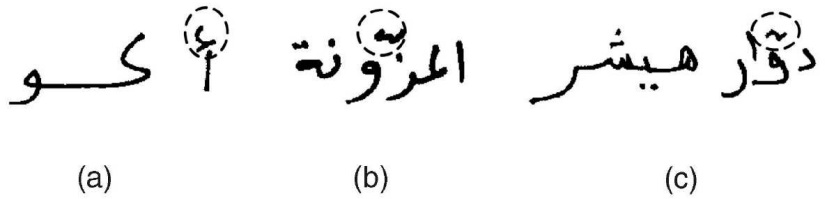
Un groupe de trois points peut donner lieu à une, deux ou trois composantes connexes, en fonction du style d'écriture.

3.1.2.2.2 Les voyelles En arabe, les voyelles ne sont pas des lettres, mais des signes diacritiques associés aux lettres sur lesquelles ils s'appliquent (voir figure 3.2).

En général on ne représente pas les voyelles, sauf dans les manuels scolaires.

L'absence de voyelles peut toutefois être source de confusions. Comme le rappellent Y. Bahou et al dans [193], un mot peut avoir plusieurs voyellations possibles et par conséquent plusieurs catégories grammaticales. Dans certains cas, une phrase peut donc avoir deux voyellations différentes, ce qui nous donne deux structures syntaxiques possibles. Par exemple, dans le cas :

يخشى الأستاذ الطلبة



(a) (b) (c)

FIG. 3.3 – Autres signes diacritiques : (a) hamza, (b) chadda, (c) madda.

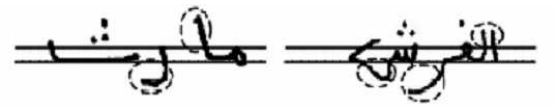


FIG. 3.4 – Les ascendants et descendants sont entourés. La bande de base est donnée à titre indicatif

qui peut se voyeller des deux manières suivantes :

يَخْشَى الأُسْتَاذُ الطَلَبَةَ : L'enseignant craint les étudiants.

يَخْشَى الأُسْتَاذَ الطَلَبَةُ : Les étudiants craignent l'enseignant.

La première voyellation correspond à la structure (Verbe + Sujet + Complément), tandis que la deuxième voyellation donne la structure (Verbe + Complément + Sujet).

Les voyelles peuvent parfois être mentionnées sur certaines lettres pour lever l'ambiguïté et faciliter la lecture. Mais en général, les scripteurs les omettent purement et simplement, et c'est au lecteur qu'est réservé le soin d'interpréter correctement le sens de la phrase en fonction du contexte.

3.1.2.2.3 Les autres signes diacritiques Les autres signes diacritiques sont la hamza, la chadda et la madda (voir figure 3.3). La chadda est une accentuation de la lettre (c'est l'équivalent d'une consonne doublée). Hamza et madda suivent des contraintes morpho-syntaxiques plus complexes.

3.1.2.3 Ascendants et descendants

Comme dans l'écriture latine, l'écriture arabe contient des ascendants et des descendants (voir figure 3.4). En arabe, les descendants peuvent se prolonger horizontalement sous la bande de base, ce qui introduit une superposition verticale entre la lettre qui comprend le descendant et la lettre suivante.

FIG. 3.5 – Un mot peut être composé de plusieurs composantes connexes (pseudo-mots). Le nombre de pseudo-mots ne dépend pas de la longueur du mot, mais des lettres qui le composent.

3.1.2.4 Une ou plusieurs composantes connexes par mot

6 lettres ne sont pas liées à leur successeur : ا, د, ذ, ر, ز, و. Ces lettres introduisent donc une coupure dans le mot. Un pseudo-mot est une unité connexe regroupant une ou plusieurs lettres sous forme d'une séquence. Un mot peut être composé d'un ou plusieurs pseudo-mots (voir figure 3.5).

En manuscrit, l'espacement entre les différents pseudo-mots d'un même mot n'est pas forcément systématiquement supérieur à l'espacement entre deux mots différents, ce qui pose parfois des problèmes de segmentation.

Lorsqu'une des 22 lettres (28 moins les 6 qui ne se lient pas avec la suivante) apparaît dans sa forme "fin de mot" ou "isolée", cela signifie obligatoirement que l'on arrive à la fin d'un mot.
Remarque : les lettres (ط et ظ) sont les seules parmi les 22 à ne pas prendre une forme différente lorsqu'elles sont isolées ou en fin de mot.

Par ailleurs, les articles (le, la, les) font partie du mot auquel ils sont rattachés. La séquence ال (un pseudo mot qui contient la lettre ا isolée, suivi d'un autre pseudo-mot qui commence par la lettre ل) correspond nécessairement au début d'un mot.

3.1.2.5 Ligatures verticales

En écriture arabe, il n'y a pas de liaisons hautes comme le 'v' ou le 'o' en latin : les ligatures se situent au niveau de la ligne support de l'écriture (ligne de base). En revanche, les scripteurs sont libres de constituer certains groupes de deux ou trois lettres liées verticalement en début de pseudo-mot. Ce sont les ligatures verticales (figures 3.7 et 3.8). En général très complexes à segmenter, nous choisissons de les reconnaître telles quelles.

FIG. 3.6 – ﻻ sous forme d'une ou de deux composantes connexes.

FIG. 3.7 – Ligatures verticales et inversion de l'ordre du tracé.

3.1.2.5.1 Ligatures verticales connectées ou non Par définition, les éléments qui constituent une ligature verticale sont connectés. Ils appartiennent donc tous au même pseudo-mot. Toutefois, en fonction du style d'écriture et en raison des phénomènes de levés de plume, il est possible qu'une ligature verticale soit composée de plusieurs composantes connexes (voir figure 3.6).

3.1.2.5.2 Recouvrements partiels La forme de certaines ligatures, combinée au style d'écriture du scripteur (en particulier lorsque le mot est écrit rapidement), peuvent parfois donner des résultats fâcheux, comme des recouvrements partiels, voire même une inversion de l'ordre de lecture (voir figure 3.7). Des situations très problématiques qui heureusement restent rares.

Un moyen de traiter des données problématiques serait d'avoir recours à un algorithme de reconstitution du tracé dynamique, qui permettrait de retrouver l'ordre de l'écriture. Ce problème n'est pas traité dans cette thèse.

3.1.2.5.3 Statistiques des ligatures sur IFN/ENIT A titre indicatif, la base IFN/ENIT (qui sera détaillée dans la section 3.2.1.4), fournit une analyse statistique détaillée du nombre de ligatures verticales et de leur distribution (cf figure 3.8).

Les expériences seront menées sur cette base. Nous considérerons que ces statistiques sont représentatives d'un problème de reconnaissance de noms de villes tunisiennes.

Nous exploiterons donc ces informations pour établir l'alphabet de corps de lettres qui sera présenté dans la section 4.2.

Label	Quantity	Shape	Label	Quantity	Shape
ا_EJ_B	789		ح_MJ_B	363	
ا EJ M	1066		ح Mم MJ B	64	
ا EJ B	120		ح MJ B	100	
أ_EJ_B	357		خ_MJ_B	304	
ج_MJ_B	530		م_MJ_B	450	

FIG. 3.8 – Analyse statistique des ligatures verticales dans la base IFN/ENIT.

(a) (b)

FIG. 3.9 – Un même mot écrit avec et sans ligature verticale.
(a) écriture avec ligature verticale : حلق الجمل.
(b) écriture sans ligature verticale : حلق الـجمل

3.1.2.6 Différentes manières d'écrire un mot

La possibilité pour le scripteur d'utiliser des ligatures verticales (voir paragraphe 3.1.2.5), ainsi que la possibilité d'écrire la lettre ة de deux manières différentes lorsqu'elle est en fin de mot (selon qu'elle soit reliée à la lettre précédente ـتـ / ـةـ , ou isolée ت / ة), ont pour conséquence le fait qu'un même mot arabe peut s'écrire de différentes manières. Il est donc possible qu'une même séquence de lettres donne lieu à des mots ayant la même signification, mais écrits de façons différentes (voir figure 3.9).

En fin de mot, la lettre ة peut être "ouverte" ("TA maftouha" : ـت) ou "fermée" ("TA marbouta" : ـة). : تونكت ≡ تونكة

3.2 Etude bibliographique

3.2.1 Principales bases de données existantes

Dans [59], Atwell et al. listent un ensemble d'outils pour l'analyse de corpus de textes arabes appliqués au traitement automatique de la langue naturelle écrite ou orale. Des corpus de textes arabes conséquents [66] sont disponibles pour l'analyse morphologique

de la langue arabe.

En revanche, dans le domaine de la reconnaissance automatique de l'écriture manuscrite, les bases de données d'images annotées sont moins nombreuses.

Le domaine de la reconnaissance automatique de l'écriture arabe manuscrite a ainsi péché pendant de nombreuses années de l'absence d'une base de données de référence, qui permette des comparaisons objectives entre les différents systèmes. Des bases plus conséquentes mais payantes comme la base Kharma/Ahmed/Ward ou la base de chèques du CENPARMI sont apparues. Mais c'est la base IFN/ENIT, gratuite pour la recherche académique, qui s'est imposée comme la base de données de référence pour la comparaison des performances des systèmes de reconnaissance de l'écriture arabe manuscrite.

3.2.1.1 Kharma/Ahmed/Ward 1999

En 1999, N. Kharma et al [83] proposent une base de données, obtenue à partir des contributions de 500 étudiants :

- 37000 mots arabes
- 10000 chiffres, à la fois arabes (maghreb) et indiens (moyen orient)
- 2500 signatures
- 500 phrases

Les images sont disponibles à la fois en niveaux de gris et sous forme binarisées. Cette base est disponible pour les chercheurs Canadiens, mais son accès est soumis à restrictions pour les chercheurs du reste du monde.

3.2.1.2 AHDB

En 2002, Almaadeed et al. présentent une base de données collectées à l'aide d'une centaine de scripteurs [5].

Chacun des scripteurs était invité à écrire :

- Chacun des mots du vocabulaire des montants numériques.
- Trois montants numériques écrits en toutes lettres.
- Quelques lignes de texte libre.

Les 20 mots les plus utilisés par tous les scripteurs sont annotés manuellement.

3.2.1.3 CENPARMI

En 2003, Y. Al-Ohali et al ont achevé la réalisation d'une base de données pour la reconnaissance de chèques arabes manuscrits [6].

Cette base se compose de 7000 images de chèques saoudiens issus de la pratique bancaire (voir figure 3.10), scannés en niveaux de gris avec une résolution de 300 dpi.

Cet ensemble d'images a permis de mettre au point plusieurs bases de données :

- Une base de 1547 montants littéraux de chèques.
- Une base de 1547 montants numériques de chèques.
- Une base de 23325 pseudo-mots.
- Une base de 9865 chiffres indiens isolés.

Il serait également possible de mettre au point une base de données de dates à partir de ces images.

FIG. 3.10 – Exemple de chèque saoudien, et son montant littéral.

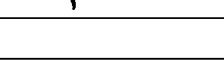
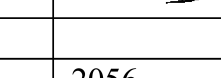
Image		
Ground truth:		
Postcode	6132	2056
Global word	حمّام بياضة	روّاد
Character shape sequence	م_A|ا_E|مـ_M|ح_B| ة_E|ض_B|ا_E|ي_M|ب_B|	و_A|ـّ_L|ر_A| د_A|ا_A|
Baseline y1,y2	70,50	46,39
Baseline quality	B1 (B1=OK; B2=bad)	B1
Quantity of words	2	1
Quantity of PAWs	4	4
Quantity of characters	9	4
Writing quality	W1 (W1=OK; W2=bad)	W1

FIG. 3.11 – Deux images et leurs annotations respectives dans la base IFN/ENIT. B pour Begin (début), M pour Middle (milieu), A pour Alone (isolé) et F pour Final (finale).

3.2.1.4 IFN/ENIT

M. Pechwitz et al introduisent la base IFN/ENIT en 2002 [147].

Il s'agit d'une base de données d'images de noms de villes tunisiennes. Outre la séquence de lettres, sont également annotées la forme que prend chacune des lettres au sein du mot, la présence des signes diacritiques secondaires (voir section 3.1.2.2), et une approximation de la ligne de base.

411 scripteurs ont été mis à contribution, pour collecter environ 26400 noms de villes (dans un lexique de 937 villes) et plus de 210000 caractères.

Dans cette base, l'annotation est réalisée de telle sorte qu'une séquence de lettres contienne également l'information de la forme que prend chacune des lettres au sein du mot (voir figure 3.11).

Cette base, par sa taille et sa disponibilité (gratuite pour des travaux académiques) s'est imposée comme la base de référence des travaux en reconnaissance de l'écriture manuscrite arabe. Elle sert de support à l'organisation des compétitions internationales dans le domaine de la reconnaissance de l'écriture arabe manuscrite : ICDAR Arabic Handwritten Competition, 2005 et 2007 (voir section 4.3).

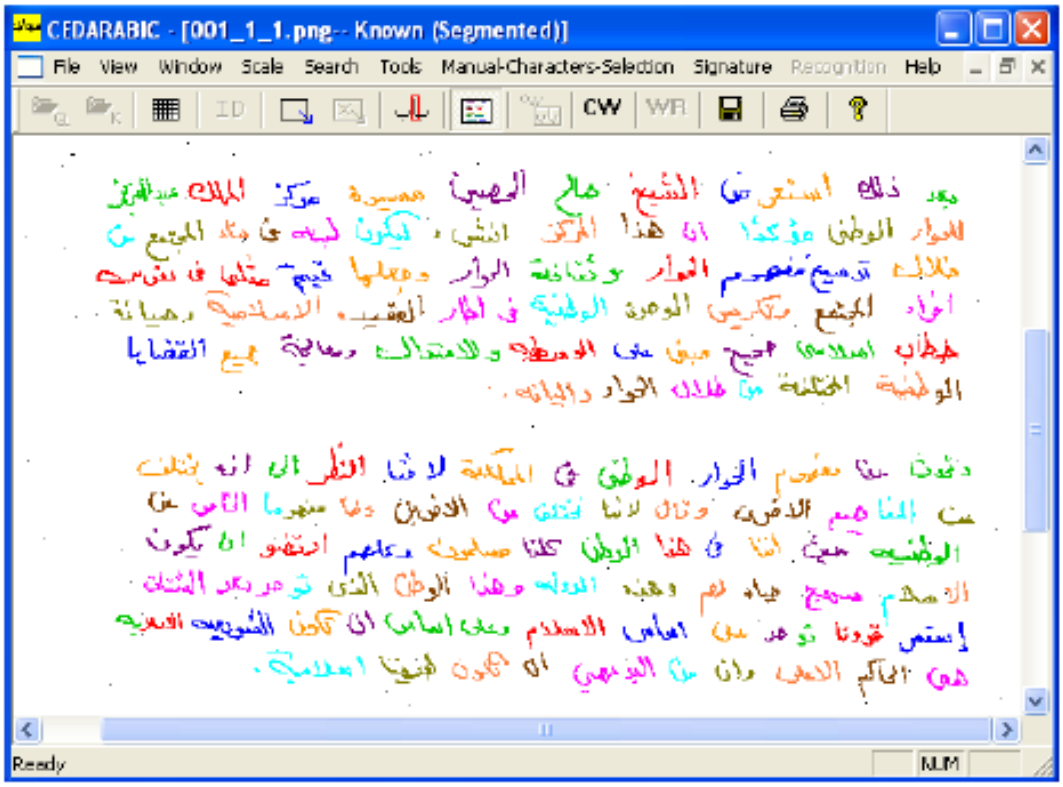


FIG. 3.12 – Exemple de segmentation en pseudo-mots qui porte à confusion (figure extraite de [158]).

3.2.1.5 CEDARABIC

La base de données CEDARABIC [158] comprend 10 scripteurs différents. Chacun d'entre eux a écrit 10 pages complètes différentes de texte manuscrit. Chaque page comprend entre 150 et 200 mots, soit un total d'environ 20000 mots. Les documents sont scannés en 300 dpi. Une annotation complète des documents est faite manuellement : segmentation en mots, mais aussi séquence des lettres arabes, ainsi que la prononciation et la traduction en anglais.

Cette base de données est donc particulièrement adaptée pour des travaux sur la segmentation en mots et pour la recherche de mots clés dans une page d'écriture manuscrite en langue arabe.

3.2.1.6 HODA farsi digits

Cette base de données a été rendue publique en février 2007 par H. Khosravi et E. Kabir [88]. Il s'agit d'une base conséquente de chiffres farsi isolés, d'une taille comparable à la base MNIST. Elle est composée de 102352 chiffres extraits à partir d'environ 12000 formulaires, remplis par des étudiants et des personnels universitaires. Les documents

sont scannés en 200dpi, binarisés, et la base est prédécoupée en un ensemble d'apprentissage (60000 exemples), un ensemble de test (20000 exemples), plus un troisième ensemble qui contient des données restantes (22352 exemples).

Nous avons effectué des expériences sur cette base de données. Ces travaux sont présentés dans la section 4.5.

3.2.1.7 Applied Media Analysis : Arabic-Handwriting-1.0

Cette base de données, qui a été rendue publique en septembre 2007, n'a pour l'instant fait l'objet d'aucune publication scientifique de la part de ses auteurs.

Ses principales caractéristiques sont :

- 5000 pages manuscrites
- plusieurs types de documents : diagrammes, mémos, formulaires, listes (comprenant des chiffres indiens et arabes), poèmes
- différents types de stylos : crayons à papier, marqueurs larges ou fins, pointes fines, stylos à bille. Ecriture en noir ou en couleur.
- images disponibles en binaire ou en niveaux de gris
- annotations XML : mots ou pseudo-mots, format UTF-8

Son coût est de 500$ pour une recherche académique, 1500$ pour l'industrie.

Les principaux avantages de cette base sont donc :

- les différents types de stylos avec lesquels les documents ont été écrits, qui forcent à prendre en compte cette variabilité de styles, soit dans le reconnaisseur, soit dans une phase de prétraitements de normalisation.
- le mélange imprimé/manuscrit présent dans les formulaires, qui permet de traiter des problématiques d'extraction : recherche de mot-clé et information associée à un champ particulier.
- le fait de disposer d'images en niveaux de gris permet d'envisager la possibilité d'une chaîne de reconnaissance en niveaux de gris, et donc un système dans lequel il n'est pas nécessaire de prendre une décision de seuillage brutal (noir ou blanc) dès les premières étapes du traitement.

Son principal défaut semble toutefois être lié à la manière dont cette base est annotée :

- les boîtes de pseudo-mots sont tronquées lorsque les composantes connexes appartenant à deux pseudo-mots différents se superposent verticalement. Voir figure 3.13.
- il n'y a pas d'annotation au niveau forme des lettres. En effet, la forme des lettres se déduit de leur position dans le mot, sauf dans le cas des alias (voir paragraphe 4.2.2.1), par exemple les lettres ﺖ et ﺔ. Il est donc plus compliqué d'entraîner les modèles de formes de lettres (voir paragraphe 4.2) sur cette base : annotations

FIG. 3.13 – Exemples de segmentations en pseudo-mots qui portent à confusion.

manuelles supplémentaires pour différencier ces classes de formes qui appartiennent à la même classe de lettres, ou annotations automatiques à partir d'un système déjà entraîné en conservant le meilleur candidat.

3.2.2 Différentes approches et systèmes existants

3.2.2.1 Surveys existants

Les surveys suivants sont triés par ordre chronologique. Pour chacun d'entre eux, nous présenterons ses principaux apports, et les conclusions des auteurs.

3.2.2.1.1 ICDAR 1997 Dans [11] (1997), Adnan Amin analyse différentes approches pour la reconnaissance automatique de l'écriture arabe. Il présente les approches On-Line/OffLine, ainsi que les approches Globale/Analytique. Il pose quelques unes des problématiques fondamentales de la reconnaissance de l'écriture arabe (alphabet, signes diacritiques, styles d'écriture, codage de l'écriture, pseudo-mots, ...). Il liste un nombre important de systèmes, sans toutefois entrer dans le détail de leur fonctionnement.

Pour l'auteur, le sujet reste ouvert et de nombreuses voies d'amélioration sont possibles. L'évolution des performances des systèmes de reconnaissance de l'écriture arabe au cours des 10 dernières années montre qu'il avait raison.

3.2.2.1.2 CIFED 2000 Dans [22] (2000), Najoua Ben Amara et al. s'intéressent aux approches markoviennes qui ont été mises en oeuvre pour résoudre le problème de la reconnaissance de l'écriture arabe. Concernant l'écriture arabe manuscrite, Najoua Ben Amara et al. passent notamment en revue les HMM-1D, les HMM-planaires.

Selon Ben Amara et al., les HMM-1D donnent des résultats encourageants, tout comme dans le cas de l'écriture latine. Les HMMs sont indépendants de l'alphabet considéré. Mais Ben Amara et al. soulignent également les limitations classiques d'une modélisation linéaire, qui là encore, ne sont pas propres à l'écriture arabe.

Pour pallier ce problème, ils introduisent les PHMM, qui prennent en compte des variations bidirectionnelles de l'écriture. Les auteurs justifient en particulier ce choix sur l'arabe en raison du degré de difficulté supplémentaire que représente la présence de ligatures verticales. Ce dernier argument concernant la présence de ligatures verticales, n'est

pas le plus pertinent. Les symboles qui correspondent aux ligatures verticales peuvent en effet être reconnus comme tels dans leur globalité, sans qu'il soit nécessaire de les segmenter (voir paragraphe 4.2.1.3).

3.2.2.1.3 PAMI 2006 Le Survey de Liana M. Lorigo et Venu Govindaraju [104] (2006), est le plus complet sur la reconnaissance de l'écriture arabe à l'heure actuelle.

L. M. Lorigo et V. Govindaraju passent en revue la plupart des problèmes spécifiques à la reconnaissance de l'écriture arabe. Ils consacrent un paragraphe à la reconnaissance de l'écriture imprimée, mais la plus grosse partie du document est réservée à la reconnaissance de l'écriture manuscrite.

Ils présentent quelques bases de données, qui sont également décrites ici aux paragraphes 3.2.1.2, 3.2.1.3 et 3.2.1.4.

Ils découpent la chaîne de traitement d'un système de reconnaissance en fonctionnalités :

- Représentation
- Segmentation
- Extraction de primitives
- Moteur de reconnaissance, où ils distinguent :
  - Système à base de règles
  - Réseau de Neurones
  - Modèles de Markov Cachés
  - Systèmes hybrides.

Les auteurs réservent également un paragraphe à la compétition ICDAR 2005 (voir paragraphe 4.3.1), dans lequel ils présentent brièvement les participants et les résultats.

En conclusion, les auteurs expliquent que des systèmes de reconnaissance de l'écriture arabe donnent des performances satisfaisantes sur des applications contraintes (petite taille de lexique, forme des mots relativement contrainte...). L'avenir de la discipline se situe dans la capacité des systèmes à traiter de l'écriture libre, comme des courriers manuscrits. De telles applications nécessitent des modèles de langage développés, qui sont pour l'instant des voies largement inexplorées en langue arabe. L'intégration de contraintes morphologiques (analyse de la formation des mots à partir d'affixes, suffixes, racines) permettrait également de reconnaître des mots hors-vocabulaire.

3.2.2.1.4 SACH 2006 Dans [20] (2006), A. Belaïd et Ch. Choisy s'appuient sur le modèle de McClelland et Rumelhart (voir figure 3.14) pour structurer leur analyse des systèmes de reconnaissance de l'écriture arabe manuscrite.

A. Belaïd et Ch. Choisy insistent sur le fait que ce modèle est également applicable à la reconnaissance de l'écriture manuscrite arabe, à condition de lui rajouter un niveau

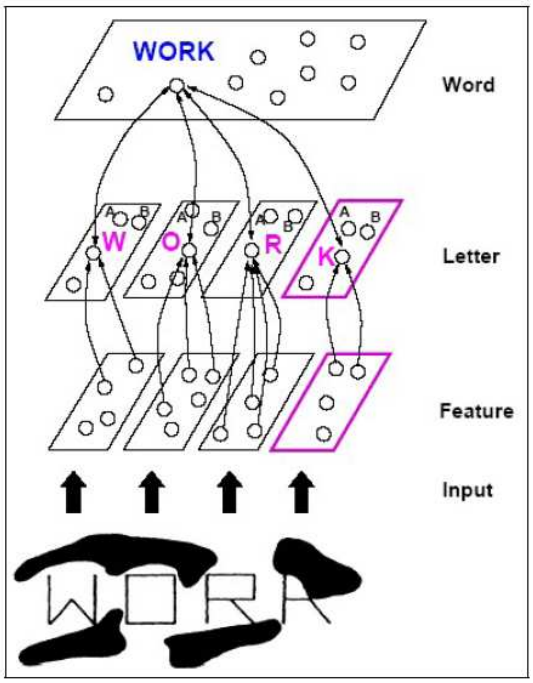


FIG. 3.14 – Modèle de McClelland et Rumelhart [115]

intermédiaire : le niveau pseudo-mot.

Les auteurs soulignent le fait que les primitives de bas niveau sont indépendantes du langage. En revanche, les primitives de plus haut niveau dépendent de la langue considérée, et nécessitent le développement de procédures spécifiques à la langue. Ces deux types de primitives sont complémentaires.

Ils mettent en évidence la difficulté de la segmentation en mots [160], qui est selon eux l'une des raisons qui expliquent l'absence de système commercial de reconnaissance de l'écriture arabe manuscrite.

Ils mettent également en évidence le rôle crucial que jouent les pseudo-mots dans la langue arabe. Contrairement à l'écriture latine, en arabe l'entité élémentaire n'est pas le mot. Les approches globales devraient donc s'appuyer sur les pseudo-mots plutôt que sur les mots complets. Les approches analytiques devraient également exploiter cette particularité. Les auteurs déplorent le faible nombre de travaux qui exploitent cette particularité de la langue arabe dans leur chaîne de reconnaissance.

Les auteurs insistent également sur le fait qu'il est raisonnablement admis depuis longtemps dans la communauté de la reconnaissance de l'écriture cursive latine, que la segmentation en lettres est un problème mal posé. Rechercher une segmentation idéale en lettres dans un mot cursif est un problème qui n'a pas de solution. Or la segmentation d'un mot manuscrit arabe est un problème plus complexe que la segmentation d'un mot latin [161]. Les auteurs déplorent donc la quantité de travaux qui s'attachent à optimiser la reconnaissance de lettres extraites à partir d'une segmentation manuelle, compte tenu qu'il semble illusoire qu'une segmentation automatique puisse fournir une segmentation

en lettres idéale en entrée d'un reconnaisseur de caractères isolés.

Finalement, A. Belaïd et Ch. Choisy montrent un vif intérêt pour les systèmes hybrides qui, selon eux, semblent très prometteurs : ils combinent efficacement différents niveaux perceptifs, permettant ainsi de discriminer des mots sans avoir accès à une description complète. L'ajout d'informations locales à un système global permet d'étendre le vocabulaire en limitant les confusions. Les approches hybrides ne nécessitent pas une segmentation complète, et sont moins sujettes aux perturbations induites par les problèmes de perte d'information.

3.2.2.1.5 ISSPA 2007 Dans [47] (2007), Mohamed Cheriet fait une synthèse des différentes approches, avec la volonté d'ouvrir la réflexion sur de futures applications industrielles. M. Cheriet part du constat suivant : les systèmes de reconnaissance de l'écriture manuscrite arabe, même s'ils obtiennent des performances encourageantes, n'ont pour l'instant fait leurs preuves que sur des données académiques, dans le cadre d'applications à l'environnement contraint. Il n'y a pas de système commercial de reconnaissance de l'écriture arabe manuscrite.

Il pose un certain nombre de questions ouvertes :
- Nécessité de procéder à une segmentation ?
- Quel paradigme de reconnaissance utiliser ?
- Quelle technique de reconnaissance utiliser ?
- Faire un post-traitement, ou essayer d'intégrer les contraintes au sein de la chaîne de reconnaissance
- Redéfinir de nouvelles architectures, ou s'appuyer sur des techniques existantes ?

M. Cheriet insiste également sur l'importance de l'analyse morphologique. Il s'interroge sur le meilleur endroit pour intégrer l'analyseur morphologique : au sein de la chaîne, ou à la fin en tant que post-traitement ?

Il s'interroge également sur les moyens de constituer un lexique pour guider un système de reconnaissance à vocabulaire ouvert, et conclut par l'intérêt d'utiliser le Coran en tant que corpus, pour en dériver un lexique de manière automatique ou semi-automatique.

En s'appuyant sur le système développé par H. Miled et al. [120] [124] [122], il présente une étude de cas dans laquelle il combine trois niveaux perceptifs :
- le niveau global, qui considère uniquement des indices visuels, qui sont issus des signes diacritiques et des tracés. Chaque mot est représenté par une série d'indices visuels qui sont traités par un système à base de Modèles de Markov Cachés.
- un deuxième niveau, dans lequel les mots sont segmentés en graphèmes. Chaque graphème est alors transformé en un vecteur d'observation discret. Des Modèles de Markov Cachés sont entraînés pour décoder les séquences d'observations ainsi générées. A ce niveau, les signes diacritiques ne sont pas pris en compte, ce qui permet de réduire le nombre de modèles à estimer.
- le niveau pseudo-mots, qui est un niveau intermédiaire entre les deux premiers. Des Modèles de Markov Cachés sont à nouveau utilisés, cette fois pour modéliser les

probabilités de transition inter-pseudo-mots.

Une combinaison de ces trois modèles est ainsi effectuée pour évaluer la classe d'un mot inconnu.

Pour terminer, M. Cheriet conclut sur l'intérêt de travaux de recherche qui viseraient à mieux intégrer le traitement de la langue naturelle dans une chaîne de reconnaissance de l'écriture arabe.

En reprenant l'idée du découpage du Survey de L. M. Lorigo et V. Govindaraju [104], nous présenterons un état de l'art de la reconnaissance de l'écriture arabe manuscrite, en regroupant les articles par fonctionnalités.

Nous passerons ainsi en revue les prétraitements, la segmentation, la reconnaissance, et les post-traitements.

3.2.2.2 Prétraitements

3.2.2.2.1 Correction du skew et ligne de base Dans [1], Sehad et al s'appuient sur le fait que la transformée de Hough donne de bons résultats pour déterminer des alignements. Cette technique est suffisamment robuste pour détecter des angles entre 0 et 180°. Toutefois, son utilisation est mal adaptée à la détection de l'inclinaison du texte dans les documents en raison du temps de calcul important que nécessite cette méthode. L'idée de l'article est de chercher à limiter la quantité d'informations sur laquelle on effectue la transformation de Hough. Dans un premier temps, les auteurs cherchent à localiser des liaisons entre lettres. Ils déterminent les centres de gravité des boîtes englobantes des liaisons détectées. Ensuite, ils appliquent une transformée de Hough sur ces centres de gravité. Cette méthode donne des résultats intéressants : 96% pour un seuil de tolérance de 1° sur de l'imprimé. Les auteurs ne parlent pas de l'application de cette méthode au cas manuscrit. Dès le début, ils écartent l'idée de travailler sur les centres de gravité des pseudo mots, en justifiant ce choix par le fait qu'en écriture arabe les ascendants et les descendants ne sont pas symétriques par rapport à la bande de base. Mais ils ne donnent pas de résultats comparatifs des deux approches (travailler sur le centre de gravité des liaisons VS travailler sur le centre de gravité des pseudo mots) pour valider cette hypothèse.

Dans [38], Peter Burrow propose 3 méthodes pour la correction de l'inclinaison

- histogrammes calculés successivement sur des petites variations d'angle. Le pic maximal correspond à la bonne inclinaison.
- transformée de Hough : chaque point dans l'espace de Hough est un accumulateur associé à un angle et à une distance de l'origine. La valeur de ce point dans l'espace de Hough correspond au nombre de points qui sont sur cette ligne dans l'espace image. Les zones de valeurs importantes dans l'espace de Hough correspondent aux alignements les plus importants dans l'espace image. Cette technique permet de déterminer l'alignement du texte, et donc de le corriger si nécessaire.
- Une ACP sur l'image permet de déterminer l'axe principal de la distribution des

pixels. Le vecteur principal de l'ACP correspond à la direction de la ligne de base. Cette technique donne un angle, mais pas la position verticale de la ligne. Pour retrouver la position de la ligne de base, il faut effectuer une rotation de l'angle correspondant et récupérer la position du pic de l'histogramme horizontal.

Néanmoins, ces méthodes sont sensibles aux mots courts qui possèdent beaucoup d'ascendants et de descendants. La présence de nombreux signes diacritiques peut également perturber l'évaluation de la ligne de base. Pour traiter le problème des ascendants et des descendants, l'auteur propose de calculer le squelette du mot, puis de retirer les pixels qui correspondent aux extrémités des strokes (spur removal). En répétant cette opération plusieurs fois (par expérience, l'auteur choisit une dizaine de fois), on réduit d'autant l'influence des ascendants et des descendants. Cette technique semble intéressante, mais pourtant elle dégrade les résultats. Selon l'auteur, le problème vient du fait qu'appliquée telle quelle, cette méthode ne supprime pas uniquement les ascendants et les descendants, mais va également rogner des parties de tracés qui sont dans la bande de base, ce qui va au final dégrader les performances.

Dans [149], Pechwitz et al. présentent une méthode basée sur le squelette pour extraire la ligne de base. Chaque composante connexe du mot donne lieu à un squelette. Des primitives sont extraites sur chaque squelette, comme des mesures sur la boîte englobante (hauteur, largeur, aire, ratio largeur/hauteur) ou sur le squelette lui-même (longueur du squelette, nombre de noeuds, nombre d'arcs, degré des noeuds, longueur de chaque arc, angles, ...). Des mesures statistiques sont également extraites sur l'ensemble des squelettes pour une image donnée (moyenne et variance de la taille des boîtes englobantes, moyenne et variance de la longueur des squelettes, ...). A partir de ces primitives, le système extrait d'autres caractéristiques de plus haut niveau, à l'aide d'un ensemble de règles : détection des points, des chadda, des structures simples ou des courbes qui correspondent aux descendants (voir figure 3.15 pour quelques exemples). Ces caractéristiques de plus haut niveau sont utilisées pour extraire une première estimation de la ligne de base. Certaines de ces caractéristiques sont écartées. On considère uniquement les lignes qui sont plus ou moins horizontales, et on évalue la position du point central du tracé, auquel on attribue un poids en fonction de la longueur du tracé. On calcule alors une somme pondérée de ces candidats pour estimer la position y de la ligne de base. Cette première estimation de la ligne de base permet de définir une bande dont la hauteur correspond à 1/3 de la hauteur du mot, dans laquelle on va chercher des points supports plus précis. Des bons candidats sont les minimums des boucles qui sont dans cette bande, et les points supérieurs des longs tracés courbes qui correspondent aux descendants. Une ligne de base est obtenue par régression linéaire de tous ces candidats.

La ligne de base étant annotée manuellement dans la base IFN/ENIT, les auteurs évaluent leur algorithme d'extraction de la ligne de base par rapport à l'annotation. Visuellement, les auteurs estiment qu'un décalage de 7 pixels par rapport à l'annotation

donne un résultat tout à fait satisfaisant. Jusqu'à 15 pixels de décalage, la ligne de base est acceptable. L'algorithme proposé extrait une ligne de base dont l'erreur est inférieure à 15 pixels dans 95% des cas.

En reprenant cette procédure d'évaluation, l'analyse de la qualité de notre extraction de ligne de base est donnée dans la section 4.1.2.4.

Mais au delà d'une mesure de la qualité basée sur des pixels, une mesure plus pertinente de la qualité de la ligne de base est le taux de reconnaissance obtenu par le système de reconnaissance dans lequel est intégrée cette extraction de la ligne de base. C'est ce que font Pechwitz et al. dans [150]. Ils comparent la méthode précédente (basée sur le squelette [149]) avec une autre méthode basée sur des projections et la transformée de Hough. Ils montrent ainsi la supériorité de la méthode basée sur le squelette. Le taux de récupération de la ligne de base correcte (un écart inférieur à 7 pixels par rapport à l'annotation) est de 82,7% pour histogrammes, contre 87.8% pour le squelette. Après la reconnaissance, l'écart se tasse et la différence est moins importante : 82% avec les histogrammes, 84% avec les squelettes (la limite max étant 89,5% avec la bande de base annotée).

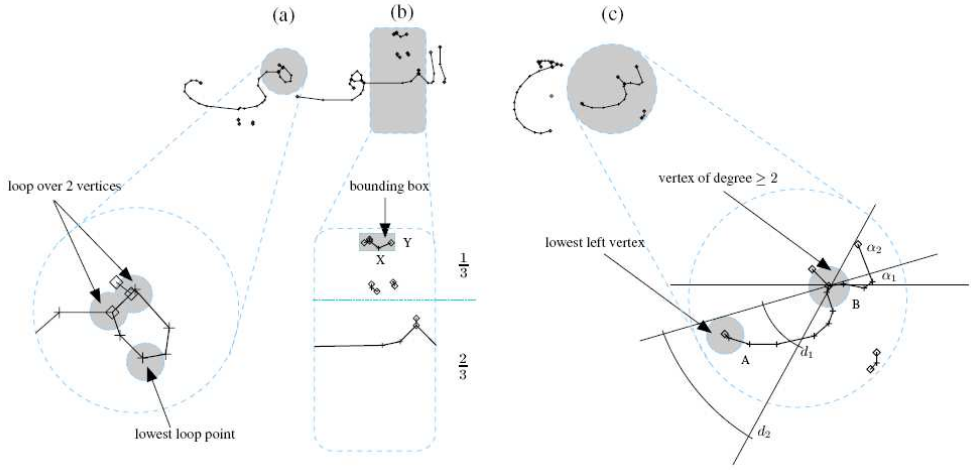


FIG. 3.15 – Détection de la ligne de base (figure extraite de [149]). (a) détection de boucle et extraction du point de la boucle le plus bas ; (b) détection de chadda ; (c) détection d'un tracé courbe qui correspond à un descendant

L'extraction de la partie supérieure de la bande de base est plus erratique. Les auteurs contournent ce problème de la détection de la ligne de base supérieure du tracé : une fois la ligne de base inférieure estimée, ils considèrent que la ligne de base supérieure est parallèle à la ligne de base inférieure, et qu'elle se situe à 40% de la hauteur entre la ligne de base inférieure et le sommet du mot. Dans [148], les mêmes auteurs montrent qu'une méthode basée sur la transformée de Hough pour évaluer la partie supérieure de la bande de base donne de moins bons résultats que la méthode naïve ci-dessus.

Les histogrammes sont couramment utilisés pour extraire la bande de base. Dans [114], S. Masmoudi et H. Amiri localisent la bande de base à l'aide d'un seuil fixé sur l'histogramme de projection horizontale.

- Limite haute : 18% de la valeur max de l'histogramme horizontal, au dessus du pic.
- Limite basse : 30% de la valeur max de l'histogramme horizontal, en dessous du pic.

Cette méthode est sensible à la présence de signes diacritiques et aux successions de descendants qui peuvent affecter l'histogramme et donc perturber l'extraction de la bande de base. Mais comme nous le décrivons dans le paragraphe 4.1.2, il est possible d'introduire des contraintes qui limitent l'influence de ces facteurs.

Dans [161], T. Sari et al utilisent des histogrammes de projection pour diviser le mot en 3 bandes horizontales, mais ne détaillent pas les seuils utilisés pour définir ces trois zones.

- zone haute : ascendants
- zone médiane : correspond à la bande de base
- zone basse : zone des descendants et des diacritiques inférieurs

Les diacritiques peuvent potentiellement se situer dans la zone haute et dans la zone médiane. Les auteurs ne mentionnent pas cet aspect.

3.2.2.2.2 Normalisation des caractères Dans [108], Maddouri et al normalisent des caractères à l'aide d'une caractérisation elliptique du contour de l'écriture par coefficients de Fourier. La transformée de Fourier est calculée sur à partir de la chaîne de Freeman du contour. La normalisation se fait dans l'espace de Fourier, et une transformée inverse permet de récupérer le texte normalisé. Les normalisations portent sur l'angle d'inclinaison, la taille et la position.

Il est possible d'évaluer une distance dans l'espace de Fourier entre l'image normalisée et celle de référence, donc de faire de la classification.

Dans [150], Pechwitz et al proposent une normalisation de l'écriture qui s'appuie sur la ligne de base. Les auteurs comptent le nombre de transitions noir/blanc le long de trois lignes parallèles à la ligne de base. Ils en déduisent un nombre approximatif de caractères dans le mot. Ils normalisent le mot en largeur afin d'obtenir une taille moyenne par caractère constante. Une normalisation verticale est également effectuée de manière à avoir une taille des ascendants/descendants qui soit constante.

Ces normalisations sont effectuées sur le squelette. La normalisation de l'épaisseur du tracé est obtenue en appliquant un filtre gaussien sur le squelette normalisé. L'image normalisée du mot est donc en niveaux de gris (voir figure 3.16). Cet article est à notre connaissance le seul à se baser sur des images en niveaux de gris pour effectuer la reconnaissance.

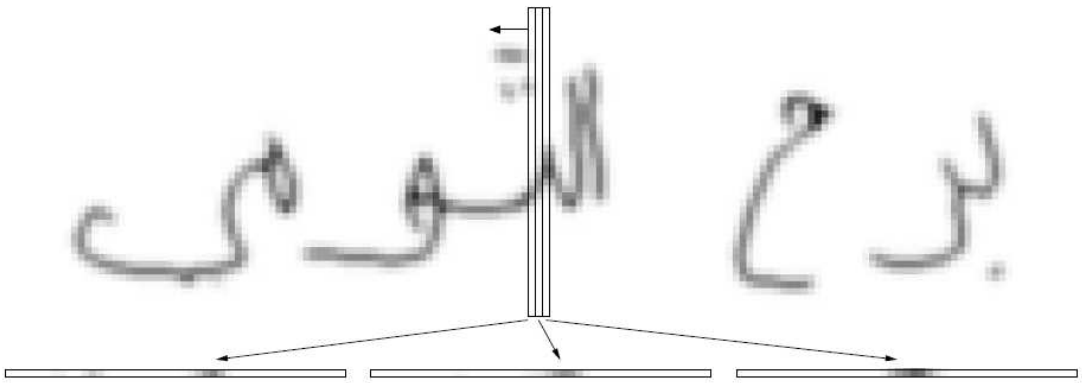


FIG. 3.16 – Squelette normalisé en hauteur et en largeur de l'image figure 3.15, sur lequel on applique un filtre gaussien pour obtenir une image du mot dont l'épaisseur du trait est normalisée. Une fenêtre glissante de 3 pixels de large extrait les vecteurs de primitives qui seront soumis en entrée du reconnaisseur (figure extraite de [150]).

FIG. 3.17 – On centre un cercle sur le point d'origine, et on récupère les pixels de la forme qui intersectent ce cercle. Les points trop près des bords du contour sont écartés. Le point est choisi parmi les candidats restants

FIG. 3.18 – En cas d'ambiguïté comme c'est le cas lorsqu'on rencontre une boucle, on choisit le candidat qui minimise le changement de direction du tracé (critère de régularité)

Reconstitution de l'ordre du tracé [38]

3.2.2.2.3 Reconstitution de l'ordre du tracé Dans [38], Peter Burrow adapte un algorithme de reconstitution du tracé afin d'extraire des primitives Online. Il définit un point extrême comme étant un point qui termine un tracé. L'algorithme part du point extrême le plus en haut à droite dans le mot, et suit le tracé jusqu'à atteindre le point extrême le plus en bas à gauche. Du point de départ, on trace un cercle. On prend un cercle centré sur ce point et on détermine le point suivant, qui doit être à la fois sur le cercle et dans le tracé (voir figure 3.17). Puis on centre le cercle sur ce nouveau point et on progresse ainsi de proche en proche. En cas d'ambiguïté on applique un principe de régularité (voir figure 3.18). Pour que l'algorithme se comporte correctement vis-à-vis des boucles, il convient également d'effacer le contenu de la composante connexe au fur et à mesure du parcours, pour éviter de passer deux fois au même endroit.

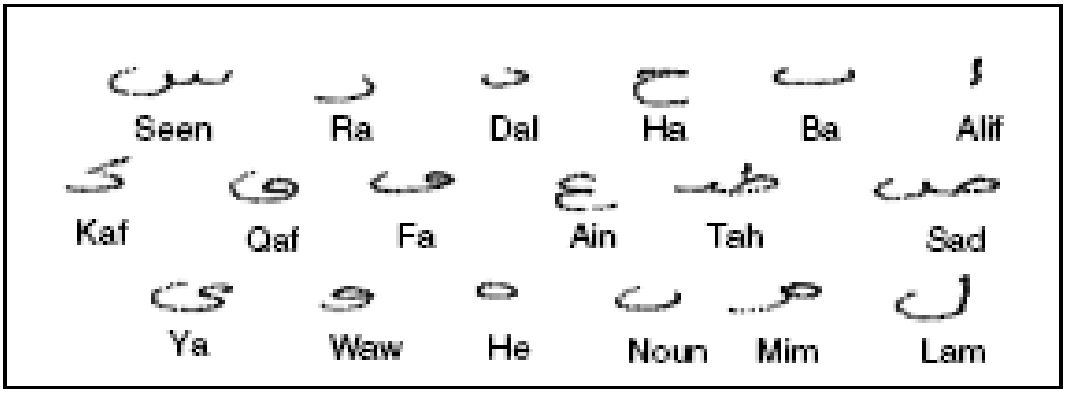


FIG. 3.19 – Alphabet de formes de lettres sans les points pour un système de reconnaissance on-line (figure extraite de [119]).

3.2.2.3 Segmentation et alphabets de symboles

3.2.2.3.1 Segmentation en caractères Comme l'indiquent A. Belaïd et Ch. Choisy dans [20], il est admis depuis longtemps dans la communauté de la reconnaissance automatique de l'écriture manuscrite (en particulier en écriture latine) qu'une segmentation idéale en lettres d'un texte cursif est un problème insoluble.

Dans [161], Sari et al proposent un système de segmentation en lettres sur du texte arabe manuscrit. Les points de segmentation potentiels se situent au niveau des minimums locaux de contours extérieurs bas. Ces points de segmentation sont soumis à un ensemble de règles afin de les valider ou au contraire de les rejeter.

Les auteurs ne traitent pas un certain nombre de problèmes :
- ligatures verticales
- caractères qui se touchent alors qu'ils ne devraient pas :
  - fin de mot (ou de pseudo mot) relié avec le mot (pseudo mot) suivant.
  - Doubles ligatures : lorsque deux caractères successifs ont plusieurs points de contact (normalement deux caractères successifs ne sont en contact qu'au niveau de la ligne de base).

Les auteurs parlent également d'une validation de leur outil de segmentation en utilisant un reconnaisseur de caractères, mais cette idée reste à l'état de projet, qui n'a semble-t-il pas été réalisé par la suite.

Dans [119], Mezghani et al définissent un alphabet de formes élémentaires pour leur système de reconnaissance on-line de caractères isolés (voir figure 3.19). Ils retirent les signes diacritiques afin de factoriser les formes. Une idée assez proche de cette notion d'alphabet de symboles, mais cette fois-ci sur du texte manuscrit off-line sera présentée

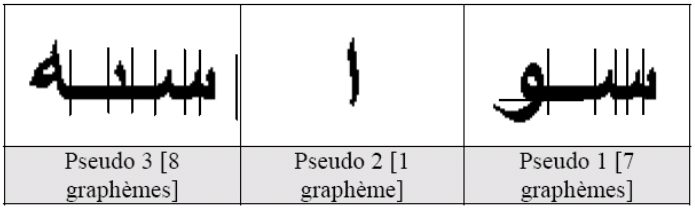


FIG. 3.20 – Segmentation en graphèmes sur du texte imprimé (figure extraite de [78]).

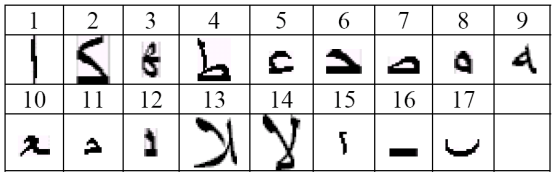


FIG. 3.21 – Alphabet de graphèmes (figure extraite de [78]).

dans la section 4.2.

3.2.2.3.2 Segmentation en graphèmes Dans [78], W. Kammoun et A. Ennaji proposent un système de reconnaissance de textes arabes imprimés. Ils segmentent le texte en mots en utilisant des projections verticales, en s'appuyant sur le fait qu'en imprimé l'espace séparant les mots est plus important que celui séparant les pseudo-mots qui appartiennent au même mot. Ils éliminent les signes diacritiques, mais mémorisent leurs coordonnées. Cette information qualitative de la présence ou non de signes diacritiques sera utilisée par la suite, sans pour autant procéder à leur reconnaissance (le système ne détermine pas s'il s'agit d'un point, deux points, trois points, un autre signe diacritique). Les auteurs procèdent à une segmentation graphèmes, avec des coupures à la fois verticales et horizontales (voir figure 3.20).

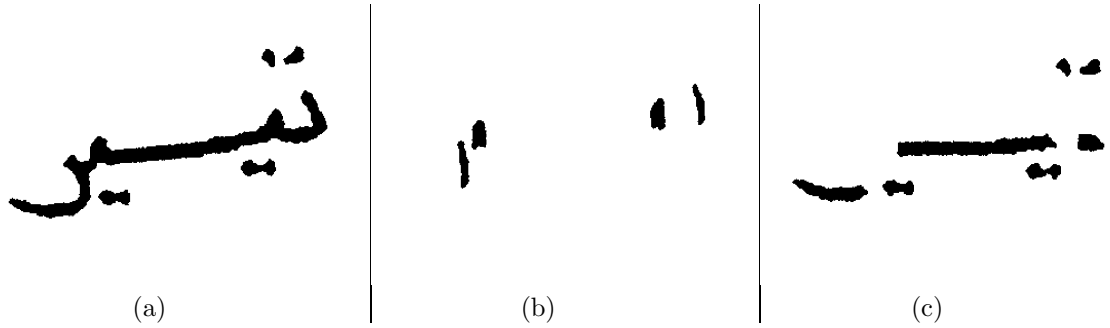
Les auteurs proposent un Alphabet de 17 graphèmes, dont ils se servent pour reconstruire les lettres (voir figure 3.21).

A partir de la reconnaissance d'une séquence de graphèmes, et compte tenu de l'information qualitative portée par les signes diacritiques, le système génère tous les candidats possibles, et sélectionne le meilleur (voir figure 3.22).

Bien que décrivant un système de reconnaissance de l'écriture imprimée, ce système comporte un certain nombre d'idées utiles dans le système de reconnaissance de l'écriture manuscrite que nous décrivons dans le chapitre 4.

تثقايلون	تتقايلون	تنقايلون	تثفايلون	تنفايلون	تثقابلون	تثقابلون	تثفابلون	تنفابلون
تثقايلون	تتقايلون	تثفايلون	تتفايلون	تنفايلون	تثقابلون	تتقابلون	تثفابلون	تنفابلون
تنقايلون	تثقايلون	تثفايلون	تتفايلون	تثقابلون	تتقابلون	تنقابلون	تثفابلون	تنفابلون
تثقايلون	تتقايلون	تنفايلون	تثفايلون	تثقابلون	تنقابلون	تثفابلون	تتفابلون	ننفابلون

FIG. 3.22 – Tous les candidats possibles pour le mot : تتقابلون (figure extraite de [78])



(a) (b) (c)

FIG. 3.23 – D. Motawa [128] : Régularités / Singularités. (a) Image initiale. (b) Une ouverture sur l'image permet d'obtenir les singularités. (c) En soustrayant l'image des singularités à l'image originale, on obtient les régularités.

Dans [128], Motawa et al utilisent des techniques de morphologie mathématique pour faire de la segmentation en graphèmes sur des mots manuscrits. Leur algorithme est basé sur le principe des régularités/singularités [170, 169].

Par exemple sur l'image donnée à la figure 3.23-a, en effectuant une ouverture sur l'image, D. Motawa et al obtiennent les singularités (voir figure 3.23-b). Les auteurs ne précisent pas la forme de l'élément structurant utilisé. En soustrayant ces singularités à l'image, ils obtiennent les régularités (voir figure 3.23-c). Ces régularités correspondent aux zones qui contiennent les points candidats de coupure de la segmentation en graphèmes. Bien qu'apparemment prometteurs, ces travaux n'ont pas donné lieu à d'autres articles.

Dans [140], C. Olivier et al proposent une méthode pour segmenter un mot en graphèmes. Leur technique s'appuie sur la recherche des minimums locaux des contours supérieurs des mots. Ces minimums locaux définissent des zones d'intérêt. Les points de segmentation sont choisis dans ces zones d'intérêt, en fonction d'un certain nombre de règles :

– Si un point candidat est au-dessus d'une boucle, il est éliminé
– L'épaisseur du tracé à l'endroit du point candidat doit être inférieure à un seuil
– Si plusieurs candidats sont voisins, on choisit le plus proche de la ligne de base.

Dans [124, 120], Miled et al utilisent la segmentation graphème ci-dessus. Ils définissent

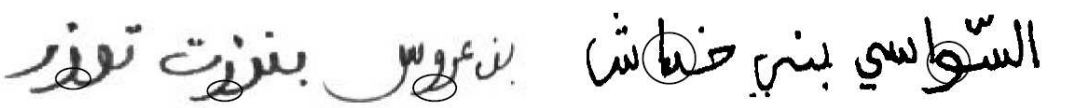
Latin Code	main body of the characters	Arabic Characters	Latin Code	main body of the characters	Arabic Characters	Latin Code	main body of the characters	Arabic Characters
a	ا	ا	h	ط	ظ ط	o	هـ	هـ
b	ب	ب ت ث	i	ع	غ ع	p	و	و
c	ح	ح ج خ	j	ف	ف	q	ى	ى ي
d	د	د ذ	k	ك	ك	r	ء	ء
e	ر	ر ز	l	ل	ل	s	ق	ق
f	س	س ش	m	م	م			
g	ص	ص ض	n	ن	ن			

FIG. 3.24 – Alphabet de 19 symboles pour représenter les 29 lettres arabes (figure extraite de [124]).

un alphabet de symboles qui leur permet d'éviter de prendre en compte les signes diacritiques (voir figure 3.24).

3.2.2.3.3 Segmentation en pseudo-mots Dans [122, 123], Miled et al travaillent au niveau pseudo mot. Ils décrivent un certain nombre de problèmes liés à la segmentation en pseudo mots :

- causes de sous segmentations en pseudo mots (voir figure 3.25) :
  - succession de caractères avec jambes (prolongements sous la bande de base) qui peuvent se toucher. Cette situation fait apparaître un minimum local du contour supérieur en dessous de la bande de base. Attention toutefois à ne pas le confondre avec un caractère valide (ن ou ي en fin de mot).
  - surcharge de pixels noirs dans la zone médiane : lorsque l'un des 6 caractères qui ne devraient pas être reliés avec son successeur touche le caractère suivant (lettres trop rapprochées). Aucune réponse n'est apportée à ce problème.
- sur segmentations en pseudo mots (voir figure 3.26) :
  - problèmes de numérisation, une composante connexe est découpée en deux.
  - levée de plume : le début de certaines lettres s'écrit de gauche à droite (lettres ه,



(a) succession de caractères à extension basse (b) causes accidentelles de sous-segmentation

FIG. 3.25 – Exemples de sous-segmentations en pseudo-mots (figure extraite de [123]).

(a) phénomène de la levée de plume (b) artefacts de l'acquisition

FIG. 3.26 – Exemples de sur-segmentations en pseudo-mots (figure extraite de [123]).

ص, ط, ح, ...), ce qui oblige le scripteur à lever son stylo. Lorsque la lettre qui suit un levé de plume est "trop courte", elle n'est pas liée à la lettre précédente et fait apparaître dans le tracé une coupure indésirable. Ce phénomène crée donc une composante connexe supplémentaire.

Le système développé par A. AbdulKader [2] s'appuie également sur une segmentation en pseudo-mots. L'auteur ne détaille pas la segmentation en pseudo-mots pour les cas difficiles. La stratégie de reconnaissance utilisée dans ce système est présentée dans le paragraphe 3.2.2.4.6

3.2.2.3.4 Segmentation en mots Dans [20], A. Belaïd et al. insistent sur la difficulté de la segmentation en mots en arabe, et déplorent la trop faible quantité de travaux de recherche qui vont dans cette direction.

A notre connaissance, les seuls travaux qui traitent cette tâche (segmentation en mots dans une page de texte manuscrit) sont ceux de Srihari et al [160] [159]. S. Srihari et al mettent au point un système qui utilise un réseau de neurones pour déterminer les points de coupures d'une segmentation en mots. Les auteurs extraient les composantes connexes qui correspondent aux pseudo-mots, et considèrent que tous les espaces entre deux pseudo-mots successifs sont des candidats de séparation inter-mots. Un vecteur de 9 primitives qui dépendent du pseudo-mot précédent et du pseudo-mot suivant est ainsi extrait pour qualifier chaque espace. Un réseau de neurones est entraîné pour classer ces candidats en deux classes : séparateur de mots ou non. Les auteurs remarquent que la présence de alif donne une information importante pour guider une segmentation en mots (beaucoup de mots commencent par la lettre alif). Sur l'ensemble des 10 scripteurs

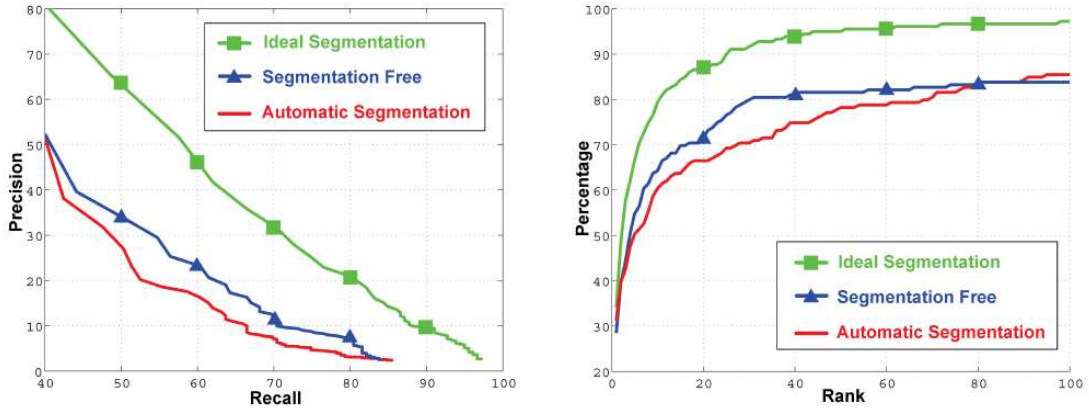


FIG. 3.27 – Courbes de Rappel/Précision. Extrait de [18]

de la base, le système segmente correctement 60% des mots. Ces performances semblent relativement faibles. Ces travaux sont à notre connaissance les seuls qui s'attaquent au problème de la segmentation en mots dans une page de texte manuscrit arabe. Cette tâche est plus complexe que la segmentation en mots dans une page de texte cursive en alphabet latin. En effet, le découpage en pseudo-mots multiplie les candidats, et complexifie considérablement la tâche de segmentation en mots.

Les auteurs soulignent l'importance des alif dans la segmentation en mots. Mais nous insistons également sur le fait que les jambages (voir paragraphe 4.2.1.1) semblent également particulièrement importants. Ces jambages, que l'on peut regrouper sous forme de trois classes, sont discriminants : la présence de l'un d'eux indique la fin d'un mot. Il convient sans doute d'exploiter cette particularité de la langue arabe pour améliorer cette segmentation en mots. Une piste de réflexion est donnée dans la section 4.4.2, dans laquelle nous présentons un automate des pseudo-mots bien formés. Cet automate a deux états terminaux. L'un de ces deux états signifie nécessairement une fin de mot.

3.2.2.3.5 Segmentation en bandes verticales Plusieurs travaux utilisent une segmentation implicite par fenêtre glissante et s'appuient sur un reconnaisseur à base de HMM. Les systèmes de reconnaissance qui obtiennent les meilleurs taux de reconnaissance sur le manuscrit arabe sont basés sur cette architecture. En particulier les travaux de El Hajj et al [57] [58] [4] qui utilisent une fenêtre glissante de 4 ou 8 pixels de large ; et de Pechwitz et al [150] [148] [131] qui utilisent une fenêtre glissante de 3 pixels de large. Ces systèmes seront détaillés dans la section 3.2.2.4.4.

G. R. Ball et al. [18] indiquent qu'une segmentation par fenêtre glissante donne de meilleures performances que leur segmentation en graphèmes. En revanche, ils montrent également qu'une segmentation idéale donnerait de meilleurs résultats que la segmentation par fenêtre glissante (voir figure 3.27). Ce résultat suggère qu'une segmentation graphèmes de meilleure qualité pourrait offrir de meilleures performances que l'approche par fenêtre glissante

FIG. 3.28 – Segmentation en bandes uniformes et non-uniformes (figure extraite de [24])

Dans [23, 24], A. Benouareth et al. mettent en concurrence deux stratégies de segmentation en bandes verticales (voir figure 3.28). La première est uniforme (toutes les bandes ont la même largeur) : les auteurs fixent empiriquement la largeur d'une bande à 10 pixels. La deuxième stratégie de segmentation est non uniforme : le système découpe l'image en bandes verticales dont la largeur varie. Ce découpage s'appuie sur une analyse de l'histogramme de projection vertical. Mais les auteurs ne donnent pas précisément la procédure qu'ils utilisent pour segmenter l'image en bandes à partir de l'histogramme. Leur système de reconnaissance est présenté plus en détails dans le paragraphe 3.2.2.4.4.

3.2.2.3.6 Segmentation en lignes Zahour et al proposent une approche de segmentation en lignes sur du texte ancien arabe [200]. L'approche consiste à découper l'image en bandes de largeurs uniformes. Dans chacune des bandes, un histogramme de projection sur l'axe des ordonnées est calculé. Les petits blocs de texte correspondent généralement aux signes diacritiques. Les blocs de taille moyenne correspondent aux corps de lettres. Les gros blocs correspondent aux lignes qui sont fusionnées du fait que des ascendants/descendants se touchent d'une ligne à l'autre (voir figure 3.29). Ces gros blocs seront découpés de manière à trouver une segmentation en lignes acceptable (voir figure 3.30).

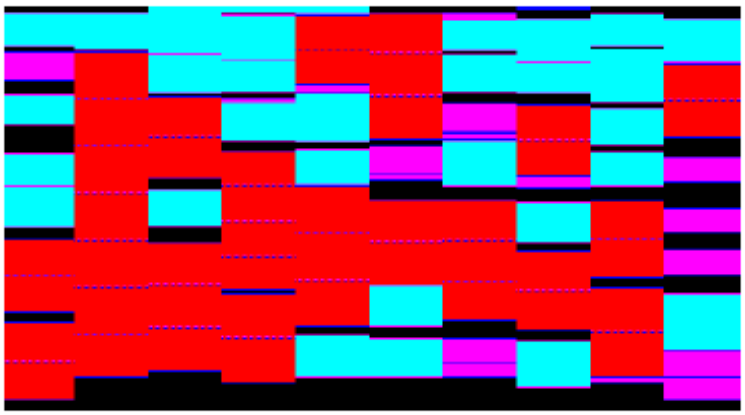


FIG. 3.29 – Blocs de texte. Rouge : régions à segmenter. Bleu : zones de texte. Violet : zones attribuées aux signes diacritiques. [200]

FIG. 3.30 – Résultat de la segmentation en lignes. [200]

3.2.2.4 Reconnaissance

3.2.2.4.1 Globale Dans [114], S. Masmoudi et H. Amiri mettent au point un système de reconnaissance globale. Ils entraînent un modèle HMM par mot.

Pour cela, ils extraient des primitives globales dans le mot, qui sont basées sur les :

- occlusions
- traits saillants
- espaces vides
- diacritiques

Ils obtiennent 96% de reconnaissance sur 10 classes. Mais ce type d'approche devient problématique lorsque la taille du vocabulaire augmente.

Les techniques de reconnaissance globales au niveau mot ont également été appliquées à la reconnaissance de l'écriture arabe imprimée.

Par exemple, dans [86] [87], M. S. Khorsheed et W. F. Clocksin travaillent sur des images de mots arabes cursifs imprimés dans plusieurs polices de caractères, sur lesquelles ils extraient des primitives globales sans segmentation du mot. Ils commencent par exprimer l'image en coordonnées polaires normalisées, puis appliquent une transformée de Fourier 2D sur cette nouvelle image. Ils extraient ainsi un jeu de primitives invariantes en taille, translation et rotation. La classification se fait par mesure d'une distance euclidienne par rapport à un modèle.

Dans [13], Adnan Amin utilise des primitives globales et des arbres de décision appliqués à des mots imprimés. L'application utilise un vocabulaire de 1000 mots et différentes polices de caractères. Les primitives globales extraites sont au nombre de 7 : nombre de pseudo-mots, nombre de pics de l'histogramme de projection horizontale pour chacun des pseudo-mots, nombre de boucles, nombre et position des signes diacritiques, hauteur et largeur de chaque pic de l'histogramme de projection horizontale.

L'auteur obtient des résultats qui semblent intéressants sur de l'écriture imprimée. Mais le recours à une telle technique (approche holistique avec un vocabulaire de 1000 mots) ne pourrait pas donner de bons résultats sur du manuscrit.

3.2.2.4.2 En-ligne Relativement peu de travaux ont été menés en reconnaissance automatique de l'écriture arabe manuscrite en-ligne.

Dans [10], A. Amin et al divisent les caractères arabes en deux catégories : les caractères stables et les caractères instables. Ils adoptent une représentation syntaxique particulière pour chaque catégorie. La classification est basée sur l'appariement des chaînes.

Dans [3], S. Al-Emami et M. Usher divisent les caractères en un groupe indépendant du scripteur et un autre dépendant de celui-ci. Les vecteurs de caractéristiques sont formés par les codes de Freeman. Une structure d'arbre est utilisée pour la classification.

Dans [8], A. M. Alimi présente un système de reconnaissance basé sur une approche neuro-floue, dans laquelle un réseau de neurones est entraîné à l'aide d'un algorithme génétique.

Plus récemment, dans [119, 133, 118], N. Mezghani et al proposent un système de reconnaissance en-ligne de caractères arabes isolés. Leur système de reconnaissance s'appuie sur des réseaux de Kohonen.

3.2.2.4.3 Caractères Dans [181, 182], S. Touj et al utilisent une transformée de Hough généralisée (THG). Un tableau de référence nommé R-table définit la correspondance entre l'espace de définition de la forme recherchée (l'espace image), et l'espace paramétrique. La THG peut être utilisée non seulement pour reconnaître les caractères, mais aussi pour déterminer leurs positions dans l'image grâce à la localisation de leurs points de référence. L'idée est de réaliser une reconnaissance de l'écriture arabe imprimée sans avoir recours à une étape de segmentation (en caractères ou en graphèmes).

Images de caractères isolés → THG → R-Tables.

En phase de reconnaissance on fournit l'image d'un pseudo mot à reconnaître et la liste des R-Tables de référence. Le système renvoie la liste des caractères reconnus et leurs positions dans le pseudo mot. Leur système commet des erreurs de reconnaissance dues au fait que certaines formes de caractères sont totalement incluses dans d'autres caractères. Dans certains cas, il arrive que la sous-partie de caractère répondre à la place du caractère correct, ce qui peut entraîner des erreurs de reconnaissance.

A. Amin mène plusieurs travaux sur la reconnaissance de caractères manuscrits arabes isolés [12]. Dans [16], donne le détail des primitives structurelles utilisées (lignes, courbes, boucles, ...), et utilise un réseau de neurones.

Dans [14], il travaille sur une base de 6000 caractères imprimés. Le classifieur est un arbre de décisions de type C4.5.

Dans [15], il utilise un système à base de règles pour reconstruire les lettres à partir de primitives géométriques simples (lignes, courbes, boucles).

Dans [68], N. Hassan propose une segmentation en lettres basée sur les histogrammes de projections verticales. Puis chaque lettre est à son tour décomposée en trois parties : ascendant, zone médiane, et descendant. Ce système extrait des primitives sur chacune de ces trois zones. L'auteur introduit une grammaire non-contextuelle qui permet de recomposer les lettres à partir des primitives qui les composent.

Il ne donne pas de performances sur une base de référence, et n'indique pas les limitations du système, qui semble pourtant très dépendant d'une bonne segmentation en lettres.

3.2.2.4.4 Mots à l'aide de HMM 1-D *Graphèmes*

Dans [84], Khorsheed propose une approche à base de Modèles de Markov Cachés. Il utilise la méthode d'extraction du squelette proposée par T. Y. Zhang et C. Y. Suen [201], squelette qu'il simplifie en fusionnant certains points singuliers [103]. Les boucles sont extraites à partir de ce squelette simplifié, puis les tracés restants sont linéarisés (segments) et traduits en symboles à l'aide d'un codebook. Deux primitives sont extraites sur chacun de ces symboles : sa longueur et son angle.

Des modèles de HMM de lettres sont entraînés pour décoder ces séquences de vecteurs de primitives. L'apprentissage est réalisé à l'aide de la méthode de Baum-Welch. Les modèles de mots sont obtenus par concaténation de modèles de lettres.

Dans [120], H. Miled travaille au niveau des pseudo mots, supprime les diacritiques pour factoriser les formes équivalentes aux points près, regroupe les pseudo mots qui appartiennent à des mots différents (permet de disposer de plus d'exemples pour entraîner les modèles HMM de pseudo mots, segmente un pseudo mot en graphèmes et extrait des primitives sur ces graphèmes. Les modèles HMM de mots sont construits comme une concaténation des modèles de pseudo mots dont on estime les probabilités de transition. On définit un symbole de transition entre pseudo mots. Cette modélisation, appelée "modélisation pseudo analytique" améliore les performances par rapport à une méthode

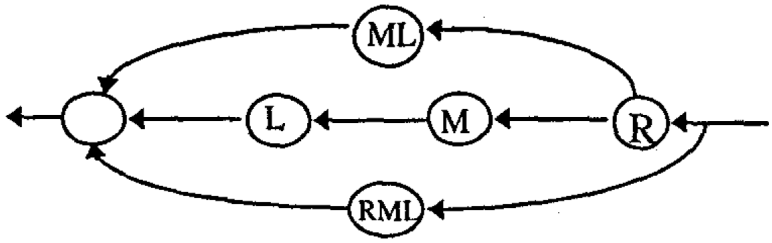


FIG. 3.31 – Modèle de lettre (figure extraite de [124]).

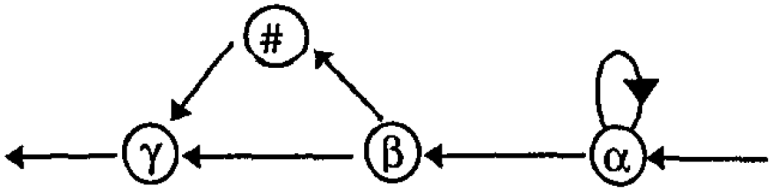


FIG. 3.32 – Transitions inter-caractères (figure extraite de [124]). 'α' , 'β' , 'γ' sont des lettres. '#' correspond à un espace inter pseudo-mots.

classique de modélisation du mot complet sans passer par le niveau pseudo mot (67.8% $\rightarrow$ 72.5%).

Dans [124], H. Miled et al utilisent une segmentation en graphèmes. Un vecteur de 19 primitives est extrait sur chaque graphème. Il s'appuie sur :

- des primitives topologiques : boucles, ouvertures, tailles relatives (hauteur, largeur), position relative à la bande de base, ...
- les 10 premiers descripteurs de Fourier

Ces primitives sont normalisées entre -1 et 1.

La séquence de graphèmes et exprimée sous forme d'une séquence de vecteurs de primitives. Un KPPV permet d'attribuer une classe d'observation à chaque vecteur de primitives. Le KPPV permet donc de transformer la séquence de vecteurs de primitives en une séquence d'observations dans un alphabet de classes d'observations prédéfini. Expérimentalement, les auteurs déterminent que 34 classes d'observation donnent le meilleur résultat.

Des modèles de HMM de lettres (voir figure 3.31) sont entraînés pour décoder les séquences d'observations.

Les auteurs modélisent explicitement les pseudo-mots en rajoutant un modèle HMM qui émet un caractère transparent (voir figure 3.32).

Les performances obtenues sont de 82.5% de reconnaissance sur un lexique de 232 noms de villes tunisiennes.

Fenêtre glissante

Dans [65], A. M. Gouda et M. A. Rashwan utilisent des HMMs à fenêtre glissante pour reconnaître (et segmenter a posteriori) l'écriture imprimée. Les modèles de mots sont construits par concaténation des modèles de lettres. Les auteurs se servent du résultat de la reconnaissance pour déduire les meilleurs points de segmentation en lettres : la transition entre la fin d'un HMM et le début du suivant marque un point de segmentation. Les auteurs utilisent une fenêtre glissante d'un pixel de large qu'ils déplacent latéralement de la droite vers la gauche (dans le sens de la lecture). Des primitives sont extraites sur chacune de ces bandes à l'aide des moments invariants. Une Quantification Vectorielle permet de transformer la séquence de vecteurs de primitives en une séquence d'observations. La segmentation en lettres est déduite à partir de la reconnaissance.

Dans [150], M. Pechwitz et V. Maergner déplacent une fenêtre glissante de trois pixels de large sur une image normalisée (voir figure 3.16). Cette normalisation est décrite dans le paragraphe 3.2.2.2.2.

Chacune de ces trois bandes donne lieu à un vecteur composé des valeurs en niveaux de gris des pixels. Ces trois vecteurs sont concaténés pour en former un seul. Les auteurs appliquent une transformation de Loeve-Karhunen sur les valeurs afin de réduire la dimension des primitives.

Les auteurs utilisent des Modèles de Markov Cachés semi-continus pour modéliser les lettres. Chaque modèle de caractère contient 7 états. Chacun de ces états a 3 transitions possibles :

- Une boucle
- Une transition vers l'état suivant
- Une transition qui saute l'état suivant (cette transition ne permet de sauter qu'un seul état).

Les modèles de mots sont obtenus par concaténation de modèles de lettres.

Ce système donne de bonnes performances : 89.5% de reconnaissance sur la base IFN/ENIT.

R. Al-Hajj et al. [57, 58, 4] proposent un système de reconnaissance à base de HMM à fenêtre glissante. Les primitives sont extraites dans une bande verticale de 8 pixels de large. Cette bande verticale est découpée en cellules homogènes. Le détail des 26 primitives utilisées est donné dans [58]. Ces primitives dépendent de la bande de base. Certaines d'entre elles s'appuient sur :

- des caractéristiques de densité : nombre de transitions, position relative des centres de gravité dans deux fenêtres consécutives, position normalisée du centre de gravité par rapport à la bande de base, densité des pixels au-dessus et en dessous de la

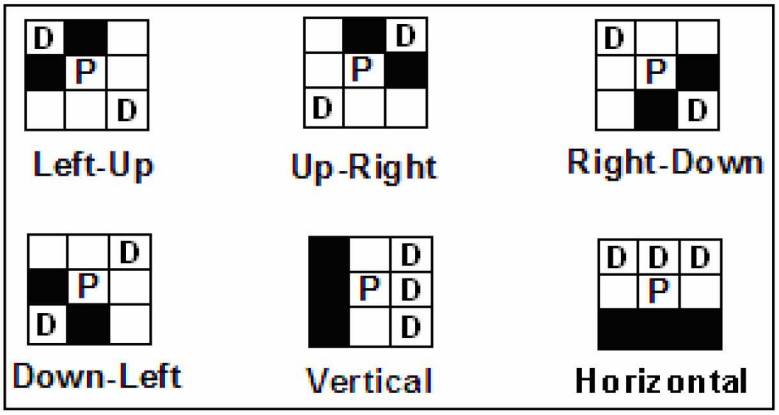


FIG. 3.33 – Six types de configurations locales pour un pixel de fond P. Les pixels marqués D peuvent être indifféremment noirs ou blancs (figure extraite de [58]).

bande de base, nombre de transitions noir/blanc, zone à laquelle appartient le centre de gravité (soit 16 primitives)
- des caractéristiques de configurations locales : recherche de motifs particuliers (voir figure 3.33) dans l'ensemble de la fenêtre, et également spécifiquement dans la bande de base (12 primitives).

Mais la particularité de ce système vient du fait qu'il comporte 3 types de fenêtres (voir figure 3.34) :
- une fenêtre verticale
- une fenêtre inclinée d'un angle $+alpha$
- une fenêtre inclinée d'un angle $-alpha$

Ces 3 fenêtres permettent d'entraîner 3 classifieurs, dont les réponses sont combinées pour calculer la réponse du système global. Cette approche présente l'avantage d'éviter d'avoir à estimer et à corriger l'inclinaison de l'écriture. Elle permet également une plus grande tolérance quant à l'alignement des signes diacritiques sur la lettre à laquelle ils sont associés (voir figure 3.34).

La combinaison des 3 classifieurs permet ainsi d'obtenir des performances meilleures que chacun des trois pris séparément.

Pour réaliser la fusion des résultats, trois approches sont proposées à partir des listes de résultats : une somme, un vote, ou l'utilisation d'un réseau de neurones de type MLP qui renvoie le classifieur à sélectionner à partir des scores des trois listes de reconnaissance.

Durée d'état explicite

Dans [23, 24], Benouareth et al. commencent par effectuer une segmentation en bandes verticales (découpées uniformément ou non, voir paragraphe 3.2.2.3.5), puis extraient un

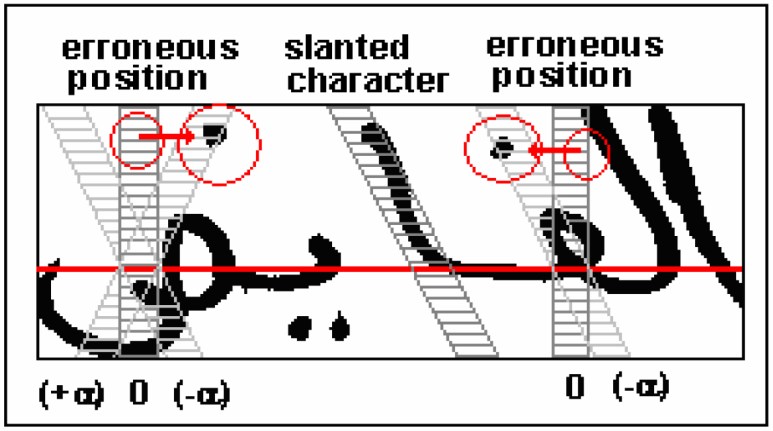


FIG. 3.34 – Les fenêtres inclinées peuvent prendre en compte l'inclinaison des caractères et les décalages dans la position des points (figure extraite de [58]).

vecteur de 33 caractéristiques pour chaque bande. Chaque bande est divisée en cellules de 4 pixels de hauteur. Les primitives sont basées sur :

- les densités de pixels noirs
- boucles
- les points singuliers
  - points extrêmes et points de jonctions
  - les points d'inflexion : changement de courbure
  - les points de rebroussement
  - les points diacritiques
- ...

Certaines de ces caractéristiques dépendent de la bande de base. Les modèles de mots sont obtenus par concaténation de modèles de lettres.

Benaroueth et al. comparent les résultats obtenus en modélisant explicitement la durée d'état dans les HMMs avec plusieurs lois de distribution. Par ces expériences, ils montrent d'une part que la segmentation non-uniforme donne de meilleurs résultats que la segmentation en bandes uniformes. Et d'autre part, ils montrent également que l'utilisation de DHMM (HMM à durée d'état explicite) se révèle plus performante que l'utilisation de HMM semi-continus standards.

Benouareth et al. réalisent leurs expériences sur la base IFN/ENIT, et obtiennent un taux de reconnaissance de 89,79% pour le meilleur système (loi Gamma sur segmentation en bandes non-uniformes). Leurs expériences suggèrent que la loi gamma se comporte mieux que la loi gaussienne et la loi de poisson pour modéliser la durée dans les états des HMM sur une tâche de reconnaissance de l'écriture arabe manuscrite.

3.2.2.4.5 Mots à l'aide de HMM planaires Les HMM planaires ont été introduits dans la section 2.4.3.2.

Appliqués à l'arabe, les HMM planaires ont été utilisés en reconnaissance de l'écriture imprimée [9, 136, 121], et en reconnaissance de l'écriture manuscrite [121, 157, 180].

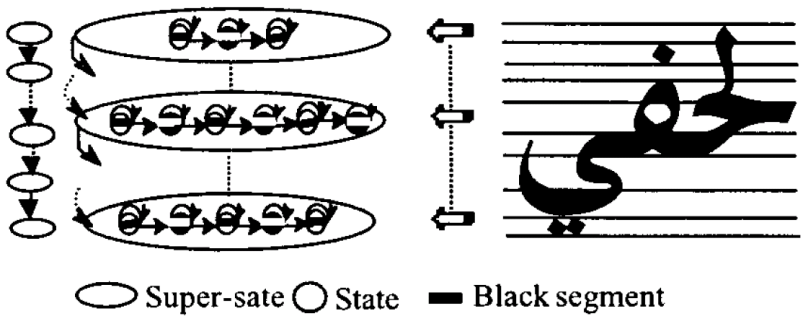


FIG. 3.35 – PHMM : Découpage en bandes. Un HMM-1D par bande horizontale, et un HMM vertical de 'super-états' (figure extraite de [121]).

Dans les deux cas, le mot est découpé en bandes verticales (voir figure 3.35). Un HMM-1D est associé à chacune des bandes horizontales. 1 HMM-1D vertical de 'super-états' gère les transitions verticales.

Dans le cas du manuscrit, on commence par détecter la bande médiane. On définit alors 2 autres bandes : la bande des ascendants, et la bande des descendants. On identifie les diacritiques (comme étant des composantes connexes plus petites au-dessus ou en dessous de la bande de base). On crée 2 autres bandes : une pour les diacritiques supérieurs, et une pour les diacritiques inférieurs. On décale ces diacritiques verticalement de façon à les sortir des zones ascendants/descendants. Au final on obtient donc 5 bandes, de haut en bas (voir figure 3.36) :

- diacritiques supérieurs
- ascendants
- zone médiane (bande de base)
- descendants
- diacritiques inférieurs

Les symboles de la zone médiane sont segmentés en graphèmes (voir figure 3.37).

Dans [180], sur un lexique de 25 classes :

- 73% de reconnaissance en se basant uniquement sur le contenu de la bande de base.
- 88.7% de reconnaissance lorsqu'on exploite également l'information contenue dans les bandes des diacritiques et des extensions.

Toutefois, les résultats de la compétition ICDAR 2005 sur la reconnaissance de l'écriture arabe manuscrite [113] montrent que sur des vocabulaires plus importants, les systèmes à base de PHMM appliqués à l'écriture arabe ne parviennent pour l'instant pas à atteindre des performances satisfaisantes.

3.2.2.4.6 Pseudo-Mots à l'aide de Réseau de neurones Dans [2], A. AbdulKader décompose le problème en deux parties :

- Un premier problème, qui consiste à reconnaître un pseudo-mot à partir des lettres qui le constituent.
- Un deuxième problème, qui consiste à reconnaître un mot à partir des pseudo-mots qui le constituent.

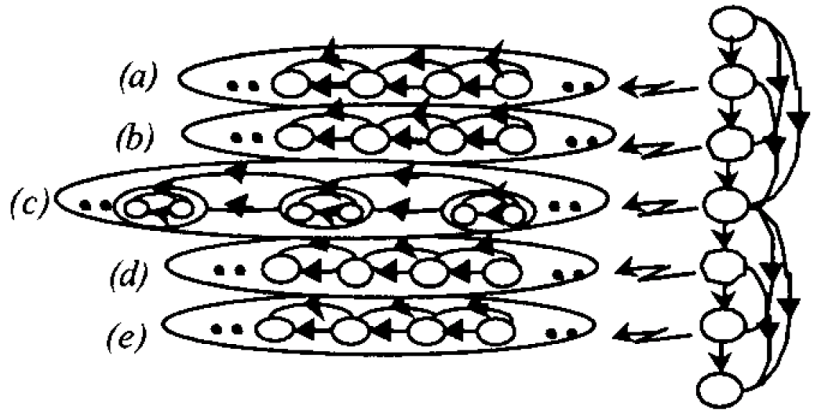


FIG. 3.36 – PHMM pour l'écriture arabe manuscrite (figure extraite de [121]) : (a) signes diacritiques supérieurs ; (b) ascendants ; (c) zone médiane ; (d) descendants ; (e) signes diacritiques inférieurs.

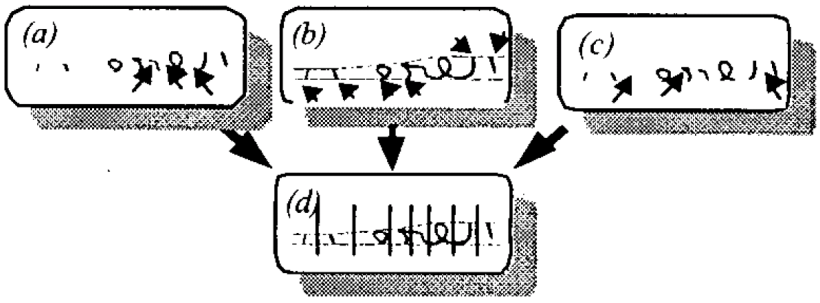


FIG. 3.37 – PHMM : segmentation graphèmes de la zone médiane (figure extraite de [121]).

Ces deux problèmes sont liés et sont résolus de concert.

A. AdbulKader exploite le fait que les pseudo-mots ont leur propre distribution de probabilités d'apparence. Même si le mot est mal écrit et que certains pseudo-mots prêtent à confusion, leur recomposition pour former des mots du lexique devrait permettre de lever les ambiguïtés.

Les composantes de pseudo-mots sont découpées en deux catégories : les primaires et les secondaires. Les primaires correspondent aux corps de lettres, tandis que les secondaires portent sur les diacritiques. Leurs relations sont définies en fonction des recouvrements horizontaux. Une composante secondaire ne peut exister seule. Un pseudo-mot se compose à la fois de sa composante primaire et de ses composantes secondaires (voir figure 3.38).

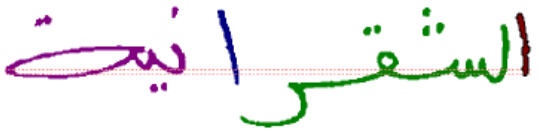


FIG. 3.38 – Segmentation en pseudo-mots correcte. Les composantes secondaires (signes diacritiques) sont rattachées à la composante primaire (corps de lettres) auxquelles elles se rapportent (figure extraite de [2]).

Le vocabulaire de 937 villes donne lieu à un vocabulaire de 762 pseudo-mots. Un réseau de neurones de type Réseau à Convolution à 762 sorties permet de classifier un pseudo-mot parmi cette liste de candidats de pseudo-mots. La taille des pseudo-mots peut varier selon un facteur de 1 à 8. Les pseudo-mots sont donc normalisés en taille de telle sorte qu'une image de pseudo-mot corresponde à la taille de la fenêtre d'entrée du réseau de neurones, tout en conservant son aspect. Il n'y a pas de notion de fenêtre glissante dans ce système.

Etant donnés les problèmes de liaisons basses ou autres pseudo-mots connectés (voir figure 3.40), et de levés de plume (voir figure 3.39), le système évalue plusieurs options de segmentation en pseudo-mots.

Pour retrouver les mots à partir des candidats de pseudo-mots, AbdulKader utilise un Beam search, qui permet de prendre en compte toutes les hypothèses de reconnaissance de pseudo-mots à partir des différentes options de segmentation, en contraignant la reconnaissance à l'aide d'un vocabulaire de mots exprimés dans un alphabet de pseudo-mots.

Malgré un taux d'erreur de reconnaissance de pseudo-mots relativement élevé de 44,86% d'erreur en première position, la combinaison des pseudo-mots contrainte par le vocabulaire de mots permet de réduire le taux d'erreur global à 11,06% d'erreur. Les sous-segmentations en pseudo-mots restent la principale source d'erreurs.

3.2.2.5 Traitement de la langue naturelle

3.2.2.5.1 Réduction du vocabulaire Dans [130], S. Mozzafari et al proposent une approche de réduction du vocabulaire, préliminaire à une reconnaissance plus précise.

En pratique, les auteurs extraient des primitives simples sur les signes diacritiques, qui sont regroupés à l'aide d'un certain nombre d'heuristiques. Ce travail préliminaire sur

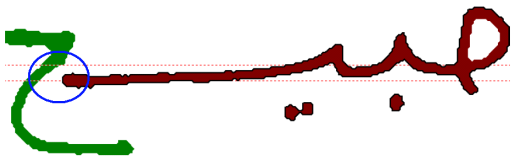


FIG. 3.39 – Segmentation en pseudo-mots : levé de plume (figure extraite de [2]).

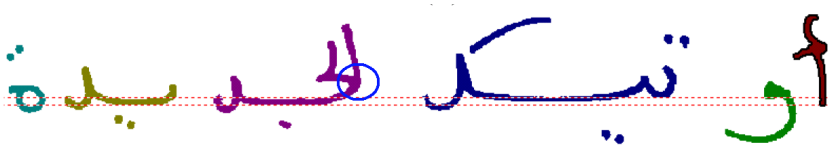


FIG. 3.40 – Segmentation en pseudo-mots : pseudo-mots connectés (figure extraite de [2]).

les signes diacritiques permet de restreindre la taille du vocabulaire considéré. La tâche de reconnaissance à proprement parler est effectuée à l'aide de ce vocabulaire restreint. Cette technique permet de réduire le temps de reconnaissance, au prix d'une diminution mineure du taux de reconnaissance.

3.2.2.5.2 Analyse Morphologique Dans [79], S. Kanoun et al utilisent une approche affixale, qui s'appuie également sur les contraintes morpho-syntaxiques de la langue arabe, pour un système de reconnaissance de l'écriture imprimée. Le système décompose le mot en préfixe, suffixe, infixe et radical et reconnaît ces différentes parties. En pratique, on commence par extraire les candidats de préfixe et de suffixes qui sont cohérents les uns avec les autres. Cette première sélection permet de déterminer la zone possible pour le radical, réduisant ainsi l'espace de recherche.

Dans [78], W. Kammoun et al étendent les travaux de S. Kanoun [80], et proposent sur un système de reconnaissance morpho-syntaxique de la langue arabe pour filtrer les hypothèses d'un système de reconnaissance de mots imprimés à vocabulaire ouvert. Le système de vérification linguistique est utilisé en phase de post-traitement, et permet des interactions avec l'utilisateur en cas d'échec de la reconnaissance [80].

I. A. Al-Sughaiyer et I. A. Al-Kharashi ont publié un survey des différentes techniques d'analyses morphologiques appliquées à l'arabe [7]. Ils déplorent le fait que les différents systèmes développés au sein de la communauté sont trop rarement comparables. Ils insistent sur la nécessité de standardiser les différentes tâches ayant trait à l'analyse morphologique, à développer et à diffuser des bases communes à différents systèmes, de manière à rendre les travaux comparables et ainsi permettre une progression du domaine.

3.2.2.6 Domaines d'application industriels

3.2.2.6.1 reconnaissance de montants littéraux Dans [172, 134], L. Souci et al proposent une approche par combinaison de classifieurs (réseau de neurones, KPPV et fuzzy-KPPV) pour la reconnaissance de montants littéraux, sur une base de 4800 mots collectée

FIG. 3.41 – Quelques échantillons de montants numériques de chèques Saoudiens (figure extraite de [132]).

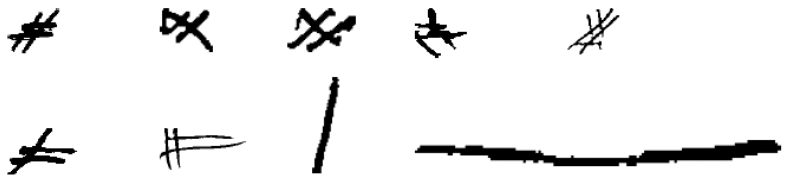


FIG. 3.42 – Quelques exemples de délimiteurs dans des montants numériques de chèques Saoudiens (figure extraite de [132]).

par les auteurs. Ils définissent également une grammaire qui permet de contraindre l'interprétation d'un champ.

Dans [132] N. E. Ayat et al présentent un système neuro-flou pour la reconnaissance de montants numériques sur des chèques arabes. Un réseau de neurones est utilisé pour la reconnaissance des chiffres indiens. Des primitives de type morphologique (boucles, courbures) et de type statistique (basées sur les chaînes de Freeman associées au contour). Un ensemble de règles qui s'appuient sur la logique floue permettent de localiser et d'extraire les délimiteurs d'un champ composé de chiffres (voir figure 3.41). Les délimiteurs sont des caractères supplémentaires avant et/ou après la séquence de chiffres indiens qui constitue le montant (voir figure 3.42 pour quelques exemples que peuvent prendre les délimiteurs).

Dans [203], M. Ziaratban et al proposent un système automatique de reconnaissance de chèques farsi. Des modèles spéciaux sont entraînés pour reconnaître les chiffres 0 et 1. Un autre reconnaisseur est utilisé pour l'ensemble des autres chiffres. Si la reconnaissance de chiffres obtient un score de confiance inférieur à un certain seuil, la reconnaissance sur le montant littéral est lancée également. Et une combinaison des deux résultats est réalisée.

3.2.2.6.2 OCR imprimé Cette partie traite de la reconnaissance de l'écriture arabe imprimée. Il n'existe pas à ce jour de système de reconnaissance de l'écriture arabe qui soit clé en main.

Dans [67], L. Hamami et D. Berkani proposent un système complet de reconnaissance de l'écriture arabe imprimée (multi-fontes).

Dans [81], T. Kanungo et al comparent deux produits OCR commerciaux de reconnaissance de l'écriture arabe imprimée : OmniPage et Sakhr. Sakhr obtient de meilleures performances (90% de reconnaissance, contre 86 % pour OmniPage).

Et enfin plus récemment dans [85], Khorsheed propose un système de reconnaissance multi-fontes à base de HMM à fenêtres glissantes, qui s'appuie sur la librairie de traitement de parole HTK. Une fenêtre glissante est déplacée de droite à gauche (avec recouvrement partiel d'une bande à l'autre). Cette fenêtre est divisée en cellules, et des primitives simples sont extraites sur ces cellules (densités, gradient vertical et gradient horizontal, ...). Les modèles de lettres sont concaténés pour former des modèles de mots. Khorsheed montre ainsi qu'il est possible de réaliser un système de reconnaissance de l'écriture imprimée arabe en s'appuyant sur des techniques standard de reconnaissance de l'écriture manuscrite.

Chapitre 4

Contributions à la reconnaissance de l'écriture arabe manuscrite

Nous avons mis au point un système de de reconnaissance de l'écriture arabe, que nous évaluons sur la base IFN/ENIT (voir section 3.2.1.4).

Dans la section 4.1, nous présenterons l'architecture générale du système mis en oeuvre. Dans la section 4.2, nous détaillerons un nouvel alphabet de formes, qui permet de mieux exploiter les redondances de formes des lettres arabes. Nous analyserons les ambiguités introduites par cet alphabet, et nous verrons que sur l'application considérée, il est possible de proposer un système de reconnaissance sans prendre en compte les signes diacritiques.

Dans la section 4.4, nous proposerons une stratégie qui permet de réintroduire les signes diacritiques dans le processus de reconnaissance.

Dans la section 4.5, nous présenterons un reconnaisseur de chiffres farsis (persans) isolés à l'aide de réseaux de neurones à convolutions. Nous verrons que ce type d'architecture, qui donne de très bons résultats sur les chiffres latins, est également tout à fait adaptée à une tâche de reconnaissance de chiffres arabes.

4.1 Description générale du système de reconnaissance utilisé

De la même manière que d'autres systèmes de reconnaissance de l'écriture arabe manuscrite ([57, 58, 4] ou [150, 131]), le système que nous utilisons est issu d'un système de reconnaissance de l'écriture cursive latine. Le schéma synoptique de notre système est donné à la figure 4.1.

L'hypothèse que nous faisons est celle de considérer que la reconnaissance de l'écriture arabe manuscrite est en soi une tâche qui n'est pas fondamentalement plus complexe que la reconnaissance de l'écriture latine manuscrite cursive. Néanmoins, les spécificités de l'écriture arabe présentent un certain nombre de difficultés supplémentaires (voir section 3.1.2), mais également un certain nombre de particularités dont il conviendra de tirer parti (voir section 4.2).

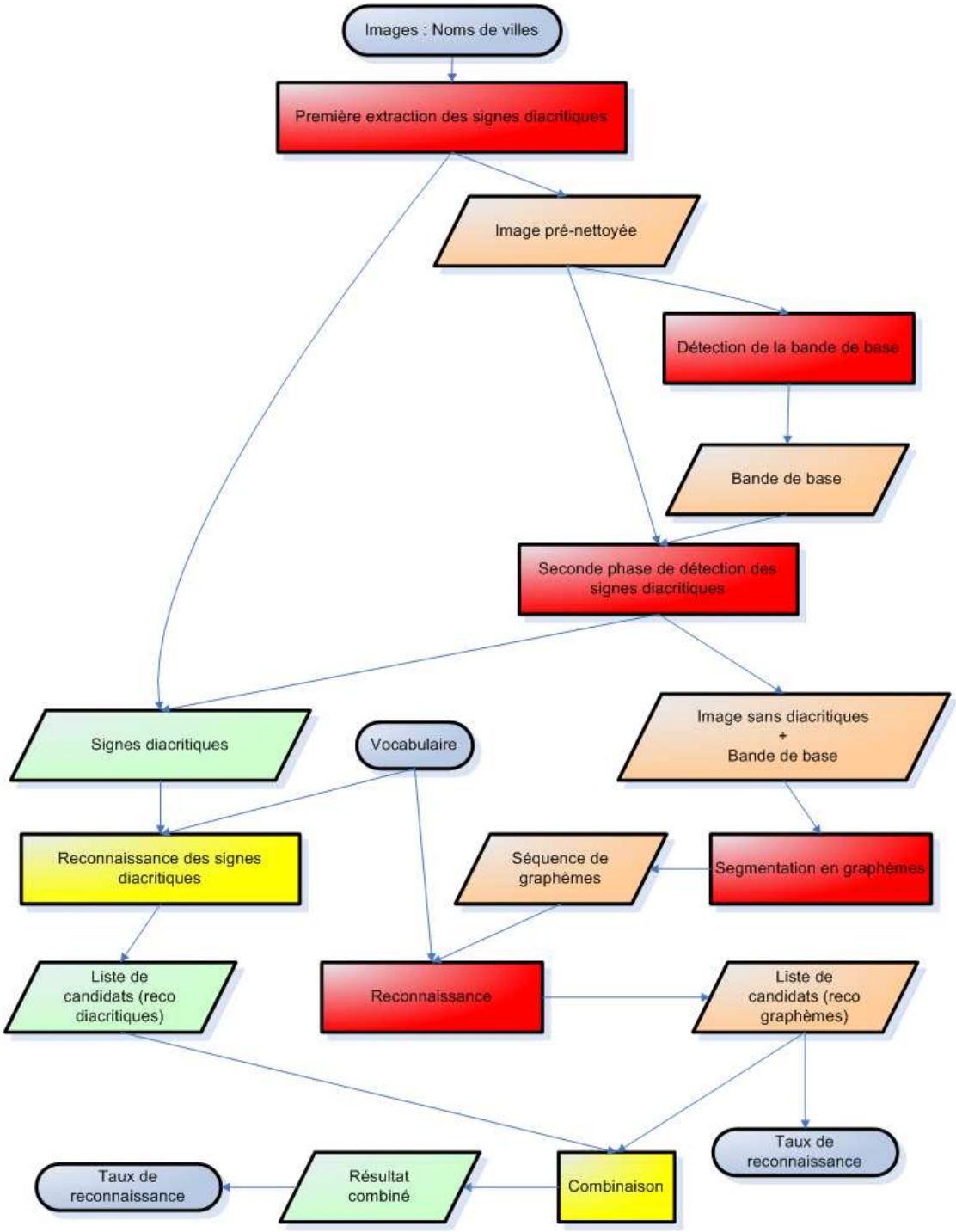


FIG. 4.1 – Système de reconnaissance de l'écriture arabe.
En rouge : étapes du système de reconnaissance sans diacritiques.
En orange : données intermédiaires du système de reconnaissance sans diacritiques.
En jaune : étapes supplémentaires pour la reconnaissance des diacritiques.
En vert : données intermédiaires utilisées pour la reconnaissance avec diacritiques.
En bleu : entrées et sorties du système.

Les éléments du système de reconnaissance appliqué à l'arabe sont donc relativement proches de ceux utilisés dans le cadre de la reconnaissance de l'écriture latine cursive. La segmentation en graphèmes est identique à celle développée pour l'écriture latine cursive. Nous verrons toutefois que cette segmentation est parfois source d'erreurs (voir section 4.1.3), et qu'il pourrait être intéressant de la modifier pour réduire la quantité de sous-segmentations. L'extraction de primitives est également la même que celle utilisée pour la reconnaissance de l'écriture latine cursive. Le jeu de primitives utilisé (voir section 2.3) est adapté pour la reconnaissance de l'écriture latine. Il semble assez naturel d'utiliser le même jeu de primitives pour la reconnaissance de l'écriture arabe.

En revanche, certaines adaptations sont indispensables pour mettre en oeuvre un système de reconnaissance de l'écriture arabe. L'extraction de la bande de base en particulier doit être adaptée. Notamment en raison de la présence de nombreux signes diacritiques, et également à cause de la forme des ascendants/descendants (voir section 3.1.2.3) qui, contrairement à l'usage dans l'écriture latine, peuvent être étendus horizontalement sous la bande de base. Le type de classifieur est également le même que celui utilisé dans le cadre de la reconnaissance latine : il s'agit d'un système hybride HMM/MLP [17, 55]. Le nombre de classes d'observations des HMM est déterminé empiriquement (cette partie est présentée en annexe B).

4.1.1 Détection des signes diacritiques

La détection des signes diacritiques a deux utilités. D'une part, un trop grand nombre de signes diacritiques risque de perturber l'histogramme de projection horizontale, qui sert a évaluer la bande de base (l'évaluation de la bande de base est détaillée dans la section 4.1.2). Et d'autre part, les signes diacritiques ainsi détectés pourront par la suite être utilisés pour améliorer les résultats de la reconnaissance. Ce deuxième point est abordé dans la section 4.4.

La détection des signes diacritiques se fait en deux étapes.

La première étape consiste à effectuer un filtrage des composantes connexes en s'appuyant sur des critères assez simples : taille des boîtes englobantes, aire, superposition verticale. L'objectif est de rejeter la plupart des signes diacritiques, sans rejeter de composante connexe correspondant à un corps de lettre ou à un pseudo-mot. La figure 4.2 décrit cette procédure de filtrage. Les seuils sont fixés empiriquement.

Le premier test se base sur l'aire : les composantes connexes trop grosses ne peuvent pas être des signes diacritiques.

Les deuxième et troisième tests permettent de conserver les barres de alif (ا), qui contiennent peu de pixels et sont étendues verticalement.

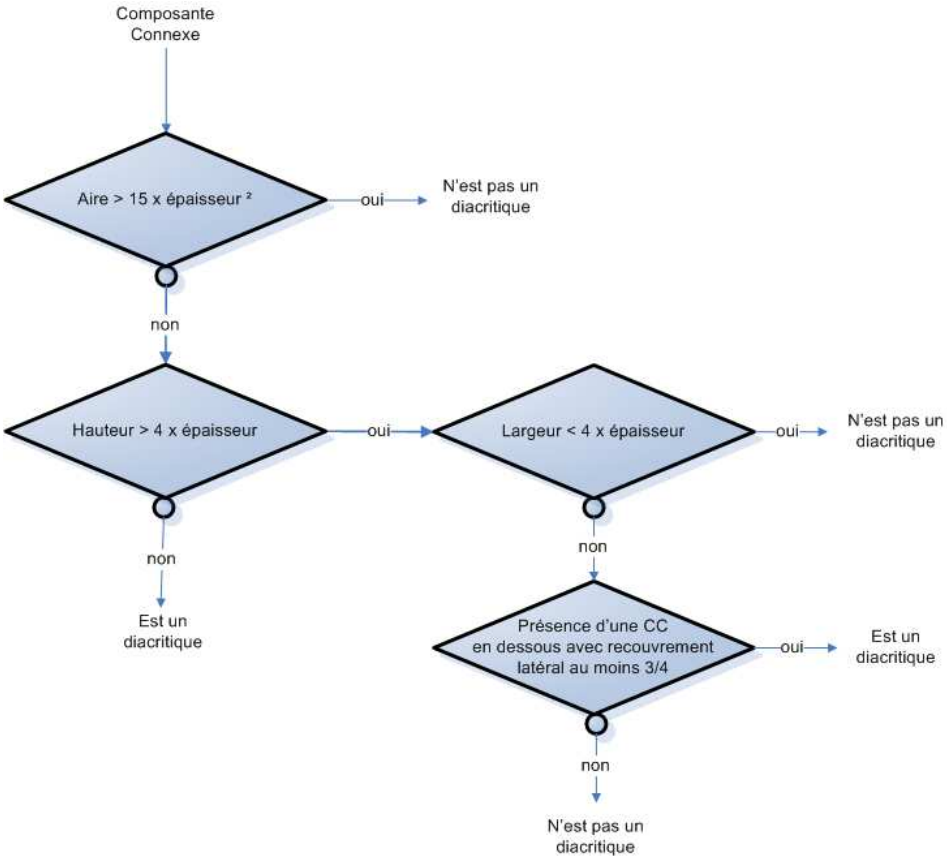


FIG. 4.2 – Procédure de filtrage des signes diacritiques : des seuils heuristiques définis à partir de l'épaisseur du tracé permettent d'effectuer un premier filtrage grossier.

Le 4ème test permet de conserver les caractères د isolés, qui sont parfois facilement assimilables à des signes diacritiques par leur forme. Cette règle repose sur la position relative de la composante connexe et de ses voisines : si une composante connexe C_1 d'aire réduite est située au dessus d'une autre composante connexe C_2, et que C_2 recouvre verticalement C_1 à plus de 75%, alors C_1 est un signe diacritique.

Ce premier filtrage permet de détecter et d'écarter un certain nombre de composantes connexes. On peut alors réaliser une approximation de la bande de base sur l'image ainsi filtrée (voir section 4.1.2). Cette estimation de la bande de base permet d'effectuer un second filtrage plus précis.

Ce second filtrage prend en compte le fait que les signes diacritiques se situent soit en dessous, soit au-dessus de la bande de base.

Par exemple, dans la figure 4.3, une chadda n'a pas été filtrée lors de la première étape. Elle n'est pas supprimée car son recouvrement vertical n'est que partiel (inférieur

au seuil fixé empiriquement) avec la lettre ر à laquelle elle est associée. Cette composante connexe a donc été conservée lors de la première étape. Mais après l'évaluation de la bande de base, il apparaît clairement qu'il s'agit bien d'un signe diacritique, puisque cette composante connexe est entièrement au dessus de la bande de base. On peut donc l'ajouter à la liste des signes diacritiques détectés lors de la première étape.

FIG. 4.3 – Pic maximal de l'histogramme (ligne jaune) et seuils pour déterminer une approximation de la ligne haute et de la ligne basse. On se sert de cette approximation de la bande de base pour retirer les signes diacritiques résiduels (ici une chadda entourée en bleu).

Dans certains cas, il arrive que les signes diacritiques ne forment pas des composantes connexes séparées du corps du texte. Un exemple est donné figure 4.4. Ces cas nécessitent la mise en oeuvre de mécanismes plus complexes, qui permettraient de générer plusieurs alternatives. Ce type de problèmes n'est pas traité dans cette thèse.

(a) (b)

FIG. 4.4 – Présence d'un signe diacritique collé au corps de texte (chadda). Ces cas de figure ne sont pas traités.
(a) image initiale.
(b) image obtenue après application de l'algorithme de suppression des signes diacritiques.

4.1.2 Extraction de la bande de base

On utilise un histogramme de projection horizontale pour déterminer une première approximation de la bande de base. Pour éviter que cet histogramme ne soit perturbé par la présence d'un trop grand nombre de signes diacritiques, on effectue au préalable un filtrage de ces composantes (voir section 4.1.1).

Pour éviter les pics parasites qui peuvent apparaître sous la bande de base, nous utilisons les boucles pour restreindre la zone de recherche du maximum de l'histogramme. Cette partie sera présentée dans la section 4.1.2.1.

Une fois la première approximation de la bande de base obtenue à l'aide de seuils sur l'histogramme (voir section 4.1.2.2), on utilise le squelette pour déterminer plus précisément la ligne support de l'écriture (voir section 4.1.2.3).

4.1.2.1 Les boucles

En arabe, les descendants peuvent avoir des portions horizontales suffisamment longues pour perturber l'histogramme, et faire apparaître un pic parasite. La figure 4.5 illustre ce phénomène : en raison d'un grand nombre de descendants, le maximum de l'histogramme (pic principal) se situe sous la bande de base.

Une fois la plupart des signes diacritiques supprimés, on pré-localise la bande de base en utilisant les boucles, s'il y en a. En arabe, les boucles se situent dans la bande de base (ة , ص , ض , ط , ظ , ع , ف , ق , ه , ھ , ه, و), à l'exception de la lettre (ڤ) pour laquelle les boucles dépassent légèrement en haut et en bas. Localiser les boucles permet donc de déterminer une zone candidate pour la bande de base.

Mais en s'appuyant sur la localisation des boucles dans l'image, on définit une zone élargie centrée sur ces boucles (qui sont supposées dans la bande de base). C'est dans cette zone élargie qu'on recherche le maximum.

FIG. 4.5 – Utilisation des boucles pour déterminer une zone de recherche du maximum de l'histogramme de projection horizontale. Cette zone de recherche est une bande horizontale de trois fois la hauteur maximale des boucles, centrée sur les boucles. La recherche du maximum dans cette zone permet d'éviter le pic du bas, qui est dû à une succession de descendants.

4.1.2.2 Histogramme horizontal

A partir du maximum de l'histogramme, on applique un seuil haut et un seuil bas, qui permettent de déterminer une approximation de la bande de base (lignes rouge et verte de la figure 4.6).

Empiriquement, ces seuils sont fixés à :
seuil haut = 20% de la valeur du pic principal
seul bas = 45% de la valeur du pic principal.

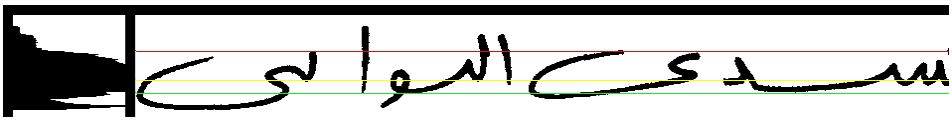


FIG. 4.6 – Histogramme pour estimation grossière de la bande de base. Ligne jaune : pic maximal de l'histogramme. Ligne verte : correspond au seuil bas. Ligne rouge : correspond au seuil haut.

Cette approximation de la bande de base permet d'effectuer un deuxième filtrage des composantes connexes pour achever la détection des signes diacritiques résiduels (deuxième étape de filtrage des signes diacritiques, décrite dans la section 4.1.1, un exemple est donné figure 4.6).

La hauteur de l'écriture (qui sera la hauteur de la bande de base précise) est supposée constante sur l'ensemble de la ligne. Elle est déterminée par la moyenne des distances verticales entre les extremums locaux du contour situés dans cette première estimation de la bande de base (entre les lignes rouge et verte de la figure 4.6).

4.1.2.3 Squelette

Le squelette est extrait à l'aide de l'algorithme de Hilditch [71]. Une image du squelette est donnée figure 4.7.

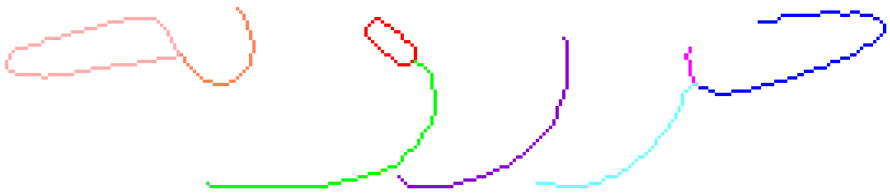


FIG. 4.7 – Extraction du squelette et des traits, pour déterminer les points singuliers.

Les points singuliers du squelette sont les points dont le voisinage est supérieur à 2. Ce sont les points d'embranchement, points de croisement ou boucles. Ils sont marqués en vert dans la figure 4.8. Certains de ces points singuliers sont utilisés pour estimer la ligne d'écriture (ligne de base) de façon plus précise.

Les minimums globaux des arcs de squelette sont également extraits. Les points singuliers retenus sont ceux qui se situent dans la bande de base, et dont l'un des arcs de squelette auxquels ils sont reliés ont un minimum global qui se situe sous la bande de base.

FIG. 4.8 – Utilisation des points singuliers pour trouver une évaluation plus fine de la bande de base.

Un exemple est donné figure 4.8. Dans cet exemple, trois points singuliers sont candidats.

- Celui de gauche appartient à une boucle. Il n'est pas retenu, car les arcs de squelette auxquels il est relié ne vont pas sous la bande de base.
- Celui du milieu appartient également à une boucle. Il est retenu, un trait noir est marqué pour le relier au minimum global de l'arc de squelette considéré.

– Celui de droite appartient à un embranchement. Il est retenu, un trait bleu est marqué pour le relier au minimum global de l'arc de squelette considéré.

Les minimums locaux du contour inférieur sont également retenus s'ils se situent dans la bande de base. Ces minimums locaux et les points singuliers retenus sont de bons estimateurs locaux de la position de la ligne de base. La ligne de base affinée est donc construite comme une interpolation linéaire à partir de ces points retenus. La partie supérieure de la bande de base est donnée par la hauteur du tracé, supposée constante sur toute la ligne.

Au final, on obtient une détection de la ligne de base qui s'adapte bien aux légères variations de l'inclinaison de l'écriture au sein d'une même ligne (voir figure 4.9).

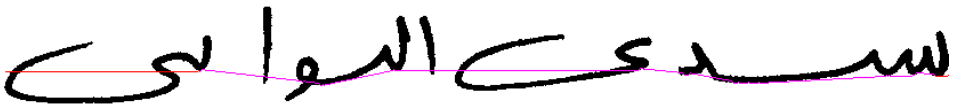


FIG. 4.9 – Détection fine de la bande de base.

4.1.2.4 Evaluation sur la base IFN/ENIT

Dans la section 3.2.2.2.1, nous avons décrit la procédure utilisée par Pechwitz et al. [149] pour extraire la ligne de base. Ils estiment qu'un décalage de 15 pixel par rapport à l'annotation est acceptable. Leur système extrait la bande de base avec un décalage d'au plus 15 pixels dans 95% des images.

Selon une procédure d'évaluation similaire, nous rapportons un taux de 88,4 %. Toutefois, cette manière d'évaluer la qualité de la ligne de base n'est pas toujours adaptée. Il arrive que des lignes de base satisfaisantes soient sanctionnées avec un décalage important. Un exemple de ce type est donné figures 4.10 et 4.11. Dans cet exemple, bien que la bande de base extraite semble satisfaisante, la partie gauche de la ligne de base est décalée de 20 pixels par rapport à l'annotation.

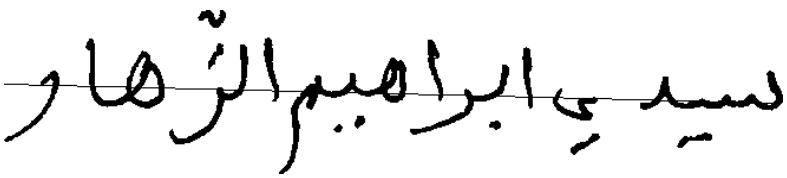


FIG. 4.10 – Ligne de base annotée.

Cet exemple illustre la difficulté d'évaluer la qualité de la bande de base selon un critère réellement significatif. Au delà d'une mesure basée sur des pixels, une mesure plus pertinente de la qualité de la ligne de base est le taux de reconnaissance obtenu par le système de reconnaissance dans lequel est intégrée cette extraction de la ligne de base.

Un système dont la ligne de base serait évaluée de façon très naïve comme étant une ligne droite au milieu de l'image obtient 82,3% de reconnaissance. Le même système utilisant la ligne de base annotée obtient 87,5% de reconnaissance. Le même système

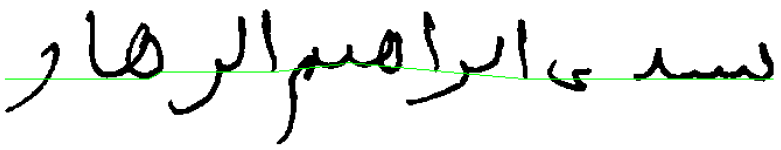


FIG. 4.11 – Ligne de base extraite par notre système.

avec la détection de la ligne de base présenté ci-dessus obtient 87,2% de reconnaissance (les résultats de ce dernier système sont détaillés dans la section 4.2.4.2).

La bande de base annotée est une ligne droite, ce qui n'est pas adapté pour certains documents dont le skew varie au sein de la ligne. On pourrait donc imaginer qu'un algorithme d'extraction de la ligne de base plus précis que l'annotation puisse obtenir de meilleurs résultats. L'approche très naïve montre que le reconnaisseur situé en aval est robuste aux décalages de la ligne de base, et absorbe une partie de la variabilité.

4.1.3 Segmentation en graphèmes

La segmentation graphèmes utilisée dans ce système est la même que celle utilisée dans la reconnaissance de l'écriture cursive latine [17, 55]. Cette segmentation est présentée dans la section 2.2.3.1.

Des exemples de segmentation en graphèmes sont donnés figures 4.12 et 4.13.

Cette segmentation dépend de nombreux seuils, et donne parfois des résultats erronés. Des exemples de sous-segmentations sont donnés figures 4.14 et 4.15. D'autres erreurs de segmentation sont dues aux liaisons basses, qui sont des points de contacts enrte descendants (un exemple de ce type est illustré figure 4.16).

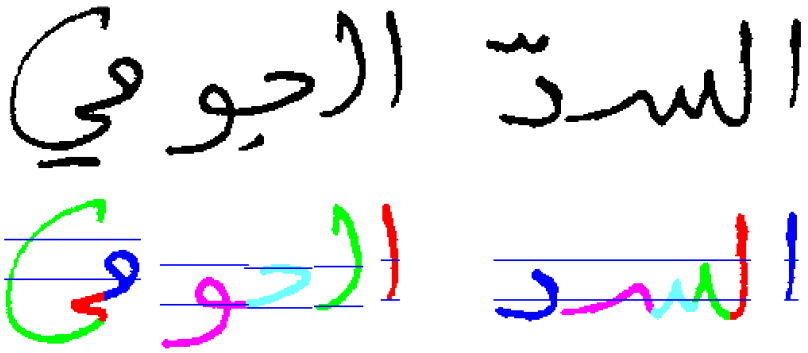


FIG. 4.12 – Exemple de segmentation en graphèmes : السدّ الجوفي.

Les graphèmes sont parcourus de la droite vers la gauche, selon le sens de lecture de l'écriture arabe. Sur chaque graphème, on extrait un vecteur de primitives de 74 dimensions. Cette extraction de primitives est identique à celle utilisée par E. Augustin pour la reconnaissance de l'écriture manuscrite latine [17]. Elle ne sera pas décrite ici.

Un champ à reconnaître est ainsi sous forme d'une séquence de vecteurs d'observations

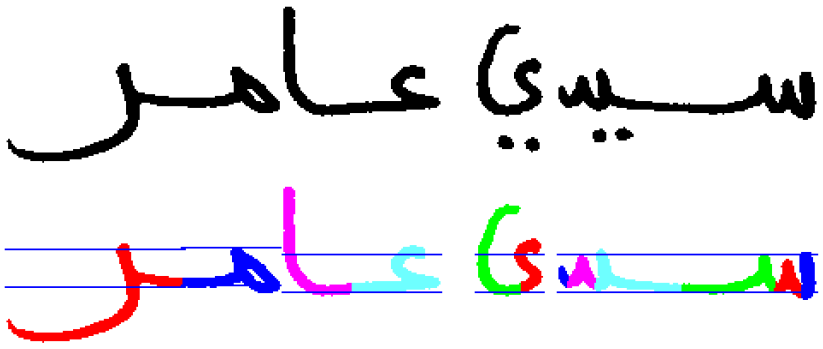


FIG. 4.13 – Exemple de segmentation en graphèmes : سيدي عا مر.

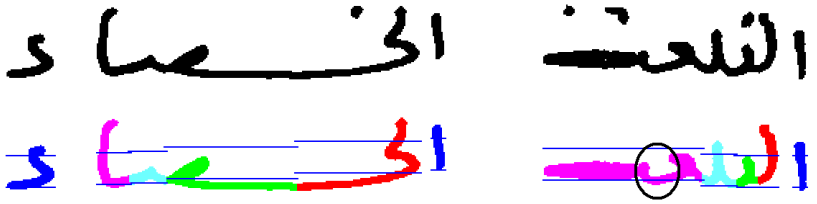


FIG. 4.14 – Exemple d'erreur de segmentation (sous segmentation entourée) : القلعة الخصبا ء

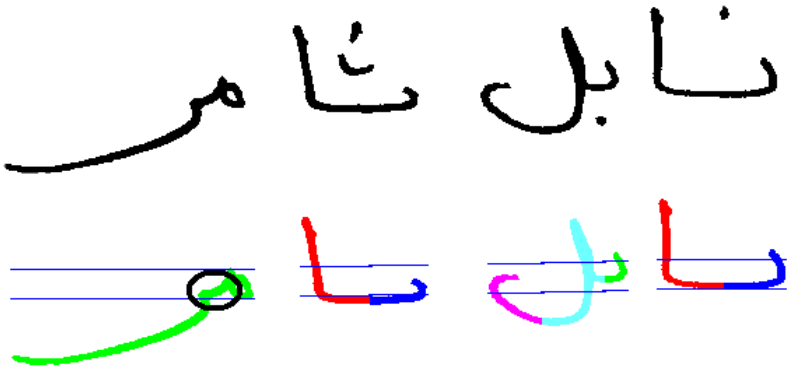


FIG. 4.15 – Exemple d'erreur de segmentation (sous segmentation entourée) : نابل ثامر

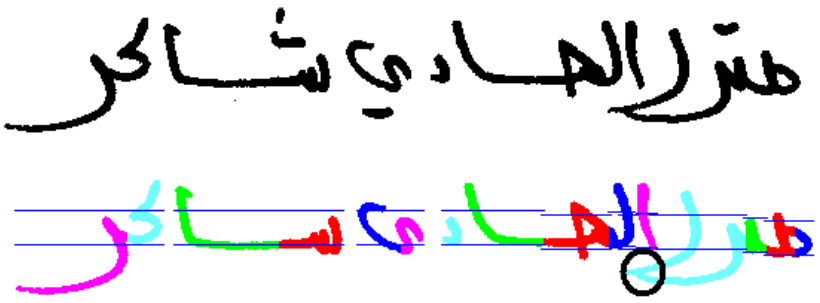


FIG. 4.16 – Exemple d'erreur de segmentation (sous segmentation entourée) : منـزل الهادي شاكر

$O = o_1, \ldots, o_T$ où T est le nombre de graphèmes. Cette séquence est décodée par le reconnaisseur présenté dans la section suivante.

4.1.4 Système de reconnaissance et initialisation

Le reconnaisseur utilisé est du même type que celui utilisé pour la reconnaissance de l'écriture latine. Il s'agit d'un système hybride à base de Modèles de Markov Cachés et d'un réseau de neurones de type Perceptron MultiCouches [17, 55]. Les états sont regroupés en colonnes, et chaque état ne peut émettre qu'une seule classe d'observations (voir figure 4.17).

Soit une séquence d'observations $O = o_1, \ldots, o_T$ de longueur T obtenue par l'extraction de primitives sur une séquence de graphèmes.

La tâche de reconnaissance consiste à trouver la classe du mot correct, dans un vocabulaire prédéfini W. C'est trouver W^* tel que :

$$w^* = \arg\max_{w_i \in W} P\left(w_i|O\right)$$

La règle de Bayes donne :

$$w^* = \arg\max_{w_i \in W} P\left(w_i|O\right) = \arg\max_{w_i \in W} \frac{P\left(O|w_i\right) P\left(w_i\right)}{P\left(O\right)} = \arg\max_{w_i \in W} P(O|w_i)P(w_i)$$

$P(O)$ est indépendant de w_i. Et $P(w_i)$ est la probabilité a priori de la classe de mot w_i. Ici, nous nous passons de cette information : nous considérons que tous les mots du vocabulaire sont équiprobables. Dans notre cas, maximiser la probabilité a posteriori ($P(w_i|O)$) revient à maximiser la vraisemblance des données ($P(O|w_i)$).

Les modèles de mots sont construits par la concaténation des modèles de lettres qui les composent : $P(O|w_i) = P(O|M_i)$, où M_i est le modèle HMM obtenu par la concaténation des HMM des lettres qui composent le mot w_i.

Lors de la phase de reconnaissance, on construit un modèle de mot par entrée du vocabulaire, on a donc N_w modèles M_1, ..., M_{N_w}, où N_w est la taille du vocabulaire.

$$c^* = \arg\max_{i=1,\ldots,N_w} P(O|M_i)$$

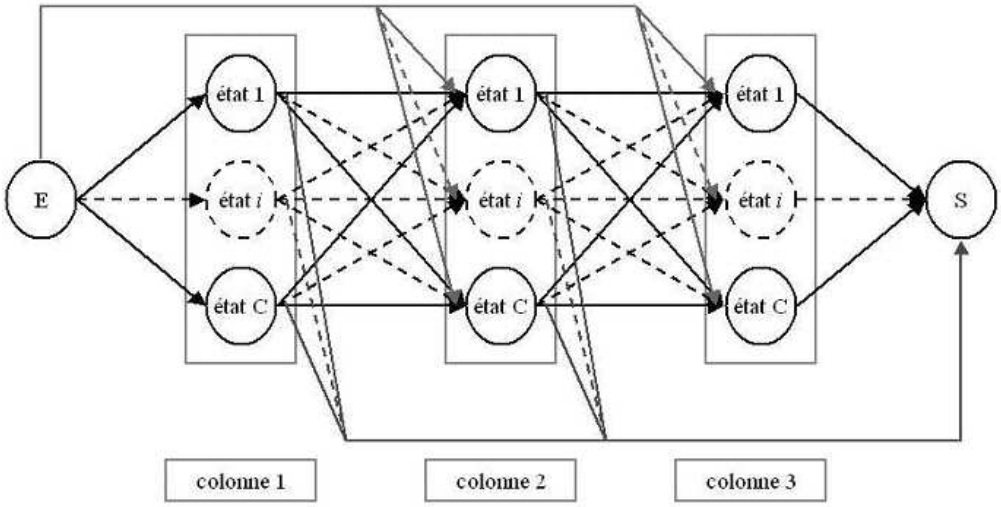


FIG. 4.17 – HMM colonnes d'états.

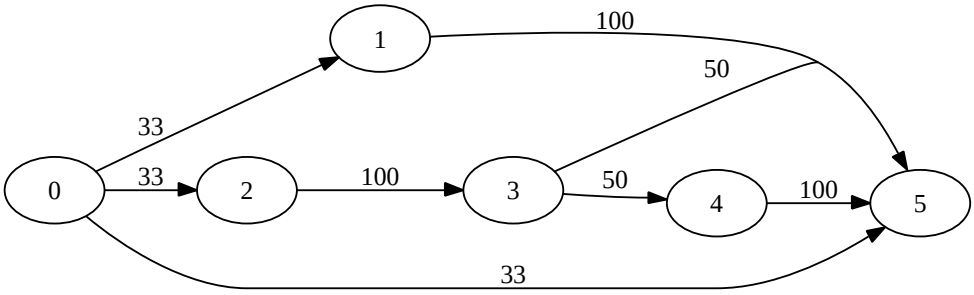


FIG. 4.18 – Topologie d'un modèle de lettre arabe. Il s'agit d'un modèle gauche-droite du même type que ceux utilisés en reconnaissance de l'écriture latine.

c^* est l'indice de classe de mot w^*.

Les modèles de lettres sont entraînés à l'aide de l'algorithme de Baum-Welch, qui permet d'adapter les paramètres des modèles pour maximiser la vraisemblance sur l'ensemble de la base d'apprentissage.

Pour résoudre le problème que constituait l'initialisation du système, nous avons procédé de la façon suivante :

– La topologie des modèles HMM est une topologie gauche/droite du même type que celle utilisée en reconnaissance de l'écriture latine (voir figure 4.18). C'est la séquence d'observation qui est générée de manière différente : en latin les graphèmes sont parcourus de la gauche vers la droite, en arabe les graphèmes sont parcourus de la droite vers la gauche. Les probabilités de transition sont initialisées de manière uniforme.
– Pour initialiser le réseau de neurones, on commence par procéder à un K-Means sur

l'ensemble des vecteurs de primitives de la base d'apprentissage. Le résultat de ce K-Means sert à initialiser un réseau de neurones de type Perceptron Multi-Couches, dont le nombre de classes de sortie sera le même que le nombre K de partitions du K-Means. Cette partie est détaillée plus longuement en Annexe B.

Ce premier système est ensuite entraîné de manière itérative à l'aide de l'algorithme de Baum-Welch. Cette procédure d'apprentissage est décrite en annexe A.

4.2 Alphabet de corps de lettres

Nous avons proposé un nouvel alphabet [116]. Cet alphabet exploite un certain nombre de spécificités de l'écriture arabe, en particulier la redondance des formes que peut prendre une même lettre en fonction de sa position dans le mot ; et la redondance des formes des lettres qui ne se différencient que par la présence, la position, ou le nombre de points.

Les expériences confirment que l'utilisation de cet alphabet modifié est utile pour la stratégie de reconnaissance.

4.2.1 Description de l'alphabet

L'analyse de l'alphabet arabe révèle certaines particularités intéressantes, qui peuvent être exploitées pour établir une stratégie de reconnaissance. Ces particularités sont les suivantes :

- les lettres s'écrivent en général sous forme : radical + terminaison
- la similarité des formes des lettres qui ne se différencient que par les points
- les ligatures verticales

4.2.1.1 Radical et terminaison

On a souvent coutume de dire que les lettres arabes peuvent prendre 4 formes différentes en fonction de leur position dans le mot. C'est vrai pour les lettres (ع ـعـ ـع ع), (غ ـغـ ـغ غ) et (ه ـهـ ـه ه). Mais c'est faux dans le cas général. Pour la plupart des lettres, la forme est la même selon qu'elle soit en début ou en milieu de mot. Comme pour les langues qui reposent sur l'alphabet latin, en écriture cursive, une lettre qui se situe en milieu de mot se voit attribuer une ligature avec la lettre qui la précède. En français par exemple, on ne dit pas que la lettre 'e' prend deux formes en fonction de sa position dans le mot. Il en est de même avec les lettres arabes. Une lettre en milieu de mot n'a pas une forme supplémentaire. C'est la même forme avec une ligature.

En revanche, lorsque la lettre est en fin de mot, elle est complétée par une terminaison,

FIG. 4.19 – نحّال : classe de ligature verticale ـحـ

FIG. 4.20 – المنزه : classe de ligature verticale ـلـ

à l'exception des formes ا د ر ط و qui sont invariantes. La forme d'une lettre en fin de mot est donc la forme en début/milieu de mot, à laquelle on rajoute une "jambe". Une lettre en fin de mot est du type : radical + terminaison.

4.2.1.2 Des lettres identiques aux points près

Plusieurs sous-ensembles de lettres ne se différencient que par le nombre et/ou la présence et/ou la position de points. Ce qui signifie que pour différencier ces lettres, il est nécessaire de prendre en compte les points. Mais ceci signifie également qu'en écartant les points et en regroupant les corps de lettres dont les formes sont similaires (bien qu'elles proviennent de lettres différentes), on dispose de plus d'exemples pour entraîner des modèles à reconnaître ces formes particulières.

4.2.1.3 Modélisation explicite des ligatures verticales

Les ligatures verticales sont une concaténation verticale de plusieurs lettres, qui sont empilées les unes sur les autres. La segmentation de ces formes en lettres est très complexe, et ne se justifie pas : les transitions entre lettres sont floues, et ces formes sont reconnues en tant que telles par le lecteur. Il conviendra donc de modéliser ces formes explicitement.

Un exemple pour chaque classe de ligature est donné aux figures 4.19 (ـحـ), 4.20 (ـلـ), 4.21 (ـلـه) , 4.22 (ـلـحـ) et 4.23 (لا).

4.2.1.4 Notre alphabet

Pour prendre en compte les trois points précédents, nous proposons un alphabet de corps de lettres, mieux adapté à la reconnaissance automatique de l'écriture arabe cursive que ne l'est l'alphabet arabe standard.

FIG. 4.21 – الجم : classe de ligature لحـ

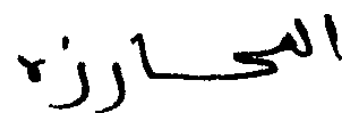


FIG. 4.22 – Ligature de verticale de trois lettres : لـ مـ حـ

FIG. 4.23 – القلّال : classe de ligature لا

ا	ر	ه	ه	حـ	ح	د	ر	سـ	س
صـ	ص	ط	ء	عـ	ع	ع	و	كـ	لـ
ل	م	م	ں	هـ	ھ	و	ى	Tail 1 : ـ	
لا	لحـ	لمـ	لـ	لـ					

TAB. 4.1 – Liste complète des 34 symboles qui constituent notre alphabet.

Cet alphabet comporte 34 symboles, pour représenter les 28 lettres de l'alphabet arabe.

4.2.2 Traduction et Ambiguïtés

Un mot arabe (séquence de lettres de l'alphabet arabe) sera traduit en un mot dans notre alphabet (une séquence de symboles de notre alphabet). Cette traduction fait naître deux types d'ambiguïtés, que nous appellerons : 'alias' et 'confusions'.

4.2.2.1 Les alias

En fonction de la tolérance de l'écriture arabe, un mot peut s'écrire de différentes façons : utilisation ou non de ligatures verticales, utilisation de la forme ـت ou ـة pour représenter la lettre ـتـ en fin de mot. Un même mot arabe (séquence de lettres dans l'alphabet arabe) pourra donc donner lieu à plusieurs séquences de formes dans notre alphabet, en fonction des différentes manières dont ce mot est écrit. Toutes ces séquences de formes sont des alias, elles sont synonymes en ce sens qu'elles correspondent toutes à un mot unique dans l'alphabet initial.

Ce premier type d'ambiguïté ne pose pas de problème : il suffit de cumuler les probabilités de toutes les séquences de formes synonymes pour exprimer une liste de résultats dans l'alphabet des lettres arabes.

La quantité d'alias dépend de la variabilité des styles d'écriture des scripteurs dans la base.

4.2.2.2 Les confusions

Les confusions sont des ambiguïtés plus problématiques. Ce sont les cas pour lesquels une séquence de formes (lettres dans le nouvel alphabet) correspond à deux mots distincts du vocabulaire exprimé dans l'alphabet initial (lettres arabes). Dans ce cas là, étant donné une séquence de formes, il est impossible de déterminer à quelle séquence de lettres elle se rapporte, sans analyser les signes diacritiques. Dans ces cas là, les signes diacritiques contiennent une information dont il est impossible de se passer.

4.2.2.3 Application à la base IFN/ENIT

La quantité de confusions est déterminée par le vocabulaire de l'application considérée.

Par exemple, dans le cas de la base IFN/ENIT (937 villes tunisiennes), étant donné l'alphabet mentionné ci-dessus, la taille du vocabulaire obtenu devient 1280. La taille du

vocabulaire augmente, du fait de la variabilité des styles d'écriture des scripteurs : un même mot, qui correspond à une séquence de lettres données, peut s'écrire de différentes manières, et peut donc donner lieu à plusieurs séquences de formes (voir section 3.1.2.6).

En reconnaissance de formes, et a fortiori en reconnaissance de l'écriture, on cherche en général à réduire au maximum la taille du lexique. Or le traitement effectué produit l'effet inverse : la taille du lexique augmente. Dans 4.2.5, nous verrons que malgré cela, les performances sont améliorées.

Les classes qui portent à confusion sont les suivantes :

– classes 7027 (مثليـن) et 7034 (متليـن) : 45 images

– classes 1240 (فريانة) et 8140 (فرنانة) : 15 images

– classes 1220 (فوسانة) et 2082 (فوشانة) : 16 images

Un autre type de confusion peut provenir de la similarité entre les ـسـ et les successions de ـنـ. Ce type de confusion concerne les classes suivantes :

– classes 3010 (الـحنشة) et 2012 (الـحبيبيّة) : 15 images

La confusion porte donc sur 0,34% de la base de données. C'est très largement inférieur aux taux d'erreurs de reconnaissance rapportés sur cette base. Donc malgré les confusions introduites, l'utilisation de cet alphabet de formes n'est pas problématique pour une tâche de reconnaissance sur la base IFN/ENIT.

4.2.2.4 Application à un vocabulaire de montants

خـمسة	خـمس	اربعة	اربع	ثلاثة	ثلاث	اثنيـن	اثنان	وحد
عشـر	تسعة	تسع	ثـمانية	ثـمان	سبعة	سبع	ستة	ست
اربعون	ثلاثيـن	ثلاثون	عشـرين	عشـرون	اثنـن	احدى	احد	عشـرة
ثـمانيـن	ثـمانون	سبعيـن	سبعون	ستيـن	ستون	خـمسيـن	خـمسون	اربعيـن
اربعمئة	ثلاثـمائة	ثلاثـمئة	مائتيـن	مئتيـن	مائة	مئة	تسعيـن	تسعون
ثـمانـمائة	ثـمانـمئة	سبعمائة	سبعمئة	ستمائة	ستمئة	خـمسمائة	خـمسمئة	اربعمائة
		مليون	الاف	الفيـن	الفان	الف	تسعمائة	تسعمئة

TAB. 4.2 – Vocabulaire des nombres arabes en masculin et en féminin.

Le tableau 4.2 donne le vocabulaire des mots utilisés dans les montants. Les mots sont donnés dans leur forme masculine et féminine. L'étude de ce vocabulaire laisse supposer que des confusions peuvent apparaître entre les couples 7 et 9, 70 et 90, ainsi que 700 et 900.

En effet, il semble très difficile de différencier une séquence "سـ" d'une séquence "تـسـ" sans avoir recours aux points, en particulier avec la variabilité de l'écriture manuscrite. On retrouve ce problème dans :

- 7 (تسعة ou تسع) et 9 (سبعة ou سبع)
- 70 (سبعون ou سبعيـن) et 90 (تسعون ou تسعيـن)
- 700 (سبعمائة ou سبعمئة) et 900 (تسعمائة ou تسعمئة)

Les couples 7/9, 70/90 et 700/900 semblent donc difficiles à différencier à l'aide de notre alphabet sans diacritiques. Néanmoins, dans le cadre d'une application de reconnaissance automatique de chèques, où la reconnaissance du montant littéral est combinée à la reconnaissance du montant numérique, le fait de fusionner les couples de classes problématiques pourrait tout de même apporter de l'information. Cette solution est envisageable dans le cas de documents très bruités pour lesquels il serait difficile de faire la différence entre des bruits de fond et des signes diacritiques.

En revanche, si la qualité des documents permet d'utiliser les signes diacritiques, l'approche qui sera présentée dans la section 4.4 pourra être appliquée.

Etablir le vocabulaire exact des séquences de corps de lettres n'est possible que parce que la base IFN/ENIT est annotée au niveau formes de lettres. Dans le cas plus général, il sera nécessaire de déduire un vocabulaire plus général à partir des annotations de mots (séquences de lettres), en dérivant tous les cas possibles. Ce point est développé dans le paragraphe suivant.

4.2.3 Générer le vocabulaire de formes à partir des séquences de lettres

La base IFN/ENIT (voir paragraphe 3.2.1.4) fournit une annotation de la forme que prend chaque lettre au sein d'un mot (forme début, milieu, finale ou isolée). A partir de l'ensemble des annotations, il est donc possible d'établir un vocabulaire des séquences de formes possibles exprimées dans notre alphabet.

Ce vocabulaire de séquences de formes dérivé à partir des annotations est minimal, puisqu'il n'intègre que la variabilité des exemples que contient la base de données. Mais on pourrait imaginer que d'autres scripteurs auraient introduit d'autres types de variabilités. Pour travailler dans un cas plus général, il est donc nécessaire de dériver toutes les séquences de formes possibles à partir d'une séquence de lettres données (ie toutes

les manières possibles d'écrire un même mot).

Il convient également de s'assurer que le nombre de confusions (voir paragraphe 4.2.2.2) n'augmente pas significativement.

4.2.3.1 Alternatives possibles

Pour dériver toutes les séquences de formes possibles à partir d'un mot donné, il faut prendre en compte les alternatives suivantes :

– ا en fin de mot peut s'écrire : ا ou ى

– ـتـ en fin de mot (lié à la lettre précédente) peut s'écrire : ـتـ ou ـة .

– ـتـ isolé (en fin de mot et non lié à la lettre précédente) peut s'écrire : ت ou ة .

Il faut également rajouter les alternatives qui proviennent du fait qu'on peut ou non utiliser des ligatures verticales (voir paragraphe 4.2.1.3) :

– ـبـحـ éventuellement remplacés par : ـبحـ

– ـلـم éventuellement remplacés par : ـلم

– ـلـحـ éventuellement remplacés par : ـلحـ

– ـلـمـحـ éventuellement remplacés par : ـلمحـ

Le fait de rajouter toutes ces variantes accroît substantiellement la taille du vocabulaire : on obtient 2229 manières différentes d'écrire les 937 mots du vocabulaire de la base IFN/ENIT.

On passe donc ainsi de 937 villes à 1280 séquences de formes différentes dans la base de données, pour une limite théorique de 2229 séquences de formes totales.

4.2.3.2 Nouvelles confusions sur la base IFN/ENIT

Le fait de dériver l'ensemble des séquences de formes possibles sans tenir compte de l'information portée par l'annotation ne fait apparaître qu'un seul cas d'ambiguïtés supplémentaires : les classes 4232 (تنبيب) et 3083 (تينة).

Dériver toutes les formes possibles de la classe 3083 donne les cas de figure suivants : { تينة , تينت }.

La classe 3083 dérive donc les formes : { سب , سه }

Tandis que la classe 4232 dérive la forme : { سب }.

Il y a donc confusion entre ces deux classes. Mais là encore, ce problème ne touche qu'une très faible partie de la base de données, ce qui provoque un impact négligeable sur le résultat final :

– la classe 3083 contient 8 images

– la classe 4232 contient 33 images

Ce qui représente 41 images sur 26459, soit 0.15% de la base de données, qui s'ajoutent aux 0.34% de confusions qui sont décrits dans le paragraphe 4.2.2.3.

4.2.4 Expériences et résultats

Comme nous l'avons vu dans le paragraphe 4.2.2.3, la confusion porte sur 0,34% de la base de données, ce qui rend donc possible l'utilisation de notre alphabet de formes sur cette application.

La base IFN/ENIT (voir paragraphe 3.2.1.4) est prédécoupée en sous-bases. Historiquement, cette base a été rendue publique en 2003 sous la forme de quatre sous-bases : les ensembles {a,b,c,d}. Afin de rendre les résultats des différents systèmes comparables, les chercheurs sont invités à se servir des ensembles {a,b,c} en apprentissage, et de l'ensemble {d} en test. Les expériences suivantes respectent cette procédure.

4.2.4.1 Objectif

L'objectif de ces expériences est de montrer l'intérêt de l'alphabet de corps de lettres par rapport à une approche naïve, qui consisterait à prendre un alphabet de 28 lettres. Pour cela, nous comparerons les résultats des quatre systèmes de reconnaissance suivants :

Le système de référence est appris de la manière suivante : chacune des 28 lettres de l'alphabet arabe est modélisée par un HMM. De même que pour l'écriture latine, les points et autres signes diacritiques sont filtrés et ne sont pas utilisés pour la reconnaissance. Un modèle de mot est obtenu par concaténation de modèles de lettres.

Un deuxième système est construit selon le même principe que le système de référence : chacune des 28 lettres de l'alphabet est modélisée par un HMM. Mais cette fois-ci, les signes diacritiques ne sont pas filtrés. Les signes diacritiques sont réaffectés aux lettres selon un critère de proximité.

Un troisième système est composé de 112 modèles de HMM : une classe par forme de lettre en fonction de sa position dans le mot.

Un quatrième système utilise l'alphabet de corps de lettres décrit dans les paragraphes précédents : les lettres qui correspondent à des formes identiques sont regroupées au sein d'une même classe, qui dispose ainsi de plus de données en apprentissage par la mise en commun de symboles qui sont attribués à des classes différentes dans les autres expériences.

Et enfin un quatrième système, identique au troisième, au vocabulaire près. Dans le troisième système, le vocabulaire des séquences de formes est déduit des exemples disponibles dans l'ensemble d'apprentissage et dans l'ensemble de validation. Ceci n'est possible que parce que la base IFN/ENIT est annotée au niveau formes de lettres (voir paragraphe 4.2.2.3). Dans le quatrième système, le vocabulaire de séquences de formes (utilisé en reconnaissance) est dérivé automatiquement à partir du vocabulaire de mots, en prenant en compte toutes les variabilités possibles que peuvent introduire les scripteurs. La taille de ce vocabulaire sera donc plus importante, et inclura les manières d'écrire les mots qui n'ont que peu de chances d'être réalisées, et qui ne se produisent pas dans la base IFN/ENIT.

4.2.4.2 Résultats

Les résultats des expériences mentionnées ci-dessus sont reportés dans le tableau 4.3.

La comparaison des tests I et III valide l'apport de l'alphabet de 34 symboles que nous proposons, par rapport à l'alphabet de lettres standard de la langue arabe. Le fait de regrouper les lettres qui ne se différencient qu'à l'aide des signes diacritiques, et d'ajouter des classes de symboles pour modéliser explicitement les ligatures verticales, améliore les performances de façon significative.

La comparaison des tests I et II montre que le fait d'associer les signes diacritiques aux graphèmes les plus proches est une technique trop naïve, qui génère plus de bruit qu'elle n'apporte d'information. D'autres alternatives pour traiter les signes diacritiques seront présentées dans la section 4.4.

La comparaison des tests III et IV montre la dégradation des performances lorsqu'on ne dispose d'aucune connaissance a priori sur la manière dont sont écrits les mots, et qu'on génère toutes les alternatives de séquences de formes possibles pour chacun des mots. En particulier avec les ـة et les ـت, on génère de nombreuses alternatives qui n'ont que très peu de chances d'être réalisées. La taille du vocabulaire passe ainsi de 1280 séquences de formes possibles à 2229 séquences de formes possibles. Plus de 40% des séquences ainsi générées ne correspondent à aucune image dans la base IFN/ENIT. La figure 4.24 montre les courbes de reconnaissance / substitution (présentées dans la section 2.6.1) qui comparent les résultats des tests III et IV.

Les quatre dernières lignes du tableau 4.3 donnent les résultats des principaux systèmes dont les auteurs ont publié sur cette base de données. Ces systèmes prennent en compte les signes diacritiques. Ce n'est pas le cas du système III du tableau 4.3. Dans la section 4.2.2.3, nous avons montré que les signes diacritiques ne sont pas indispensables pour une tâche de reconnaissance sur la base IFN/ENIT. Néanmoins, ils apportent une information qui peut aider la reconnaissance. C'est ce que nous montrerons dans la section 4.4.

	1ère pos	2ème pos	10ème pos	Taille vocabulaire
Apprentissage {a,b,c} III	90.3	94.1	97.5	1280
Test {d} I	81.6	87.6	96.0	937
Test {d} II	73.2	80.2	91.6	937
Test {d} III	87.2	91.9	96.4	1280
Test {d} IV	86.3	91.1	96.3	2229
UOB [4] {d}	90.96	92.95	94.44	937
ARAB-IFN [148] {d}	89.1	91.7	95.9	937
SCHMMs [24] {d}	89.79	92.25	96.78	937
Microsoft Research [2] {d}	88.94		95.01	937

TAB. 4.3 – Résultats de reconnaissance en 1ère, 2ème et 10ème position :
I : Alphabet arabe (28 lettres) sans les points
II : Alphabet arabe (28 lettres) avec les points
III : Notre alphabet de corps de lettres
IV : Notre alphabet de corps de lettres dérivé depuis l'alphabet de lettres (toutes les alternatives possibles, même celles qui ne sont pas réalisées dans la base).

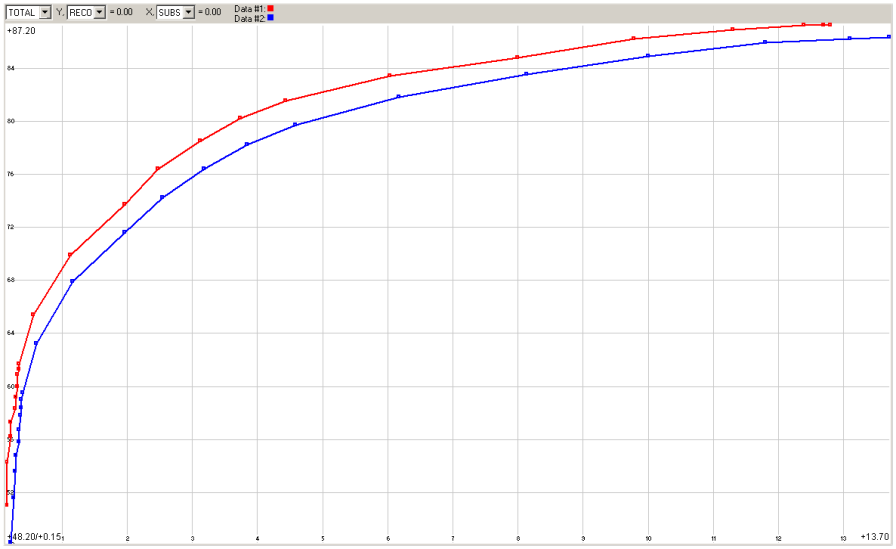


FIG. 4.24 – Courbes de Reconnaissance / Substitution. En rouge : Test III. En bleu : Test IV.

$T_{reco(seuil)}$	61.7	69.9	73.7	76.4	78.5	81.5	83.4	84.8	86.2	86.9	87.2	87.2	87.2	87.2
$T_{sub(seuil)}$	0.34	1.13	1.96	2.48	3.13	4.44	6.04	8.00	9.79	11.3	12.4	12.7	12.8	12.8
Seuil	990	950	900	850	800	700	600	500	400	300	200	100	50	0

TAB. 4.4 – Tableau reconnaissance / substitution : Test III. Correspond à la figure 4.24

La figure 4.24 et le tableau 4.4 montrent que le reconnaisseur est fiable pour les seuils de confiance élevés. Cette caractéristique est intéressante dans le cadre d'une utilisation dans un système permettant le rejet des réponses en cas de manque de fiabilité.

Un exemple de mot correctement reconnu est donné figure 4.25.

Des exemples de mots mal reconnus sont également donnés figures 4.26 et 4.27.

4.2.5 Conclusion

Nous avons proposé un alphabet de symboles pour améliorer la reconnaissance de l'écriture manuscrite. Bien que le nombre de symboles de cet alphabet soit plus important que le nombre de lettres de l'alphabet arabe standard (34 au lieu de 28), et bien que le vocabulaire de mots qui en découle soit également plus important (1280 au lieu de 937), cet alphabet de symboles permet une amélioration significative du taux de reconnaissance. Le fait de regrouper certains symboles dont la forme est identique aux points près permet d'entraîner un même modèle HMM à partir de caractères provenant de lettres différentes. Et le fait de rajouter de nouveaux symboles pour modéliser explicitement les principaux types de ligatures verticales facilite leur reconnaissance.

Néanmoins, le fait d'écarter volontairement les signes diacritiques prive le reconnaisseur d'une partie des informations. Ce premier système, basé uniquement sur les corps de lettres, pourra être combiné avantageusement avec un reconnaisseur de signes diacritiques. Cet aspect sera présenté dans la section 4.4.

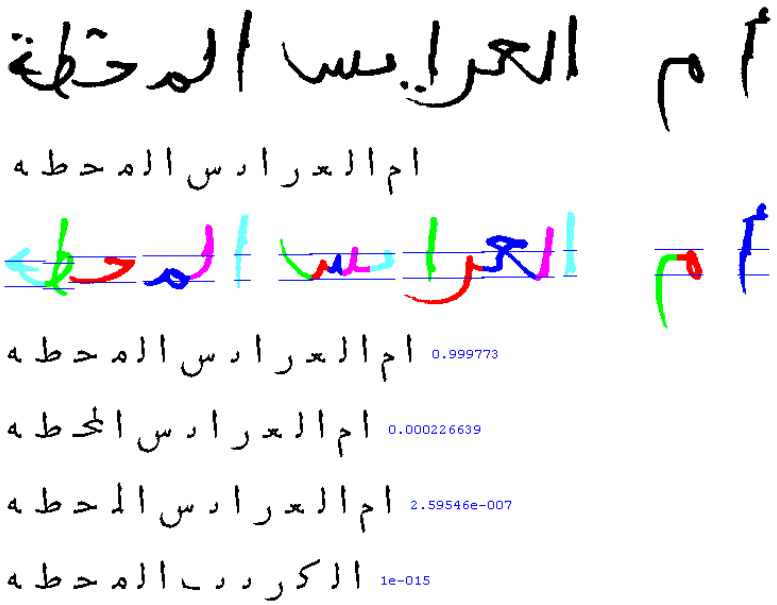


FIG. 4.25 – Exemple de reconnaissance correcte : 1ère ligne : image initiale. 2ème ligne : annotation du mot exprimée dans notre alphabet de corps de lettres. 3ème ligne : segmentation en graphèmes. Lignes suivantes : meilleurs candidats de reconnaissance.

FIG. 4.26 – Erreur de reconnaissance : un exemple de la classe 6122 a été reconnu comme un 5135. ح reconnu comme un ف. La réponse correcte est en deuxième position.

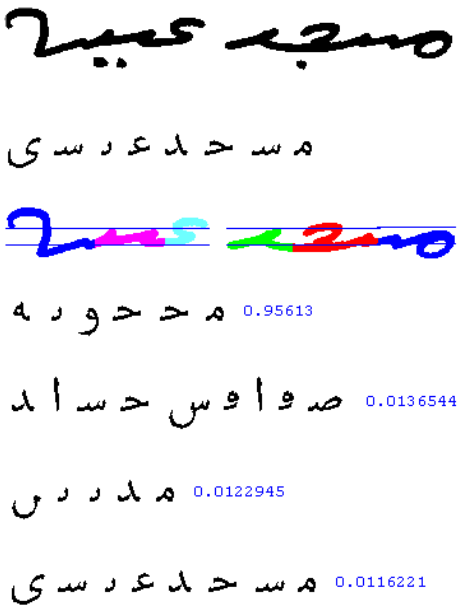


FIG. 4.27 – Erreur de reconnaissance due à une sous-segmentation : un exemple de la classe 5042 a été reconnu comme un 7132. La réponse correcte est en quatrième position.

D'autres pistes d'améliorations sont également possibles, en particulier :
- la segmentation graphème pourrait être adaptée pour réduire la quantité de sous-segmentations (cet aspect est présenté dans la section 4.1.3).
- un mécanisme permettant de générer plusieurs candidats lors de la phase d'extraction des signes diacritiques éviterait de prendre une décision trop brutale dès le début du traitement, ce qui serait préférable à la solution actuelle (cet aspect est présenté das la section 4.1.1).

4.3 Compétition ICDAR 2007 sur la reconnaissance de l'écriture arabe

4.3.1 Historique : compétition 2005

En 2005, ICDAR a accueilli la première compétition de reconnaissance de l'écriture manuscrite arabe [113]. Elle s'est déroulée sur la base IFN/ENIT, rendue publique deux ans plus tôt.

La base IFN/ENIT était prédécoupée en quatre sous-ensembles : {a,b,c,d}. Les participants ont entraîné leurs systèmes sur ces 4 sous-ensembles, tandis que l'évaluation portait sur un cinquième sous-ensemble {e} jusque là inconnu. Ce sous-ensemble {e} a par la suite été rendu publique en 2007. Ce sous-ensemble {e} était donc également à la disposition des participants pour entraîner les systèmes soumis lors de la compétition ICDAR 2007.

Cinq systèmes ont été évalués. Tous présentaient des performances assez proches sur la base d'apprentissages, mais certains d'entre eux souffraient d'un problème de sur-apprentissage manifeste, comme par exemple le système SHOCRAN, passant de 100% de reconnaissance sur la base d'apprentissage à 35.70% sur la base d'évaluation.

UOB est le système qui a remporté la compétition 2005. Il est le fruit d'une collaboration de Chafic Mokbel de l'Université de Balamand au Liban, et Laurence Likforman-Sulem de l'ENST-Paris, qui ont encadré les travaux de Ramy El-Hajj. Il s'agit d'un système à base de Modèles de Markov initialement développé pour la parole, adapté à la reconnaissance de l'écriture. L'extraction de caractéristiques est faite par fenêtre glissante. Les Modèles de Markov de lettres sont concaténés pour former des modèles de mots.

Les systèmes ayant présenté les meilleures performances étaient basés sur des Modèles de Markov Cachés et des Réseaux de Neurones, montrant ainsi que les techniques standards de reconnaissance de l'écriture manuscrite pouvaient être adaptées à la reconnaissance de l'écriture arabe.

Mais comme l'indique la conclusion de l'article récapitulatif de la compétition 2005 : certains systèmes à base de Modèles de Markov Cachés ou de réseaux de neurones ont également fourni de très mauvaises performances. Ce résultat montre que le choix d'une famille de reconnaisseurs n'est pas suffisant pour obtenir de bonnes performances et un système de reconnaissance robuste. Les normalisations et le choix des primitives utilisées jouent également un rôle très important dans la qualité du reconnaisseur.

4.3.2 Participants

Suite au succès de la première campagne d'évaluation de la reconnaissance de l'écriture arabe manuscrite, ICDAR a renouvelé l'opération en 2007 [112]. La compétition 2007 a vu apparaître de nouveaux participants, dont le système de reconnaissance présenté dans cette thèse.

Huit participants différents ont soumis un total de quatorze systèmes de reconnaissance. Deux de ces participants figuraient déjà lors de la compétition 2005 : ICRA et UOB. Les nouveaux venus étaient : MITRE, CACI, le CEDAR, MIE, SIEMENS et PARIS-5 (notre système tel que présenté dans la section 4.2.4.2. Cette version n'intègre pas la combinaison avec les signes diacritiques qui sera présentée dans la section 4.4).

4.3.3 Résultats

Le système UOB avait fourni les meilleures performances lors de l'édition précédente. En modifiant les primitives extraites et en combinant plusieurs classifieurs, ses auteurs ont nettement amélioré leur système. Mais en deux ans, le niveau de l'état de l'art a été considérablement relevé. Siemens, en adaptant leur système de reconnaissance de l'écriture manuscrite latine [77, 40, 163], a obtenu les meilleures performances. Ils ont ainsi montré à leur tour que contrairement à une idée véhiculée pendant des années dans la communauté de l'écriture manuscrite arabe, la reconnaissance de l'écriture manuscrite arabe n'est pas fondamentalement plus complexe que la reconnaissance de l'écriture latine cursive : moyennant quelques adaptations spécifiques, les techniques de reconnaissance peuvent être transférées de l'une à l'autre avec profit.

C'est également l'approche de la Mie University du Japon. Les auteurs ont proposé un système qui s'appuie sur celui développé dans la cadre de la reconnaissance postale en anglais [60, 89], avec quelques adaptations pour l'écriture arabe.

Les organisateurs de la compétition ont prévu d'écrire un article de journal qui présentera plus longuement les différents systèmes ayant participé à cette compétition. Cet article est actuellement en cours de rédaction.

Pour la prochaine édition (ICDAR 2009), les organisateurs envisagent deux pistes supplémentaires :

- L'évaluation du score de confiance, afin de comparer les taux de lecture à taux de substitution constant
- La mesure précise du temps de calcul, et son intégration dans l'évaluation de la performance des reconnaisseurs.

4.4 Reconnaissance des signes diacritiques

Nous avons vu dans la section précédente (4.2.2.3) que la reconnaissance des signes diacritiques (en particulier les points) n'était pas indispensable, selon le vocabulaire utilisé. En particulier, sur la base IFN/ENIT, nous avons montré que les ambiguïtés concernent une très faible portion des données, et qu'il est avantageux d'utiliser un alphabet qui écarte les signes diacritiques et qui permet de prendre en compte la redondance des formes des corps de lettres.

Sur des images bruitées, le fait de pouvoir se passer des signes diacritiques semble également être un avantage important, du fait de la difficulté qu'on pourrait avoir pour discriminer les points des artefacts de numérisation.

Toutefois, pour des applications dont la qualité des documents est bonne, comme c'est le cas pour la base IFN/ENIT, les signes diacritiques sont porteurs d'une information, qu'il pourrait être intéressant d'intégrer dans la chaîne de reconnaissance. Nous proposons donc deux stratégies qui permettent de réintégrer les signes diacritiques dans la procédure de reconnaissance :

- Une première stratégie consiste à définir un alphabet de symboles de diacritiques. Un ensemble de règles simples permet d'associer ces symboles et de constituer une séquence. Une distance d'édition permet de générer des candidats de classes. Cette liste de candidats est ensuite fusionnée avec la liste de reconnaissance obtenue à partir des graphèmes (voir section 4.2). Nous verrons que cette stratégie permet d'améliorer significativement les performances du système.
- La deuxième stratégie, plus complexe à mettre en oeuvre, mais plus naturelle et qui sera sans doute plus efficace, devrait permettre de mieux intégrer des contraintes de la langue arabe. Elle exploite le formalisme des automates à états finis pondérés (WFSM). Nous présenterons cette approche d'un point de vue uniquement théorique, sa mise en oeuvre fera l'objet de travaux ultérieurs.

4.4.1 Règles d'associations des symboles et distance d'édition

Dans cette partie, nous présentons la stratégie qui consiste à utiliser les signes diacritiques pour améliorer les performances du système présenté dans la partie 4.2.4.2. Nous

continuons donc à travailler sur la base IFN/ENIT. Sauf mention contraire, les notions d'ensemble d'entraînement et de test, ainsi que le système de reconnaissance de corps de lettres, feront référence à ceux présentés dans la section 4.2.4.2.

4.4.1.1 Introduction

Dans [130], S. Mozaffari et al. proposent de s'appuyer sur les signes diacritiques pour réduire la taille du vocabulaire. Cette réduction de la taille du vocabulaire se fait en amont du système de reconnaissance, et permet de réduire le temps de calcul au prix d'une baisse du taux de reconnaissance. Ils définissent trois classes de signes diacritiques :

- ensemble 1 : points isolés
- ensemble 2 : couples de points (isolés ou reliés), madda
- ensemble 3 : triplets de points (isolés ou reliés), chadda, hamza

Ils définissent un ensemble de règles heuristiques et de seuils pour regrouper les diacritiques qui doivent l'être (regroupement de 2 ou 3 points isolés), selon des critères de tailles et de positions relatives.

Dans notre approche, nous proposons également des règles d'association pour regrouper les diacritiques selon des critères de taille et de positions relatives. Mais contrairement à l'approche de S. Mozaffari et al, la séquence de signes diacritiques obtenus sera utilisée non pas en amont pour restreindre le vocabulaire et ainsi réduire le temps de calcul au prix d'une réduction du taux de reconnaissance. Au contraire, dans notre approche la séquence de signes diacritiques sera utilisée en aval du reconnaisseur, pour valider ou infirmer les résultats de reconnaissance. Le système basé uniquement sur les corps de lettres présenté dans la section 4.1 est porteur de beaucoup plus d'informations que celui basé uniquement sur les signes diacritiques. Nous avons vu dans la section 4.2.2.3 que les corps de lettres étaient suffisants dans 99,7% des cas dans la base IFN/ENIT. Intuitivement (nous le vérifierons dans la section 4.4.1.2), on sent bien qu'un système basé uniquement sur les signes diacritiques va provoquer beaucoup plus de confusions, car de nombreuses classes ne pourront pas être différentiées uniquement à partir de leurs signes diacritiques. La grosse partie du travail est donc confiée au reconnaisseur de corps de lettres, tandis que la séquence de diacritiques sera utilisée en post-traitement : son rôle sera de faire remonter en première position les candidats mal reconnus par le reconnaisseur de corps de lettres.

Dans la suite de cette section, nous détaillerons les alphabets de corps de lettres que nous utilisons, puis nous expliciterons les règles heuristiques utilisées pour recomposer les diacritiques complexes (deux points et trois points). Nous présenterons ensuite la distance d'édition utilisée et ses variantes. Nous verrons également les différentes stratégies de combinaisons possibles. Puis enfin, nous terminerons cette partie en présentant des pistes d'amélioration possibles.

4.4.1.2 Alphabet de signes diacritiques

Les types de diacritiques (présentés dans le paragraphe 3.1.2.2) sont au nombre de 9, que l'on nommera de (A) à (I) :

- (A) Un point en haut
- (B) Deux points en haut

– (C) Trois points en haut
– (D) Un point en bas
– (E) Deux points en bas
– (F) Chadda
– (G) Madda
– (H) Hamza en haut
– (I) Hamza en bas

Selon les scripteurs, les classes B et E peuvent être écrites sous forme d'une ou deux composantes connexes. La classe C peut être écrite sous forme d'une, deux ou trois composantes connexes.

Il est possible de décrire chaque classe de mots sous forme d'une séquence de signes diacritiques. Cette nouvelle représentation des mots sous forme de séquences de signes diacritiques génèrera un grand nombre de confusions. Exemple le plus simple : tous les mots ne possédant aucun signe diacritique seront équivalent à la chaîne vide, et seront impossibles à différencier. C'est le cas des classes de mots : 1245, 2015, 3046, 6022, 2183.

D'autres exemples de confusions sont donnés dans le tableau 4.5.

Séquence de diacritiques	Mots correspondants
EAFB	4092 (عين الرّحمة) , 2140 (الرديف المحطّة)
EEE	2045 (حي المهيري) , 1231 (سيدي سهيل), 7133 (سيدي مطير), 6174 (سيدي سعيد)
FE	2021 (واد اللّيل) , 4031 (سوسة الزّهور)
DAFB	7064 (بو زرّاعة) , 7153 (جرّة)

TAB. 4.5 – Exemples de confusions : une même séquence de signes diacritiques peut correspondre à différentes classes de mots.

La proportion des confusions est considérable : là où une approche basée sur l'alphabet de corps de lettres présenté dans la section 4.2 introduit une confusion sur 0,3% de la base, une approche basée uniquement sur les signes diacritiques introduit une confusion sur plus de 60% des images de la base (voir figure 4.28).

Un tel degré de confusion interdit toute utilisation autonome du reconnaisseur de signes diacritiques. Il devra nécessairement être couplé a un reconnaisseur basé sur les corps de lettres.

4.4.1.3 Reconnaissance d'une séquence de diacritiques

La forme des signes diacritiques peut varier de façon considérable selon les scripteurs. En tant que caractères de petite taille, leur forme est parfois erratique. Il est souvent difficile, voir parfois impossible de déterminer leur classe sans s'aider du contexte (la forme de la lettre à laquelle ils sont associés). C'est ce contexte qui permet de désambiguïser la lecture des signes diacritiques. Le tableau 4.6 donne quelques exemples de signes diacritiques impossibles à différentier sans le contexte.

Or dans cette première approche simplifiée, nous allons tenter de résoudre ce problème malgré tout. Pour cela, nous allons introduire un nouvel alphabet de symboles diacri-

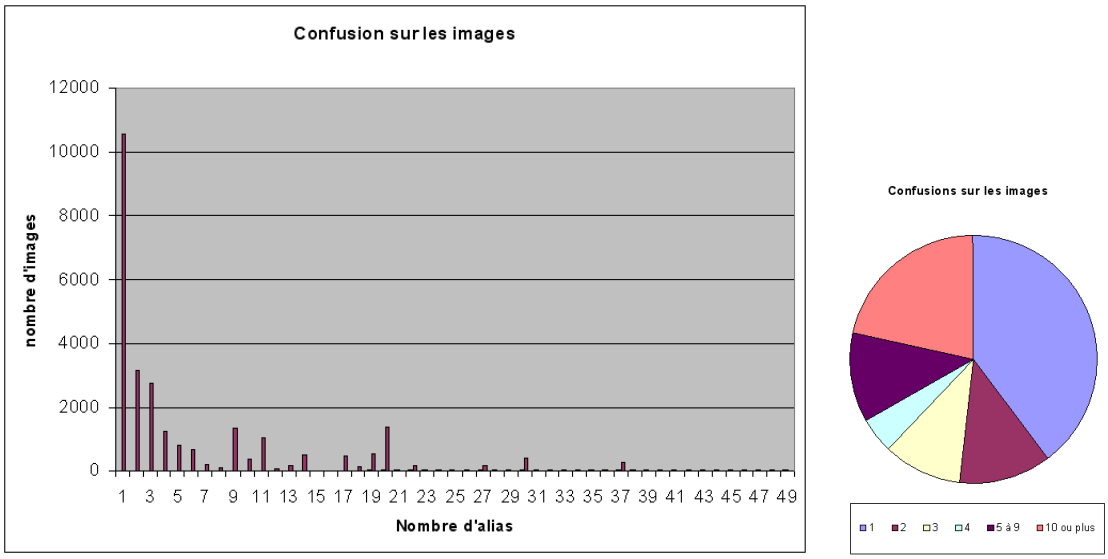


FIG. 4.28 – Répartition des confusions pour les 26459 images de la base IFN/ENIT. Seules 40% des images peuvent être classées sans confusion uniquement à partir des signes diacritiques.

Signe diacritique	classe	Image dont il est extrait
	B	ae07_020
	F	af10_021
	I	af35_037
	A	ai68_034
	B	ae69_029
	I	af34_019
	F	ai14_054
	F	aq27_017

TAB. 4.6 – Difficulté pour discriminer les classes de signes diacritiques sans avoir recours au contexte.

tiques, plus ambigus que celui défini dans la section précédente (4.4.1.2). Dans la section suivante (4.4.1.4), nous verrons comment une distance d'édition bien choisie permettra de faire le lien entre ces deux alphabets de signes diacritiques.

Ce nouvel alphabet de signes diacritiques se compose de 10 symboles :

- Su : 1 point en haut
- Du : 2 points en haut
- Tu : 3 points en haut
- Uu : forme inconnue en haut
- Cu : chadda
- Sd : 1 point en bas
- Dd : 2 points en bas
- Ud : forme inconnue en bas
- Ru : 1 ou 2 points en haut
- Rd : 1 ou 2 points en bas

Les classes Su/Sd, Du/Dd et Uu/Ud sont déterminées selon les mêmes critères de tailles et de hauteurs/largeurs que les Group1, Group2 et Group3 introduits par S. Mozaffari et al dans [130].

Ces classes sont ensuite raffinées en fonction de la taille du plus petit signe diacritique correspondant à un point (marqué Su ou Sd) :

- Si un symbole marqué Su (resp. Sd) a une aire supérieure à 2 fois l'aire du plus petit symbole marqué Su ou Sd, et que sa largeur est au moins 2 fois plus grande que l'aire du plus petit symbole marqué Su ou Sd, ce symbole est transformé en Ru (resp. Rd).
- Si un symbole marqué Uu (resp. Ud) a une aire inférieure à 2 fois l'aire du plus petit symbole marqué Su ou Sd, ce symbole est transformé en Su (resp. Sd).

A partir de ces symboles unitaires, vient ensuite la phase de regroupement des symboles selon des critères de proximité et de positions relatives :

- regroupement de 3 symboles Su pour former un symbole Tu.
- regroupement de 2 symboles Su (resp. Sd) côte à côte pour former un symbole Du (resp. Dd).
- un symbole Su situé au-dessus d'un symbole Du ou Uu donne un symbole Tu.
- un symbole Uu situté au-dessus d'un symbole Su, Du ou Tu donne un symbole Cu.

La liste des symboles ainsi constitués est parcourue de droite à gauche. Dans le paragraphe suivant, nous présentons la distance d'édition qui permet d'établir une mesure de proximité entre cette séquence et chacun des mots du vocabulaire.

4.4.1.4 Distance d'édition et alternatives

Dans cette partie, nous présentons la distance d'édition utilisée pour établir une mesure de similarité entre la séquence de diacritiques reconnue, et la séquence de diacritiques correspondant à un mot donné. La distance d'édition que nous utilisons est du même type que la distance de Levenshtein présentée dans le paragraphe 2.5.1.

Comme nous l'avons montré dans le paragraphe précédent, il est parfois impossible de déterminer la classe d'un signe diacritique lorsqu'on ne dispose pas de son contexte.

C'est la raison pour laquelle nous avons introduit un nouvel ensemble de symboles { Su, Du, Tu, Uu, Cu, Sd, Dd, Ud, Ru, Rd }.

Une mesure de similarité définie de façon empirique permet d'établir un critère de similarité entre une séquence de diacritiques reconnus et une séquence de diacritiques correspondant à une classe de mot.

- Su correspond à la classe A (un point en haut)
- Du correspond à la classe B (deux points en haut)
- Tu correspond à la classe C (trois points en haut)
- Cu correspond à la classe F (chadda)
- Sd correspond à la classe D (un point en bas)
- Dd correspond à la classe E (deux points en bas)
- Uu peut correspondre aux classes C (trois points en haut), F (chadda), G (madda) ou H (hamza en haut). Plus rarement à la classe B (deux points en haut), ou encore à la classe A (un point en haut).
- Ud correspond le plus souvent à la classe I (hamza). Plus rarement à la classe E (deux points en bas) ou encore D (un point en bas).
- Ru peut correspondre à un A (un point en haut) ou à un B (deux points en haut).
- Ru peut correspondre à un D (un point en bas) ou à un E (deux points en bas).

De plus, certains signes diacritiques sont parfois omis par les scripteurs :

- les deux points au dessus d'un ة en fin de mot
- les deux points en dessous d'un ي en fin de mot
- les hamza en haut ou en bas

Ces omissions de certains types de diacritiques peuvent également être modélisées dans la distance d'édition. Les symboles correspondants pourront avoir un coût de suppression plus faible. Pour que ce coût de suppression plus faible s'applique uniquement aux deux points qui sont au dessus des ة, on rajoute la classe de diacritique J similaire à B. Et pour que le coût de suppression plus faible ne s'applique qu'aux deux points qui sont en dessous des ي, on rajoute une classe de diacritique K similaire à E. Les coûts de suppression des symboles H, I, J, et K pourront donc être plus faibles que les autres pour modéliser la possibilité d'omission de ces symboles par les scripteurs.

Au final, la matrice de substitutions utilisée pour le calcul de la distance d'édition est donnée dans le tableau 4.7.

Par exemple, si l'annotation correspond à la séquence de symboles : "ACABAJ", et que la séquence de symboles reconnue est : "Su Tu Uu Su", la distance d'édition sera de "2,3" (suppression du symbole A et du symbole J et substitution de B par Uu).

En revanche, si on accorde une tolérance sur la suppression des deux points au dessus

	Su	Du	Tu	Uu	Cu	Sd	Dd	Ud	Ru	Rd
A	0	0,5	1	0,5	1	1	1	1	0,3	1
B	0,5	0	1	0,3	1	1	1	1	0,3	1
C	1	1	0	0,1	1	1	1	1	1	1
D	1	1	1	1	1	0	0,5	0,3	1	0,1
E	1	1	1	1	1	0,5	0	0,3	1	0,1
F	1	0,1	1	0,1	0	1	1	1	1	1
G	1	0,1	1	0,1	1	1	1	1	1	1
H	1	0,3	1	0,1	1	1	1	1	1	1
I	1	1	1	1	1	1	0,3	0,1	1	1
J	0,5	0	1	0,3	1	1	1	1	0,3	1
K	1	1	1	1	1	0,5	0	0,3	1	0,1

TAB. 4.7 – Matrice de coûts de substitutions utilisée pour le calcul de la distance d'édition entre séquences de diacritiques. Par défaut, les coûts d'insertion et de suppression sont égaux à 1.

des ـَـة ـًـة, et qu'on fixe ce coût à "0,1", la distance d'édition entre les deux chaînes précédentes devient "1,4".

Ce coût de "0,1" est fixé de façon empirique. Néanmoins, les expériences rapportées dans le paragraphe 4.4.1.6 montrent que le fait de prendre en compte les omissions de cette manière ne permet pas d'améliorer les performances.

4.4.1.5 Combinaison et résultats

Comme nous l'avons vu dans le paragraphe 4.4.1.2, un système uniquement basé sur les signes diacritiques ne permet pas de prendre une décision. Il doit être couplé au reconnaisseur de corps de lettres. L'utilisation des signes diacritiques devra permettre de faire passer en tête la bonne réponse parmi les 10 meilleurs candidats fournis par le reconnaisseur de corps de lettres. Le taux de reconnaissance en 10ème position étant de 97,5% sur la base d'entraînement et 96,4% sur la base de test (voir section 4.2.4.2), ce chiffre constitue la borne supérieure de ce que l'on peut obtenir dans le cas idéal.

Une combinaison trop naïve dégrade les performances. Ainsi, sur la base d'apprentissage, si l'on considère que résultat de la combinaison répondra systématiquement la classe dont la distance d'édition est la plus faible parmi les 10 candidats renvoyés par le reconnaisseur de corps de lettres, cette combinaison permet de corriger 903 erreurs, mais introduit 2409 nouvelles erreurs, soit un différentiel négatif de -1506 documents.

Quatre variables semblent pertinentes pour fixer un seuil permettant de limiter le nombre d'apparition de nouvelles erreurs, en posant les hypothèses suivantes :

- Hypothèse 1 : on considère le coût de la meilleure distance d'édition de la séquence de diacritiques parmi les 10 candidats. On suppose que si le coût d'appariement est trop important, le reconnaisseur de diacritique n'est pas fiable. Il ne faut alors pas modifier la liste retournée par le reconnaisseur de corps de lettres.
- Hypothèse 2 : on considère le score de confiance du premier élement de la liste rendue par le reconnaisseur de corps de lettres. On suppose que si ce score est faible, le reconnaisseur de corps de lettres est peu fiable, et on peut donc faire confiance au reconnaisseur de diacritiques pour sélectionner le meilleur candidat.

– Hypothèse 3 : on considère le rang de la classe renvoyée par le reconnaisseur de diacritiques. Cette hypothèse suppose que l'hypothèse n°1 est fausse : on suppose que la valeur de la distance d'édition n'est pas un critère absolu : dans certains cas une valeur de distance d'édition faible n'est pas un bon critère car de nombreux candidats ont une distance d'édition encore plus faible. Et a contrario, dans d'autres cas, un candidat peut être l'un des tous premiers de la liste tout en ayant une valeur de distance d'édition plus importante. Dans ce cas, le rang du candidat retenu pourrait être un critère plus pertinent que le coût.
– Hypothèse 4 : on considère le rang du candidat retenu dans la liste de reconnaissance de corps de lettres. Cette hypothèse postule que la profondeur de liste considérée (10 candidats) est trop importante. Il faut se limiter à une profondeur plus faible car au delà d'un certain indice la combinaison devient erratique.

L'analyse de ces quatre critères est faite en annexe C.1. Les résultats montrent que seule la deuxième hypothèse est pertinente. Comme le montre le tableau 4.8, un seuil sur cette grandeur permet d'obtenir une combinaison qui améliore notablement les performances.

Le reconnaisseur de corps de lettres retourne une liste de 10 candidats dont les scores de confiance correspondent aux vraisemblances normalisées (scaled-likelihoods) du décodage des mots par les HMM de corps de lettres. Un score élevé pour le premier candidat est souvent synonyme de reconnaissance fiable. Dans ce cas, il vaut mieux faire confiance au reconnaisseur et ignorer le résultat de la séquence de diacritiques. En revanche, lorsque le score est faible, la reconnaissance n'est pas fiable, et dans ce cas les signes diacritiques peuvent apporter de l'information.

Le seuil optimal se situe autour de 0.65 sur la base d'entraînement, et 0.85 sur la base de validation. Un seuil plus prudent fixé entre 0.7 et 0.75 permet d'obtenir 90,35% de reconnaissance sur l'ensemble de test (le sous-ensemble D). La prise en compte des signes diacritiques par cette technique permet donc de passer de 87,2% à 90,35% de reconnaissance, ce qui constitue une amélioration intéressante (réduction de 25% du taux d'erreur).

En fixant ce seuil à 0.7, on réitère l'opération précédente pour évaluer la pertinence des trois autres hypothèses. Mais là encore, leur apport n'est pas concluant. Les détails de cette analyse sont donnés dans l'annexe C.2.

4.4.1.6 Omissions courantes et inversion de chadda

Dans le paragraphe 4.4.1.4, nous avons introduit différents types d'omissions.

Dans cette partie, nous testons les omissions de { ي ة ة } seuls, des hamza seuls { ء

	Score	corrections	nouvelles erreurs	différentiel	Taux reco
Train	0.1	0	0	0	90,3
	0.15	5	0	5	90,3
	0.2	18	1	17	90,4
	0.25	43	4	39	90,5
	0.3	93	11	82	90,7
	0.35	159	15	144	91,0
	0.4	226	24	202	91,3
	0.45	306	40	266	91,6
	0.5	389	60	329	92,0
	0.55	466	81	385	92,3
	0.6	532	112	420	92,4
	0.65	**579**	**143**	**436**	**92,5**
	0.7	623	190	433	92,5
	0.75	669	237	432	92,5
	0.8	723	295	428	92,5
	0.85	766	360	406	92,4
	0.9	804	479	325	91,9
	0.95	848	600	248	91,6
	1.0	903	2409	-1506	82,7
Test	0.1	0	0	0	87,2
	0.15	5	0	5	87,3
	0.2	14	1	13	87,4
	0.25	27	2	25	87,6
	0.3	42	3	39	87,8
	0.35	65	7	58	88,1
	0.4	97	10	87	88,5
	0.45	130	15	115	88,9
	0.5	163	29	134	89,2
	0.55	201	36	165	89,7
	0.6	225	50	175	89,8
	0.65	251	62	189	90,0
	0.7	276	70	206	90,3
	0.75	299	84	215	90,4
	0.8	322	98	224	90,5
	0.85	**344**	**119**	**225**	90,5
	0.9	356	162	194	90,1
	0.95	377	206	171	89,7
	1.0	411	781	-370	81,7

TAB. 4.8 – Analyse du score de la meilleure réponse fournie par le reconnaisseur de corps de lettres comme critère de seuil de la combinaison avec le reconnaisseur de diacritiques.

}, et des deux à la fois.

Parallèlement, notons que le symbole chadda peut être superposé verticalement avec un autre signe diacritique. Etant donné la variabilité de l'écriture manuscrite, ce symbole peut être décalé vers la droite ou vers la gauche. Sans information contextuelle (corps de lettres auxquelles ce symboles sont associés), il est parfois impossible de déterminer avec certitude l'ordonnancement des symboles de droite à gauche. Pour régler ce problème, nous proposons de traduire systématiquement les mots qui contiennent une chadda de deux manières différentes lorsque cette chadda est associée à une lettre qui possède déjà un signe diacritique.

Par exemple, la classe de mots 6036 (كتّانة) sera traduit à la fois en "FBAJ" et en "BFAJ". Chacune de ces deux chaînes sera utilisée pour le calcul de la distance d'édition.

	Sans inversion de chadda					Avec inversion de chadda				
	seuil	aucun	ة ي	ء	ة ي ء	seuil	aucun	ة ي	ء	ة ي ء
Train	0.65	436	408	435	409	0.65	446	422	444	422
	0.7	433	399	431	403	0.7	443	413	440	416
	0.75	432	394	430	398	0.75	**445**	410	442	414
	0.80	428	389	427	393	0.80	443	407	441	411
	0.85	406	362	367	426	0.85	426	382	424	387
	seuil	aucun	ة ي	ء	ة ي ء	seuil	aucun	ة ي	ء	ة ي ء
Test	0.65	189	176	189	178	0.65	196	188	195	189
	0.7	206	190	206	192	0.7	213	202	212	203
	0.75	215	193	216	196	0.75	221	205	221	207
	0.80	224	199	225	202	0.80	**230**	213	230	215
	0.85	225	197	227	200	0.85	231	213	232	215

TAB. 4.9 – Analyse différentiel nouvelles correction - nouvelles erreurs, en fonction de la tolérance aux omissions et à l'inversion des chadda.

Comme le montre le tableau 4.9, l'inversion des chadda dans la séquence de diacritiques permet d'améliorer légèrement le différentiel des nouvelles corrections moins les nouvelles erreurs, et améliore donc les performances. En revanche, on constate que le fait d'attribuer un coût de suppression plus faible à certains symboles de diacritiques (H, I, J, et K qui correspondent respectivement à Hamza haut, Hamza bas, deux points haut de ة , et deux points bas de ي) dégrade les performances.

	1ère pos	2ème pos	10ème pos
Apprentissage {a,b,c} sans diacritiques	90.3	94.1	97.5
Apprentissage {a,b,c} avec diacritiques (seuil 0.8)	92.5	95.5	97.5
Test {d} sans diacritiques	87.2	91.9	96.4
Test {d} avec diacritiques (seuil 0.8)	90.6	93.8	96.4

TAB. 4.10 – Apport des signes diacritiques. Les résultats du système "sans diacritiques" sont ceux du tableau 4.3 de la section 4.2.4.2.

Un seuil bien adapté (0.8) et l'inversion des chadda dans la séquence de diacritiques permet de réduire le nombre d'erreurs de 230 sur la base de test par rapport au système

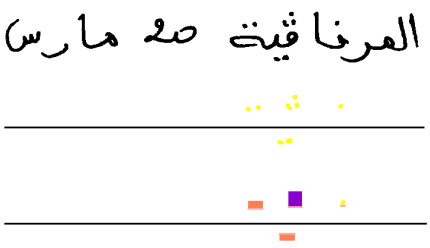


FIG. 4.29 – Détection des points et regroupements.
Le parcours de la séquence de droite à gauche donne : Su Tu Dd Du

FIG. 4.30 – Détection des points et regroupements.
Le parcours de la séquence de droite à gauche donne : Ru Su Du Sd Su Ru Uu Dd Ru Du

initial, ce qui entraine un taux de reconnaissance de 90,6%, soit un taux de réduction d'erreur de 26,6%.

Ce seuil permet de réduire le nombre d'erreurs de 443 sur la base d'entraînement, et implique donc un taux de reconnaissance de 92,5%, soit un taux de réduction d'erreur de 22,7%.

Ces résultats sont comparés à ceux du système précédent (sans diacritiques) dans le tableau 4.10.

4.4.1.7 Exemples

Les figures 4.29 à 4.32 illustrent l'extraction des signes diacritiques et leur étiquetage. Dans chacune de ces figures sont indiquées :

- Première ligne : image initiale.
- Deuxième ligne : points détectés.
- Troisième ligne : points regroupés.

La figure 4.33 illustre la combinaison entre le reconnaisseur de corps de lettres et le reconnaisseur de signes diacritiques.

4.4.1.8 Influence de la combinaison sur le rejet

Dans la section 4.4.1.5, nous avons montré que la combinaison entre le reconnaisseur de corps de lettres et le reconnaisseur de diacritiques permettait d'améliorer le taux de reconnaissance. Cette amélioration provient du différentiel entre les corrections apportées et les nouvelles erreurs produites. Nous avons vu que sur la base de test, la

FIG. 4.31 – La chadda est correctement étiquetée.
Le parcours de la séquence de droite à gauche donne : Su Cu Su Ud Uu

FIG. 4.32 – La hamza collée au corps de texte n'est pas extraite dans la liste des signes diacritiques.
Le parcours de la séquence de droite à gauche donne : Uu Sd Su Su Su

valeur optimale était obtenue pour un seuil de 0,8. En revanche, l'étude des courbes de reconnaissance/substitution montrent que l'utilisation d'un seuil plus élevé permet d'améliorer significativement les résultats pour les taux de substitution faibles.

Dans la section 4.2.4.2, nous avons vu que la mesure de score du premier élément était pertinente, car la proportion d'erreurs de classification est faible lorsque le score de confiance est élevé. Si la combinaison avec le reconnaisseur de diacritiques contredit le meilleur candidat de la liste de reconnaissance sur les corps de lettres, le score de cette combinaison est faible, qu'il s'agisse d'une correction ou d'une nouvelle erreur. En d'autres termes, le reconnaisseur de diacritiques agit comme un filtre : si le reconnaisseur de diacritiques contredit le résultat du reconnaisseur de corps de lettres, le document ne sera plus comptabilisé dans les scores élevés.

Ce phénomène est visible sur la figure 4.34. Cette figure illustre bien le phénomène décrit ci-dessus : plus la combinaison avec les signes diacritiques est appliquée sur les documents de score élevés, et plus la proportion de documents ayant un score élevé après combinaison est faible.

Une valeur de seuil élevée "filtre" davantage de documents. Ce filtrage écarte des documents qui étaient bien reconnus (nouvelles erreurs), mais filtre également des documents erronés (soit en les corrigeant, soit en les remplaçant par une nouvelle erreur, mais dans tous les cas en leur attribuant un score faible). Au final, le degré de pureté est amélioré pour les scores élevés qui n'ont pas été filtrés par le reconnaisseur de diacritiques. C'est ce qu'indique la courbe figure 4.35. Lorsque le seuil devient trop élevé, tous les documents sont concernés, et la proportion de documents filtrés alors qu'ils étaient correctement reconnus devient trop importante par rapport à la proportion de

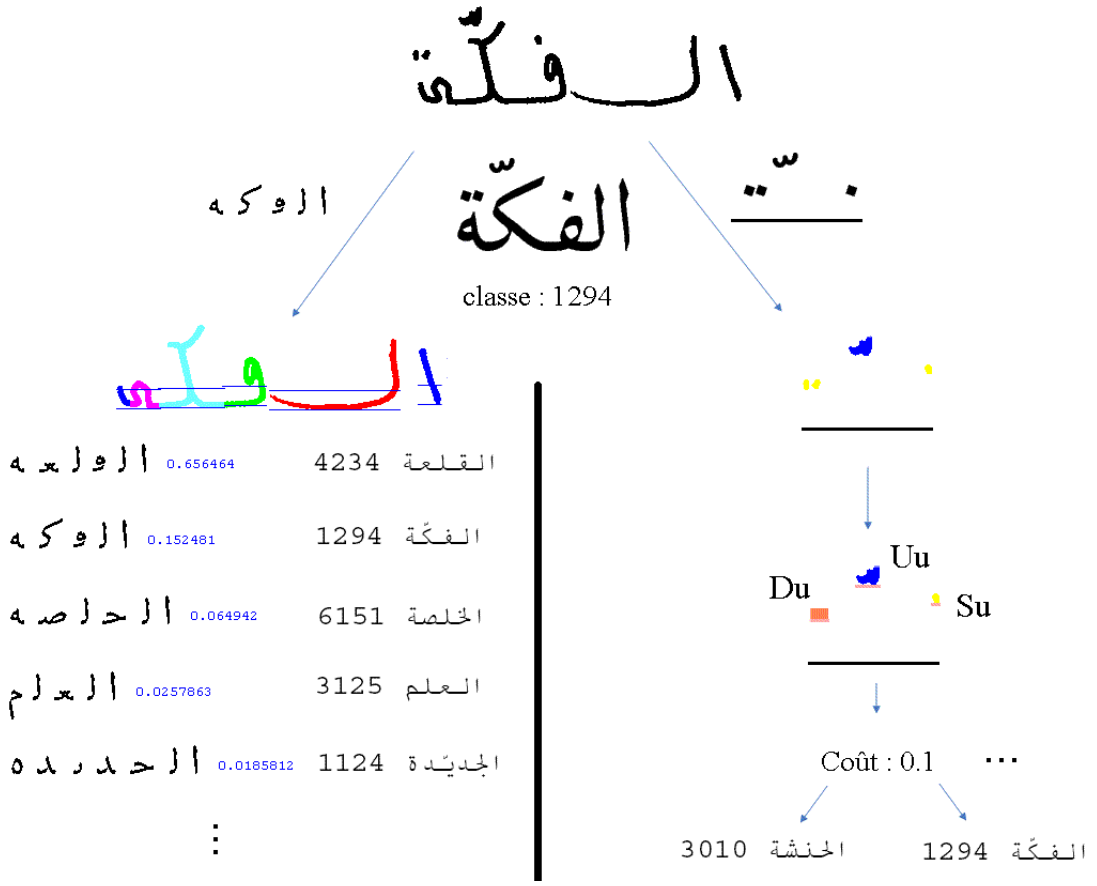


FIG. 4.33 – Combinaison entre la reconnaissance des corps de lettres et la reconnaissance des signes diacritiques pour le mot الفكّة.
Première ligne : le mot à reconnaître.
Deuxième ligne : les réponses attendues.
Colonne de gauche : résultat de la reconnaissance des corps de lettres.
Colonne de droite : résultat de la reconnaissance des diacritiques. Seul le coût d'édition le plus faible (0.1) est représenté.

documents filtrés alors qu'ils étaient mal reconnus.

Au final, cette méthode permet de passer de 69% à 75% de reconnaissance pour 1% de substitution.

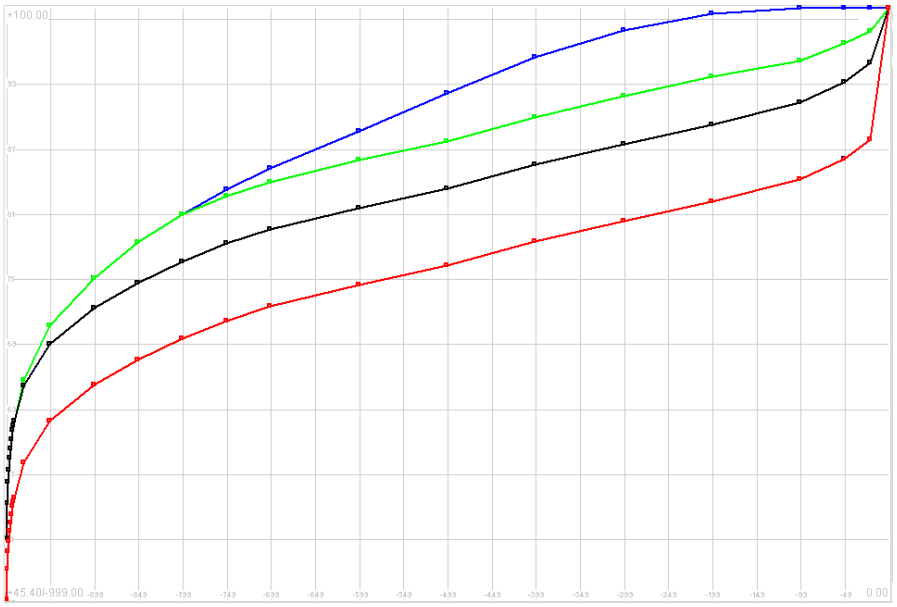


FIG. 4.34 – Evolution du taux de lecture en fonction du score de confiance. Le score est donné en abscisse, par valeurs décroissantes de 0.99 à 0.0. La courbe indique la proportion de documents ayant un score de confiance supérieur au seuil donné.
Courbe bleue : reconnaissance des corps de lettres seuls, sans combinaison avec les signes diacritiques (résultat 4.2.4.2)
Courbe verte : combinaison avec un seuil à 0.8 (optimum du taux de reconnaissance global)
Courbe noire : combinaison avec un seuil à 0.98
Courbe rouge : combinaison avec un seuil à 0.99 (tous les documents)

4.4.1.9 Amélioration du reconnaisseur de corps de lettres

Lorsqu'on améliore les performances du reconnaisseur de corps de lettres, la combinaison continue d'apporter de l'information. On aurait pu craindre que l'amélioration du premier étage du système (le reconnaisseur de corps de lettres) se traduise par une stagnation des performances en sortie (après combinaison), réduisant ainsi l'intérêt d'une combinaison avec les signes diacritiques. Il est intéressant de constater que ce phénomène ne se produit pas.

En faisant varier les topologies du réseau de neurones du système hybride (nombres de neurones sur la couches cachée et nombre de sorties), et en testant différentes initialisations, on parvient à améliorer les performances du reconnaisseur donné dans la section 4.2.4.2. Des tableaux récapitulatifs de ces expériences sont donnés en annexe B.

Ainsi, un nouveau système hybride RN + MMC avec 200 classes d'observations et 650 neurones sur la couche cachées permet d'obtenir un taux de reconnaissance de 89.97% de

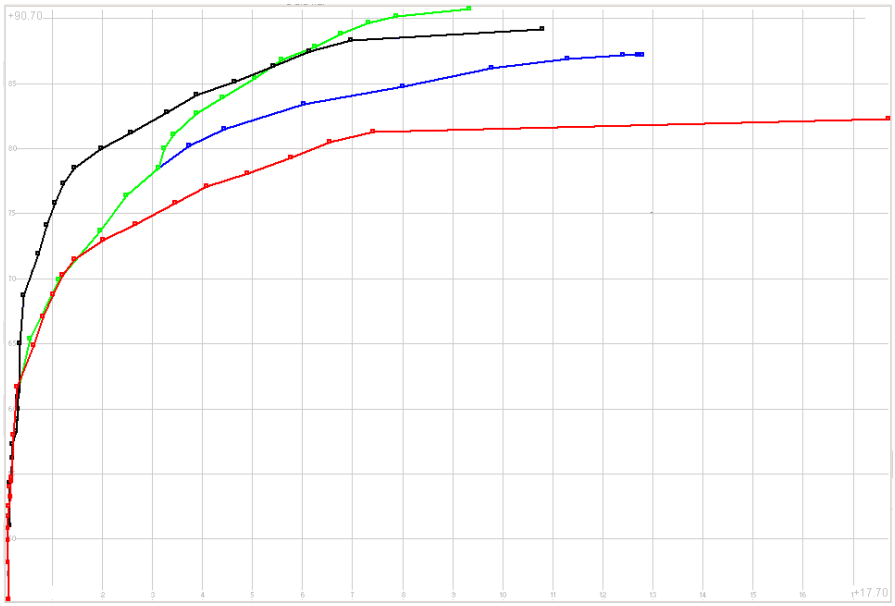


FIG. 4.35 – Evolution du taux de reconnaissance en fonction du taux de substitution.
Courbe bleue : reconnaissance des corps de lettres seuls, sans combinaison avec les signes diacritiques
Courbe verte : combinaison avec un seuil à 0.8 (optimum du taux de reconnaissance global)
Courbe noire : combinaison avec un seuil à 0.98
Courbe rouge : combinaison avec un seuil à 0.99 (tous les documents)

reconnaissance là où le système précédent (système hybride RN + MMC avec 100 classes d'obversations et 500 neurones sur la couche cachée) permettait d'obtenir un taux de reconnaissance de 87,24% (résultats présentés dans la section 4.2.4.2).

Nous appliquons la même stratégie de combinaison avec les signes diacritiques que celle décrite dans les paragraphes précédents. Le tableau 4.11 indique l'évolution du taux de reconnaissance sur la base d'apprentissage (ABC) et sur la base de test (D), pour le nouveau système, en fonction du seuil de combinaison. Le seuil optimal de combinaison est obtenu pour un score de confiance de 0.7, à la fois sur la base d'apprentissage et sur la base de test. La combinaison avec le reconnaisseur de diacritiques permet d'obtenir une réduction du taux d'erreurs de 18,8% sur la base d'apprentissage (on passe de 92,92% à 94,25% de reconnaissance), et de 24,9% sur la base de test (on passe de 89,98% à 92,47% de reconnaissance).

La figure 4.36 indique l'influence du seuil sur les courbes reconnaissance/substitution. Comme dans la section 4.4.1.8, la courbe optimale pour 1% de substitution est obtenue pour un seuil de confiance de 0.98

Pour 1% de substitution, on passe ainsi de 75,1% de reconnaissance avec le système sans diacritiques, à 80,7% de reconnaissance pour la combinaison avec un seuil de 0.98

Le tableau 4.12 compare notre système avec les principaux autres systèmes ayant publié suivant cette procédure d'évaluation (apprentissage sur l'ensemble {A, B, C}, test sur l'ensemble { D }).

4.4.1.9.1 Performances sur le sous-ensemble E

Ici, nous donnons les performances en reconnaissance sur l'ensemble E à titre indicatif. Notre système a été entraîné sur l'ensemble {A,B,C}. Ces résultats ne sont pas directement comparables avec ceux de la compétition ICDAR 2005 (voir section 4.3.1), pour laquelle les systèmes étaient entraînés sur {A,B,C,D}. Ils ne sont pas non plus comparables avec ceux de la compétition ICDAR 2007 (voir section 4.3), pour laquelle les systèmes étaient entraînés sur {A,B,C,D,E} et testés sur une autre sous-base.

Nous pouvons néanmoins étudier l'influence de la combinaison avec le reconnaisseur de diacritiques, avec l'ancien système (présenté dans la section 4.2.4.2), qui sera noté I ; et le nouveau (présenté dans la section 4.4.1.9), qui sera noté II.

Les différents systèmes de l'état de l'art présentent généralement une baisse de 7 à 15% de reconnaissance entre les performances annoncées sur la base D et sur le base E, même lorsque ces dernières sont utilisées en apprentissage comme c'était le cas lors de la compétition ICDAR 2007 [112].

Nous constatons le même phénomène.

Le tableau 4.13 donne l'évolution du taux de reconnaissance en fonction de seuil de combinaison avec le reconnaisseur de diacritiques. Le seuil optimal est obtenu pour 0.75, ce qui est proche des 0.70 qu'on obtenait dans le tableau 4.11.

Le tableau 4.14 compare les performances du système I et du système II, avec et sans combinaison avec le reconnaisseur de signes diacritiques, en rappelant les taux de reconnaissance sur la base {D} et en donnant les taux obtenus sur la base {E}.

	Score	corrections	nouvelles erreurs	différentiel	Taux reco
Train	0.1	0	0	0	92,92
	0.15	1	0	1	92,92
	0.2	13	0	13	92,98
	0.25	31	3	28	93,06
	0.3	56	6	50	93,17
	0.35	87	11	76	93,30
	0.4	134	26	108	93,46
	0.45	179	31	148	93,67
	0.5	224	45	179	93,82
	0.55	275	64	211	93,99
	0.6	330	91	239	94,13
	0.65	370	114	256	94,22
	0.7	**408**	**146**	**262**	**94,25**
	0.75	448	205	243	94,15
	0.8	495	266	229	94,08
	0.85	532	332	200	93,93
	0.9	561	459	102	93,43
	0.95	586	583	3	92,93
	1.0	612	2507	-1895	83,31
Test	0.1	0	0	0	89,98
	0.15	0	1	-1	89,96
	0.2	4	1	3	90,02
	0.25	16	2	14	90,19
	0.3	41	7	34	90,48
	0.35	62	9	53	90,76
	0.4	87	10	77	91,12
	0.45	110	18	92	91,34
	0.5	140	20	120	91,76
	0.55	166	31	135	91,98
	0.6	189	38	151	92,22
	0.65	208	48	160	92,35
	0.7	**228**	**60**	**168**	**92,47**
	0.75	237	75	162	92,38
	0.8	249	105	144	92,12
	0.85	270	132	138	92,03
	0.9	282	175	107	91,57
	0.95	295	224	71	91,03
	1.0	312	827	-515	82,33

TAB. 4.11 – Evolution du taux de reconnaissance sur la base d'apprentissage et sur la base de test en fonction du seuil.

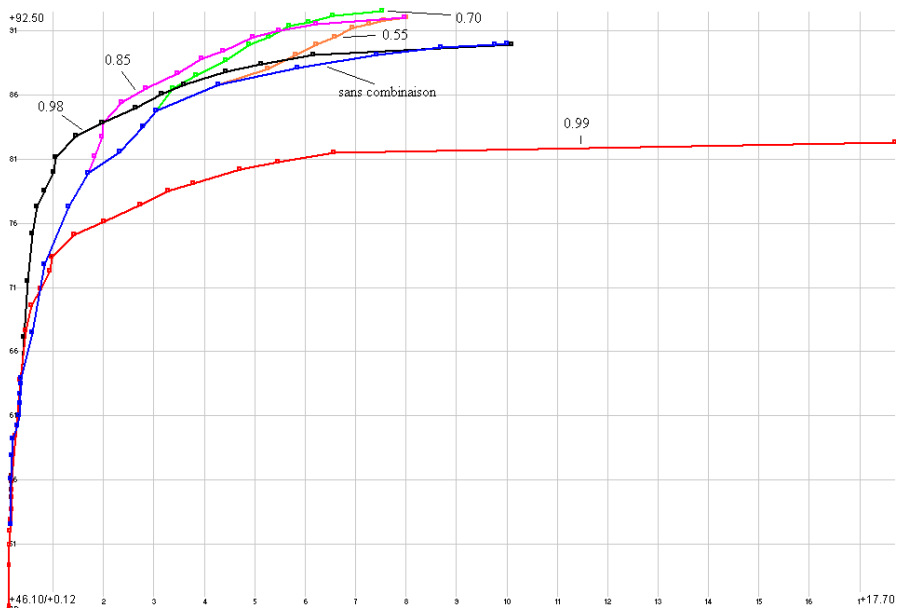


FIG. 4.36 – Evolution du taux de reconnaissance en fonction du taux de substitution.
Courbe bleue : reconnaissance des corps de lettres seuls, sans combinaison avec les signes diacritiques
Courbe orange : combinaison avec un seuil à 0.55
Courbe verte : combinaison avec un seuil à 0.70 (optimum du taux de reconnaissance global)
Courbe mauve : combinaison avec un seuil à 0.85
Courbe noire : combinaison avec un seuil à 0.98 (optimum du point de fonctionnement pour 1% de substitution)
Courbe rouge : combinaison avec un seuil à 0.99 (tous les documents)

	1ère pos	2ème pos	10ème pos
Apprentissage {a,b,c} sans diacritiques	92.92	95.60	98.15
Apprentissage {a,b,c} avec diacritiques (seuil 0.7)	94.25	96.37	98.15
Test {d} sans diacritiques	89.98	93.54	97.45
Test {d} avec diacritiques (seuil 0.7)	**92.47**	**94.76**	**97.45**
UOB [4] {d}	90.96	92.95	94.44
ARAB-IFN [148] {d}	89.1	91.7	95.9
SCHMMs [24] {d}	89.79	92.25	96.78
Microsoft Research [2] {d}	88.94		95.01

TAB. 4.12 – Apport des signes diacritiques et comparaison avec les autres systèmes de l'état de l'art.

	Score	corrections	nouvelles erreurs	différentiel	Taux reco
Test E	0.1	0	0	0	79,81
	0.15	2	0	2	79,84
	0.2	9	0	9	79,96
	0.25	26	1	25	80,23
	0.3	53	3	50	80,64
	0.35	90	8	82	81,17
	0.4	119	21	98	81,44
	0.45	145	31	114	81,70
	0.5	185	45	140	82,13
	0.55	217	60	157	82,41
	0.6	252	71	181	82,81
	0.65	277	86	191	82,98
	0.7	302	112	190	82,96
	0.75	**331**	**125**	**206**	**83,23**
	0.8	342	147	195	83,04
	0.85	358	179	179	82,78
	0.9	373	215	158	82,43
	0.95	395	279	116	81,73
	1.0	412	1032	-620	69,53

TAB. 4.13 – Evolution du taux de reconnaissance sur la base de test {E} en fonction du seuil.

	1ère pos	2ème pos	10ème pos	% reduc err
I. Test {d} sans diacritiques	87.24	91.88	96.41	
I. Test {d} avec diacritiques (seuil 0.7)	90.30	93.51	96.41	24
I. Test {e} sans diacritiques	76.99	84.28	93.57	
I. Test {e} avec diacritiques (seuil 0.7)	80.74	86.46	93.57	16.3
II. Test {d} sans diacritiques	89.98	93.54	97.45	
II. Test {d} avec diacritiques (seuil 0.7)	92.47	94.76	97.45	24.9
II. Test {e} sans diacritiques	79.81	86.60	94.51	
II. Test {e} avec diacritiques (seuil 0.7)	82.96	88.31	94.51	15.6

TAB. 4.14 – Comparaison du système I (présenté dans la section 4.2.4.2 et du système II (présenté dans la section 4.4.1.9), tous deux entraînés sur {A,B,C}, testés sur {D} et sur {E} avec et sans combinaison avec le reconnaisseur de diacritiques. La dernière colonne donne le pourcentage de réduction du taux d'erreur en première position.

4.4.1.10 Conclusion et prolongements

Nous avons présenté une approche qui permet de combiner avantageusement le système de reconnaissance sans diacritiques présenté dans la partie 4.1. Cette combinaison permet d'obtenir l'un des meilleurs systèmes publiés sur cette base en terme de taux de reconnaissance (voir tableau 4.12). Dans l'optique de l'utilisation d'un système intégrant le rejet, cette technique permet également d'améliorer le taux de reconnaissance obtenu pour les taux de substitution faibles, malgré une baisse du taux de reconnaissance du même système utilisé sans rejet.

Dans la section 4.4.1.9, nous avons également montré que la combinaison entre le reconnaisseur de corps de lettres et le reconnaisseur de diacritiques permet une réduction significative du taux d'erreur, même lorsque le reconnaisseur de corps de lettres est amélioré. L'amélioration du premier étage du système se traduit par une amélioration globale des performances après combinaison.

L'une des pistes d'améliorations possibles de ce système serait d'entraîner les coûts de la distance d'édition, qui sont pour l'instant fixés de façon heuristique. Ces poids pourraient être adaptés à l'aide de la base d'apprentissage, en utilisant des techniques comme celles mises en oeuvre par M. Sebban et al. [25, 141, 26, 27].

Néanmoins, cette approche trouve ses limites dans le fait qu'elle n'exploite pas le contexte (les corps de lettres environnantes), et n'exploite pas les distances entre les différents signes diacritiques (on traite une séquence, sans modéliser la distance inter-diacritiques). Autre défaut de cette approche, le fait de regrouper des diacritiques élémentaires (les points) pour former des diacritiques plus complexes (les couples ou triplets de points) est une décision basée sur des seuils. C'est une décision brutale qui est prise au début de la chaîne, et qui peut être sources d'erreurs. Il pourrait être utile d'examiner plusieurs alternatives en parallèle pour éviter de prendre une mauvaise décision au début de la chaîne.

Une approche basée sur les automates à états finis pondérés pourrait combiner tous ces avantages. Nous proposons une étude théorique de cette approche dans la section suivante.

4.4.2 Approche Automates à états finis pondérés sur pseudo-mots

Dans cette partie, l'analyse est uniquement théorique. Pour des raisons de temps, cette partie n'a pas pu être expérimentée. Nous ne présenterons donc aucun résultat appuyé sur des chiffres. Néanmoins, nous présentons ici un formalisme prometteur qui permettra de redistribuer les signes diacritiques sur les formes de lettres, tout en intégrant des contraintes morphologiques propres à l'écriture arabe.

L'automate des mots bien formés que nous introduisons ici constitue également une piste de réflexion pour apporter une solution au problème de segmentation en mots.

Dans un mot, les lettres respectent certaines contraintes morphologiques. Ces contraintes (découpage en pseudo mots) dépendent de la forme des lettres plutôt que de l'information portée par les signes diacritiques. Les lettres د et ذ sont de la même forme, et elles suivent le même comportement : ces deux lettres ne se lient pas avec la lettre suivante, et provoquent la création d'un nouveau pseudo-mot. Idem pour les lettres ر et ز.
Autre exemple, les lettres ص et ض sont de la même forme, et elles suivent également un comportement similaire : elles se lient toutes les deux avec la lettre suivante, et leur forme isolée/en fin de mot est identique : ص et ض respectivement.
Idem pour le triplet { ج ، ح ، خ } qui devient { ج ، ح ، خ }.

En intégrant les contraintes morphosyntaxiques, il est donc possible de construire un graphe des pseudo-mots bien formés à partir des formes élémentaires.

- Dans un premier temps, exclure les signes diacritiques. Ils seront utilisés ultérieurement.
- A partir d'une segmentation en graphèmes, construire un graphe de segmentation qui correspond aux différentes façons d'assembler les graphèmes.
- Un reconnaisseur de corps de lettres construit un graphe de reconnaissance pour lequel chaque arc a une probabilité d'appartenir à une classe de forme.
- Faire une composition entre le graphe de reconnaissance de formes, et le graphe des pseudo-mots bien formés, afin d'exclure tous les chemins qui ne correspondent pas à des interprétations valides d'un point de vue morphologique.
- Réaliser plusieurs hypothèses de reconnaissance des diacritiques (détection des hamza, chadda, comptage du nombre de points). Ces hypothèses sont représentées sous forme d'un graphe transducteur.
- Plusieurs hypothèses de redistribution des diacritiques sur les formes sont alors possibles, en respectant à la fois les contraintes de proximité et les contraintes de la langue arabe (il n'est pas possible d'associer n'importe quel couple de diacritique et de corps de lettre). Ces associations sont réalisées par le biais d'une composition

entre le graphe des hypothèses de corps de lettres et le graphe transducteur des hypothèses de diacritiques.
– Le résultat de cette transduction est un graphe. La réponse du système est déterminée par le calcul du meilleur chemin dans ce graphe.

Sur un exemple :

Tunis : تونس

En supprimant les diacritiques : بوںس

La segmentation est donnée figure 4.37

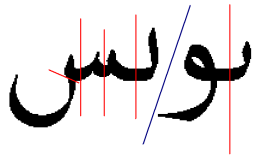
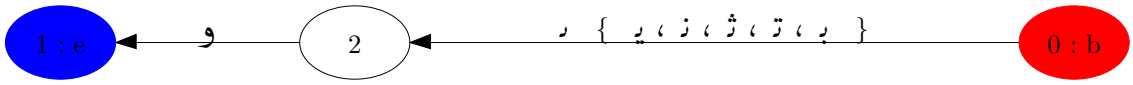
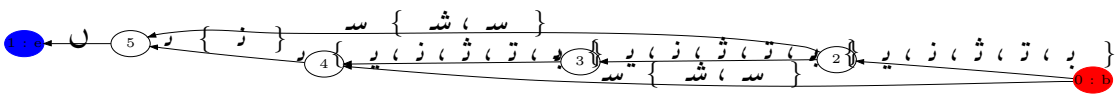


FIG. 4.37 – Segmentation graphème : en bleu les points de coupures inter-pseudo-mots, en rouge les points de coupures de inter-graphèmes.

Pour chaque pseudo-mot, on applique le reconnaisseur de formes, et on réalise la composition du résultat avec le graphe des pseudo-mots bien formés. On obtient respectivement :



et



Dans le premier graphe, la forme د (lettres { د ، ذ }) est rejetée puisqu'elle imposerait une coupure entre la première et la deuxième lettre du mot.

Dans le deuxième graphe, une succession de 3 ر (ر ـر ـر) peut donner lieu à un سـ
Par ailleurs, la forme qui relie les noeuds 4 à 5 est un ر mais la seule lettre valide associée à cette forme est un ز : la lettre suivante est un ن, parmi toutes les lettres qui sont associées à la forme ر, seul le ز peut être suivi d'un ن. Les autres éventualités sont élaguées au moment de la composition avec le graphe des pseudo-mots bien formés (voir graphe suivant).

Ce qui signifie qu'à cette étape, avant d'avoir recours à l'information portée par les diacritiques, les mots acceptés sont toutes les combinaisons de ces deux graphes pour les lettres autorisées (lettres entre accolades).

بوسن , نوسن ,ثوبيتن , يوبس , etc ...
Remarque : le 3ème mot prend une forme différente avec cette police de caractères, mais en manuscrit on peut s'attendre à voir tous ces mots prendre une forme très proche.

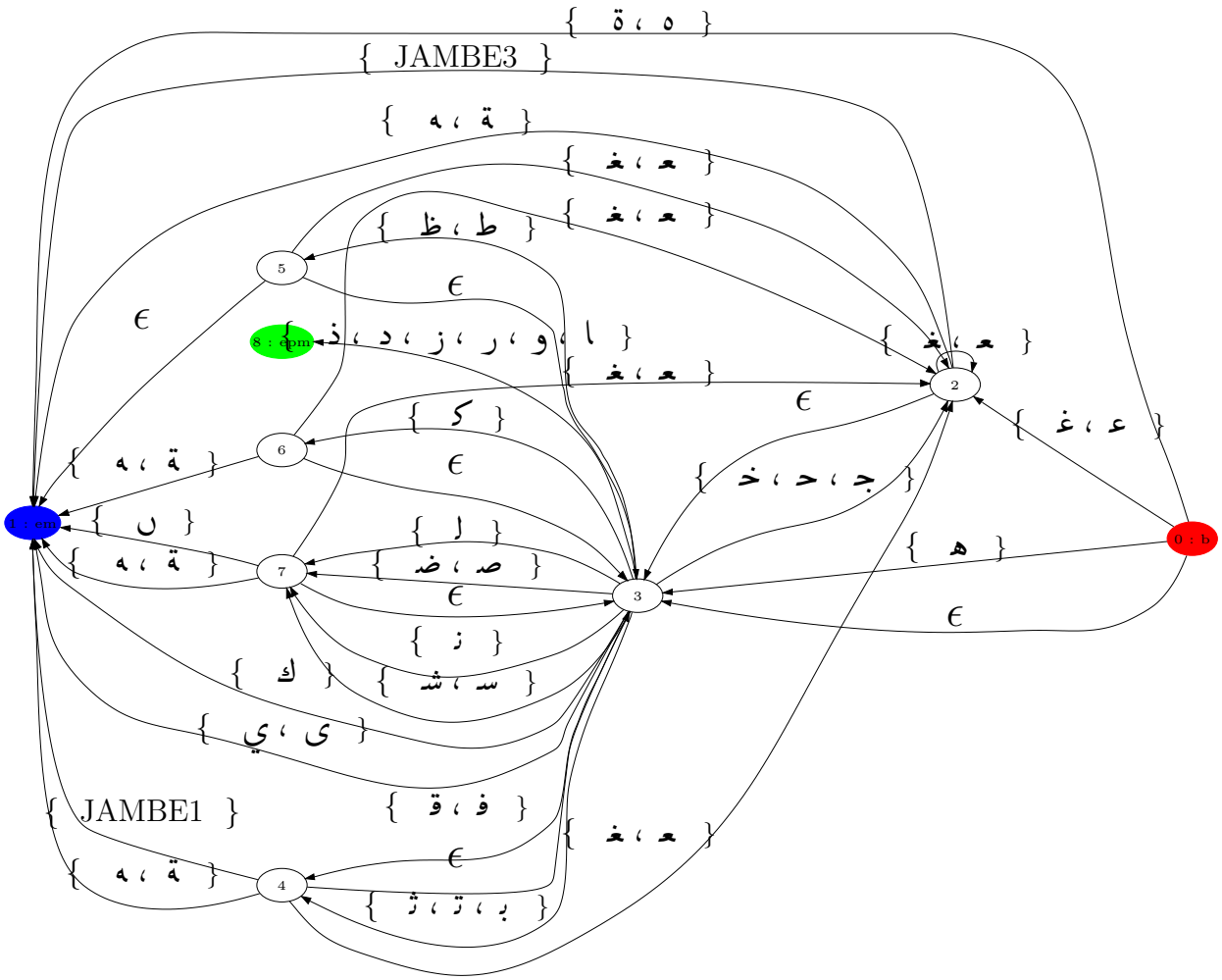
Reste alors à distribuer les points sur les formes en choisissant parmi les lettres entre accolades. Cette redistribution peut se faire par le biais d'une composition entre le graphe des hypothèses de lettres, et le graphe des hypothèses de signes diacritiques. Le coût de la composition serait fonction de critères de proximité et dépendrait également des contraintes de la langues (certains diacritiques ne sont applicables qu'à certains types de lettres).

Dans l'exemple donné ci-dessus, il n'y a pas d'ambiguïté : dans le premier pseudo-mot, le و ne prend pas de point. Dans le deuxième pseudo mot, on ne dispose que d'un point. Les chemins possibles sont donc : 0-2-5-1 (نس) et 0-4-5-1 (سن). Mais pour des raisons de proximité spatiale, seule la première solution est acceptable.

4.4.2.1 Graphe des pseudo-mots bien formés

Les arcs portant la lettre : م ـم ـم م ne sont pas représentés pour éviter de surcharger le graphe. Une sortie sur le noeud bleu correspond nécessairement à une fin de mot. Une sortie sur le noeud vert peut correspondre soit à une fin de pseudo-mot, soit à une fin de mot.

Le caractère ϵ représente la chaîne vide. Il sera possible de rendre cet automate déterministe pour réaliser une composition sans ϵ.



L'écriture arabe est plus contrainte que l'écriture latine au niveau de l'enchaînement des séquences de lettres. L'automate précédent pourrait donc également être utilisé pour guider et contraindre une reconnaissance de l'écriture arabe sans vocabulaire : dans le cadre de l'imprimé par exemple.

Le formalisme des automates à états finis pondérés, et leur représentation sous forme de graphes, pourraient être utilisés pour la reconnaissance de l'écriture arabe. Ce formalisme permettra de fusionner les hypothèses de reconnaissance sur les corps de lettres et sur les signes diacritiques de façon plus efficace que l'approche présentée dans la section 4.4.1.

Dans le cadre de cette thèse, les automates à états finis pondérés ont été utilisés pour une tâche de reconnaissance de noms de familles écrits en français. Cette approche est présentée dans le chapitre 5.

4.5 Reconnaissance de chiffres farsis isolés

Dans cette partie, nous présentons un système de reconnaissance de chiffres farsis (persans) isolés. Nous utilisons un réseau de neurones à convolutions (cf section 2.4.2.5), et nous montrerons que ce type de réseau de neurones s'adapte bien à la tâche de reconnaissance de chiffres farsis isolés.

Europe	0	1	2	3	4	5	6	7	8	9
Monde Arabe (Moyen Orient)	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Iran/Pakistan (farsi)	٠	١	٢	٣	۴	۵	۶	٧	٨	٩

TAB. 4.15 – Chiffres utilisés en Europe, dans le monde Arabe, et en Iran/Pakistan.

Comme le montre le tableau 4.15, les alphabets de chiffres arabes et farsis sont proches. Ils partagent les mêmes symboles, à l'exception du '4' et du '6'. Les symboles '0','1','2','3','7','8','9' sont rigoureusement identiques. Le 5 est très légèrement différent, mais compte tenu de la variabilité importante de l'écriture manuscrite, on peut s'attendre à ce que les '5' arabes et farsis soient quasiment identiques.

Disposant d'une base de chiffres farsis isolés (cf paragraphe 3.2.1.6), dans cette section nous décrivons la procédure d'entraînement d'un système de reconnaissance de chiffres farsis (Persan) isolés. Compte tenu de la proximité avec l'alphabet de chiffres arabes, nous pouvons raisonnablement estimer qu'un système de reconnaissance appris sur une base similaire de chiffres arabes fournirait probablement des résultats très proches.

4.5.1 Base de données

La base utilisée est présentée dans la section 3.2.1.6. Il s'agit d'une base de chiffres farsis isolés, prédécoupée en un ensemble d'apprentissage (60000 documents) et un ensemble de test (20000) documents.

Les images sont stockées dans une base de données au format CDB. Du code C++ ou Matlab [88] permet d'extraire les chiffres isolés.

0	1	2	3	4	5	6	7	8	9
									

TAB. 4.16 – Exemples de caractères farsi manuscrits.

Le tableau 4.16 donne une liste de chiffres farsis manuscrits.

4.5.2 Prétraitement des données

Les données sont fournies sous forme d'images bitonales de tailles variables. La taille maximale est de 51x62 pixels, tandis que la taille minimale est de 3x4 pixels. Une telle disparité s'explique par le fait que le caractère '0' est un point, et est donc beaucoup plus petit que les autres caractères.

Outre le cas particulier du zéro, les hauteurs des autres caractères de la base de sont pas normalisées, et leur taille peut varier du simple au double en fonction des exemples.

La manière dont sont stockées les données ne permet pas de retrouver les documents dont sont extraites les images. Il n'est donc pas possible de renormaliser leur taille en fonction des scripteurs. Ceci peut être gênant en particulier à cause de la ressemblance des symboles '0' et '5', qui ne se différencient que par la taille : '0' est un point, tandis que '5' est un cercle. Dans certains cas, on aura une confusion réelle entre des images de ces deux classes. Ce problème est discuté dans la partie résultats (4.5.5).

Pour prendre en compte la variabilité de la taille de l'écriture, il conviendra donc soit de normaliser ces caractères en hauteur, soit d'extraire des primitives invariantes, soit d'entraîner un système robuste aux variations de hauteur de l'écriture. C'est cette troisième méthode qui sera mise en oeuvre. Les images sont simplement sous-échantillonnées de telle sorte que leur hauteur et leur largeur soient divisées par deux. Elles sont ensuite centrées dans des imagettes de taille 32x32 en niveaux de gris.

Ces imagettes de taille réduite et en niveaux de gris seront directement soumises en entrée d'un réseau de neurones à convolutions.

4.5.3 Système de reconnaissance

Dans [94], LeCun et al. présentent une topologie de réseau de neurones à convolutions adapté à la reconnaissance de l'écriture manuscrite (voir paragraphe 2.4.2.5). Dans [168], P. Y. Simard et al. proposent une autre topologie de réseaux de neurones à convolutions.

En s'inspirant de ces travaux, nous proposons une autre topologie de réseaux de neurones à convolution, similaire à celle d'un LeNet5, mais pour lequel le dernier étage (classifieur à base de combinaison de gaussiennes) est remplacé par un perceptron multicouches à sorties softmax.

Le système prend en entrées des images en niveaux de gris de taille 32x32. Un tel réseau est entraîné par rétropropagation du gradient d'erreur. Il apprend sa propre extraction de caractéristiques conjointement à l'apprentissage de la tâche de classification.

P. Y. Simard et al. [168] ont montré qu'un tel réseau permettait d'obtenir d'excellents taux de reconnaissance sur la base MNIST. Nous allons vérifier que ce type d'architecture donne également de bons résultats sur l'alphabet de chiffres farsis.

4.5.4 Déformations élastiques

Souvent, dans les systèmes de reconnaissance, on travaille sur l'extraction de caractéristiques, de manière à rendre les primitives extraites robustes par rapport à la variabilité de l'écriture. De cette manière, on simplifie la tâche du reconnaisseur. Mais cette simplification se fait au prix d'une complexité accrue de la tâche d'extraction de primitives, qui risque d'être sur-adaptée pour un type de données particulier, et de mal se généraliser à d'autres données.

Le pendant de cette méthode consiste à entraîner le modèle sur davantage de données, de telle sorte que le reconnaisseur dispose d'une couverture d'exemples plus large, de manière à mieux prendre en compte la variabilité de l'écriture. C'est le sens des techniques qui consistent à augmenter artificiellement la taille de la base de données d'entraînement.

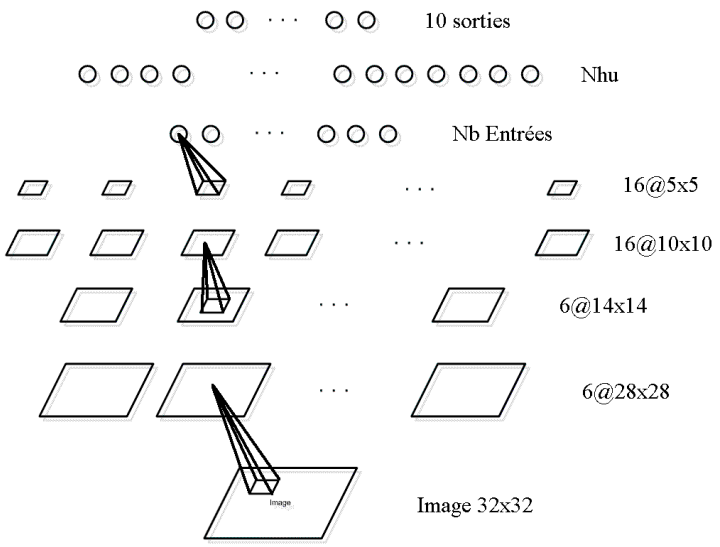


FIG. 4.38 – Système de reconnaissance utilisé pour la reconnaissance des chiffres farsis. Topologie inspirée des travaux de [94] et [168]. L'image en niveaux de gris est soumise en entrée du système. Une alternance de couches de convolution et de sous-échantillonnage réalisent l'extraction de primitives, et un réseau de neurones de type Perceptron Multicouches à sorties softmax effectue la classification. Le système complet est entraîné par rétropropagation du gradient.

On pourrait craindre que le reconnaisseur soit sur-adapté pour prendre en compte ces types de variations pseudo-aléatoires. Mais en pratique, on constate que le fait de rajouter des données déformées visuellement réalistes permet effectivement d'améliorer la robustesse du reconnaisseur.

Image initiale	1	2	3	4	5
					

TAB. 4.17 – Déformations élastiques. Cinq exemples d'images synthétiques obtenues pseudo-aléatoirement par déformations élastiques de l'image initiale.

Dans [168], Simard et al. décrivent un algorithme de déformation des données. Un champ de déformations est initialisé aléatoirement. Ce champ aléatoire est lissé par le biais d'une convolution à l'aide d'un filtre Gaussien. Ce champ de déformation est alors appliqué à l'image initiale pour obtenir une nouvelle image déformée. En fonction de la variance de la gaussienne et de l'intensité du champ de déformation aléatoire, on peut obtenir différents types de champs de déformations (voir figure 4.39).

Simard et al. proposent des paramètres de telle sorte que le modèle de distorsions élastiques offre un rendu proche de la variabilité de l'écriture manuscrite. Cette méthode permet de générer des exemples de chiffres réalistes à partir des données déjà existantes (voir figure 4.17).

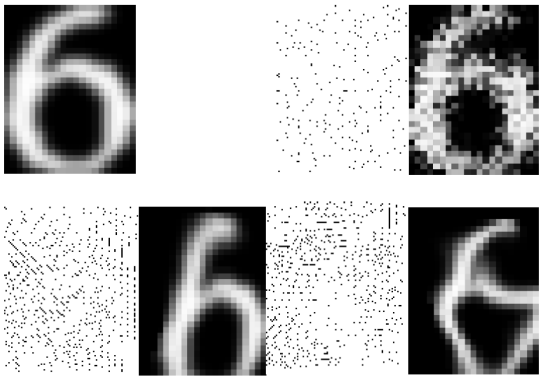


FIG. 4.39 – En haut à gauche : images initiale. En haut à droite : champ de déformation aléatoire non lissé, et image résultante. En bas : champs de déformations lissés et images résultantes. Figure extraite de [168]

Entrées	Couche cachée	% reco base classique	% reco base étendue
120	200	98.34	**99.02**
120	100	**98.65**	98.97
80	100	98.54	98.83
50	100	98.46	98.75

TAB. 4.18 – Résultats de reconnaissance.

4.5.5 Résultats

La base de données utilisée est prédécoupée en un ensemble d'apprentissage (60000 images chiffres) et un ensemble de test (20000 images de chiffres). Les classes sont équiprobables.

Nous utiliserons un réseau de neurones à convolutions, qui est appliqué directement sur l'image et ne nécessite pas de phase d'extraction de caractéristiques. Nous appliquons également la technique proposée par Simard et al. pour augmenter la taille de l'ensemble d'apprentissage à l'aide de données pseudo-aléatoires obtenues en déformant les données d'entraînement existantes. Chaque image d'entraînement permet ainsi de générer 5 images supplémentaires.

Nous essayons différentes topologies de réseaux de neurones pour déterminer la meilleure configuration. Les performances sont stables. Comme le montre le tableau 4.18, l'architecture dont le premier étage extrait un vecteur de 120 primitives et dont le perceptron multicouches contient 100 neurones sur la couche cachée permet d'obtenir les meilleures performances en utilisant la base d'entraînement classique. L'augmentation artificielle de la taille de la base d'entraînement permet d'améliorer sensiblement les performances. Le système avec 200 neurones sur la couche cachée devient légèrement meilleur avec le jeu de données étendues : les performances sont très proches, et se situent autour de 99% de reconnaissance sur la base de test.

L'analyse de la matrice de confusion (table 4.19) permet de déterminer deux types de confusions :

	0	1	2	3	4	5	6	7	8	9
0	98.35	0.00	0.00	0.00	0.05	1.25	0.05	0.30	0.00	0.00
1	0.30	99.50	0.00	0.00	0.05	0.00	0.00	0.00	0.05	0.10
2	0.00	0.10	98.00	1.35	0.15	0.00	0.10	0.15	0.00	0.15
3	0.00	0.00	1.00	98.15	0.75	0.05	0.00	0.00	0.05	0.00
4	0.00	0.00	0.10	1.45	98.25	0.05	0.05	0.00	0.00	0.10
5	0.05	0.00	0.00	0.05	0.05	99.75	0.05	0.00	0.05	0.00
6	0.00	0.15	0.10	0.00	0.00	0.10	98.95	0.05	0.00	0.65
7	0.00	0.05	0.05	0.00	0.05	0.00	0.10	99.75	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.80	0.20
9	0.00	0.05	0.00	0.00	0.05	0.00	0.20	0.00	0.00	99.70

TAB. 4.19 – Matrice de confusion pour le cas 120 entrées et 200 neurones sur la couche cachées, entraîné sur la base de données étendue.

– certains chiffres '0' sont reconnus comme des '5'
– des confusions entre les chiffres '2', '3' et '4'.

Des exemples de confusions sont donnés dans le tableau 4.20.

Les confusions entre '0' et '5' sont dues à l'absence de normalisation en hauteur entre les caractères. Sur un champ numérique complet, les '0' sont faciles à différencier des autres symboles. Il est possible de les reconnaître durant une phase de prétraitement du champ numérique. C'est l'approche que proposent M. Ziaratban et al dans [203] : leur système reconnaît les '0' et les '1' sur la base de règles très simples, puis utilise un reconnaisseur plus élaboré pour différencier les autres symboles. Il n'est toutefois pas possible d'appliquer une telle approche sur cette base : elle ne contient que des images de chiffres isolés, et il est impossible de reconstituer les champs dont les images sont extraites.

La confusion entre les chiffres '2', '3' et '4' est plus problématique. Selon les scripteurs, on voit (tableau 4.20) que ces symboles sont parfois visuellement très proches. La décision est difficile, même pour un opérateur humain.

4.5.6 Conclusion

Ces expériences confirment l'intuition selon laquelle les réseaux à convolution, qui donnent de très bons résultats de reconnaissance sur la base de chiffres MNIST, pourraient avantageusement être appliqués à la reconnaissance de chiffres farsis sans aucune modification et sans nécessiter une phase d'extraction de primitives ad hoc.

Ces expériences confirment également le fait que les techniques de déformations élastiques pour augmenter artificiellement la taille de l'ensemble d'apprentissage s'appliquent bien aux chiffres farsis.

Classe réelle	Classe reconnue	Exemples
0	5	
2	3	
3	2	
3	4	
4	3	

TAB. 4.20 – Errerurs de reconnaissance.

Comme nous l'avons vu au paragraphe 4.4.2, le formalisme des automates à états finis pondérés semble bien adapté à la modélisation de l'écriture arabe, que ce soit pour définir un mécanisme de redistribution des signes diacritiques sur les formes de lettres ; ou encore pour modéliser des contraintes morphologiques inhérentes à l'écriture arabe (découpage en pseudo-mots) qui pourraient être utilisées pour contraindre une reconnaissance de l'écriture imprimée sans vocabulaire prédéfini. Ces deux pistes seront explorées dans de futurs travaux.

Néanmoins, avant d'appliquer le formalisme des automates à états finis pondérés à la reconnaissance de l'écriture arabe, il semblait intéressant d'évaluer un tel système sur un problème plus simple, comme la reconnaissance de champs de noms de familles français écrits en lettres capitales. Ces expériences sont présentées dans le chapitre suivant.

Chapitre 5

Automates à états finis et Apprentissage Global

Le formalisme des Réseaux Transformateurs de Graphes (GTN pour Graph Transformer Networks) a brièvement été introduit dans la section 2.4.5.

Dans ce chapitre, nous allons les appliquer à une tâche de reconnaissance de noms de familles en français manuscrits en lettres capitales. Dans la section 5.1, nous présenterons les données et les objectifs de ce travail. Dans la section 5.2, et en particulier la sous-section 5.2.2, nous présenterons l'architecture du système de reconnaissance à base de graphes. Nous verrons également comment l'adaptation du reconnaisseur s'intègre dans ce formalisme.

Cette tâche de reconnaissance se fait sans dictionnaire. Dans la section 5.4, nous verrons de quelle manière ce formalisme permet d'introduire des connaissances supplémentaires, à travers l'utilisation d'un modèle de N-gram.

5.1 Base de données et objectifs

L'objectif de ce chapitre est de mettre en oeuvre un système de reconnaissance à base d'automates à états finis pondérés, et nous allons l'appliquer à la reconnaissance de noms de familles français écrits en lettres capitales (voir figure 5.1).

Le champ est localisé à l'aide du mot clé imprimé "NOM :". La détection de ce champ

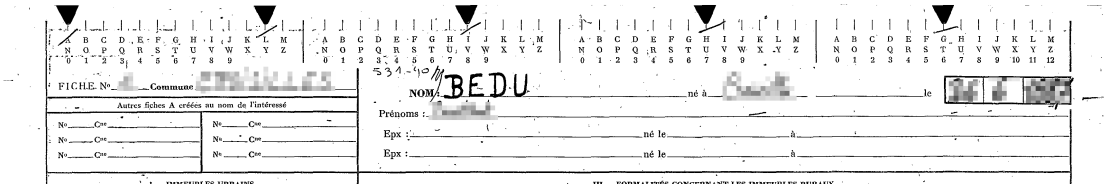


FIG. 5.1 – Exemple de document à traiter. On cherche à reconnaître le champ NOM. Les autres champs ont été floutés.

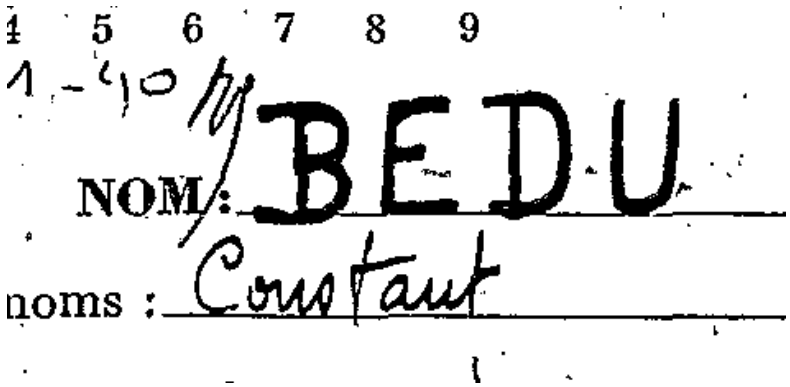


FIG. 5.2 – Champ nom à reconnaître : "BEDU"

ne fait pas l'objet de notre travail, aussi le détail de cette localisation n'est pas donné ici.

Le champ à reconnaître est bruité (voir figure 5.2). Certains de ces bruits de fond sont trop gros pour être nettoyés de façon suffisamment fiable. Le reconnaisseur de caractères devra donc être suffisamment robuste aux bruits.

Dans cette application, la reconnaissance se fait sans dictionnaire. Nous disposons d'une procédure de segmentation qui propose plusieurs hypothèses de segmentation en caractères, et d'un reconnaisseur de caractères isolés (ici un réseau de neurones de type Perceptron MultiCouches). Il est appliqué sur chacune des hypothèses pour déterminer la meilleure interprétation du champ.

Pour résoudre un problème donné, il est parfois utile de réentraîner un reconnaisseur existant pour optimiser ses performances sur ce problème spécifique. C'est ce que nous allons faire ici : en partant d'un reconnaisseur de caractères isolés, nous allons chercher à le réentraîner pour optimiser ses performances sur une tâche de reconnaissance de noms de familles, en utilisant la base de donnée présentée ci-dessus.

Pour cela, nous disposons d'une quantité (limitée) de données annotées. Ces données sont de plusieurs types : pour certains documents, les caractères sont découpés et annotés manuellement. Mais cette procédure est coûteuse, et la quantité de données ainsi disponible est limitée : 106384 caractères annotés. Nous disposons également de 43620 documents annotés au niveau champs, pour lesquels la valeur du champ à reconnaître est connue, mais pas l'annotation individuelle des caractères qui composent ces champs (ces 43620 documents représentent 310971 caractères).

L'objectif de ce chapitre est de comparer deux procédures d'optimisation :

- dans la première, le reconnaisseur initial est réentraîné à l'aide des caractères annotés dont nous disposons. Cette procédure et les résultats obtenus seront décrits dans la section 5.3.1.
- dans la seconde, le reconnaisseur initial est réentraîné à l'aide des informations au niveau champs, plus nombreuses mais moins précises (pas d'annotation des différents caractères qui composent ces champs). Pour cela, nous construirons un système de reconnaissance à base de graphes, inspiré de celui présenté par Y. LeCun et al. dans [94]. Ce système sera décrit dans la section 5.2.2. La procédure

d'entraînement et les résultats obtenus seront décrits dans la section 5.3.2.

Nous verrons ainsi que le système de reconnaissance à base de graphes, bien qu'étant entraîné sans avoir recours à une annotation précise au niveau caractères, permet d'obtenir de meilleures performances, car le reconnaisseur ainsi entraîné parvient à mieux rejeter les mauvais candidats de segmentation.

Dans un second temps, dans la section 5.4, nous étudierons l'apport d'un modèle de langage sur la tâche de reconnaissance des noms de familles. Nous verrons que dans le cas de l'utilisation d'un modèle de langage en reconnaissance, il est plus intéressant de réentraîner le reconnaisseur en incluant le modèle de langage dans la chaîne (optimisation globale) plutôt que de réentraîner le reconnaisseur indépendamment du modèle de langage (section 5.4.2).

5.2 Présentation du système de reconnaissance

5.2.1 Système de reconnaissance de référence

Le système de reconnaissance initial est un réseau de neurones de type perceptron multi-couches. Ses sorties sont normalisées à l'aide d'une fonction de transfert de type softmax :

$$S_i = \frac{\exp^{-V_i}}{\sum_j \exp^{-V_j}}$$

Le nombre de sorties de ce réseau de neurones correspond au nombre de classes à reconnaître, plus trois classes supplémentaires de rejet : bruits de fond, sur-segmentations et sous-segmentations.

Il comporte 652 entrées. Les primitives utilisées ne sont pas décrites.

Ce réseau de neurones est entraîné par rétro propagation du gradient, à l'aide d'une base annotée de caractères isolés. La fonction de coût utilisée est la divergence de Kullback-Leibler : ($\sum_i t_i . \log\left(\frac{t_i}{s_i}\right)$, t_i étant la cible (la valeur désirée) de la sortie i, et s_i la valeur réelle de la sortie i).

Outre les classes de caractères valides, le reconnaisseur dispose également de trois classes de rejets :

- la classe "bruit de fond" : correspond à des traits superflus ou des taches parasites qui proviennent de la phase de binarisation et qui n'ont pas été supprimées lors de la phase de nettoyage.
- la classe "sur-segmentation" : correspond à un candidat de segmentation trop petit (une portion de lettre). Le module de segmentation en caractères propose une hypothèse de segmentation erronée.
- la classe "sous-segmentation" : correspond à un candidat de segmentation trop gros (une lettre plus un morceau d'une deuxième lettre). Le module de segmentation en caractères propose une hypothèse de segmentation erronée.

La classe "bruit de fond" ne disqualifie pas l'hypothèse de segmentation en lettres. Son rôle est simplement d'écarter les symboles qui correspondent aux bruits dans l'image, mais qui n'ont pas d'interprétation.

Par opposition à la classe "bruit de fond", les classes "sur-segmentation" et "sous-segmentation" vont disqualifier l'hypothèse de segmentation dans laquelle des symboles de ce type apparaissent, car ces symboles correspondent à des erreurs de segmentation.

En plus des caractères valides, des exemples de caractères de rejet (bruits de fond, sous-segmentations et sur-segmentations) sont également annotés manuellement en tant que tels pour permettre l'apprentissage du reconnaisseur.

La reconnaissance se fait sans connaissance du vocabulaire. La réponse du système correspond à la meilleure hypothèse de segmentation-reconnaissance.

5.2.2 Système de reconnaissance à base d'automates à états finis

5.2.2.1 Principe général

L'utilisation des Automates à Etats Finis Pondérés (WFSM pour Weighted Finite State Machines) est très répandue en reconnaissance automatique de la parole et en traitement automatique de la langue naturelle [61, 32, 127].

Ces modèles peuvent également être utilisés dans le cadre de la reconnaissance automatique de l'écriture manuscrite, comme l'ont montré Y. LeCun et al. dans [94]. Ils décrivent un système de reconnaissance de l'écriture basé sur les Réseaux Transformateurs de Graphes (GTN pour Graph Transformer Networks). Le système présenté ici est largement inspiré de la méthodologie qu'ils décrivent dans [94].

Définitions

Un automate à états finis de type *transducteur* est un 7-uplet $(Q, I, F, \sum, \Delta, \delta, \sigma)$
- Q l'ensemble des états,
- $I \subseteq Q$ l'ensemble des états initiaux,
- $F \subseteq Q$ l'ensemble des états finaux,
- $\sum$ l'ensemble des symboles qui constituent l'alphabet d'entrée,
- Δ l'ensemble des symboles qui constituent l'alphabet de sortie,
- δ l'ensemble des transitions, qui vont de $Q \times \sum$ vers Q,
- σ la fonction de sortie, qui va de $Q \times \sum$ vers Δ.

Un *accepteur* est un transducteur qui ne possède qu'un seul alphabet (l'alphabet de sortie est le même que l'alphabet d'entrée).

En pratique, nous représenterons ces automates sous forme de graphes. Dans le cas des automates à états finis pondérés (que nous utiliserons ici), les transitions sont pondérées. Chaque arc du graphe porte une probabilité (ou un coût, les deux approches sont duales).

Dans ce formalisme, les résultats intermédiaires (graphe de segmentation, graphe de reconnaissance) sont des graphes accepteurs, tandis que les grammaires sont des graphes transducteurs. De façon générale, chaque élément de la chaîne de reconnaissance (segmenteur, extracteur de primitives, reconnaisseur, et même modèle de langage) peut être vu comme une fonction de transfert d'un graphe accepteur vers un autre graphe accepteur.

La composition d'un graphe accepteur dans $\sum$ et d'un graphe transducteur $\sum$ vers Δ donne un graphe accepteur dans Δ, est définie par :

$$(S \otimes T)(t) = \sum_{s \in \sum} (S(s) \times T(s,t))$$

La composition $S \otimes T$ assigne une probabilité w à l'accepteur d'une suite t s'il existe une suite s telle que S accepte s avec une probabilité u, T transduit s vers t avec une probabilité v, et $w = u \cdot v$

Une telle composition permet d'extraire tous les chemins qui sont compatibles aux deux graphes, et assigne à chaque arc du graphe résultat une probabilité (resp. un coût) égale au produit (resp. la somme) des probabilités (resp. coûts) de ses deux arcs parents.

Par exemple, la figure 5.3 issue de [94] illustre une procédure de composition entre un graphe accepteur résultat de reconnaissance, et un vocabulaire représenté sous forme de transducteur. Dans ce cas particulier, les alphabets d'entrée et de sortie du transducteur sont identiques, et les coûts des arcs du transducteur sont nuls.

Pour mieux comprendre l'opération de transduction, imaginons qu'un jeton soit positionné sur l'état initial du graphe accepteur et qu'un autre jeton soit positionné sur l'état initial du graphe transducteur. Le jeton peut se déplacer d'arc en arc sur le graphe accepteur à condition que, dans le même temps, l'autre jeton puisse se déplacer sur le graphe transducteur en parcourant les mêmes symboles d'entrées. Un chemin est dit *acceptable* si les deux jetons parviennent simultanément dans un état terminal. Dans ce cas, un chemin est construit dans le graphe accepteur résultat. Chacun des arcs de ce chemin porte le symbole émis par le graphe transducteur (alphabet de sortie), et sa probabilité (resp. son coût) est égal au produit des probabilités (resp. à la somme des coûts) des deux arcs qui l'ont produit. L'opération de transduction consiste ainsi à extraire tous les chemins compatibles entre les deux graphes.

La figure 5.4 décrit l'architecture du système de reconnaissance à base de graphes que nous avons utilisé. Dans les paragraphes suivants, nous décrivons les différents modules qui composent ce système.

5.2.2.2 Graphe de segmentation

L'étape de segmentation génère une liste d'options de segmentation (voir figure 5.5). Chaque option de segmentation est une manière différente de découper une ligne en lettres. Chaque élément d'une option de segmentation est un 'candidat de lettre'. Chaque morceau d'encre du champ à reconnaître ne doit être présent qu'une et une seule fois dans chaque option de segmentation.

Une liste d'options de segmentation peut aisément être convertie en un graphe de segmentation (voir figure 5.6). Ce graphe permet de représenter l'information contenue dans la liste sous une forme plus compacte.

Les noeuds correspondent aux points de coupure, et les arcs portent les candidats de lettres. Dans ce **graphe de segmentation**, chaque candidat est représenté une et une seule fois.

Un chemin dans le graphe de segmentation correspond à une option de segmentation. La probabilité de ce chemin est la même que la probabilité de l'option de segmentation

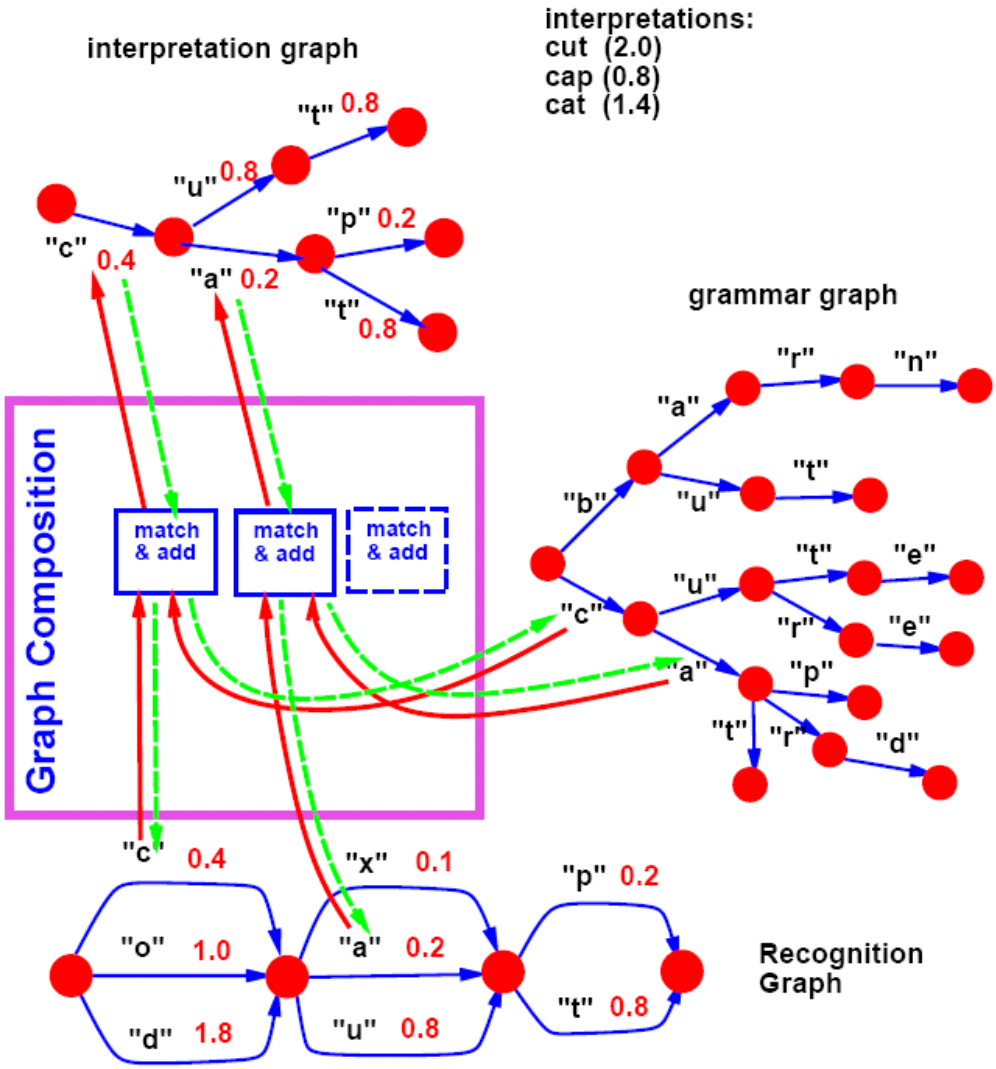


FIG. 5.3 – Exemple de composition : composition entre un résultat de reconnaissance et un vocabulaire, figure extraite de [94]. Le graphe de reconnaissance (recognition graph) est un graphe accepteur. Les valeurs numériques associées aux arcs sont des coûts. Le graphe qui correspond au vocabulaire (grammar graph) est un transducteur. Dans ce cas particulier, les alphabets d'entrée et de sortie du transducteur sont identiques, et les coûts des arcs du transducteur sont nuls. Le résultat est un graphe accepteur (interpretation graph), dont les chemins sont les interprétations compatibles entre le graphe de reconnaissance et le graphe du vocabulaire.

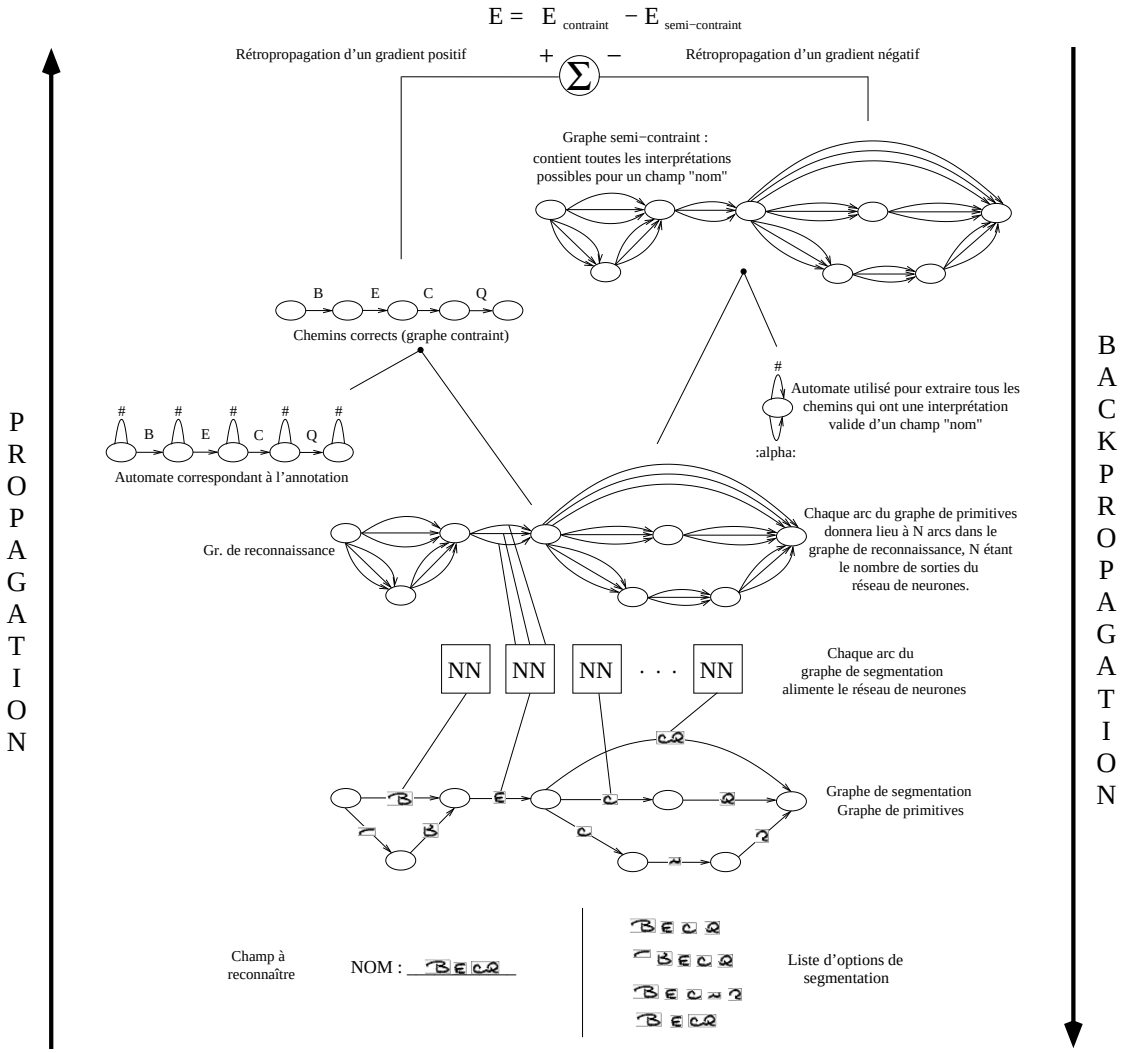


FIG. 5.4 – Architecture du système de reconnaissance de noms de familles à base de graphes.

FIG. 5.5 – Liste des options de segmentation pour le champ 'DUCAMPS'.

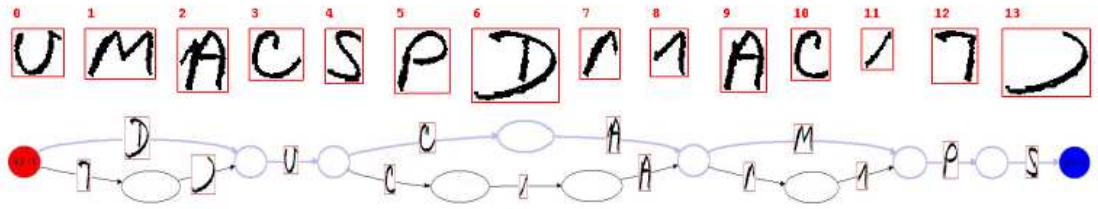


FIG. 5.6 – Les 14 candidats de caractères (issus de la figure 5.5) correspondant au champ 'DUCAMPS', et les différentes manières de les combiner sous forme de graphe.

correspondante. Comme chaque morceau d'encre n'est représenté qu'une et une seule fois, le graphe est acyclique. Comme tous les autres graphes de résultats intermédiaires (qui seront présentés dans les paragraphes suivants) sont dérivés à partir du graphe de segmentation, cette propriété est conservée : tous ces graphes seront également acycliques.

Contrairement à la solution proposée dans [94], nous décidons de travailler sur des probabilités plutôt que sur des coûts. Ces deux approches sont duales.

Il est possible d'exprimer un coût sous forme de probabilité, et inversement :

$$C_{arc} = -\log\left(P_{arc}\right)$$

5.2.2.3 Graphe de primitives

A partir du graphe de segmentation, il est possible de construire un graphe de caractéristiques. Ce graphe de primitives a exactement la même topologie que le graphe de segmentation. Chaque candidat lettre (information portée par un arc dans le graphe de segmentation) donne lieu à un vecteur d'attributs (information qui est également portée par un arc dans le graphe de primitives).

Dans le cas qui nous intéresse, la taille de l'espace d'entrée est 652. La probabilité associée à chaque arc dans le graphe de primitives est la même que la probabilité de l'arc parent dans le graphe de segmentation.

5.2.2.4 Graphe de reconnaissance

A partir du graphe de primitives, on construit un nouveau graphe (le graphe de reconnaissance), qui contient le même nombre de noeuds que le graphe de primitives. Chaque arc du graphe de primitives donne lieu à N arcs parallèles dans le graphe de reconnaissance, où N est le nombre de sorties du réseau de neurones (défini dans la section 5.2.1).

Chacun de ces arcs correspond à une seule sortie du réseau de neurones. Il porte l'étiquette de la classe associée à la sortie du réseau de neurones dont il est issu. La probabilité de cet arc est la probabilité a posteriori (valeur de la sortie considérée) multipliée par la probabilité associée à l'arc parent dans le graphe de primitives.

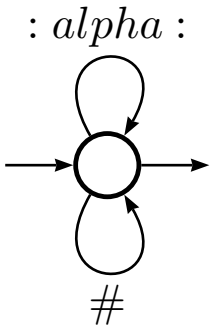


FIG. 5.7 – Automate utilisé pour écarter les arcs (sorties du réseau de neurones) qui n'ont pas d'interprétation valide dans un champ de type 'nom de famille'. '#' est le symbole qui désigne un bruit de fond. Cet automate n'autorise que les chemins qui contiennent des lettres (:alpha :) et des bruits de fond, pas les symboles de sous-segmentation ni sur-segmentation.

5.2.2.5 Graphe semi-contraint

Contrairement à LeCun et al. [94], nous utilisons des classes explicites de rejet pour les sur segmentations et les sous segmentations. Ces deux classes n'ont pas d'interprétation valide dans l'évaluation d'un champ. Par conséquent, il est possible d'élaguer le graphe de reconnaissance en supprimant tous les chemins qui contiennent au moins l'un de ces symboles. On construit donc un sous graphe qui ne contient que des arcs qui ont une interprétation valide (les caractères alphabétiques, ainsi que les bruits de fond qui sont considérés comme symboles non émetteurs).
Pour ce faire, on effectue une composition entre le graphe de reconnaissance et un automate. Cet automate introduit une contrainte sémantique sur le résultat, et permet de rejeter les classes de sur segmentation, de sous segmentation, et les autres classes de caractères qui n'ont pas de sens dans l'interprétation d'un champ de type 'Nom de famille'.
Dans notre cas, cet automate n'apporte pas une contrainte très importante. Mais on pourrait tout à fait imaginer des applications dans lesquelles il pourrait être utile d'introduire des informations sémantiques plus importantes : par exemple un reconnaisseur de plaques d'immatriculation, qui pourrait être contraint à l'aide d'un automate qui autoriserait : 1 à 5 chiffres, puis 2 ou 3 lettres, puis 2 chiffres.

Le graphe résultant de la composition du graphe de reconnaissance avec cet automate est appelé **graphe semi contraint**. C'est ce graphe (réduit) qui sera utilisé pour extraire le meilleur chemin, et évaluer la réponse du système (la valeur du champ à reconnaître).

5.2.2.6 Graphe contraint

Pour entraîner le système, il est nécessaire d'introduire un autre graphe, qui se dérive également du graphe de reconnaissance. C'est le graphe des réponses correctes : il contient tous les chemins du graphe de reconnaissance qui correspondent à l'annotation.

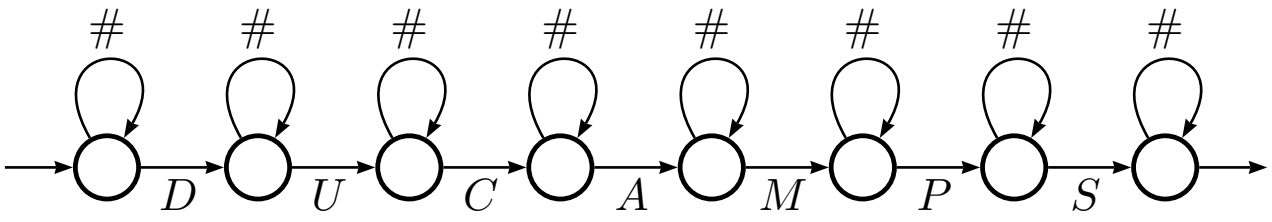


FIG. 5.8 – Automate qui correspond à l'annotation : il permet d'extraire tous les chemins qui correspondent à la cible. '#' est le caractère de bruit de fond. Cet automate autorise un nombre arbitraire de symboles de bruits entre deux caractères consécutifs, et également avant le premier et après le dernier.

Comme nous l'avons dit précédemment, une image peut être bruitée, même après l'opération de nettoyage. Le graphe qui correspond aux réponses correctes doit lui aussi contenir des chemins qui contiennent des bruits de fond (mais pas des caractères de sur segmentation et sous segmentation). Par conséquent, l'automate qui correspond à l'annotation doit prendre en compte cette contrainte en incluant ces bruits de fond. Nous choisissons un modèle qui permet un nombre quelconque de caractères de bruit entre deux caractères valides consécutifs.

Le résultat de la composition entre cet automate et le graphe de reconnaissance est appelé le **graphe contraint**. Il contient tous les chemins qui correspondent à la cible (la valeur réelle du champ à reconnaître, donnée par l'annotation).

En résumé :

- On construit le graphe de segmentation à partir d'une liste d'options de segmentation. Chaque option de segmentation correspond à une manière de découper le champ en lettres.
- A partir du graphe de segmentation, on construit le graphe de primitives. Ce graphe a la même topologie que le graphe de segmentation. L'information portée par ses arcs sont les vecteurs de primitives issues de la segmentation.
- Chaque arc du vecteur de primitives est appliqué en entrée du réseau de neurones. Chaque sortie du réseau de neurones donnera lieu à un arc dans le graphe de reconnaissance. Par conséquent, le graphe de reconnaissance a le même nombre de noeuds que le graphe de primitives, et N fois plus d'arcs, où N est le nombre de classes de sortie du réseau de neurones.
- A partir de ce graphe de reconnaissance, on construit deux sous-graphes :
  - le graphe contraint, qui contient tous les chemins qui correspondent à l'annotation, en absorbant les possibles bruits de fond
  - le graphe semi-contraint, qui ne garde que les arcs qui permettent une interprétation valide (conserve les caractères alphabétiques et les bruits de fond, mais rejette les sous-segmentations et sur-segmentations).

Nous allons maintenant présenter la procédure d'apprentissage du réseau de neurones intégré dans ce système de reconnaissance. Le critère d'optimisation est choisi de telle sorte que pour chaque document, les chemins inclus dans le graphe contraint soient également les meilleurs chemins du graphe semi-contraint. De cette manière, les

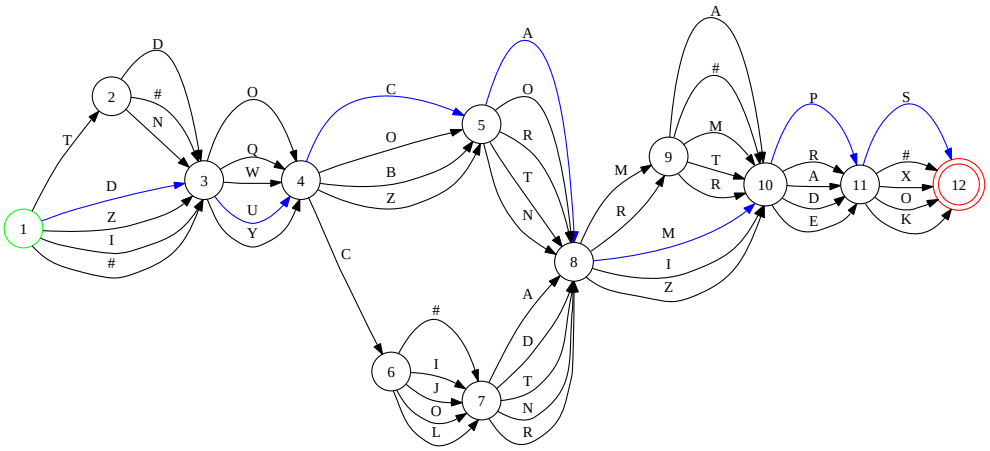


FIG. 5.9 – Graphe de reconnaissance du champ 'DUCAMPS'. Sur cette figure, on affiche seulement les cinq meilleurs arcs sortant de chaque noeud. Dans le graphe complet, il y a autant de sorties en parallèle qu'il y a de classes de sorties dans le réseau de neurones. Le meilleur chemin (chemin ayant la probabilité la plus importante) est affiché en bleu.

meilleurs chemins du graphe semi-contraint seront ceux qui correspondent à l'annotation. La procédure d'apprentissage est détaillée dans la section suivante.

5.2.2.7 Apprentissage forward discriminant

Comme nous l'avons vu dans les paragraphes précédents, le graphe de reconnaissance donne lieu à deux sous-graphes :

- Le graphe contraint, qui contient tous les chemins qui donnent une interprétation valide au sens de l'annotation.
- Le graphe semi-contraint, qui contient tous les chemins qui ne contiennent que des caractères alphabétiques et des bruits de fond.

La fonction de coût est la même que celle utilisée dans [94] :

$$E = E_c - E_{sc}$$

E_c : coût forward sur le graphe contraint.
E_{sc} : coût forward sur le graphe semi-contraint.

En utilisant le formalisme probabiliste :

$$E = -\log(\alpha_c) + \log(\alpha_{sc}) = -\log(\frac{\alpha_c}{\alpha_{sc}})$$

α_c : probabilité forward du graphe contraint
α_{sc} : probabilité forward du graphe semi-contraint.
La procédure de calcul de la probabilité forward est donnée dans le paragraphe 5.2.2.7.1.

E est toujours positif, puisque le graphe contraint est inclus dans le graphe semi-contraint. Lorsque le système renvoie une réponse proche de la réponse attendue (c'est à dire lorsque les meilleurs chemins du graphe semi-contraint sont également les meilleurs chemins du graphe contraint, les coûts forward E_c et E_{sc} sont proches, donc le coût $E = E_c - E_{sc} \approx 0$.

Une valeur de E faible indique que le champ est déjà correctement reconnu (les meilleurs chemins du graphe semi-contraint sont les interprétations correctes, et les chemins qui correspondent aux erreurs de reconnaissance ont déjà une faible probabilité). Dans ce cas, le coût est faible, et l'influence de ces exemples sur l'apprentissage l'est également.

On calcule le gradient de coût sur chacun des deux graphes : gradient positif sur le graphe contraint, et gradient négatif sur le graphe non-contraint. Et on fait la somme de ces deux quantités sur chacun des arcs du graphe de reconnaissance.
De cette manière, les arcs qui appartiennent aux deux graphes se compensent.
Les arcs qui sont importants dans le graphe contraint et qui ne le sont pas dans le graphe semi-contraint verront leurs coûts décroître (leurs probabilités augmenter).

5.2.2.7.1 Fonction forward

En utilisant les mêmes notations que dans [94], on a :
C_{ij} : coût sur un arc du noeud i au noeud j
U_j ensemble des noeuds prédécesseurs de j

$$\log_{add}(x_1, x_2, \ldots, x_n) = -\ln\left(\sum_{i=1}^{n} e^{x_i}\right)$$

Coût forward du noeud j :

$$f_j = \log_{add}\left((f_i + C_{ij})_{i \in U_j}\right) = -\ln\left(\sum_{i \in U_j} e^{-(f_i + C_{ij})}\right)$$

avec $f_{init} = 0$

En utilisant le formalisme probabiliste, on définit également :
La probabilité d'un arc du noeud i au noeud j :

$$O_{ij} = e^{(-C_{ij})}$$

Et la probabilité forward d'un arc j :

$$\alpha_j = e^{(-f_j)} = e^{\ln\left(\sum_{i \in U_j} e^{-(f_i + C_{ij})}\right)}$$

$$\alpha_j = \sum_{i \in U_j} e^{-(f_i + C_{ij})} = \sum_{i \in U_j} \left(e^{-f_i} \times e^{-C_{ij}}\right) = \sum_{i \in U_j} \alpha_i \times O_{ij}$$

Avec : $\alpha_{init} = e^{-f_{init}} = 1$

On retrouve la forme de la fonction Forward utilisée dans l'algorithme Baum-Welch (voir annexe A.1.2.

5.2.2.7.2 Fonction Backward

On définit également :

$$\beta_i = \frac{\partial \alpha_N}{\partial \alpha_i} = \sum_j \left(\frac{\partial \alpha_N}{\partial \alpha_j} \times \frac{\partial \alpha_j}{\partial \alpha_i} \right) = \sum_j \left(\beta_j \times \frac{\partial \left(\sum_k \alpha_k \times O_{kj} \right)}{\partial \alpha_i} \right) = \sum_j \left(\beta_j \times O_{ij} \right)$$

avec $\beta_N = 1$

5.2.2.7.3 Rétro-propagation

Comme dans [94], soit : $E = f_N$, le coût forward du graphe.

Nous avons alors :

$$\frac{\partial E}{\partial \alpha_i} = \frac{\partial f_N}{\partial \alpha_N} \times \frac{\partial \alpha_N}{\partial \alpha_i} = \frac{\partial \left(-\ln \left(\alpha_N \right) \right)}{\partial \alpha_N} \times \frac{\partial \alpha_N}{\partial \alpha_i} = -\frac{1}{\alpha_N} \times \frac{\partial \alpha_N}{\partial \alpha_i} = -k \times \beta_i$$

On peut également exprimer le gradient sur chaque arc du graphe :

$$\frac{\partial E}{\partial O_{ij}} = \frac{\partial E}{\partial \alpha_N} \times \frac{\partial \alpha_N}{\partial O_{ij}} = -\frac{1}{\alpha_N} \times \frac{\partial \alpha_N}{\partial \alpha_j} \times \frac{\partial \alpha_j}{\partial O_{ij}} = -k \times \beta_j \times \frac{\partial \left(\sum_i \alpha_i \times O_{ij} \right)}{\partial O_{ij}} = -k \times \alpha_i \times \beta_j$$

Où k est une constante pour le graphe considéré : $k = \frac{1}{\alpha_N}$

Chaque arc du graphe de reconnaissance a un gradient d'erreur (obtenu en faisant la somme du gradient positif qui provient du graphe contraint et du gradient négatif qui provient du graphe semi-contraint).

Pour chaque arc du graphe de reconnaissance, il est possible de retrouver la classe de réseau de neurones qui lui est associée, et également l'arc parent dans le graphe de primitives. Nous disposons donc des sorties, des gradients d'erreur, et des entrées. Il est donc possible de rétro propager le gradient d'erreur à travers le réseau de neurones, pour tous les candidats de lettres d'un champ donné.

Pour un document donné, on calcule autant de rétropropagations qu'il y a d'arcs dans le graphe de primitives (qui correspond également au nombre de candidats de caractères). Pour chaque candidat de caractère ayant été soumis en entrée du réseau de neurones, on calcule le gradient d'erreur par rapport aux paramètres du réseau de neurones (poids et biais). On accumule ces gradients sur l'ensemble des candidats de caractère d'un champ donné. Puis on met à jour les poids du réseau lorsque l'ensemble des gradients a été accumulé sur le champ en question.

La mise à jour des poids s'effectue à l'aide de la formule suivante :

$$\omega(t+1) = \omega(t) - \gamma \times \frac{\partial E}{\partial \omega}$$

γ est le coefficient d'apprentissage

Remarque : les classes de caractères qui sont absentes du graphe semi-contraint (en particulier les classes de sur-segmentation et de sous-segmentation) sont entraînées *par défaut*. Pour les options de segmentation qui sont erronées, toutes les sorties valides (lettres et bruit de fond) vont recevoir un gradient négatif : toutes ces sorties vont décroître. Or, la normalisation Softmax impose que les sorties du réseau de neurones somment à 1. Donc pour les exemples pour lesquels toutes classes valides se voient attribuer un gradient négatif, les sorties des classes invalides (sur-segmentation et sous-segmentation) vont croître. Par conséquent, le réseau de neurones va également être entraîné à mieux rejeter les erreurs de segmentation.

5.2.2.7.4 Score et décision La réponse du système est le meilleur chemin du graphe semi-contraint. Le score de confiance associé à cette réponse est la probabilité du meilleur chemin (produit des probabilités des arcs qui le composent), normalisée par la probabilité forward du graphe.

5.3 Apprentissage caractère vs Apprentissage champ

On partira d'un reconnaisseur de caractères déjà initialisé (présenté dans la section 5.2.1), qu'on cherchera à optimiser pour l'application de reconnaissance de champs nom de famille décrite dans la section 5.1.

L'apprentissage caractères consistera à utiliser une base de 106385 caractères annotés manuellement pour réentraîner le reconnaisseur.

L'apprentissage champ en revanche, n'utilisera que l'information du contenu textuel au niveau champs. L'annotation de chacun des caractères qui composent ce champ ne sera pas accessible. En revanche, l'annotation au niveau champs étant moins coûteuse et plus facile à obtenir, un plus grand nombre de documents sont disponibles : 43620 documents, qui représentent un total de 310971 caractères.

Comme nous allons le voir par la suite, bien qu'exploitant une annotation moins riche que celle utilisée par l'apprentissage caractères, et bien qu'obtenant de moins bonnes performances en terme de reconnaissance de caractères, l'apprentissage au niveau champs permet néanmoins d'obtenir de meilleurs résultats sur la tâche qui est visée : la reconnaissance de champs.

5.3.1 Apprentissage caractère

5.3.1.1 Méthodologie

Le reconnaisseur utilisé est un réseau de neurones de type Perceptron-Multicouches de 652 entrées et 39 classes de sorties (voir section 5.2.1). Son réapprentissage se fait sur

Classe	Apprentissage	Test	Total	%
A	7614	3803	11417	10,73%
B	2713	1374	4087	3,84%
C	3077	1507	4584	4,31%
D	2632	1333	3965	3,73%
E	4487	2235	6722	6,32%
F	1825	928	2753	2,59%
G	2725	1368	4093	3,85%
H	2832	1402	4234	3,98%
I	3042	1549	4591	4,32%
J	465	219	684	0,64%
K	727	351	1078	1,01%
L	4123	2050	6173	5,80%
M	2771	1414	4185	3,93%
N	1561	761	2322	2,18%
O	2901	1474	4375	4,11%
P	1301	653	1954	1,84%
Q	1245	634	1879	1,77%
R	2553	1263	3816	3,59%
S	3182	1596	4778	4,49%
T	2437	1224	3661	3,44%
U	2893	1409	4302	4,04%
V	1756	854	2610	2,45%
W	192	93	285	0,27%
X	914	460	1374	1,29%
Y	1471	732	2203	2,07%
Z	1373	655	2028	1,91%
bruit de fond	1967	906	2873	2,70%
sur-segmentation	5796	2928	8724	8,20%
sous-segmentation	234	129	363	0,34%
Nombre de caractères annotés	70984	35401	106385	

TAB. 5.1 – Distribution des caractères annotés dans les bases utilisées lors de l'apprentissage caractères.

la base décrite dans la section 5.1. Un récapitulatif de la distribution des données par classe de caractère est donnée dans le tableau 5.1.

Le réseau de neurones est appris de manière itérative par descente de gradient stochastique. Les exemples sont présentés en entrée du réseau de neurones et l'annotation correspond à la classe de sortie attendue (qui correspond à une valeur de sortie attendue de 1 pour cette classe, et 0 pour toutes les autres). Une fonction de transfert Softmax assure que les sorties du réseau de neurones sont homogènes à des probabilités. La fonction de coût utilisée est la divergence de Kullback-Leibler.

$C_{KL} = \sum_i P_i log(\frac{P_i}{Q_i})$

Les classes de rejet en sur-segmentation et sous-segmentation sont également annotées manuellement et utilisées en apprentissage, au même titre que les autres classes de caractères.

Classe	Reco init %	Reco réentraîné %	Pos Moy init	Pos Moy réentraîné
A	98.79	99.61	1.0284	1.0047
B	93.23	99.49	1.1630	1.0116
C	97.68	98.74	1.0796	1.0504
D	98.20	98.72	1.0585	1.0773
E	98.61	99.28	1.0174	1.0139
F	97.09	98.60	1.0916	1.0334
G	97.44	98.39	1.1418	1.0599
H	98.64	98.93	1.0478	1.0756
I	98.64	96.45	1.0297	1.0581
J	66.21	95.89	1.9087	1.0731
K	90.31	98.01	1.4758	1.1624
L	97.95	98.59	1.0478	1.0390
M	99.36	99.50	1.0262	1.0347
N	99.47	99.61	1.0092	1.0092
O	98.03	98.51	1.0353	1.0217
P	92.19	98.16	1.4257	1.1194
Q	80.28	95.58	1.7634	1.1672
R	97.55	98.81	1.0277	1.0388
S	99.06	99.06	1.0639	1.0351
T	98.94	98.86	1.0433	1.0351
U	98.72	98.51	1.0383	1.0646
V	93.68	97.07	1.1499	1.0995
W	72.04	94.62	2.1290	1.2688
X	98.70	98.48	1.1217	1.0478
Y	96.86	98.36	1.2623	1.1079
Z	95.27	98.32	1.2595	1.0962
bruit de fond	63.47	80.40	1.6104	1.2511
sur-segmentation	71.00	91.25	1.3999	1.1059
sous-segmentation	22.48	63.57	8.0155	1.8992

TAB. 5.2 – Comparaison entre le système initial et le système réentraîné au niveau caractères. Colonnes de gauche : taux de reconnaissance caractères. Colonnes de droite : position moyenne de la réponse correcte (1 $\Rightarrow$ réponse correcte).

5.3.1.2 Apprentissage caractères : résultats

Cette partie contient les résultats du système réentraîné à l'aide d'annotations caractères. Nous présenterons d'abord les performances du systèmes en terme de taux de reconnaissance sur des caractères isolés. Puis nous présenterons les performances de ce système appliqué à une tâche de reconnaissance d'un champ.

5.3.1.2.1 Reconnaissance au niveau caractères Le tableau 5.2 indique la comparaison des résultats au niveau caractères. Réentraîner le système permet de l'adapter sur ce type de données, et améliore les performances en terme de reconnaissance de caractères.

5.3.1.2.2 Reconnaissance au niveau champs Les résultats donnés figure 5.10 montrent l'amélioration des performances au niveau champs qui découlent de l'optimisation du

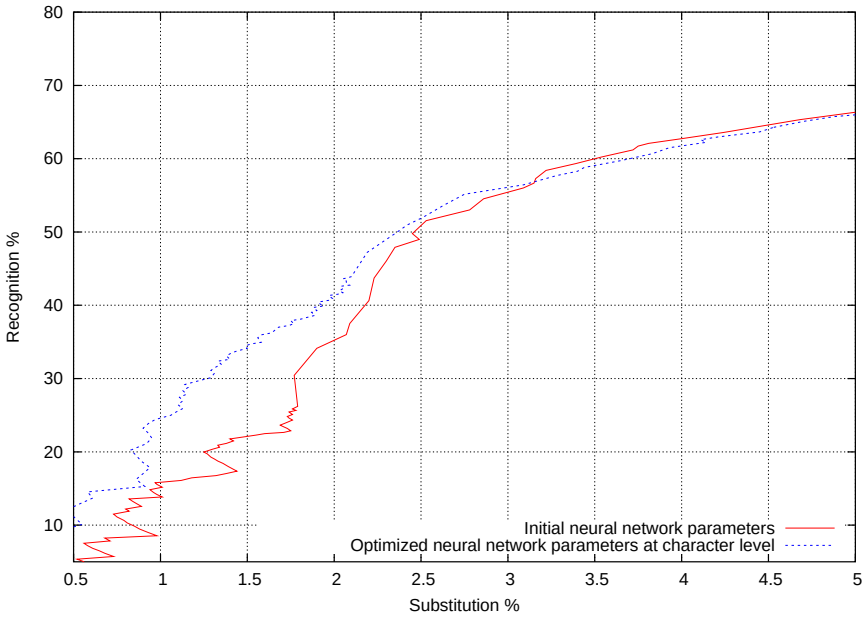


FIG. 5.10 – Reconnaissance au niveau champs. Courbe rouge : système initial. Courbe bleue : système optimisé au niveau caractères.

système de reconnaissance à l'aide d'annotations de caractères isolés.

5.3.2 Apprentissage champ

5.3.2.1 Méthodologie

A partir de l'annotation au niveau champs, il est possible d'évaluer la quantité et la proportion de lettres dans la base utilisée (voir table 5.3). Ces informations sont déduites à partir des annotations champs : dans cette partie, le système est réentraîné sans avoir recours aux annotations de lettres.

La procédure d'apprentissage utilisée est décrite dans la section 5.2.2.7. Nous partons du même système de reconnaissance initial que dans la section 5.3.1. Ce système sera réentraîné pour être adapté sur cette base.

5.3.2.2 Apprentissage champs : résultats

De la même manière que dans la section 5.3.1.2, nous présenterons les performances du système en terme de taux de reconnaissance sur des caractères isolés, et nous présenterons également les performances de ce système sur une tâche de reconnaissance de champs.

5.3.2.2.1 Reconnaissance au niveau caractères Bien que n'ayant pas été appris sur des données annotées au niveau caractères, il est possible d'évaluer les performances du reconnaisseur sur la même base de caractères que celle utilisée dans la section 5.3.1.

Lettre	Apprentissage	Test	Total	%
A	28942	6229	35171	11,31 %
B	12006	1486	13492	4,34 %
C	11131	2671	13802	4,44 %
D	12507	2653	15160	4,88 %
E	36167	7118	43285	13,92 %
F	3003	617	3620	1,16 %
G	4753	1114	5867	1,89 %
H	5752	981	6733	2,17 %
I	15127	2661	17788	5,72 %
J	802	131	933	0,30 %
K	1387	306	1693	0,54 %
L	22865	4063	26928	8,66 %
M	5776	1223	6999	2,25 %
N	16091	2916	19007	6,11 %
O	13662	2523	16185	5,20 %
P	2115	406	2521	0,81 %
Q	2268	339	2607	0,84 %
R	20637	4280	24917	8,01 %
S	7911	1388	9299	2,99 %
T	13133	2359	15492	4,98 %
U	14224	2296	16520	5,31 %
V	2920	501	3421	1,10 %
W	443	79	522	0,17 %
X	1624	571	2195	0,71 %
Y	3392	553	3945	1,27 %
Z	2311	558	2869	0,92 %
Nb total de caractères	260949	50022	310971	
Nombre de documents	36509	7111	43620	

TAB. 5.3 – Distribution des lettres qui composent les champs annotés. Les lettres ne sont pas annotées en tant que caractères. Ces informations sont déduites des annotations champs.

Classe	Reco init %	Reco réentraînté %
A	98.79	92.11
B	93.23	65.43
C	97.68	91.57
D	98.20	83.20
E	98.61	92.13
F	97.09	93.10
G	97.44	74.34
H	98.64	90.73
I	98.64	38.15
J	66.21	42.01
K	90.31	52.42
L	97.95	90.73
M	99.36	85.15
N	99.47	97.24
O	98.03	92.61
P	92.19	88.51
Q	80.28	62.30
R	97.55	88.20
S	99.06	89.54
T	98.94	91.83
U	98.72	91.48
V	93.68	66.04
W	72.04	1.08
X	98.70	86.74
Y	96.86	73.50
Z	95.27	74.96
bruit de fond	63.47	18.87
sur-segmentation	**71.00**	**95.83**
sous-segmentation	**22.48**	**86.82**

TAB. 5.4 – Comparaison entre le système initial et le système réentraîné au niveau champs.

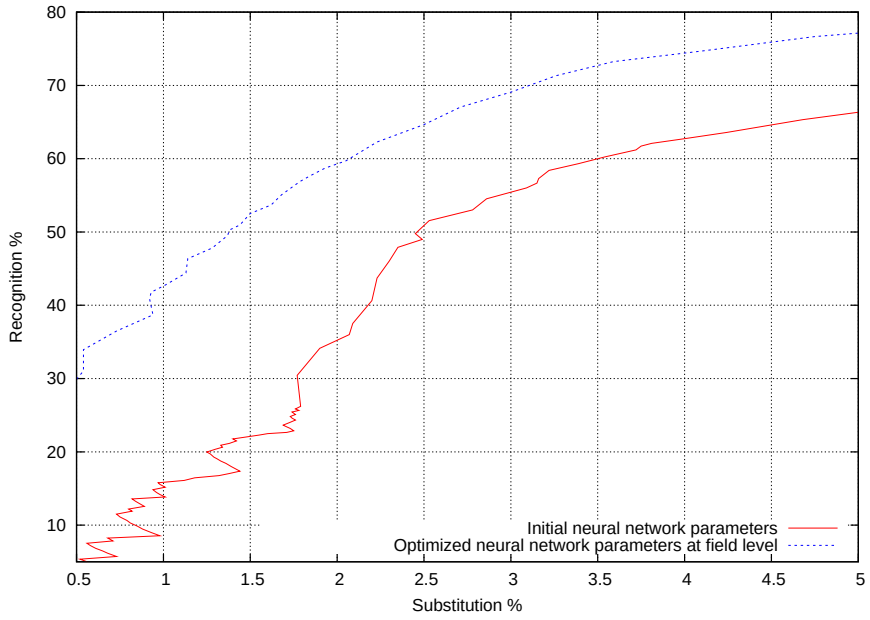


FIG. 5.11 – Reconnaissance au niveau champs. Courbe rouge : système initial. Courbe bleue : système optimisé au niveau champs.

L'optimisation au niveau champs dégrade considérablement les performances du reconnaisseur en terme de reconnaissance de caractères. Les caractères peu représentés (en particulier 'W') sont fortement dégradés. Le caractère 'I' (dont une forme proche apparaît souvent lors de sur-segmentations de 'B' ou de 'K') voit également ses performances se dégrader de façon très nette.

En revanche, les caractères de sur-segmentation et de sous-segmentation sont beaucoup mieux reconnus. Ces classes de caractères sont apprises par défaut (voir section 5.2.2.7), en prenant en compte les différentes hypothèses de segmentation du champ en lettres. De cette manière, sans annotation explicite de ces classes de caractères, le système parvient à beaucoup mieux rejeter les hypothèses de caractères qui proviennent de segmentations erronées.

5.3.2.2.2 Reconnaissance au niveau champs Les résultats donnés à la figure 5.11 montrent l'amélioration de la reconnaissance au niveau champs, bien que les performances au niveau caractères soient fortement dégradées pour la plupart des classes de lettres. L'amélioration du rejet des classes de sur-segmentation et sous-segmentation permet de compenser cette dégradation des performances au niveau caractères.

Au final, les performances du système sont améliorées sur la tâche à optimiser (la reconnaissance au niveau champs).

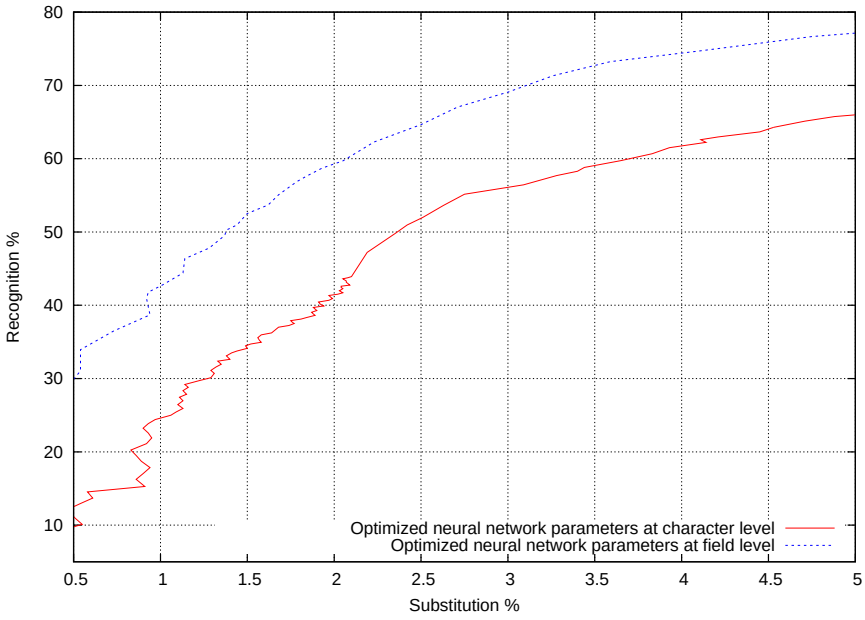


FIG. 5.12 – Reconnaissance au niveau champs. Courbe rouge : système optimisé au niveau caractères. Courbe bleue : système optimisé au niveau champs.

5.3.3 Conclusion

Les résultats donnés à la figure 5.12 montrent que le système optimisé au niveau champs (sans avoir d'annotation pour chacun des caractères pris individuellement) permet d'obtenir de meilleures performances que le même système optmisé sur des données annotées au niveau caractères, sur une tâche de reconnaissance de champs.

Ces résultats soulignent l'importance de disposer d'un système capable de bien rejeter les mauvaises hypothèses de caractères, qui proviennent des mauvais candidats de segmentation (sur-segmentations et/ou sous-segmentations). Mieux rejeter permet au final de mieux reconnaître.

5.4 Apports des N-gram

Dans cette partie, nous allons étudier l'apport d'un modèle de langage pour la tâche de reconnaissance de noms de familles. Ces expériences vérifient le fait qu'utiliser un modèle de langage permet d'améliorer significativement les performances de reconnaissance. Ces expériences montrent également que le fait d'intégrer un modèle de langage (N-gram) durant la phase d'apprentissage permet d'obtenir de meilleurs résultats que lorsque ce même modèle de langage est rajouté après l'apprentissage du reconnaisseur.

Les N-gram ont été introduits dans la section 2.5.2. Leur information peut être représentée sous forme de graphe. Les N-gram s'intègrent donc parfaitement dans le formalisme des automates à états finis pondérés.

Pour réévaluer les probabilités à l'aide du modèle de langage, il suffit de calculer

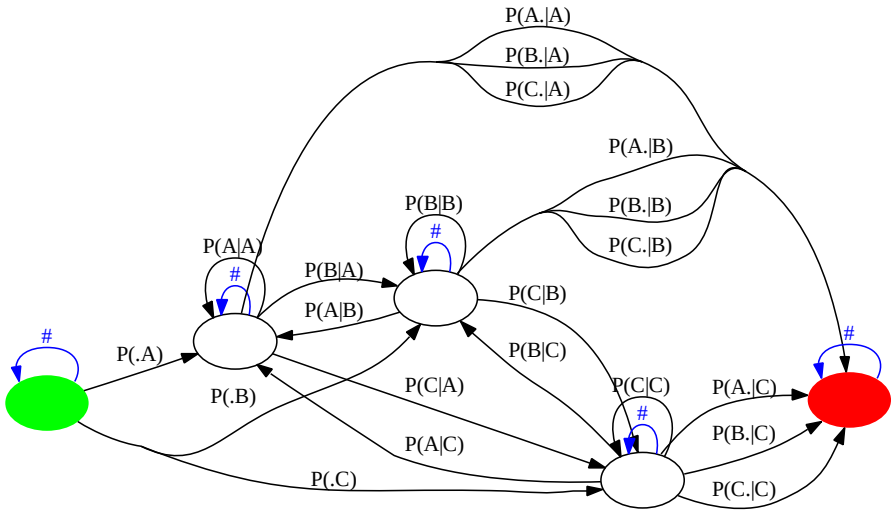


FIG. 5.13 – Exemple de modèle de bi-gram. Vert : état initial. Rouge : état final. Bleu : arcs rajoutés pour prendre en compte les bruits inter-caractères.

la composition entre le graphe de reconnaissance (voir section 5.2.2.4) et le modèle de N-grams, afin d'extraire les chemins communs aux deux graphes, et dont les arcs seront pondérés comme le produit des probabilités des deux arcs sources.

L'intégration du modèle de langage dans le système de reconnaissance est très simple : il suffit de remplacer l'automate qui permet d'obtenir le graphe semi-contraint (voir section 5.2.2.5) par le graphe qui porte l'information des N-gram. Pour permettre l'absorption des 'bruits de fond' de la même manière que le permettait l'automate, il suffit d'introduire un arc rebouclant sur le même état pour chaque état du graphe de N-gram. Le modèle de langage pourra ainsi absorber les symboles de bruit de la même façon que l'automate dans le système précédent.

Dans un modèle de bi-gram sur un alphabet de 3 symboles $\{A, B, C\}$, cette modification consisterait à rajouter les arcs bleus avec une probabilité de 1 (voir figure 5.13).

Dans notre application, un modèle de tri-gram est calculé à partir d'un lexique de 1,4 millions de noms de familles français. Ce lexique est considéré comme représentatif de la tâche à effectuer.

On peut distinguer deux types d'utilisation des N-gram

- Apprentissage du reconnaisseur sans N-gram, et ajout des N-gram durant la phase de reconnaissance.
- Apprentissage du reconnaisseur en cascade avec le modèle de N-gram. Dans cette stratégie, le coût sera rétropropagé à travers le modèle de N-gram puis à travers le reconnaisseur. Le reconnaisseur sera donc optimisé non pas isolément, mais en conjonction avec le modèle de N-gram.

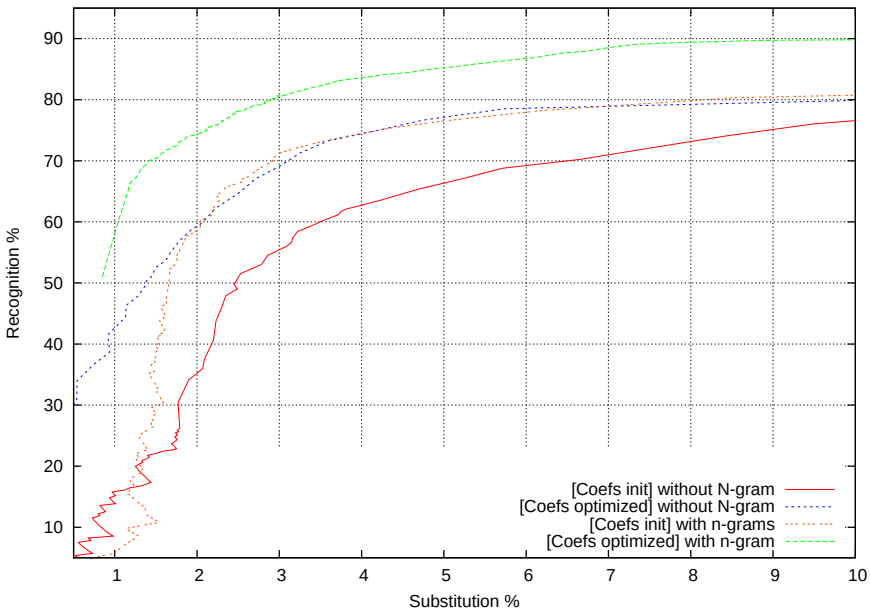


FIG. 5.14 – Courbes de Reconnaissance/Substitution : Apport du modèle de langage.

5.4.1 N-gram après optimisation

La figure 5.14 montre l'apport du modèle de langage, à la fois sur le système initial et sur le système optimisé au niveau champs.

Le modèle de langage permet d'améliorer les performances dans les deux cas.

5.4.2 Optimisation à travers les N-gram

La figure 5.15 montre que le système entraîné en cascade avec les N-gram offre de meilleures performances que le système entraîné seul auquel on rajouterait le même modèle de N-gram en cascade uniquement durant la phase de reconnaissance.

5.4.3 Conclusion sur les N-gram

Les résultats obtenus sur les modèles de langages sont cohérents avec les travaux de Lecun et al [94], ou encore ceux de G. Boulianne et al. [32]. Ils valident l'hypothèse selon laquelle il est plus profitable d'entraîner un système de façon globale plutôt que d'optimiser ses éléments pris séparément puis assemblés pour effectuer une tâche donnée.

5.5 Conclusion

Les automates à états finis pondérés constituent un cadre qui permet d'unifier les différents modules qui composent un système de reconnaissance, en les regroupant au sein d'un formalisme unique.

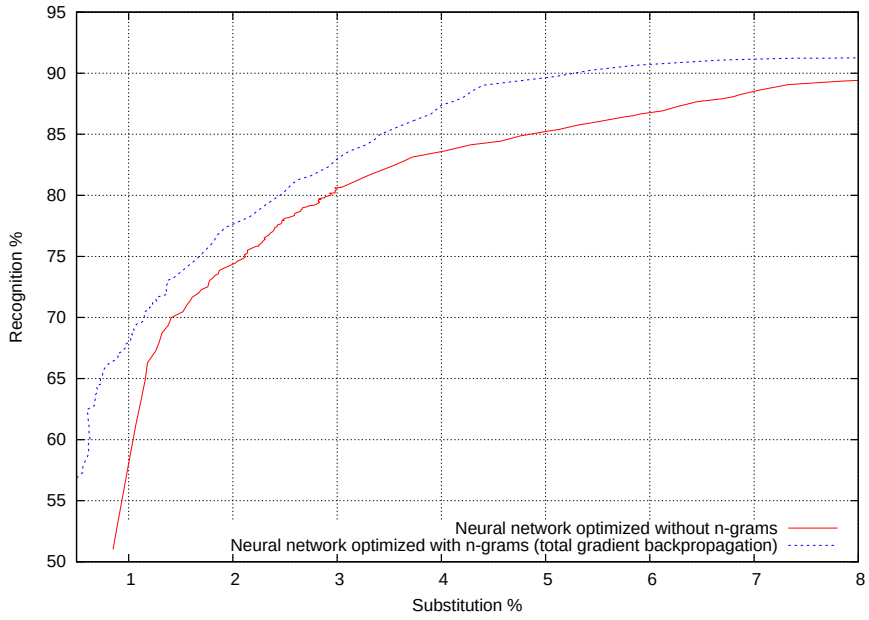


FIG. 5.15 – Courbes de Reconnaissance/Substitution. Courbe rouge : système appris seul auquel on rajoute un modèle de langage. Courbe bleue : système appris en incluant le modèle de langage.

Dans cette partie, nous avons exploité ce formalisme, en l'appliquant à une tâche de reconnaissance de champs de noms de familles français écrits en lettres capitales. Nous avons montré que ce système pouvait avantageusement être réentraîné sur cette application. Ce réapprentissage ne nécessite qu'une annotation au niveau champs, et ne requiert pas d'annotation des lettres individuelles.

Nous avons également montré que moyennant une modification mineure du système de reconnaissance, il était possible d'intégrer un modèle de langage. Le système de reconnaissance ainsi construit est plus efficace lorsque le modèle de langage est intégré durant la phase d'apprentissage, accréditant ainsi l'idée qu'il est plus profitable de chercher à optimiser le système globalement plutôt que d'essayer d'optimiser le module de reconnaissance séparément sur une sous-partie du problème.

Ces expériences valident l'utilisation d'automates à états finis pondérés (WFSA) dans le cadre d'une tâche de reconnaissance de noms de familles écrits en lettres capitales. Les techniques utilisées dans ces travaux pourront avantageusement être utilisées dans le cadre de la reconnaissance de l'écriture arabe manuscrite (voir section 4.4.2), en particulier le mécanisme de composition semble intéressant pour représenter les différentes alternatives et les contraintes linguistiques qui guident la fusion entre les corps de lettres et les signes diacritiques.

Chapitre 6

Conclusions et perspectives

Dans ce travail de thèse, nous nous sommes principalement intéressés à la reconnaissance de l'écriture arabe manuscrite. La base IFN/ENIT (noms de villes tunisiennes) a servi pour le développement et l'évaluation des travaux menés sur l'écriture arabe.

L'écriture arabe et l'écriture latine cursive ont de nombreux points communs, qui permettent le transfert vers l'arabe de techniques déjà éprouvées sur le latin. Nous nous sommes appuyés sur le système hybride à base de Modèles de Markov Cachés et de Réseau de Neurones développé par E. Augustin [17]. Des prétraitement spécifiques à l'arabe sont nécessaires (détection des signes diacritiques, détection de la bande de base), mais la segmentation en graphèmes, l'extraction de primitives et le moteur de reconnaissance sont les mêmes que ceux utilisés pour la reconnaissance de l'écriture latine.

A la différence d'autres systèmes, basés sur une approche par fenêtres glissantes, qui intègrent les signes diacritiques de façon implicite, nous avons fait le choix de traiter ces symboles explicitement. Pour cela, nous avons proposé un alphabet de corps de lettres qui permet de prendre en compte les redondances des formes des lettres de l'alphabet arabe. Nous avons ainsi montré que bien que les points soient indispensables pour différentier les lettres arabes, il est possible de proposer un système de reconnaissance sur une application à taille de vocabulaire moyenne (937 noms de villes tunisiennes) sans qu'il ne soit nécessaire de tenir compte de l'information portée par les signes diacritiques.

Dans un second temps, nous avons proposé un mécanisme pour traiter des signes diacritiques, et une stratégie pour combiner cette information avec le résultat de la reconnaissance sans diacritique. Cette combinaison permet de réduire le taux d'erreur de 15 à 25% par rapport au système sans diacritiques. Au final, le système proposé obtient les meilleures performances publiées jusqu'à présent sur cette base selon la procédure proposée dans [147] (apprentissage sur les sous-ensembles {A,B,C} et test sur {D }).

Dans la section 4.4.1.9, nous avons montré qu'une amélioration des performances du système de reconnaissance sans diacritiques se traduit par une amélioration des performances en sortie de combinaison. Ce résultat est encourageant, car les pistes d'améliorations sont nombreuses : amélioration du processus de détection des signes diacritiques, amélioration de la détection de la bande de base, amélioration de la segmentation en graphèmes, adaptation de la distance d'édition utilisée pour déterminer

les candidats de mots à partir de la séquence de signes diacritiques, amélioration de la stratégie de combinaison entre les deux listes (reconnaissance sans diacritiques et reconnaissance à l'aide des diacritiques).

Mais la piste d'améliorations qui semble de loin la plus prometteuse est celle qui a été présentée dans la section 4.4.2. La combinaison des diacritiques et des corps de lettres apporte une information locale. Les deux types d'information se renforcent mutuellement. Un bon moyen de représenter les règles de la langue (qui autorisent tels ou tels types de combinaisons entre formes de lettres et types de signes diacritiques, et qui excluent les autres), pourrait être d'utiliser des transductions dans le formalisme des automates à états finis pondérés. Les coûts de transductions entre symboles pourraient également dépendre de la proximité spatiale entre les symboles à combiner. Ce formalisme permettrait de modéliser non plus une séquence (potentiellement ambiguë) de signes diacritiques, mais plusieurs alternatives pondérées par une mesure de confiance. La transduction suivie d'un calcul de meilleur chemin permettrait d'extraire la combinaison la plus probable.

Le système développé dans le chapitre 5 pour une tâche de reconnaissance de noms de familles français écrits en lettres capitales donne un exemple de fonctionnement de ce type de systèmes. L'utilisation de ce genre de système pour combiner reconnaissance de corps de lettres et reconnaissance de signes diacritiques semble bien adaptée pour l'écriture arabe, mais également sur l'écriture latine, sur des langues ayant une grande quantité de signes diacritiques comme le suédois ou l'allemand.

Parmi les pistes d'améliorations possibles, nous pensons que cet axe de recherche de recherche est le plus prometteur. C'est dans cette direction que nous orienterons nos prochains travaux.

Annexe A

Apprentissage du système hybride MMC/RN utilisé

A.1 Reconnaissance de mots à l'aide de Modèles de Markov Cachés

A.1.1 Introduction

Un mot est représenté sous forme d'une séquence de vecteurs (issus d'une segmentation implicite ou explicite), qu'on appelle également observations. La séquence d'observations O de longueur T est définie comme :

$$O = o_1, o_2, \ldots, o_T$$

o_t représente le vecteur observé à l'instant t. Le problème de la reconnaissance de mots peut être vu comme le calcul de :

$$\arg\max_i P(w_i|O) \tag{A.1}$$

où w_i est le i-ème mot du vocabulaire. Cette probabilité n'est pas calculable directement, mais la règle de Bayes nous donne :

$$P(w_i|O) = \frac{P(O|w_i)P(w_i)}{P(O)} \tag{A.2}$$

$P(O)$ n'intervient pas dans le calcul du maximum. Par conséquent, pour un ensemble de probabilités *a priori* $P(w_i)$, le mot le plus probable dépend seulement de la vraisemblance $P(O|w_i)$

Dans la reconnaissance à l'aide de Modèles de Markov Cachés, on considère que la séquence de symboles correspondant à un mot à reconnaître est générée par le Modèle de Markov (MMC). Un MMC est une machine à états finis qui change d'état à chaque unité de temps, et à chaque entrée dans un état s_j, un vecteur o_t est généré selon une densité de probabilité $b_j(o_t)$ propre à chaque état. La transition d'un état s_i vers un état s_j est une probabilité discrète $a_{i,j}$. Deux états particuliers initialisent et terminent la chaîne de Markov. On définit ainsi l'ensemble des probabilités de transitions initiales $\prod$

et les probabilités de transitions finales T. Un modèle de Markov Cachés de N états est donc défini par :

$$A = \{a_{i,j}\} = \{p(q_t = s_j | q_{t-1} = s_i)\}$$
$$\prod = \{\pi_i\} = \{p(q_{t=1} = s_i)\}$$
$$T = \{\tau_i\} = \{a_{i,N}\}$$
$$B = \{b_i(o_t)\} = \{p(o_t | q_t = s_i)\}$$

Avec $\forall i \sum_j a_{i,j} = 1$

La figure A.1 est un exemple qui montre comment un modèle à six états génère la séquence o_1 à o_6 en passant par la séquence d'états $X = 1, 1, 2, 3, 3, 4$. Dans cet exemple, l'état initial et l'état final sont non-émetteurs.

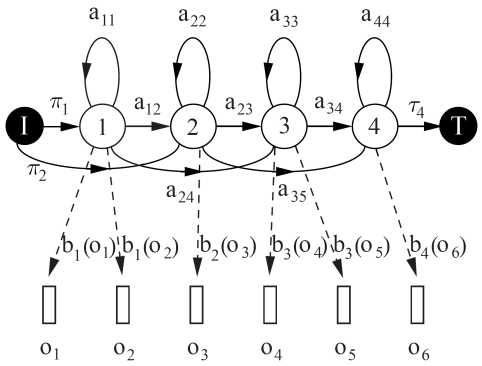


FIG. A.1 – HMM : modèles génératifs. Dans cet exemple, l'état initial (1) et l'état final (6) sont non-émetteurs.

La probabilité que le modèle M génère la séquence O à travers la séquence d'états X est donnée par :

$$P(O, X|M) = \pi_1 b_1(o_1) a_{11} b_1(o_2) a_{12} b_2(o_3) \dots$$

En pratique, la séquence d'observation O est connue, mais pas la séquence des états X. On dit que la séquence d'états est "cachée". C'est la raison pour laquelle on parle de Modèles de Markov Cachés.

Comme X est inconnu, la vraisemblance est calculée comme la somme des vraisemblances sur tous les chemins possibles :$\Gamma_T = \{(q_1, q_2, \dots, q_T)\}$

$$P(O|M) = \sum_{(q_1,\dots,q_T)\in\Gamma_T} \left(\pi_{q_1} \prod_{t=1}^{T-1} \left(b_{q_t}(o_t) a_{q_t,q_{t+1}}\right) b_{q_T}(o_T) \tau_{q_T} \right) \tag{A.3}$$

La procédure Forward, qui sera définie dans la section A.1.2, permet de calculer efficacement la quantité $P(O|M)$ de façon récursive.

La vraisemblance peut être approximée en ne considérant que la séquence d'états la plus probable :

$$\hat{P}(O|M) = \max_{(q_1,\ldots,q_T)\in\Gamma_T} \left\{ \pi_{q_1} \prod_{t=1}^{T-1} \left(b_{q_t}(o_t) a_{q_t,q_{t+1}}\right) b_{q_T}(o_T)\tau_{q_T} \right\}$$

La procédure Viterbi, qui sera définie dans la section A.1.3, permet de calculer efficacement la quantité $\hat{P}(O|M)$ de façon récursive.

La tâche de reconnaissance consiste à résoudre l'équation A.1. Etant donné un ensemble de modèles M_i correspondant aux mots w_i du vocabulaire, l'équation A.1 est résolue en utilisant l'équation A.2 et en considérant que :

$$P(O|w_i) = P(O|M_i)$$

Etant donné un ensemble d'exemples correspondant à un modèle particulier, les paramètres de ce modèle peuvent être déterminés automatiquement à travers une procédure de réestimation robuste (procédure de Baum-Welch, qui sera décrite dans la section A.1.4).

Les modèles HMMs peuvent être des modèles de mots, ou des modèles de lettres qui sont concaténés pour former des mots.

A.1.2 Forward/Backward

A.1.2.1 Forward

La somme des probabilités de tous les chemins (équation A.3) requiert de l'ordre de N^T calculs. L'algorithme Forward permet de rendre le nombre de calculs linéaire par rapport à la longueur de la séquence d'observations. On définit la fonction Forward comme étant :

$$\begin{aligned} \alpha_t(i) &= P(o_1, o_2, \ldots, o_t, q_t = s_i|M) \\ \alpha_{t+1}(j) &= \left(\sum_{i=1}^{N} \alpha_t(i) a_{i,j} \right) b_j(o_{t+1}) \\ \alpha_1(i) &= \pi_i b_i(o_1) \end{aligned}$$

A.1.2.2 Backward

De la même façon, on peut définir la fonction Backward :

$$\begin{aligned} \beta_t(i) &= P(o_{t+1}, o_{t+2}, \ldots, o_T|q_t = s_i, M) \\ \beta_t(i) &= \sum_{j=1}^{N} \beta_{t+1}(j) a_{i,j} b_j(o_{t+1}) \\ \beta_T i &= \tau_i \end{aligned}$$

La vraisemblance de la séquence complète s'exprime alors :

$$P(O|M) = \sum_i \alpha_t(i) \times \beta_t(i)$$

ou encore, en utilisant uniquement les fonctions Forward :

$$P(O|M) = \sum_i \alpha_T(i) \times \tau_i$$

A.1.3 Viterbi/Backtracking

L'algorithme de Viterbi est similaire à l'algorithme Forward, mais en remplaçant les sommes par des max. L'algorithme de Viterbi calcule la probabilité du meilleur chemin à la place de la somme sur tous les chemins. A chaque instant t, la probabilitié du meilleur chemin aboutissant à l'état j est optimisé en fonction des probabilités à l'instant précédent et des probabilités de transition a_i, j. Un pointeur $\psi_t(i)$ est gardé sur le prédécesseur de manière à pouvoir effectuer une recherche arrière du meilleur chemin (*backtracking*).

$$\delta_t(j) = \left(\max_i \delta_{t-1}(i) a_{i,j} \right) b_j(o_t)$$

$$\delta_1(j) = \pi_j b_j(o_1)$$

$$\psi_t(j) = \arg\max_i \{\delta_{t-1}(i) a_{i,j}\}$$

La recherche arrière du meilleur chemin permet d'aligner les observations o_t sur les états q_t du MMC. Le chemin se retrouve en partant de la fin et en appliquant :

$$q_{t-1} = \psi_t(q_t)$$

Lorsque qu'un modèle de mot est construit par concaténation de modèles de lettres, l'alignement viterbi permet de retrouver une segmentation en lettres comme sous-produit de la tâche de reconnaissance.

A.1.4 Algorithme EM et formules de Baum-Welch

soit K séquences d'observations

$$\left(O^k = \left(o_1^k, \ldots, o_{T_k}^k \right) \right)_{1 \leq k \leq K}$$

L'apprentissage d'une chaîne de Markov cachée consiste à trouver la solution du problème d'optimisation suivant :

$$(A_M, \pi_M, \tau_M, B_M) = \arg\max_{A,\pi,\tau,B} \underbrace{\prod_{k=1}^{K} P(o_1^k, \ldots, o_{Tk}^k | M)}_{\text{vraisemblance du modèle}}$$

$$\text{avec les contraintes} \begin{cases} \sum_i \pi_i = 1 \\ \forall j, \sum_i a_{i,j} = 1 \\ \forall j, \sum_o b_j(o) = 1 \\ \forall i, \pi_i \geq 0 \\ \forall i, \tau_i \geq 0 \\ \forall (i,j), a_{i,j} \geq 0 \\ \forall (i,o), b_i(o) \geq 0 \end{cases}$$

L'algorithme d'apprentissage des modèles de Markov cachés est basé sur les formules de Baum-Welch, qui prennent en compte les contraintes du problème (voir [19] ou [153]) Il s'agit d'un cas particulier de l'algorithme EM (Expectation-Maximisation, voir [53]).

Le principe des formules de Baum-Welch consiste à augmenter la valeur des paramètres les plus probables, et à diminuer celle de ceux peu probables. On note $l_{i,t}$ le nombre de chemins (ou séquences) partant de l'état i à l'instant t. On note $l_{i,j,t}$ le nombre de chemins partant de l'état i à l'instant t et passant à l'état j à l'instant $t+1$.

La nouvelle valeur de $\overline{a_{i,j}}$ sera :

$$\overline{a_{i,j}} = \frac{\sum_t l_{i,j,t}}{\sum_t l_{i,t}}$$

La réestimation des probabilités d'émission, ainsi que des probabilitiés initiales et terminales suit le même raisonnement.

Les formules de réestimation de Baum-Welch sont données en A.4, A.5, A.6 et A.7 :

$$\overline{a_{i,j}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k - 1} \alpha_t^k(i) a_{i,j} b_j(o_{t+1}^k) \beta_{t+1}^k(j) \right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i) \right]} \tag{A.4}$$

$$\overline{b_i(o)} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i) \gamma_{\{o_t^k = o\}} \right]}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i) \right]} \tag{A.5}$$

$$\overline{\pi_i} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{P_k} \overbrace{\alpha_1^k(i)}^{=\pi_i} \beta_1^k(i) \tag{A.6}$$

$$\overline{\tau_i} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \alpha_{T_k}^k(i) \overbrace{\beta_{T_k}^k(i)}^{=\tau_i}}{\sum_{k=1}^{K} \frac{1}{P_k} \left[\sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i) \right]} \tag{A.7}$$

avec :

$P_k = P(o_1^k, \ldots, o_{T_k}^k)$

$\alpha_t^k(i) = P(o_1^k, \ldots, o_t^k, q_t = s_i | M)$

$\beta_t^k(i) = P(o_{t+1}^k, \ldots, o_{T_k}^k | q_t = s_i, M)$

$\gamma_{\{x=y\}} \begin{cases} 1 \text{ si } x = y \\ 0 \text{ sinon} \end{cases}$

A.2 Modèle hybride MMC/RN utilisé

La section précédente présentait l'apprentissage des modèles de Markov cachés dans le cas général. Nous allons maintenant décrire l'architecture plus particulière du système hybride utilisé dans cette thèse. Il s'appuie sur le système développé par E. Augustin [17] pour la reconnaissance de l'écriture cursive latine.

A.2.1 Architecture

Le reconnaisseur est un système hybride MMC/RN. Le MMC délègue l'estimation des probabilités d'observation au réseau de neurones.

L'architecture est donnée figure A.2. Un réseau de neurones de type Perceptron Multi-Couches à N sorties calcule des probabilités a posteriori : $P(C_i|o_t)_{1 \leq i \leq N}$. Les C_i sont les sorties du réseau de neurones (la couche de sortie est de type Softmax). Or les $b_i(o_t)$ des HMM sont des vraisemblances. La règle de Bayes donne :

$$P(O_t|C_i) = \frac{P(C_i|o_t)P(o_t)}{P(C_i)}$$

$P(C_i)$ est la probabilité a priori de la classe C_i. $P(o_t)$ est indépendant de la classe, il peut donc être ignoré pour la classification.

On utilise des vraisemblances normalisées (scaled likelihoods) :

$$\tilde{P}(o_t|C_i) = \frac{\frac{P(C_i|o_t)}{P(C_i)}}{\sum_{j=1}^{N} \frac{P(C_j|o_t)}{P(C_j)}}$$

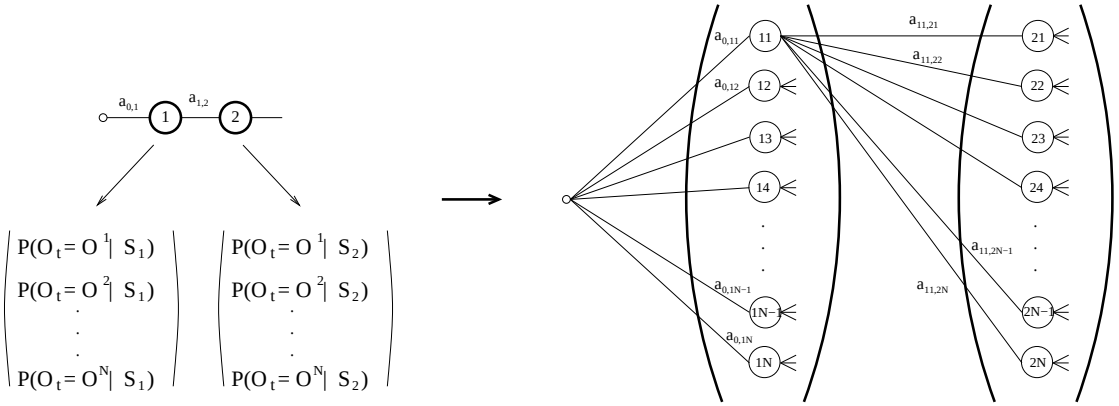


FIG. A.2 – HMM à N classes d'observations : modèle standard et modèle hybride.
A gauche : HMM standard. Les probabilité d'émission des observations dépendent des états : $\{b_j(o)\}$
A droite : HMM hybride. Chaque état du HMM initial donne lieu à une colonne de N états. Chacun des états de cette colonne ne peut émettre qu'une seule classe d'observation. Les probabilités d'émissions sont déléguées à un réseau de neurones unique.

A.2.2 Apprentissage

L'apprentissage du HMM et du RN se fait de manière séparée. Il s'agit d'un apprentissage en mode batch en quatre étapes :

1. Décoder les bases de mots avec le système RN + MMC pour créer une base de vecteurs caractéristiques annotés pour le RN.
2. Entraîner le RN par rétropropagation du gradient (gradient stochastique).
3. Utiliser le nouveau RN pour calculer les probabilités d'observation.
4. Optimiser les probabilités de transition des états des MMC par l'algorithme de Baum-Welch.

Ce processus itératif est répété sur l'ensemble de la base d'apprentissage jusqu'à saturation des performances.

L'étape n°1 consiste à utiliser le système hybride pour annoter le réseau de neurones. On distingue deux approches, qui seront détaillées dans la section suivante :
– l'alignement Viterbi, qui n'exploite que la séquence d'états qui composent le meilleur chemin.
– l'utilisation des probabilités forward-backward, qui permettent de redistribuer les cibles des probabilités a posteriori sur plusieurs classes et non plus sur une seule.

A.2.2.1 Viterbi

Dans le cas de l'annotation Viterbi, la séquence d'observations est décodée à l'aide du modèle correct. Le meilleur chemin est extrait, et chacune des observations o_t est ainsi alignée sur l'état correspondant dans le meilleur chemin. Pour chaque observation o_t de la séquence, on déduit donc la classe cible de l'annotation comme étant l'indice de l'état

q_t du meilleur chemin dans la colonne correspondante. La base d'entraînement du réseau de neurones est construite de cette manière sur l'ensemble des séquences d'observations correspondant à tous les mots de la base d'entraînement.

A.2.2.2 Forward-Backward

Plusieurs auteurs [196, 165, 17] ont proposé d'entraîner le réseau de neurones non pas avec un cible de 1 pour la classe la plus probable et 0 pour toutes les autres ("hard classification"), mais plutôt de partager cette probabilité sur plusieurs classes ("soft classification"). Tous confirment un gain de reconnaissance global grâce à une meilleure estimation des probabilités locales.

Par définition, on a :

$$\alpha_t(j)\beta_t(j) = P(O, q_t = s_j|M)$$

Or :

$$P(O, q_t = Sj|M) = P(q_t = s_j|O, M) \times P(O|M)$$

Donc :

$$P(q_t = s_j|O, M) = \frac{P(O, q_t = s_j|M)}{P(O|M)} = \frac{\alpha_t(j)\beta_t(j)}{P(O|M)} = \frac{\alpha_t(j)\beta_t(j)}{\sum_k \alpha_t(k)\beta_t(k)}$$

En désignant les états non plus par leur identifiant, mais par leur colonne et par leur indice dans la colonne considérée, l'état k_l correspond au l-ième état de la k-ième colonne :

$$P(q_t = s_{i_j}|O, M) = \frac{\alpha_t(i_j)\beta_t(i_j)}{\sum_{k=1}^{NbCol}\sum_{l=1}^{NbObs} \alpha_t(k_l)\beta_t(k_l)}$$

Où $NbCol$ est le nombre de colonnes, et $NbObs$ le nombre d'états par colonne.

L'annotation de la j-ième sortie du réseau de neurones est obtenue comme étant la somme sur toutes les colonnes :

$$Target(o_t)_j = \sum_{i=1}^{NbCol} P(q_t = s_{i_j}|O, M)$$

Annexe B

Initialisation du réseau de neurones du système hybride

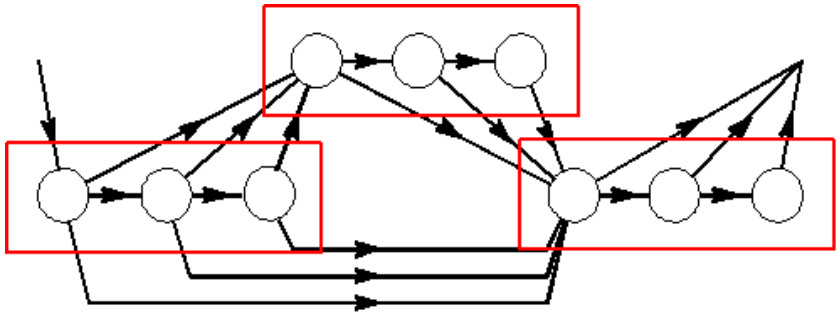


FIG. B.1 – Exemple de topologie contrainte pour l'initialisation du réseau de neurones (figure extraite de [17]).

Dans [17], E. Augustin utilisait des MMC discrets, dont la topologie contrainte est donnée figure B.1, pour annoter le réseau de neurones du premier système hybride RN + MMC. L'apprentissage du système hybride RN + MMC est ensuite poursuivi selon la précdure présentée dans l'annexe précédente (voir section A.2.2).

Ici, nous proposons d'initialiser directement le système hybride sans avoir recours à un jeu de MMC discrets. Les MMC sont initialisés de façon uniforme. En ce qui concerne le réseau de neurones, deux approches sont abordées :

- Une initialisation uniforme : le réseau de neurones est entraîné de telle sorte que les cibles soient identiques pour toutes les sorties (les valeurs désirées sont égales à $1/N$, où N est le nombre de neurones sur la couche cachée), pour tous les graphèmes de la base d'apprentissage.
- Une initialisation à l'aide d'un K-means : un premier clustering automatique est calculé sur les graphèmes de la base d'apprentissage. Le résultat de ce clustering est utilisé pour annoter le réseau de neurones, dont la fonction de transfert sera entraînée pour répliquer le résultat du K-means (une cible de 1 pour la classe désirée, et des cibles de 0 pour tous les autres). Nous verrons qu'un apprentissage complet

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	1,25	23,78	22,82	21,77	22,87	22,88	1,25	22,02	21,86	19,93	19,75	19,70
	2,32	31,67	30,84	30,37	32,09	32,06	2,52	29,95	29,47	27,16	27,75	27,75
	9,21	52,96	52,40	51,68	54,10	54,11	9,58	52,38	50,79	48,57	49,61	49,58
50	1,29	34,70	41,92	47,30	49,47	49,45	1,28	33,73	39,96	43,30	43,99	43,99
	2,63	43,23	52,40	57,77	60,05	60,05	2,63	42,21	49,37	53,33	54,45	54,45
	9,37	63,87	73,86	78,20	80,43	80,43	9,34	62,78	71,18	74,58	74,83	74,85
75	1,53	37,69	45,43	51,55	54,17	54,21	1,59	36,88	42,73	47,01	48,05	48,09
	2,83	46,64	55,93	61,99	64,02	64,05	2,95	45,33	51,94	57,18	58,46	58,47
	9,82	67,35	76,81	81,42	83,63	83,64	9,78	65,92	73,87	77,34	78,34	78,37
100	2,03	27,72	22,00	20,47	20,43	20,05	2,06	27,35	21,25	18,63	18,38	16,96
	3,52	36,69	30,35	28,13	28,31	27,77	3,79	36,08	29,09	26,44	25,57	24,50
	11,29	58,31	51,92	49,43	48,64	47,77	11,71	57,54	50,24	46,53	44,53	42,90
150	2,40	53,29	55,41	54,27	53,15	50,92	2,63	51,21	51,31	49,25	46,76	43,59
	4,08	62,87	65,38	64,53	63,17	61,11	4,02	60,59	61,38	59,33	56,59	53,01
	12,90	80,40	83,15	82,23	81,20	79,51	13,05	78,19	80,52	78,35	76,35	73,72
200	3,12	62,32	73,19	76,58	77,43	77,46	3,25	60,10	69,99	72,22	72,10	72,04
	5,07	71,16	80,97	83,48	84,33	84,34	5,00	68,64	77,79	79,64	79,67	79,64
	14,45	84,76	91,71	93,71	94,28	94,28	14,70	83,18	90,01	91,43	91,33	91,37

TAB. B.1 – Initialisation uniforme. 200 neurones sur la couche cachée.

du réseau de neurones n'est pas nécessaire : un entraînement arrêté prématurément fourni une initialisation suffisamment bonne pour permettre un bon déroulement du processus itératif d'adaptation du système hybride RN + MMC.

Les expériences sont menées avec différentes topologies du réseau de neurones :

- 35, 50, 75, 100, 150 et 200 neurones de sortie (qui correspondent au nombre de classes d'observations des MMC).
- 200, 350, 500 et 650 neurones sur la couche cachée.

Les taux de reconnaissance sur la base d'apprentissage et sur la base de test sont donnés en première, deuxième et dixième position pour chacune des 6 itérations (It0 à It5) qui composent un apprentissage.

B.1 Initialisation Uniforme

Dans le cas d'une initialisation uniforme des sorties du réseau de neurones (voir tableaux B.1 B.2 B.3 et B.4), le taux de reconnaissance du système hybride est extrêmement bas (1 à 2%). L'apprentissage reste très erratique : il peut donner de bons résultats, ou au contraire tomber rapidement dans un minimum local. Cette approche n'est pas satisfaisante.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	1,11	44,10	60,00	62,18	62,37	62,37	1,14	41,89	56,42	57,65	55,92	55,93
	1,95	53,62	69,36	71,90	72,36	72,36	2,05	51,88	66,25	67,13	65,46	65,48
	8,46	71,26	84,20	86,35	86,70	86,70	8,83	70,01	81,66	83,27	82,18	82,18
50	1,22	38,51	38,44	32,84	29,45	30,36	1,31	36,81	34,80	28,88	25,00	24,78
	2,29	47,55	47,48	41,27	37,43	38,74	2,38	45,36	43,39	35,98	31,82	31,45
	9,27	69,70	69,93	63,23	58,47	59,72	9,29	67,33	66,35	57,05	50,42	50,96
75	1,47	56,75	54,96	55,98	57,84	57,87	1,53	54,77	51,18	49,52	49,06	49,12
	2,68	65,27	64,44	65,99	66,98	67,01	2,90	63,44	60,61	59,38	58,19	58,23
	9,48	80,54	81,70	82,61	83,53	83,54	9,56	78,22	78,50	77,25	76,76	76,75
100	1,74	20,21	6,03	6,71	6,95	6,98	1,78	19,20	5,67	5,82	5,54	5,57
	3,05	27,76	9,42	10,30	10,74	10,76	3,13	25,30	8,37	8,49	8,33	8,33
	10,23	49,26	20,66	21,57	22,20	22,36	10,14	46,92	18,60	18,26	18,32	18,52
150	1,63	38,76	28,24	17,80	13,70	13,01	1,59	36,24	25,11	15,17	11,15	10,17
	2,88	48,61	37,47	24,88	18,91	18,06	2,97	46,04	33,65	20,76	15,63	14,28
	10,58	69,65	59,35	46,43	36,81	35,52	10,76	67,33	54,85	40,99	31,82	30,02
200	2,23	48,30	34,65	33,34	33,40	33,41	2,24	46,31	31,24	29,00	27,91	27,91
	3,69	58,10	44,40	43,22	43,18	43,19	3,65	56,10	40,15	37,21	35,61	35,61
	11,98	76,80	66,20	64,84	64,41	64,42	12,09	74,51	61,35	58,08	56,27	56,27

Tab. B.2 – Initialisation uniforme. 350 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	1,17	25,51	16,01	14,48	12,14	8,72	1,13	24,69	15,35	12,81	10,20	6,53
	2,12	33,82	23,32	21,22	17,90	13,31	2,26	33,08	22,33	18,49	14,49	10,19
	8,98	54,96	43,51	39,25	33,93	27,43	9,44	53,45	40,62	35,12	29,52	22,88
50	1,28	56,42	67,22	72,30	74,89	74,87	1,35	53,67	63,27	66,70	67,93	67,93
	2,66	65,80	75,67	80,26	82,38	82,37	2,70	62,51	71,98	75,47	76,63	76,61
	9,41	80,79	88,33	91,38	92,83	92,83	9,58	79,01	85,88	87,84	88,51	88,49
75	1,46	56,84	65,62	67,12	69,02	69,02	1,46	54,64	61,34	61,53	61,28	61,28
	2,65	66,13	74,65	75,98	77,12	77,12	2,72	63,61	71,02	70,60	70,16	70,14
	9,06	81,12	87,69	88,75	89,54	89,54	9,34	79,55	85,30	84,86	84,45	84,45
100	1,58	49,37	52,10	49,87	50,01	49,84	1,77	46,67	48,05	43,07	41,63	39,61
	2,83	58,25	62,35	60,22	60,27	60,10	2,91	55,46	57,91	53,65	51,79	49,31
	9,51	75,77	80,41	79,27	79,22	79,06	9,55	73,76	76,90	74,37	72,22	70,57
150	1,48	73,16	84,01	87,05	87,98	87,99	1,63	70,36	80,68	81,92	80,85	80,88
	2,71	80,78	89,60	92,05	92,73	92,75	2,84	78,34	86,44	87,85	87,20	87,25
	10,58	91,16	95,84	96,97	97,22	97,22	10,45	89,52	94,33	95,03	94,48	94,49
200	2,07	35,66	27,24	26,04	25,83	25,82	2,24	32,50	23,86	20,68	19,57	19,55
	3,79	45,26	36,55	33,95	34,22	34,23	3,83	42,32	32,69	28,23	26,43	26,43
	11,75	65,26	56,98	54,99	55,83	55,74	12,28	62,35	51,61	47,72	47,02	46,99

Tab. B.3 – Initialisation uniforme. 500 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	1,10	27,09	29,00	28,98	27,39	27,39	1,11	26,61	27,19	26,13	23,79	23,79
	2,08	36,81	38,03	38,25	36,26	36,26	2,11	35,95	35,80	35,12	32,01	32,01
	8,51	59,97	60,79	59,42	56,48	56,48	8,67	59,41	58,90	55,98	52,26	52,26
50	1,38	61,76	71,85	73,77	74,98	75,04	1,25	58,84	67,91	68,45	67,17	67,16
	2,46	70,45	80,16	82,03	82,87	82,86	2,49	67,41	76,91	76,75	75,96	76,02
	9,50	84,31	91,41	92,61	93,26	93,26	9,99	81,77	89,10	88,98	88,37	88,36
75	1,35	59,14	72,01	75,57	77,30	77,32	1,40	56,17	67,81	69,70	68,89	68,91
	2,65	68,38	79,72	83,24	84,61	84,61	2,70	65,79	76,10	78,19	77,22	77,22
	8,99	83,54	91,21	93,28	94,09	94,09	9,09	81,68	88,98	89,89	89,37	89,34
100	1,67	49,49	42,22	42,28	43,45	43,46	1,78	46,90	37,67	35,40	33,94	33,96
	3,05	59,11	52,01	52,87	53,78	53,78	3,18	56,32	47,25	45,42	43,76	43,77
	9,76	77,53	72,16	72,04	73,33	73,33	10,10	74,57	66,79	64,84	64,20	64,22
150	1,74	57,41	64,10	69,56	71,91	71,91	1,71	54,88	59,97	63,88	63,71	63,73
	2,99	66,66	73,49	78,60	80,75	80,74	3,13	64,42	69,76	73,47	73,32	73,32
	10,73	83,00	87,76	91,20	92,57	92,57	10,94	80,98	85,00	87,65	87,77	87,77
200	2,07	64,02	62,30	60,72	59,80	60,09	2,09	60,76	57,12	54,33	51,57	49,81
	3,50	72,54	71,29	70,02	69,04	69,50	3,74	70,04	66,10	63,39	60,10	58,78
	12,07	86,71	86,10	85,28	84,91	85,33	12,43	84,74	82,41	80,19	78,04	76,82

TAB. B.4 – Initialisation uniforme. 650 neurones sur la couche cachée.

B.2 Initialisation à l'aide d'un K-means

Le réseau de neurones est entraîné de telle sorte que ses sorties répliquent la classification d'une classification non-supervisée sur l'ensemble d'apprentissage.

Les tableaux B.5, B.6, B.7 et B.8 montrent les résultats des expériences dans lesquelles le réseau de neurones est entraîné pour reproduire la fonction de classification du K-means de la manière la plus proche possible (95% à 100% de taux de reconnaissance étant donné l'annotation du K-means). Les tableaux B.9, B.10, B.11 et B.12 montrent les résultats des expériences dans lesquelles l'initialisation du réseau de neurones est faite en "early stopping" : l'entraînement est arrêté beaucoup plus tôt, le réseau de neurones a un taux de reconnaissance de 80 à 90% étant donné l'annotation du K-means. Les expériences montrent que cette initialisation grossière du réseau de neurones en fonction du K-means est suffisante pour permettre un bon déroulement du processus itératif d'adaptation du système hybride RN + MMC.

Plusieurs de ces configurations offrent de meilleures performances que celles décrites dans la section 4.2.4.2. En particulier, les performances du système en gras dans le tableau B.12 sont discutées dans la section 4.4.1.9.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
	50,43	73,16	78,77	81,58	82,80	82,45	49,06	70,14	74,83	77,45	78,41	78,16
35	59,47	81,49	86,24	88,64	89,32	88,68	58,68	78,87	82,69	85,06	85,06	84,81
	76,68	92,41	94,78	95,89	95,92	95,72	75,92	90,59	92,59	93,44	93,47	93,16
	56,27	79,88	84,77	86,71	87,71	88,32	54,89	76,90	80,80	82,32	83,09	83,61
50	64,78	86,60	90,54	92,09	92,74	92,90	64,01	83,73	87,07	88,06	89,01	89,28
	80,88	94,88	96,43	96,90	97,00	97,12	81,29	93,17	94,71	95,07	95,17	95,06
	59,96	84,44	88,30	89,76	90,40	90,71	59,52	81,72	85,09	86,12	86,79	86,82
75	68,46	90,36	93,16	94,17	94,48	94,68	68,51	88,11	90,59	91,24	91,45	91,22
	83,77	96,32	97,46	97,78	97,85	97,91	84,62	95,13	96,47	96,53	96,64	96,78
	57,99	85,79	89,44	90,11	89,54	88,77	56,29	83,06	86,10	86,19	84,94	84,39
100	67,00	91,19	93,78	94,24	93,98	93,57	65,94	88,64	91,12	91,34	90,56	89,68
	82,89	96,36	97,36	97,61	97,45	97,42	82,90	95,10	96,21	96,32	95,77	95,72
	63,34	88,32	90,85	91,28	90,09	87,79	62,11	85,97	88,42	87,62	85,61	83,12
150	71,59	92,82	94,67	94,74	93,85	92,44	71,14	91,09	92,46	92,16	90,63	88,76
	85,50	96,99	97,68	97,84	97,62	96,91	85,46	96,21	96,82	96,59	95,81	94,86
	65,39	89,61	91,16	91,58	91,58	90,83	63,70	86,71	88,00	88,26	87,38	85,69
200	73,67	93,40	94,62	95,04	94,92	94,37	72,83	91,40	92,34	92,32	91,69	90,42
	86,70	97,20	97,70	97,94	97,87	97,68	86,61	96,48	96,82	96,96	96,38	95,98

TAB. B.5 – Initialisation avec K-means, apprentissage complet. 200 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
	48,99	72,98	78,76	81,38	82,69	82,41	47,31	69,76	74,89	77,05	77,92	77,09
35	58,23	81,26	86,46	88,20	89,24	89,29	56,91	78,49	82,66	84,32	84,57	84,20
	75,09	92,12	95,14	95,55	95,93	96,03	74,33	90,56	92,62	93,01	93,59	93,39
	54,92	80,08	85,22	86,97	88,24	89,32	54,27	77,22	81,60	82,48	82,81	83,07
50	63,58	86,81	90,87	92,26	93,09	93,94	63,06	84,34	87,63	88,36	88,89	89,15
	79,76	94,94	96,58	97,20	97,52	97,70	80,74	93,56	95,03	95,41	95,44	95,25
	57,32	84,16	88,05	89,51	90,47	91,18	56,61	80,94	84,16	85,18	85,17	85,49
75	65,64	89,78	92,63	93,74	94,33	94,76	65,97	87,10	89,65	90,39	90,59	90,66
	81,88	95,90	97,13	97,58	97,85	98,01	82,72	94,67	95,75	95,95	96,15	96,32
	57,89	86,40	90,07	91,15	91,86	92,14	56,44	83,44	85,79	86,40	86,12	85,86
100	66,89	91,44	94,24	95,12	95,56	95,62	65,30	89,04	91,17	91,55	91,39	91,17
	82,69	96,48	97,71	97,97	98,15	98,16	81,81	95,41	96,29	96,41	96,41	96,21
	62,56	89,16	92,26	92,83	92,12	90,59	60,53	86,06	88,20	87,57	85,86	83,47
150	70,81	93,43	95,60	95,90	95,57	94,44	69,86	90,69	92,34	91,79	90,63	88,67
	85,37	97,28	98,23	98,34	98,21	97,82	84,87	96,32	96,79	96,76	95,95	94,88
	64,54	90,11	92,66	93,20	93,36	93,35	63,96	86,27	88,17	87,53	87,07	86,24
200	73,01	93,91	95,72	96,17	96,29	96,31	72,58	91,39	92,40	92,04	91,66	90,97
	86,12	97,46	98,19	98,33	98,15	98,38	86,15	96,32	96,93	96,70	96,36	96,30

TAB. B.6 – Initialisation avec K-means, apprentissage complet. 350 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
	48,30	71,99	77,95	80,99	82,42	82,66	46,58	68,92	74,08	76,96	78,20	78,28
35	57,05	80,54	85,55	87,78	88,85	88,95	56,21	77,49	82,15	83,85	84,84	84,75
	75,08	91,98	94,50	95,45	95,83	95,96	74,27	89,80	92,38	93,18	93,54	93,33
	54,56	80,12	84,74	86,70	87,60	88,07	53,48	77,00	80,80	81,96	82,67	82,82
50	63,11	86,59	90,39	91,81	92,54	92,97	62,79	83,99	87,29	88,12	88,78	88,46
	79,15	94,52	96,20	96,81	97,19	97,34	79,67	93,18	94,61	94,83	94,98	94,79
	56,71	83,85	87,92	89,76	90,35	91,18	56,44	80,79	84,29	85,64	86,04	86,30
75	65,47	89,40	92,37	93,71	94,16	94,69	65,24	87,50	90,20	90,79	90,85	90,75
	81,79	95,77	97,04	97,52	97,60	97,81	81,86	94,91	95,84	96,17	96,08	96,10
	56,44	86,20	90,12	91,36	91,89	92,20	55,20	83,01	86,34	86,77	87,20	86,95
100	65,60	91,36	94,10	94,91	95,28	95,49	64,62	88,48	91,66	91,80	91,85	91,60
	81,85	96,61	97,57	97,94	98,07	98,14	81,71	95,19	96,32	96,48	96,44	96,20
	61,42	89,14	92,14	92,70	92,02	90,90	60,86	86,19	88,30	87,72	86,31	84,57
150	69,98	93,15	95,50	95,85	95,45	94,48	69,77	90,68	92,35	92,25	91,12	89,58
	84,49	97,10	98,15	98,29	98,18	97,92	84,57	96,38	96,84	96,60	96,14	95,29
	64,36	89,86	92,47	93,31	93,50	92,84	62,85	86,52	88,05	88,12	87,17	85,14
200	72,77	93,88	95,64	96,19	96,25	95,68	71,67	91,37	92,18	92,55	91,57	90,11
	86,26	97,35	98,15	98,38	98,45	98,24	85,95	96,27	96,78	96,78	96,42	95,58

TAB. B.7 – Initialisation avec K-means, apprentissage complet. 500 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
	47,70	72,00	77,89	80,82	81,50	81,34	46,33	69,27	74,08	76,78	77,45	77,46
35	56,52	80,03	85,28	87,72	88,18	87,84	55,53	77,82	82,14	84,13	84,32	84,10
	74,27	91,83	94,46	95,48	95,67	95,32	73,79	89,86	92,28	93,29	93,20	93,13
	54,13	79,13	83,83	85,61	86,66	87,44	53,59	76,15	80,24	81,84	81,87	82,66
50	63,12	86,08	89,57	90,99	91,80	92,48	62,45	83,40	86,64	87,90	88,14	88,46
	79,31	94,27	95,90	96,41	96,62	96,85	79,33	93,05	94,37	94,57	94,67	94,70
	56,33	83,17	87,45	89,15	89,97	90,62	56,21	80,43	83,79	85,06	85,89	86,15
75	64,74	89,19	92,32	93,38	94,10	94,28	65,26	86,93	89,96	90,56	90,73	90,73
	80,99	95,59	97,02	97,42	97,57	97,67	81,62	94,64	95,95	96,12	96,27	96,08
	56,73	85,79	89,25	90,66	91,21	91,65	55,71	82,78	85,67	86,55	86,98	86,28
100	65,63	91,14	93,72	94,55	95,00	95,18	64,87	88,70	90,94	91,51	91,34	90,99
	82,05	96,33	97,45	97,77	97,90	98,03	82,09	95,40	96,21	96,41	96,26	96,15
	62,36	89,34	92,00	91,96	90,77	88,66	60,30	86,65	88,57	87,48	84,90	81,69
150	70,72	93,39	95,35	95,37	94,66	93,34	69,50	91,55	92,78	91,97	90,26	87,72
	85,49	97,45	98,22	98,30	97,92	97,37	84,80	96,70	97,22	96,76	96,01	94,65
	62,66	89,46	91,89	92,59	93,07	93,14	61,87	86,34	88,17	87,88	87,71	87,36
200	71,22	93,46	95,27	95,86	96,21	96,16	70,26	91,31	92,32	92,20	91,89	91,37
	85,12	97,17	98,00	98,22	98,31	98,30	84,77	96,21	96,78	96,81	96,72	96,26

TAB. B.8 – Initialisation avec K-means, apprentissage complet. 650 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	68,57	81,29	83,61	83,82	84,02	84,11	68,15	79,17	80,70	80,39	79,90	79,67
	77,41	87,85	89,57	89,83	90,07	90,10	76,96	86,31	87,14	87,11	86,65	86,13
	90,18	95,53	96,34	96,51	96,42	96,27	89,81	94,48	94,60	94,28	94,14	94,02
50	73,20	85,65	88,46	89,61	90,34	90,34	72,16	83,88	86,40	86,58	87,05	86,90
	81,17	90,93	92,94	93,56	94,20	94,35	79,97	89,31	91,14	91,43	91,85	91,45
	91,74	96,41	97,25	97,51	97,77	97,68	91,46	95,49	95,99	96,32	96,36	96,24
75	74,87	87,88	89,95	90,50	91,01	90,89	74,85	86,37	87,60	87,44	87,29	86,61
	82,68	92,68	94,02	94,20	94,52	94,72	82,42	91,40	92,25	91,63	91,45	91,27
	92,49	97,02	97,64	97,73	97,91	98,04	92,10	96,41	96,66	96,64	96,47	96,17
100	74,31	89,47	91,44	91,57	91,55	91,16	74,24	87,96	89,53	88,76	88,06	86,99
	82,15	93,65	94,90	94,93	94,79	94,62	81,75	92,64	93,42	92,86	92,15	91,48
	92,20	97,46	97,95	98,11	98,01	98,05	92,13	97,09	97,42	97,16	96,73	96,69
150	73,45	88,50	90,57	91,23	91,36	91,01	73,30	86,90	88,70	88,11	87,50	86,34
	80,99	92,97	94,28	94,71	94,82	94,68	80,49	91,66	92,55	92,41	92,12	91,37
	91,57	97,02	97,72	97,94	97,95	97,92	91,36	96,76	97,19	97,15	96,84	96,63
200	69,46	88,99	91,14	91,65	91,72	91,68	69,47	87,48	88,92	88,69	87,72	87,62
	77,90	93,32	94,85	95,03	95,10	95,13	77,15	92,06	92,58	92,69	92,19	92,07
	90,12	97,41	97,91	98,04	98,03	98,14	89,74	96,87	97,15	96,94	96,88	96,73

TAB. B.9 – Initialisation avec K-means, apprentissage avec early stopping. 200 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	68,12	80,73	83,93	84,78	84,67	84,47	67,42	78,59	81,13	81,47	80,59	79,66
	76,91	87,27	89,70	90,24	90,32	90,28	76,04	85,75	87,44	87,72	86,90	86,38
	89,77	95,19	96,23	96,57	96,53	96,58	89,61	94,25	95,04	94,89	94,40	94,24
50	73,32	85,09	88,02	89,19	90,42	90,80	72,26	82,95	85,66	85,69	85,81	85,95
	81,06	90,57	92,61	93,52	94,18	94,69	79,81	89,00	90,69	90,91	91,12	90,97
	91,69	96,37	97,18	97,44	97,78	97,81	91,51	95,41	95,99	96,10	96,27	96,12
75	75,17	88,12	89,98	90,93	91,19	91,54	75,06	86,62	87,20	87,33	87,17	86,83
	82,55	92,69	93,91	94,49	94,80	94,98	82,73	91,40	92,43	92,23	91,67	91,43
	92,66	97,09	97,58	97,79	97,92	97,94	92,59	96,35	96,63	96,72	96,57	96,63
100	76,49	89,50	91,39	91,74	91,73	91,33	75,95	87,66	88,36	88,27	87,01	85,81
	83,80	93,90	94,83	95,12	95,10	95,06	83,49	92,47	92,89	92,43	91,46	90,51
	93,06	97,53	98,07	98,21	98,24	98,21	92,81	97,00	97,10	96,87	96,67	96,15
150	76,73	90,32	92,05	92,75	92,77	92,70	75,96	88,52	89,56	89,44	88,18	86,99
	83,74	93,96	95,19	95,80	95,67	95,71	83,31	92,72	93,17	92,98	92,19	91,48
	92,85	97,48	98,04	98,28	98,27	98,25	92,62	96,91	97,37	97,03	96,84	96,51
200	75,68	90,03	91,28	91,72	91,45	91,39	75,43	88,03	88,40	87,31	85,97	85,15
	82,80	93,97	94,93	94,93	94,93	94,90	82,54	92,35	92,59	91,36	90,68	90,26
	92,51	97,48	97,95	98,03	97,95	97,99	92,47	96,99	97,05	96,79	96,10	95,87

TAB. B.10 – Initialisation avec K-means, apprentissage avec early stopping. 350 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	66,68	81,31	83,84	84,88	84,81	84,92	66,15	79,90	81,84	82,38	81,84	81,10
	75,51	87,96	90,01	90,79	90,88	90,84	75,32	86,52	88,17	88,63	88,42	87,74
	88,94	95,17	96,30	96,65	96,65	96,69	88,31	94,58	95,50	95,65	95,12	95,06
50	65,79	84,70	88,16	88,54	88,49	88,71	65,61	82,87	85,73	85,57	85,02	84,88
	75,08	90,51	92,79	93,17	93,33	93,54	74,27	89,00	90,91	90,94	90,39	90,27
	88,74	96,16	97,17	97,55	97,58	97,68	88,18	95,78	96,05	96,20	96,20	95,99
75	65,09	84,04	87,65	88,96	89,77	90,49	63,82	81,80	85,05	85,63	86,07	86,21
	73,53	89,59	92,56	93,42	94,03	94,52	73,26	88,06	90,45	90,71	91,12	91,20
	86,85	95,87	97,00	97,45	97,75	97,83	86,92	94,88	95,87	96,21	96,33	96,24
100	67,88	87,41	90,15	91,03	91,58	91,57	66,68	84,86	87,45	87,72	87,77	87,75
	75,98	92,01	94,10	94,60	95,09	95,08	74,36	89,89	91,91	92,25	91,91	91,89
	88,17	96,72	97,55	97,75	97,87	97,88	88,18	95,90	96,54	96,72	96,64	96,64
150	66,30	88,68	90,70	91,36	91,18	90,77	66,37	85,75	86,59	85,97	85,24	84,22
	74,45	92,82	94,50	94,79	94,78	94,57	74,19	90,60	91,57	90,97	90,07	89,22
	86,98	97,06	97,67	97,84	97,80	97,75	86,79	95,71	96,18	96,08	95,61	95,04
200	65,29	88,88	91,19	91,20	89,92	88,16	64,17	85,23	86,44	85,18	82,79	80,46
	73,60	93,15	94,93	94,74	93,82	92,27	72,49	90,66	91,36	90,24	87,93	86,37
	86,72	97,38	98,05	98,10	97,80	97,02	86,50	96,01	96,35	95,75	95,01	94,08

TAB. B.11 – Initialisation avec K-means, apprentissage avec early stopping. 500 neurones sur la couche cachée.

	Appt ABC						Test D					
Nhu	It0	It1	It2	It3	It4	It5	It0	It1	It2	It3	It4	It5
35	66,51	79,28	82,95	83,27	83,23	83,16	65,86	77,67	80,18	79,91	79,58	79,12
	75,59	86,44	89,20	89,35	89,38	89,49	74,79	84,93	86,67	86,59	86,38	85,97
	88,64	94,76	95,95	96,21	96,23	96,19	88,57	93,73	94,57	94,40	94,11	93,88
50	72,79	84,99	87,81	89,77	90,77	90,58	71,64	83,07	85,92	87,01	86,99	86,56
	80,31	90,48	92,37	93,91	94,53	94,12	79,36	89,22	90,73	91,64	91,85	91,24
	91,24	96,08	96,97	97,47	97,66	97,44	91,21	95,41	96,14	96,33	96,47	95,84
75	74,57	87,95	90,47	91,28	91,65	91,79	73,73	86,28	88,30	88,52	88,76	87,81
	82,08	92,47	94,34	94,75	95,04	95,11	81,53	91,52	92,86	92,95	92,72	92,07
	92,21	97,00	97,75	97,99	98,10	98,11	91,77	96,44	96,85	96,75	96,81	96,69
100	76,36	89,32	91,33	91,87	92,08	92,02	76,42	87,47	88,61	88,67	88,02	86,93
	83,49	93,69	94,89	95,16	95,22	95,19	83,95	92,65	93,36	92,65	92,00	91,39
	92,87	97,41	98,05	98,18	98,09	98,11	92,78	96,85	97,03	96,96	96,59	96,45
150	78,57	90,45	92,50	92,68	92,53	92,48	77,76	89,06	89,73	89,15	87,68	86,80
	85,11	94,21	95,54	95,61	95,38	95,30	84,23	93,04	93,39	93,04	91,98	91,21
	93,63	97,54	98,11	98,18	97,90	97,82	93,38	97,00	97,27	97,12	96,38	95,87
200	78,91	91,38	**92,92**	93,11	92,83	92,56	78,54	89,64	**89,98**	89,53	88,24	87,31
	85,32	94,65	**95,60**	95,83	95,64	95,60	85,18	93,32	**93,54**	92,86	92,03	91,34
	93,52	97,73	**98,15**	98,16	98,25	98,28	93,44	97,34	**97,45**	97,19	96,47	96,32

TAB. B.12 – Initialisation avec K-means, apprentissage avec early stopping. 650 neurones sur la couche cachée.

Annexe C

Combinaison diacritiques / corps de lettres

Ce chapitre donne les détails de l'évaluation des seuils pour la combinaison décrite dans la section 4.4.1.5.

Une combinaison trop naïve dégrade les performances. Ainsi, sur la base d'apprentissage, si l'on considère que résultat de la combinaison répondra systématiquement la classe dont la distance d'édition est la plus faible parmi les 10 candidats renvoyés par le reconnaisseur de corps de lettres, cette combinaison permet de corriger 903 erreurs, mais introduit 2409 nouvelles erreurs, soit un différentiel négatif de -1506 documents.

Nous avons donc analysé quatre critères pour fixer un seuil permettant de limiter le nombre d'apparition de nouvelles erreurs. Pour cela, nous avons posé les quatre hypothèses suivantes :

Quatre variables semblent pertinentes pour fixer un seuil permettant de limiter le nombre d'apparition de nouvelles erreurs, en posant les hypothèses suivantes :

- Hypothèse 1 : on considère le coût de la meilleure distance d'édition de la séquence de diacritiques parmi les 10 candidats. On suppose que si le coût d'appariement est trop important, le reconnaisseur de diacritique n'est pas fiable. Il ne faut alors pas modifier la liste retournée par le reconnaisseur de corps de lettres.
- Hypothèse 2 : on considère le score de confiance du premier élément de la liste rendue par le reconnaisseur de corps de lettres. On suppose que si ce score est faible, le reconnaisseur de corps de lettres est peu fiable, et on peut donc faire confiance au reconnaisseur de diacritiques pour sélectionner le meilleur candidat.
- Hypothèse 3 : on considère le rang de la classe renvoyée par le reconnaisseur de diacritiques. Cette hypothèse suppose que l'hypothèse n°1 est fausse : on suppose que la valeur de la distance d'édition n'est pas un critère absolu : dans certains cas une valeur de distance d'édition faible n'est pas un bon critère car de nombreux candidats ont une distance d'édition encore plus faible. Et a contrario, dans d'autres cas, un candidat peut être l'un des tous premiers de la liste tout en ayant une valeur de distance d'édition plus importante. Dans ce cas, le rang du candidat retenu pourrait être un critère plus pertinent que le coût.
- Hypothèse 4 : on considère le rang du candidat retenu dans la liste de reconnaissance

de corps de lettres. Cette hypothèse postule que la profondeur de liste considérée (10 candidats) est trop importante. Il faut se limiter à une profondeur plus faible car au delà d'un certain indice la combinaison devient erratique.

Ces quatre critères sont analysés dans la section suivante.

C.1 Variables pour limiter le nombre d'apparitions de nouvelles erreurs

C.1.1 Coût de la distance d'édition

Comme le montre le tableau C.1, le coût d'appariement peut être un critère qui permet de limiter l'apparition de nouvelles erreurs. La meilleure valeur est 0, et correspond donc à un appariement parfait entre la séquence de diacritiques reconnue et la séquence de diacritiques de la classe correcte. L'apport de cette stratégie est limité : l'amélioration de performances apportée par cette combinaison est faible.

C.1.2 Score de confiance corps de lettres

Le score de confiance est la vraisemblance normalisée du meilleur candidat de reconnaissance sans diacritiques.

$$score_i = \frac{P(O|M_i)}{\sum_{j \in W} P(O|M_j)}$$

Où M_i représente le i-ème mot du vocabulaire W.

Le tableau C.2 montre que ce critère est pertinent. Ce critère est discuté dans la section 4.4.1.5.

C.1.3 Profondeur de la liste de diacritiques

Le tableau C.3 montre que ce critère n'est pas pertinent : même lorsqu'on prend uniquement les réponses qui sont en tête de liste des candidats de diacritiques (profondeur de 1), ce critère n'est pas suffisant. En d'autres termes, même lorsque le meilleur candidat de mot obtenu à l'aide des diacritiques fait partie de la liste des 10 candidats retenus par la reconnaissance de corps de lettres, ce candidat est souvent erroné.

C.1.4 Profondeur de la liste de reconnaissance

Le tableau C.4 montre que ce critère est pertinent. Ici, nous examinons la combinaison entre le reconnaisseur de corps de lettres et le reconnaisseur de signes diacritiques en analysant l'indice du candidat retenu dans la liste de corps de lettres. Le tableau de résultats montre que la valeur optimale est obtenue pour une profondeur de 2. En d'autres termes, lorsque le candidat issu de la reconnaissance de signes diacritiques est l'un des deux meilleurs candidats issus de la reconnaissance des corps de lettres, la combinaison améliore le taux de reconnaissance.

Néanmoins, ce critère est moins performant que celui décrit dans la section C.1.2.

	Coût	corrections	nouvelles erreurs	différentiel	Taux reco
Train	0.0	263	199	64	90,6
	0.1	415	409	6	90,3
	0.2	437	458	-21	90,2
	0.3	512	618	-106	89,8
	0.4	571	735	-164	89,5
	0.5	606	983	-377	88,4
	0.6	640	1175	-535	87,6
	0.7	657	1210	-553	87,5
	0.8	657	1214	-557	87,5
	0.9	682	1360	-678	86,9
	1.0	686	1388	-702	86,7
	1.1	755	1655	-900	85,7
	1.2	800	1828	-1028	85,1
	1.3	821	1925	-1104	84,7
	1.4	836	1980	-1144	84,5
	1.5	850	2070	-1220	84,1
	1.6	857	2122	-1265	83,9
	1.7	861	2147	-1286	83,8
	1.8	864	2184	-1320	83,6
	1.9	869	2202	-1333	83,5
	2.0	872	2244	-1372	83,3
Test	0.0	117	80	37	87,8
	0.1	169	145	24	87,6
	0.2	184	178	6	87,3
	0.3	228	233	-5	87,1
	0.4	250	264	-14	87,0
	0.5	278	344	-66	86,2
	0.6	289	415	-126	85,3
	0.7	297	423	-126	85,3
	0.8	297	424	-127	85,3
	0.9	307	475	-168	84,7
	1.0	310	479	-169	84,7
	1.1	347	546	-199	84,2
	1.2	369	613	-244	83,6
	1.3	380	645	-265	83,3
	1.4	385	663	-278	83,1
	1.5	391	679	-288	82,9
	1.6	394	693	-299	82,8
	1.7	396	699	-303	82,7
	1.8	398	706	-308	82,6
	1.9	398	712	-314	82,5
	2.0	403	723	-320	82,4

Tab. C.1 – Analyse du coût d'appariement comme critère de seuil de la combinaison avec le reconnaisseur de diacritiques.

	Score	corrections	nouvelles erreurs	différentiel	Taux reco
Train	0.1	0	0	0	90,3
	0.15	5	0	5	90,3
	0.2	18	1	17	90,4
	0.25	43	4	39	90,5
	0.3	93	11	82	90,7
	0.35	159	15	144	91,0
	0.4	226	24	202	91,3
	0.45	306	40	266	91,6
	0.5	389	60	329	92,0
	0.55	466	81	385	92,3
	0.6	532	112	420	92,4
	0.65	579	143	**436**	92,5
	0.7	623	190	433	92,5
	0.75	669	237	432	92,5
	0.8	723	295	428	92,5
	0.85	766	360	406	92,4
	0.9	804	479	325	91,9
	0.95	848	600	248	91,6
	1.0	903	2409	-1506	82,7
Test	0.1	0	0	0	87,2
	0.15	5	0	5	87,3
	0.2	14	1	13	87,4
	0.25	27	2	25	87,6
	0.3	42	3	39	87,8
	0.35	65	7	58	88,1
	0.4	97	10	87	88,5
	0.45	130	15	115	88,9
	0.5	163	29	134	89,2
	0.55	201	36	165	89,7
	0.6	225	50	175	89,8
	0.65	251	62	189	90,0
	0.7	276	70	206	90,3
	0.75	299	84	215	90,4
	0.8	322	98	224	90,5
	0.85	344	119	**225**	90,5
	0.9	356	162	194	90,1
	0.95	377	206	171	89,7
	1.0	411	781	-370	81,7

TAB. C.2 – Analyse du score de la meilleure réponse fournie par le reconnaisseur de corps de lettres comme critère de seuil de la combinaison avec le reconnaisseur de diacritiques.

	Profondeur	corrections	nouvelles erreurs	différentiel	Taux reco
Train	1	559	635	-76	89,9
	2	606	760	-154	89,5
	3	641	866	-225	89,2
	4	657	949	-292	88,8
	5	667	1013	-346	88,5
	6	680	1064	-384	88,4
	7	700	1104	-404	88,3
	8	709	1128	-419	88,2
	9	714	1174	-460	88,0
	10	735	1245	-510	87,7
	11	749	1320	-571	87,4
	12	759	1376	-617	87,2
	13	760	1409	-649	87,0
	14	772	1443	-671	86,9
	15	778	1479	-701	86,7
Test	1	246	236	10	87,3
	2	264	272	-8	87,1
	3	286	302	-16	87,0
	4	296	328	-32	86,7
	5	304	349	-45	86,5
	6	313	359	-46	86,5
	7	321	377	-56	86,4
	8	324	384	-60	86,3
	9	329	402	-73	86,1
	10	336	423	-87	85,9
	11	340	445	-105	85,6
	12	346	468	-122	85,4
	13	348	474	-126	85,3
	14	350	478	-128	85,3
	15	356	482	-126	85,3

TAB. C.3 – Analyse de la profondeur de la liste de diacritiques comme critère de seuil de la combinaison.

	Profondeur	corrections	nouvelles erreurs	différentiel	Taux reco
Train	1	0	0	0	90,3
	2	502	420	82	90,7
	3	664	749	-85	89,9
	4	749	1019	-270	88,9
	5	782	1270	-488	87,8
	6	830	1518	-688	86,8
	7	864	1777	-913	85,7
	8	882	1991	-1109	84,7
	9	892	2197	-1305	83,7
	10	903	2409	-1506	82,7
Test	1	0	0	0	87,2
	2	218	134	84	88,4
	3	292	248	44	87,9
	4	341	347	-6	87,1
	5	364	418	-54	86,4
	6	377	506	-129	85,3
	7	386	588	-202	84,2
	8	399	663	-264	83,3
	9	403	720	-317	82,5
	10	411	781	-370	81,7

TAB. C.4 – Analyse de la profondeur de la liste de reconnaissance de corps de lettres comme critère de seuil de la combinaison.

C.1.5 Conclusion

Des quatre critères analysés, celui décrit dans la section C.1.2 est le plus performant. Le score du meilleur candidat de la liste de reconnaissance des corps de lettres permet de déduire un seuil optimal pour effectuer la combinaison. Pour les documents dont le score de confiance renvoyé par le reconnaisseur de corps de lettres est supérieur à ce seuil, la combinaison avec le reconnaisseur de signes diacritiques n'est pas effectuée.

C.2 Analyse de trois variables, la 4ème étant fixée

Dans la section précédente, nous avons vu que le critère C.1.2 permettait d'améliorer le taux de reconnaissance. Le tableau C.2 montre que la quantité de nouvelles erreurs croit de manière considérable pour les valeurs de score élevées. Comme ce critère permet d'écarter un grand nombre de nouvelles erreurs, il est intéressant de le fixer et d'analyser une nouvelle fois les trois autres critères, afin d'analyser si leur conjonction permet de réduire encore la proportion de nouvelles erreurs (mauvaises corrections). Nous fixons donc ce seuil à 0,7 (la combinaison n'est appliquée que pour les documents pour lesquels le reconnaisseur de corps de lettre renvoie un score de confiance inférieur à ce seuil), et analysons à nouveau les trois autres critères.

C.2.1 Coût de la distance d'édition

Le tableau C.5 montre que le fait de fixer un seuil sur la valeur du coût de la distance d'édition ne permet pas d'améliorer la combinaison.

C.2.2 Profondeur de la liste de diacritiques

Le tableau C.6 montre que le fait de restreindre la profondeur de recherche dans la liste de diacritiques ne permet pas d'améliorer la combinaison.

C.2.3 Profondeur de la liste de reconnaissance

Le tableau C.7 montre que le fait de restreindre la profondeur de la liste des candidats issu du reconnaisseur de corps de lettres permet d'améliorer très légèrement le taux de reconnaissance sur la base d'entraînement (pour une profondeur maximale de 7), mais pas sur la base de test. Comme les deux précédents, la combinaison de ce critère avec le seuil sur le score ne permet pas réellement de réduire la proportion de nouvelles erreurs.

C.2.4 Conclusion

Parmi les quatre critères proposés, un seul est pertinent : le score du meilleur candidat de la liste renvoyée par le reconnaisseur de corps de lettres.

Si ce score est inférieur à un certain seuil (qu'on fixera entre 0.65 et 0.85), la combinaison avec le reconnaisseur de diacritiques permet d'améliorer le taux de reconnaissance. Dans ce cas là, la combinaison sélectionnera parmi les dix candidats renvoyés par le reconnaisseur de corps de lettres, celui dont la distance d'édition retournée par le reconnaisseur de signes diacritiques est la plus faible.

Dans le cas contraire, le reconnaisseur de diacritiques est ignoré.

	Coût	corrections	nouvelles erreurs	différentiel	Taux reco
Train	0	172	23	149	91,1
	0.1	286	42	244	91,5
	0.2	298	43	255	91,6
	0.3	349	53	296	91,8
	0.4	394	70	324	91,9
	0.5	419	86	333	92,0
	0.6	444	103	341	92,0
	0.7	454	109	345	92,0
	0.8	454	109	345	92,0
	0.9	475	126	349	92,1
	1.0	478	127	351	92,1
	1.1	524	146	378	92,2
	1.2	552	158	394	92,3
	1.3	568	164	404	92,3
	1.4	578	167	411	92,4
	1.5	588	171	417	92,4
	1.6	595	174	421	92,4
	1.7	598	176	422	92,4
	1.8	600	177	423	92,4
	1.9	602	178	424	92,5
	2.0	604	181	423	92,4
	3.0	618	187	431	92,5
	5.0	621	190	431	92,5
Test	0.1	115	12	103	88,7
	0.2	119	12	107	88,8
	0.3	152	19	133	89,2
	0.4	168	22	146	89,4
	0.5	184	30	154	89,5
	0.6	194	41	153	89,5
	0.7	201	43	158	89,5
	0.8	201	43	158	89,5
	0.9	209	48	161	89,6
	1.0	211	48	163	89,6
	1.1	238	53	185	89,9
	1.2	250	59	191	90,0
	1.3	256	61	195	90,1
	1.4	259	63	196	90,1
	1.5	263	66	197	90,1
	1.6	265	66	199	90,2
	1.7	266	66	200	90,2
	1.8	268	66	202	90,2
	1.9	268	68	200	90,2
	2.0	271	68	203	90,2
	3.0	276	69	207	90,3
	5.0	276	70	206	90,3

Tab. C.5 – Analyse du coût d'appariement comme critère de seuil de la combinaison avec le reconnaisseur de diacritiques, le seuil sur le score de la meilleure réponse étant fixé à 0,7.

	Profondeur	corrections	nouvelles erreurs	différentiel	Taux reco
Train	1	383	54	329	92,0
	2	413	60	353	92,1
	3	440	69	371	92,2
	4	454	72	382	92,2
	5	458	74	384	92,2
	6	468	76	392	92,3
	7	481	79	402	92,3
	8	487	80	407	92,4
	9	489	86	403	92,3
	10	501	89	412	92,4
	11	512	96	416	92,4
	12	517	102	415	92,4
	13	517	108	409	92,4
	14	523	113	410	92,4
	15	527	117	410	92,4
	35	578	163	415	92,4
	50	600	175	425	92,5
	100	623	190	433	92,5
Test	1	156	18	138	89,3
	2	171	23	148	89,4
	3	189	25	164	89,6
	4	195	26	169	89,7
	5	201	26	175	89,8
	6	209	26	183	89,9
	7	215	30	185	89,9
	8	216	32	184	89,9
	9	220	33	187	90,0
	10	224	33	191	90,0
	11	227	34	193	90,1
	12	231	37	194	90,1
	13	232	38	194	90,1
	14	233	41	192	90,1
	15	237	41	196	90,1
	35	257	58	199	90,2
	50	268	62	206	90,3
	100	276	70	206	90,3

TAB. C.6 – Analyse de la profondeur de la liste de diacritiques comme critère de seuil de la combinaison, le seuil sur le score de la meilleure réponse étant fixé à 0,7.

	Profondeur	corrections	nouvelles erreurs	différentiel	Taux reco
Train	1	0	0	0	90,3
	2	353	32	321	91,9
	3	458	58	400	92,3
	4	515	80	435	92,5
	5	540	106	434	92,5
	6	572	123	449	92,6
	7	601	146	455	92,6
	8	613	160	453	92,6
	9	617	173	444	92,6
	10	623	190	433	92,5
Test	1	0	0	0	87,2
	2	141	13	128	89,1
	3	189	23	166	89,7
	4	227	33	194	90,1
	5	242	39	203	90,2
	6	250	50	200	90,2
	7	257	58	199	90,2
	8	267	62	205	90,2
	9	271	66	205	90,2
	10	276	70	206	90,3

TAB. C.7 – Analyse de la profondeur de la liste de reconnaissance de corps de lettres comme critère de seuil de la combinaison, le seuil sur le score de la meilleure réponse étant fixé à 0,7.

Bibliographie

[1] M. T. Laskri A. Sehad, L. Mezai and M. Cheriet. Détection de l'inclinaison des documents arabes imprimés. In *8ème colloque international francophone sur l'écrit et le document (CIFED'2004)*, June 2004.

[2] A. AdbulKader. Two-tier approach for arabic offline handwriting recognition. In *The Tenth International Workshop on Frontiers in Handwriting Recognition (IWFHR 10)*, La Baule, France, October 2006.

[3] S. Al-Emami and M. Usher. On-line recognition of handwritten arabic characters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7) :704–710, 1990.

[4] R. Al-Hajj, C. Mokbel, and L. Likforman-Sulem. Combination of hmm-based classifiers for the recognition of arabic handwritten words. *icdar*, 2 :959–963, 2007.

[5] Somaya Al-Ma'adeed, Dave Elliman, and Colin A. Higgins. A data base for arabic handwritten text recognition research. *iwfhr*, 00 :485, 2002.

[6] Yousef Al-Ohali, Mohamed Cheriet, and Ching Y. Suen. Databases for recognition of handwritten arabic cheques. *Pattern Recognition*, 36(1) :111–121, 2003.

[7] Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi. Arabic morphological analysis techniques : a comprehensive survey. *J. Am. Soc. Inf. Sci. Technol.*, 55(3) :189–213, 2004.

[8] Adel M. Alimi. An evolutionary neuro-fuzzy approach to recognize on-line arabic handwriting. In *ICDAR '97 : Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 382–386. IEEE Computer Society, 1997.

[9] N. Ben Amara and A. Belaïd. Une méthode stochastique pour la reconnaissance de l'arabe imprimée. In *Forum de la Recherche en Informatique*, Tunis, juillet 1996.

[10] A. Amin, A. Kaced, J.P. Haton, and R. Mohr. Handwritten arabic character recognition by the irac system. In *ICPR*, pages 729–731, 1980.

[11] Adnan Amin. Off line arabic character recognition - a survey. In *ICDAR '97 : Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 596–599. IEEE Computer Society, 1997.

[12] Adnan Amin. Recognition of printed and handwritten arabic characters. In *BSDIA '97 : Proceedings of the First Brazilian Symposium on Advances in Document Image Analysis*, pages 40–59, London, UK, 1997. Springer-Verlag.

[13] Adnan Amin. Recognition of printed arabic text based on global features and decision tree learning techniques. *Pattern Recognition*, 33(8) :1309–1323, 2000.

[14] Adnan Amin. Prototyping structural description using decision tree learning techniques. In *ICPR (2)*, pages 76–79, 2002.

[15] Adnan Amin. Recognition of hand-printed characters based on structural description and inductive logic programming. *Pattern Recogn. Lett.*, 24(16) :3187–3196, 2003.

[16] Adnan Amin and Sameer Singh. Optical character recognition : Neural network analysis of hand-printed characters. In *SSPR '98/SPR '98 : Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 492–499, London, UK, 1998. Springer-Verlag.

[17] Emmanuel Augustin. *Reconnaissance de mots manuscrits par systèmes hybrides Réseaux de Neurones et Modèles de Markov Cachés*. PhD thesis, Université Rene Descartes - Paris V, 2001.

[18] Gregory R. Ball, Sargur N. Srihari, and Harish Srinivasan. Segmentation-based and segmentation-free methods for spotting handwritten arabic words. In Guy Lorette and Suvisoft, editors, *Tenth International Workshop on Frontiers in Handwriting Recognition*, October 2006.

[19] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3 :1–8, 1972.

[20] Abdel Belaïd and Christophe Choisy. Human reading based strategies for off-line arabic word recognition. *Summit on Arabic and Chinese Handwriting Recognition 2006 - SACH'06*, 2006.

[21] R. Belaroussi, M. Milgram, and L. Prevost. Fusion of multiple detectors for face and eyes localization. *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 24–29, 15-17 Sept. 2005.

[22] N. Ben Amara, A. Belaïd, and N. Ellouze. Utilisation des modèles markoviens en reconnaissance de l'écriture arabe état de l'art. In *Colloque International Francophone sur l'Ecrit et le Document (CIFED'00), Lyon*, Lyon, France, juillet 2000.

[23] A. Benouareth, A. Ennaji, and M. Sellami. Hmms with explicit state duration applied to handwritten arabic word recognition. In *ICPR06*, pages II : 897–900, 2006.

[24] A. Benouareth, A. Ennaji, and M. Sellami. Semi-continuous hmms with explicit state duration applied to arabic handwritten word recognition. In *The Tenth International Workshop on Frontiers in Handwriting Recognition (IWFHR 10)*, La Baule, France, 2006.

[25] M. Bernard, L. Boyer, A. Habrard, and M. Sebban. Learning probabilistic models of tree edit distance. *Pattern Recognition (accepted paper)*, 2008.

[26] M. Bernard, A. Habrard, and M. Sebban. Learning stochastic tree edit distance. In LNCS, editor, *17th European Conference on Machine Learning (ECML06)*, pages 42–53, 2006.

[27] M. Bernard, C. Janodet, and M. Sebban. A discriminative model of stochastic edit distance in the form of conditional transducer. In *International Colloquium on Grammatical Inference (ICGI06)*, 2006.

[28] J. Bernsen. Dynamic thresholding of gray-level images. *Proc. 8th Int. Conf. Pattern Recognition Paris*, pages 1251–1255, 1986.

[29] R. Bertolami, S. Uchida, M. Zimmermann, and H. Bunke. Non-uniform slant correction for handwritten text line recognition. *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 1 :18–22, 23-26 Sept. 2007.

[30] Jeff A. Bilmes. What hmms can do. *IEICE - Trans. Inf. Syst.*, E89-D(3) :869–891, 2006.

[31] Rolf-Dieter Bippus. 1-dimensional and pseudo 2-dimensional hmms for the recognition of german literal amounts. In *ICDAR '97 : Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 487–490. IEEE Computer Society, 1997.

[32] G. Boulianne, J. Brousseau, P. Ouellet, and P. Dumouchel. Le système de rapt du crim. In *12e Congrès francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle (RFIA 2000)*, volume 2, pages 149–156, Paris, France, 2000.

[33] R. M. Bozinovic and S. N. Srihari. Off-line cursive script word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(1) :68–83, 1989.

[34] Christian Brechbuhler and Sean Ho. Reconstruction of a closed curve from its elliptic descriptor.

[35] Harald Breit and Gerhard Rigoll. Improved person tracking using a combined pseudo-2d-hmm and kalman filter approach with automatic background state adaptation. In *ICIP (2)*, pages 53–56, 2001.

[36] A. Britto-Jr, R. Sabourin, E. Lethelier, F. Bortolozzi, and C. Suen. Improvement handwritten numeral string recognition by slant normalization and contextual information, 2000.

[37] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

[38] Peter Burrow. *Arabic Handwriting Recognition*. PhD thesis, Master of Science, School of Informatics University of Edinburgh, 2004.

[39] A. Harvey C. Weliwitage and A. Jennings. Whole of word recognition methods for cursive script. *APRS Workshop on Digital Image Computing (WDIC)*, febuary 2003.

[40] T. Caesar, J.M. Gloger, and E. Mandler. Preprocessing and feature extraction for a handwriting recognition system. *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 408–411, 20-22 Oct 1993.

[41] F. Carmagnac. *Classification supervisée et semi-supervisée : contributions à la classification d'images de documents*. PhD thesis, Université de Rouen, 2003.

[42] R.G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7) :690–706, Jul 1996.

[43] Chih C. Chang and Chih J. Lin. *LIBSVM : a library for support vector machines*, 2001.

[44] Yi-Kai Chen and Jhing-Fa Wang. Segmentation of handwritten connected numeral string using background and foreground analysis. *icpr*, 02 :2598, 2000.

[45] Yi-Kai Chen and Jhing-Fa Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11) :1304–1317, 2000.

[46] Ying-Nong Chen, Chin-Chuan Han, Cheng-Tzu Wang, Bor-Shenn Jeng, and Kuo-Chin Fan. The application of a convolution neural network on face and license plate detection. In *ICPR '06 : Proceedings of the 18th International Conference on Pattern Recognition*, pages 552–555. IEEE Computer Society, 2006.

[47] M. Cheriet. Strategies for visual arabic handwriting recognition : issues and case study. In *ISSPA 2007, International Symposium on Signal Processing and its Applications, 12 - 15 February 2007, Sharjah, United Arab Emirates*, Feb 2007.

[48] M. Cheriet, Y.S. Huang, and C.Y. Suen. Background region-based algorithm for the segmentation of connected digits. *Pattern Recognition, 1992. Vol.II. Conference B : Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 619–622, 30 Aug-3 Sep 1992.

[49] Mohammed Cheriet, Nawwaf Kharma, Cheng lin Liu, and Ching Suen. *Character Recognition Systems : A Guide for Students and Practitioners*. Wiley-Interscience, 2007.

[50] Yann Ricquebourg Christophe Renaudin and Jean Camillerapp. A general method of segmentation-recognition collaboration applied to pairs of touching and overlapping symbols. *Document Analysis and Recognition, 2007. ICDAR 2007 Vol. 2. Ninth International Conference on*, 2 :659–663, 23-26 Sept. 2007.

[51] R. Collobert, S. Bengio, and J. Mariéthoz. Torch : a modular machine learning software library, 2002. IDIAP Research Report 02-46, Martigny, Switzerland. (see also www.torch.ch).

[52] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[53] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) :1–38, 1977.

[54] Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange : from experimental machine learning to interactive data mining. In *PKDD '04 : Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 537–539, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[55] X. Dupre. *Contributions à la reconnaissance de l'écriture cursive à l'aide de modèles de Markov cachés*. PhD thesis, Univ Rene Descartes - Paris V, 2003.

[56] Stefan Eickeler. Face database retrieval using pseudo 2d hidden markov models. In *FGR '02 : Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, page 65. IEEE Computer Society, 2002.

[57] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel. Arabic handwriting recognition using baseline dependant features and hidden markov modeling. In *ICDAR*, pages 893–897, 2005.

[58] R. El-Hajj, C. Mokbel, and L. Likforman-Sulem. Reconnaissance de l'ecriture arabe cursive : combinaison de classifieurs mmcs à fenêtres orientées. In *CIFED*, 2006.

[59] Saleh Al-Osaimi Eric Atwell, Latifa Al-Sulaiti and Bayan Abu Shawar. A review of arabic corpus analysis tools. In *Actes de TALN'04*, Fès, Maroc, 2004.

[60] M. Shridhar F. Kimura, S. Tsuruoka and Z. Chen. Context directed handwritten word recognition for postal service applications. *Proc. Fifth US Postal Service Technology Conf.*, 1992.

[61] P. Fernando, C. Michael, and D. Riley. Speech recognition by composition of weighted finite automata, 1996.

[62] B. Gatos, I. Pratikakis, and S. J. Perantonis. Adaptive degraded document image binarization. *Pattern Recogn.*, 39(3) :317–327, 2006.

[63] M. Gilloux. Reconnaissance de chiffres manuscrits par modèle de markov pseudo-2d. In *Traitement du Signal*, 1996.

[64] J. Golenzer, C. Viard-Gaudin, and PM. Lallican. Finding regions of interest in document images by planar hmm. In *ICPR '02 : Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3*, page 30415. IEEE Computer Society, 2002.

[65] A. M. Gouda and M. A. Rashwan. Segmentation of connected arabic characters using hidden markov models. In *Computational Intelligence for Measurement Systems and Applications, 2004. CIMSA. 2004 IEEE International Conference on*, pages 115–119, 2004.

[66] Abduelbaset Goweder and Anne De Roeck. Assessment of a significant arabic corpus.

[67] Latifa Hamami and Daoud Berkani. Recognition system for printed multi-font and multi-size arabic characters. In *The Arabian Journal for Science and Engineering*, volume 27 1B. IEEE Computer Society, avril 2002.

[68] Nabil Hassan. Recognition of arabic cursive handwriting. In *GMAI*, pages 135–140, 2006.

[69] A. Hennig, N. Sherkat, and R. J. Whitrow. Zone-estimation for multiple lines of handwriting using approximating spline functions.

[70] Andreas Hennig and Nasser Sherkat. Exploiting zoning based on approximating splines in cursive script recognition. *Pattern Recognition*, 35(2) :445–454, 2002.

[71] J. Hilditch. Linear skeletons from square cupboards. *Machine Intelligence 4 (B. Meltzer and D. Michie, Eds.)*, pages 404–420, 1969.

[72] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7) :1527–1554, July 2006.

[73] M.K. Hu. Pattern recognition by moment invariants. In *Proc. IRE*, page 1428, Sept. 1961.

[74] M.K. Hu. Visual pattern recognition by moment invariants. In *IRE Trans Inform*, pages 179–187, 1962.

[75] Lei Huang, Genxun Wan, and Changping Liu. An improved parallel thinning algorithm. *icdar*, 02 :780, 2003.

[76] B.K. Jang and R.T. Chin. One-pass parallel thinning : analysis, properties, and quantitative evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(11) :1129–1140, Nov 1992.

[77] A. Kaltenmeier, T. Caesar, J.M. Gloger, and E. Mandler. Sophisticated topology of hidden markov models for cursive script recognition. *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 139–142, 20-22 Oct 1993.

[78] Wady Kammoun and Abdel Ennaji. Reconnaissance de textes arabes à vocabulaire ouvert. In *8ème colloque international francophone sur l'écrit et le document (CIFED'2004)*, June 2004.

[79] S. Kanoun, A. M. Alimi, and Y. Lecourtier. Affixal approach for arabic decomposable vocabulary recognition : A validation on printed word in only one font. In *ICDAR*, pages 1025–1029. IEEE Computer Society, 2005.

[80] Slim Kanoun, Adellatif Ennaji, Yves LeCourtier, and Adel M. Alimi. Linguistic integration information in the aabatas arabic text analysis system. In *IWFHR '02 : Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, page 389. IEEE Computer Society, 2002.

[81] T. Kanungo, G. Marton, and O. Bulbul. Omnipage vs. sakhr : Paired model evaluation of two arabic ocr products, 1999.

[82] Ergina Kavallieratou, Nikos Fakotakis, and George K. Kokkinakis. Slant estimation algorithm for ocr systems. *Pattern Recognition*, 34(12) :2515–2522, 2001.

[83] N. Kharma, M. Ahmed, and R. Ward. A new comprehensive database of handwritten arabic words, numbers, and signatures used for ocr testing. *Electrical and Computer Engineering, 1999 IEEE Canadian Conference on*, 2 :766–768 vol.2, 1999.

[84] M. S. Khorsheed. Recognising handwritten arabic manuscripts using a single hidden markov model. *Pattern Recogn. Lett.*, 24(14) :2235–2242, 2003.

[85] M. S. Khorsheed. Offline recognition of omnifont arabic text using the hmm toolkit (htk). *Pattern Recogn. Lett.*, 28(12) :1563–1571, 2007.

[86] M. S. Khorsheed and W. F. Clocksin. Spectral features for arabic word recognition. In *ICASSP '00 : Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, pages 3574–3577. IEEE Computer Society, 2000.

[87] Mohammad S. Khorsheed and William F. Clocksin. Multi-font arabic word recognition using spectral features. In *ICPR '00 : Proceedings of the 15th International Conference on Pattern Recognition*, volume 4, pages 543–546. IEEE Computer Society, 2000.

[88] Hossein Khosravi and Ehsanollah Kabir. Introducing a very large dataset of handwritten farsi digits and a study on their varieties. *Pattern Recognition Letters*, 28(10) :1133–1141, 2007.

[89] F. Kimura, M. Shridhar, and Z. Chen. Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 18–22, 20-22 Oct 1993.

[90] Alessandro L. Koerich, Robert Sabourin, and Ching Y. Suen. Large vocabulary off-line handwriting recognition : A survey. *Pattern Anal. Appl.*, 6(2) :97–121, 2003.

[91] F. P. Kuhl and C. R. Giardina. Elliptic fourier feature of a closed contour. In *Computer Vision, Graphics and Image Processing*, volume 18, pages 236–258, 1982.

[92] E. Lecolinet. Planar markov modeling for arabic writing recognition : Advancement state. In *Proceedings of the first IEEE Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 740–748, Saint Malo, France, Sept. 1991.

[93] Y. Lecun. Lenet-5, convolutional neural networks.

[94] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[95] Y. LeCun, S. Chopra, M. Ranzato, and F.-J. Huang. Energy-based models in document recognition and computer vision. In *ICDAR '07 : Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 1*, pages 337–341. IEEE Computer Society, 2007.

[96] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *cvpr*, 02 :97–104, 2004.

[97] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. In *NIPS*, 2005.

[98] Yann LeCun, Marc'Aurelio Ranzato, YLan Boureau, FuJie Huang, and Sumit Chopra. Learning a deep hierarchy of sparse and invariant features, 2006.

[99] K. F. Lee, H. W. Hon, and R. Reddy. An overview of the sphinx speech recognition system. *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, 38(1) :35–45, 1990.

[100] Seong-Whan Lee and Sang-Yup Kim. Integrated segmentation and recognition of handwritten numerals with cascade neural network. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 29(2) :285–290, 1999.

[101] E. Levin and H. Pieraccini. Dynamic planar warping for optical character recognition. In *Int. Conf. Acoustics, Speech, and Signal Processing*, pages 23–26, San Francisco, California, march 1992.

[102] Y. Li. Reforming the theory of invariant moments for pattern recognition. *PR*, 25(7) :723–730, July 1992.

[103] C.W. Liao and J.S. Huang. Stroke segmentation by bernstein-bezier curve fitting. *PR*, 23(5) :475–484, 1990.

[104] Liana M. Lorigo and Venu Govindaraju. Offline arabic handwriting recognition : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5) :712–724, 2006.

[105] Y. Lu and M. Shridhar. Character segmentation in handwritten words : An overview. *PR*, 29(1) :77–96, January 1996.

[106] Z. Lu, Z. Chi, W. Siu, and P. Shi. A backgroundthinning based approach for separating and recognizing connected handwritten digit strings, 1999.

[107] E. Lecolinet M. Cote, M. Cheriet and C. Y. Suen. Détection des lignes de référence de mots cursifs à l'aide de l'entropie, les techniques de l'i.a. appliquées aux technologies de l'information. *Cahiers Scientifiques de l'ACFAS*, pages 184–193, 1997.

[108] Samia Snoussi Maddouri, Hamid Amiri, and Abdel Belaïd. Local normalization towards global recognition of arabic handwritten script, December 2000.

[109] S. Madhvanath and V. Govindaraju. Local reference lines for handwritten phrase recognition. *PR*, 32(12) :2021–2028, December 1999.

[110] Sriganesh Madhvanath and Venu Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(2) :149–164, 2001.

[111] Sriganesh Madhvanath, Venu Krpasundar, and Venu Govindaraju. Syntactic methodology of pruning large lexicons in cursive script recognition. *Pattern Recognition*, 34(1) :37–46, 2001.

[112] Volker Märgner and Haikal El Abed. Arabic handwriting recognition competition. In *ICDAR*, pages 1274–1278. IEEE Computer Society, 2007.

[113] Volker Märgner, Mario Pechwitz, and H. El Abed. Arabic handwriting recognition competition. In *ICDAR*, pages 70–74, 2005.

[114] Sameh Masmoudi and Hamid Amiri. Reconnaissance de mots arabes manuscrits par modélisation markovienne. In *Proceedings of 2ème Colloque International Francophone sur l'Ecrit et le Document (CIFED'2000)*, Lyon (France), July 3 - 5 2000.

[115] James L. Mcclelland and David E. Rumelhart. An interactive activation model of context effects in letter perception : Part 1. An account of basic findings. *Psychological Review*, 88(5) :375–405, 1981.

[116] F. Menasri, N. Vincent, M. Cheriet, and E. Augustin. Shape-based alphabet for off-line arabic handwriting recognition. In *ICDAR '07 : Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, pages 969–973. IEEE Computer Society, 2007.

[117] David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2) :169–186, 2003.

[118] Neila Mezghani, Mohamed Cheriet, and Amar Mitiche. Combination of pruned kohonen maps for on-line arabic characters recognition. In *ICDAR '03 : Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 900. IEEE Computer Society, 2003.

[119] Neila Mezghani, Amar Mitiche, and Mohamed Cheriet. On-line recognition of handwritten arabic characters using a kohonen neural network. In *IWFHR '02 : Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, page 490. IEEE Computer Society, 2002.

[120] H. Miled. *Reconnaissance de l'ecriture semi-cursive : Application aux mots manuscrits arabes*. PhD thesis, PSI-La3i Univ Rouen, LIVIA ETS Montreal, 1998.

[121] H. Miled and N. E. B. Amara. Planar markov modeling for arabic writing recognition : Advancement state. In *ICDAR '01 : Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 69–73, 2001.

[122] H. Miled, Mohamed Cheriet, and C. Olivier. Multi-level arabic handwritten words recognition. In *SSPR '98/SPR '98 : Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 944–951, London, UK, 1998. Springer-Verlag.

[123] H. Miled, C. Olivier, and M. Cheriet. Modélisation de la notion de pseudo-mots en reconnaissance de mots manuscrits arabes. In *CIFED*, 2000.

[124] H. Miled, C. Olivier, Mohamed Cheriet, and Yves Lecoutie. Coupling observation/letter for a markovian modelisation applied to the recognition of arabic handwriting. In *ICDAR '97 : Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 580–583. IEEE Computer Society, 1997.

[125] J. Milgram. *Contribution à l'intégration des machines à vecteurs de support au sein des systèmes de reconnaissance de formes : application à la lecture automatique de l'écriture manuscrite*. PhD thesis, Ecole de Technologie Supérieure - Université du Quebec, 2007.

[126] Mehryar Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2) :269–311, 1997.

[127] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2) :269–311, 1997.

[128] Deya Motawa, Adnan Amin, and Robert Sabourin. Segmentation of arabic cursive script. In *ICDAR '97 : Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 625–628. IEEE Computer Society, 1997.

[129] Harold Mouchère and Éric Anquetil. A unified strategy to deal with different natures of reject. In *ICPR (2)*, pages 792–795. IEEE Computer Society, 2006.

[130] S. Mozaffari, K. Faez, V. Margner, and H. El Abed. Strategies for large handwritten farsi/arabic lexicon reduction. In *ICDAR07*, pages 98–102, 2007.

[131] V. Märgner, H. El Abed, and M. Pechwitz. Offline handwritten arabic word recognition using hmm - a character based approach without explicit segmentation. In *CIFED*, 2006.

[132] M. Cheriet N. E. Ayat and C.Y. Suen. Un système neuro-flou pour la reconnaissance de montants numériques des chèques arabes. In *Colloque International Francophone sur l'Ecrit et le Document (CIFED'00), Lyon*, Lyon, France, juillet 2000.

[133] A. Mitiche N. Mezghani and M. Cheriet. Reconnaissance en-ligne de caractères arabes manuscrits par un réseau de kohonen. In *Vision Interface*, pages 186–191, Calgary, Canada, 2002. IEEE Computer Society.

[134] Labiba Souici Nadir Farah and Mokhtar Sellami. Classifiers combination and syntax analysis for arabic literal amount recognition. *Engineering Applications of Artificial Intelligence*, 19(1) :29–39, 2006.

[135] P. Nagabhushan, S.A. Angadi, and B.S. Anami. Geometric model and projection based algorithms for tilt correction and extraction of acsenders / descenders for cursive word recognition. *Signal Processing, Communications and Networking, 2007. ICSCN '07. International Conference on*, pages 488–491, 22-24 Feb. 2007.

[136] Abdel Belaïd Najoua Ben Amara and Noureddine Ellouze. Modélisation pseudo bidimensionnelle pour la reconnaissance de chaînes de caractères arabes imprimés. In *CIFED*, Laval, QC, CA, mai 1998.

[137] Wayne Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, Birkeroed, Denmark, Denmark, 1985.

[138] E. Levin O. E. Agazzi, S. Kuo and R. Pieraccini. Connected and degraded text recognition using planar hidden markov models. In *ICASSP'93*, pages V–113–V–116, 1993.

[139] J. J. Oliveira, J. de Carvalho, C. Freitas, and R. Sabourin. Feature sets evaluation for handwritten word recognition, 2002.

[140] G. Olivier, H. Miled, K. Romeo, and Y. Lecourtier. Segmentation and coding of arabic handwritten words. In *ICPR96*, pages III : 264–268, 1996.

[141] J. Oncina and M. Sebban. Learning stochastic edit distance : application in handwritten character recognition. *Pattern Recognition*, 39(9) :1555–1812, 2006.

[142] Margarita Osadchy, Yann Le Cun, and Matthew L. Miller. Synergistic face detection and pose estimation with energy-based models. *J. Mach. Learn. Res.*, 8 :1197–1215, 2007.

[143] Margarita Osadchy, Matthew L. Miller, and Yann LeCun. Synergistic face detection and pose estimation with energy-based models. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1017–1024. MIT Press, Cambridge, MA, 2005.

[144] N. Otsu. A threshold selection method from gray-level histograms. In *IEEE Trans. System, Man Cybernetics9*, pages 62–66, 1979.

[145] P. Swaminathan P. W. Palumbo and S. N. Srihari. Document image binarization : Evaluation of algorithms. *Proc. SPIE*, 697 :278–285, 1986.

[146] U. Pal, A. Belaïd, and Ch. Choisy. Touching numeral segmentation using water reservoir concept. *Pattern Recogn. Lett.*, 24(1-3) :261–272, 2003.

[147] M. Pechwitz, S. Snoussi Maddouri, V. Maergner, N. Ellouze, and H. Amiri. Ifn/enit-database of handwritten arabic words. In *CIFED*, 2002.

[148] M. Pechwitz, W. Maergner, and H. ElAbed. Comparison of two different feature sets for offline recognition of handwritten arabic words. In *IWFHR*, 2006.

[149] M. Pechwitz and V. Märgner. Baseline estimation for arabic handwritten words. In *IWFHR '02 : Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, page 479. IEEE Computer Society, 2002.

[150] Mario Pechwitz and Volker Maergner. Hmm based approach for handwritten arabic word recognition using the ifn/enit- database. In *ICDAR '03 : Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 890. IEEE Computer Society, 2003.

[151] Freddy Perraud, Christian Viard-Gaudin, Emmanuel Morin, and Pierre-Michel Lallican. N-gram and n-class models for on line handwriting recognition. In *ICDAR '03 : Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 1053. IEEE Computer Society, 2003.

[152] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition, 1990.

[153] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, 1986.

[154] Marc'Aurelio Ranzato and Yann LeCun. A sparse and locally shift invariant feature extractor applied to document images. In *Proc. International Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[155] Ch. H. Reinsch. Smoothing by spline functions ii, March 1971.

[156] J. Sadri, C. Y. Suen, and T. D. Bui. Automatic segmentation of unconstrained handwritten numeral strings. In *IWFHR '04 : Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 317–322, Washington, DC, USA, 2004. IEEE Computer Society.

[157] Najoua Essoukri Ben Amara Sameh Masmoudi Touj and Hamid Amiri. Modelisation markovienne planaire pour la reconnaissance de l'écriture arabe. In *8ème colloque international francophone sur l'écrit et le document (CIFED'2004)*, June 2004.

[158] Gregory R. Ball Sargur N. Srihari and Harish Srinivasan. Versatile search of scanned arabic handwriting. *Summit on Arabic and Chinese Handwriting Recognition 2006 - SACH'06*, 2006.

[159] Pavithra Babu Sargur Srihari, Harish Srinivasan and Chetan Bhole. Spotting words in handwritten arabic documents. In *In Procs. of SPIE*, San Jose, CA, USA, Jan 2006.

[160] Pavithra Babu Sargur Srihari, Harish Srinivasan and Chetan Bhole. Handwritten arabic word spotting using the cedarabic document analysis system. In *2005 Symposium on Document Image Understanding Technology*, The Marriott Inn and Conference Center, University Maryland University College, Adelphi, Maryland, November 2-4, 2005.

[161] Toufik Sari, Labiba Souici, and Mokhtar Sellami. Off-line handwritten arabic character segmentation algorithm : Acsa. In *IWFHR '02 : Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, page 452. IEEE Computer Society, 2002.

[162] K. Sayre. Machine recognition of handwritten words : A project report. *Pattern Recognition*, 5 :213–228, 1973.

[163] Marc-Peter Schambach. Model length adaptation of an hmm based cursive word recognition system. In *ICDAR '03 : Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 109. IEEE Computer Society, 2003.

[164] Bernhard Scholkopf and Alexander J. Smola. Learning with kernels : Support vector machines, regularization, optimization and beyond. MIT Press, 2002.

[165] Andrew Senior and Tony Robinson. Forward-backward retraining of recurrent neural networks. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 743–749. The MIT Press, 1996.

[166] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation, 2004.

[167] N. Sherkat. Influence of zoning on whole word recognition. *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 1085–1089 Vol. 2, 29 Aug.-1 Sept. 2005.

[168] Patrice Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, pages 958–962. IEEE Computer Society, 2003.

[169] J. C. Simon. Off-line cursive word recognition. In *Proceedings of IEEE*, pages 1150–1161. vol 80, no. 7, 1992.

[170] J. C. Simon and O. Baret. Regularities and singularities in line pictures. In *International Journal Pattern Recognition and Artificial Intelligence*, pages 55–77. vol. 5, no. 1-2, 1991.

[171] Petr Slavik and Venu Govindaraju. An overview of run-length encoding of handwritten word images. Technical Report 2000-09, CEDAR - Buffalo, 25 2000.

[172] L. Souici-Meslati and M. Sellami. Reconnaissance de montants littéraux arabes par une approche hybride neuro-symbolique. In *RFIA'2002, 11ème Congrès francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle*, Angers, France, Janvier 2002.

[173] N.W. Strathy, C.Y. Suen, and A. Krzyzak. Segmentation of handwritten digits using contour features. *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pages 577–580, 20-22 Oct 1993.

[174] Ching Y. Suen. N-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2) :164–172, apr 1979.

[175] M. Suwa. Segmentation of connected handwritten numerals by graph representation. *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 750–754 Vol. 2, 29 Aug.-1 Sept. 2005.

[176] Eiji Taira, Seiichi Uchida, and Hiroaki Sakoe. Nonuniform slant correction for handwritten word recognition.

[177] C.C. Tappert, C.Y. Suen, and T. Wakahara. Online handwriting recognition-a survey. *Pattern Recognition, 1988., 9th International Conference on*, pages 1123–1132 vol.2, 14-17 Nov 1988.

[178] Y. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, and S. Knerr. An offline cursive handwritten word recognition system, 2001.

[179] E. Thiel. *Les distances de chanfrein en analyse d'images : fondements et applications*. Thèse de doctorat, Université Joseph Fourier, Grenoble 1, Sept 1994.

[180] Sameh Masmoudi Touj, Najoua Essoukri Ben Amara, and Hamid Amiri. A hybrid approach for off-line arabic handwriting recognition based on a planar hidden markov modeling. In *ICDAR '07 : Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, pages 964–968. IEEE Computer Society, 2007.

[181] Sofien Touj, Najoua Essoukri Ben Amara, and Hamid Amiri. Reconnaissance hors ligne de caractères arabes isolés manuscrits. In *CIFED*, 2002.

[182] Sofien Touj, Najoua Essoukri Ben Amara, and Hamid Amiri. Generalized hough transform for arabic optical character recognition. In *ICDAR '03 : Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 1242. IEEE Computer Society, 2003.

[183] Oivind Due Trier and Anil K. Jain. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12) :1191–1201, 1995.

[184] Oivind Due Trier and Torfinn Taxt. Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3) :312–315, 1995.

[185] A. Belaid U. Pal and C. Choisy. Water reservoir based approach for touching numeral segmentation. In *ICDAR '01 : Proceedings of the Sixth International Conference on Document Analysis and Recognition*, page 892. IEEE Computer Society, 2001.

[186] V. N. Vapnik. *Estimation Dependences Based on Empirical Data*. Springer-Verlag, New York, USA, 1982.

[187] C. Viard-Gaudin, P.M. Lallican, S. Knerr, and P. Binter. The ireste on/off (ironoff) dual handwriting database. *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 455–458, 20-22 Sep 1999.

[188] A. Vinciarelli and J. Luettin. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9) :1043–1050, 2001.

[189] Jiren Wang, Maylor K. H. Leung, and Siu Cheung Hui. Cursive word reference line detection. *Pattern Recognition*, 30(3) :503–511, 1997.

[190] E. Van Der Werf and R. Duin. Improved image features by training non-linear diabolo networks. *Proc. ASCI'99 5th Conf. on Advanced School for Computing and Imaging (ASCI'99)*, pages 229–234, June 1999.

[191] Wai-Hong Wong, Wan-Chi Siu, and Kin-Man Lam. Generation of moment invariants and their uses for character recognition. *Pattern Recogn. Lett.*, 16(2) :115–123, 1995.

[192] Y. Xi, Y. Chen, Q. Liao, L. Winghong, F. Shunming, and D. Jiangwen. A novel binarization system for degraded document images. *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, 1 :287–291, 23-26 Sept. 2007.

[193] C. Aloulou Y. Bahou, L. Hadrich Belguith and A. Ben Hamadou. Adaptation et implémentation des grammaires hpsg pour l'analyse de textes arabes non voyellés. In *Actes du 15e Congrès de Reconnaissance des Formes et Intelligence Artificielle (RFIA'2006), Tours*, January 2006.

[194] Y. Miyake Y. Ding, F. Kimura and M. Shridhar. Accuracy improvement of slant estimation for handwritten words. In *ICPR '00 : Proceedings of the International Conference on Pattern Recognition*, page 4527. IEEE Computer Society, 2000.

[195] Hirofumi Yamamoto, Shuntaro Isogai, and Yoshinori Sagisaka. Multi-class composite n-gram language model for spoken language processing using multiple word clusters. In *ACL '01 : Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 531–538, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[196] Y. Yan, M. Fanty, and R. Cole. Speech recognition using neural networks with forward-backward probability generated targets. In *ICASSP '97 : Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 4*, page 3241. IEEE Computer Society, 1997.

[197] B. Yanikoglu and P. Sandon. Segmentation of off-line cursive handwriting using linear programming, 1998.

[198] Ruslan Salakhutdinov Yoshua Bengio, Yann LeCun and Hugo Larochelle. Deep learning workshop : Foundations and future directions, 2007.

[199] S. Young. The htk hidden markov model toolkit : Design and philosophy, 1993.

[200] A. Zahour, L. Likforman-Sulem, W. Boussellaa, and B. Taconet. Text line segmentation of historical arabic documents. In *ICDAR '07 : Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 1*, pages 138–142. IEEE Computer Society, 2007.

[201] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3) :236–239, 1984.

[202] David X. Zhong and Hong Yan. Pattern skeletonization using run-length-wise processing for intersection distortion problem. *Pattern Recogn. Lett.*, 20(8) :833–846, 1999.

[203] M. Ziaratban, K. Faez, and M. Ezoji. Use of legal amount to confirm or correct the courtesy amount on farsi bank checks. In *ICDAR '07 : Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, pages 1123–1127. IEEE Computer Society, 2007.