Université Ibn Zohr
Faculté des Sciences d'Agadir

N° d'ordre : **101/2012**

THÈSE DE DOCTORAT EN SCIENCES

Présentée par

YOUSSEF ES-SAADY

UFR : Mathématiques et Informatique Appliquées
Spécialité : Informatique

Contribution au développement d'approches de reconnaissance automatique de caractères imprimés et manuscrits, de textes et de documents Amazighes

Soutenue le 28 janvier 2012, devant la commission d'examen composée de :

M. Driss Mammass,	Directeur de l'EST d'Agadir ;	Président
M. Ali Rachidi,	Professeur à l'ENCG d'Agadir ;	Directeur de thèse
M. Mostafa El Yassa,	Professeur à la FS d'Agadir ;	Directeur de thèse
M. El Houssine Bouyakhf,	Professeur à la FS de Rabat ;	Rapporteur
M. Mohamed Fakir,	Professeur à la FST de Beni Mellal ;	Rapporteur
M. Hassan Douzi,	Professeur à la FS d'Agadir ;	Rapporteur
M. Youssef Ait Ouguengay,	Directeur de CEISIC, IRCAM- Rabat ;	Invité

Résumé

Cette thèse a pour sujet principalement la reconnaissance automatique hors ligne de l'écriture amazighe. Dans ce cadre, nous avons d'abord construit une base de données de caractères amazighes manuscrits composée de plus de 25.000 caractères isolés écrits par 60 scripteurs différents. Cette base a été utilisée pour évaluer et tester les résultats de nos travaux.

Ce travail de thèse propose deux approches de reconnaissance automatique de l'écriture amazighe qui' ont contribué à améliorer les performances. La première approche est syntaxique, elle utilise des automates à états finis pour reconnaitre les caractères amazighes imprimés. Cette approche a de bonnes performances sur notre base de données de caractères amazighes imprimés isolés. Cependant, elle présente des limites sur le traitement des caractères circulaires. Afin de remédier ces limites, nous avons développé un système de reconnaissance de l'écriture amazighe basé sur la ligne centrale horizontale du caractère. Ce second système utilise un réseau de neurones multicouches comme classifieur. Il a montré de bonnes performances à la fois sur la base de données de patterns de la graphie amazighe et sur notre base de caractères manuscrits construite localement. Une amélioration de ce système a été proposée en intégrant d'autres caractéristiques basées sur la ligne centrale verticale du caractère. Cette amélioration a donnée de bons résultats.

Mots-clés

Reconnaissance automatique de l'écriture, Ecriture Amazighe, Tifinagh, Approches syntaxiques, Automates à états finis, Réseaux de neurones.

Abstract

This thesis is mainly on the off-line Amazigh writing recognition. In this respect, we first built a database of Amazigh handwritten characters composed of more than 25,000 individual characters written by 60 different writers. This database was used to evaluate and test the results of our work.

This work proposes two approaches of the Amazigh writing recognition which have helped improve performances. The first approach is syntactic; it uses the finite state automata to recognize the printed Amazigh characters. This approach has good results on our database of Amazigh printed characters. However, it has limitations on the treatment of non-segmented characters. To address these limitations, we have developed a system of Amazigh handwriting recognition based on the horizontal center line of the character. This second system uses a multilayer neural network as a classifier. It has shown good results on both the database spelling patterns of Amazigh and our database locally built of Amazigh handwritten characters. An improvement of this system was proposed by incorporating other features based on the vertical center line of the character. This improvement has given good results.

Keywords

Automatic handwriting recognition, Amazigh Writing, Tifinagh, Syntactic approaches, Finite state automata, Neural networks.

ملخص

ترتكز هذه الأطروحة أساسا حول التعرف التلقائي على الكتابة الأمازيغية. في هذا الإطار، أنشئنا أولا قاعدة بيانات للأحرف الأمازيغية المكتوبة بخط اليد، المحتوية على أكثر من 25،000 حرف مكتوب بواسطة 60 شخص مختلف. تم استخدام قاعدة البيانات هذه لتقييم واختبار نتائج عملنا.

هذه الأطروحة تقترح نهجين للتعرف التلقائي على الكتابة الأمازيغية اللذان ساعدا على تحسين الأداء في هذا المجال. النهج الأول يعتمد على النحوية، يستخدم نموذج الأتومات المنتهي للتعرف على الأحرف الأمازيغية المطبوعة المعزولة. هذا النهج أظهر نتائج جيدة على قاعدة بيانات مكونة من الأحرف الأمازيغية المطبوعة. ومع ذلك، فإن لهذا النهج قيود في معالجة الحروف المكونة من الدوائر. للتغلب على هذه القيود، قمنا بتطوير نظام للتعرف التلقائي على الكتابة الأمازيغية الذي يعتمد على خط الوسط الأفقي للحرف. هذا النظام الثاني يستخدم شبكة متعددة الطبقات العصبية كمصنف. و قد بين أداء جيدا على كل من قاعدة بيانات الأحرف الأمازيغية المطبوعة وقاعدة بيانات الأحرف الأمازيغية المكتوبة بخط اليد و المنشئة محليا. و لتحسين هذا النظام، اقترحنا دمج ميزات أخرى تعتمد على خط الوسط العمودي للحرف. وقد أظهر هذا التحسين نتائج جيدة.

الكلمات الرئيسية

التعرف التلقائي على الكتابة، الكتابة الأمازيغية، التيفيناغ، نهج النحوية، نموذج الأتومات المنتهي، الشبكات العصبية.

ⵜⴰⴳⴹⵡⵉⵜ

ⴰⵔ ⴱⴰⵀⵔⴰ ⵉⵙⴰⵡⴰⵍ ⵓⴷⵍⵉⵙ ⴰⴷ ⴳⴼ ⵜⵓⴽⵣⴰ ⵜⴰⵡⵜⵓⵎⴰⵜⵉⴽⵜ ⵏ ⵜⵉⵔⵔⴰ ⵙ ⵜⵎⴰⵣⵉⵖⵜ. ⴳ ⵎⵏⵉⴷ ⴰⴷ, ⵏⵙⴽⵔ ⵣⵡⴰⵔ ⵢⴰⵜ ⵜⵙⵉⵍⴰ ⵏ ⵜⵎⵓⵛⴰ ⵉⵥⵍⵉⵏ ⵙ ⵉⵙⴽⴽⵉⵍⵏ ⵏ ⵜⵎⴰⵣⵉⵖⵜ ⵜⵜⵢⴰⵔⴰⵏⵉⵏ ⵙ ⵓⴼⵓⵙ, ⵉⵍⵉⵏ ⴳⵉⵙ ⵓⴳⴳⴰⵔ ⵏ 25.000 ⵏ ⵉⵙⴽⴽⵉⵍⵏ ⵥⵍⵉⵏⵉⵏ, ⵜⵜⵢⴰⵔⴰⵏⵉⵏ ⵙ 60 ⵏ ⵉⵎⴰⵔⴰⵜⵏ. ⵏⵙⵙⵎⵔⵙ ⵜⴰⵙⵉⵍⴰ ⴰⴷ ⴳ ⵓⵙⵜⴰⵍ ⴷ ⵢⵉⵔⵉⵎ ⵏ ⵜⵏⴰⵟⵟⵓⴼⵉⵏ ⵏ ⵜⵡⵓⵔⵉⵡⵉⵏ ⵏⵖ.

ⴰⵔ ⵜⴰⴽⴽⴰ ⵜⵡⵓⵔⵉ ⴰⴷ ⵙⵏⴰⵜ ⵜⵎⵢⴰⴷⴰⵙⵉⵏ ⵏ ⵜⵓⴽⵣⴰ ⵜⴰⵡⵜⵓⵎⴰⵜⵉⴽⵜ ⵏ ⵜⵉⵔⵔⴰ ⵙ ⵜⵎⴰⵣⵉⵖⵜ, ⵍⵍⵉ ⵜⵜⴰⵡⵙⵏⵉⵏ ⴳ ⵓⵙⴼⵓⵍⴽⵉ ⵏ ⵜⵓⴽⵣⴰ ⴰⴷ. ⵜⴰⵎⵢⴰⴷⴰⵙⵜ ⵜⴰⵎⵣⵡⴰⵔⵓⵜ ⵜⴳⴰ ⵜⴰⵎⵙⵏⵉⵢⵜ, ⴰⵔ ⵜⵙⵙⵎⵔⴰⵙ ⵉⵡⵓⵔⵎⴰⵏ ⵍⵍⴰⵏⵉⵏ ⴳ ⵡⴰⴷⴷⴰⴷ ⵉⵙⴰⵍⴰⵏ, ⵃⵎⴰ ⴰⴷ ⵜⴰⴽⵣ ⵉⵙⴽⴽⵉⵍⵏ ⵙⵉⴳⴳⵥⵏⵉⵏ ⵏ ⵜⵎⴰⵣⵉⵖⵜ. ⴰⵔ ⵜⴰⴽⴽⴰ ⵜⵎⵢⴰⴷⴰⵙⵜ ⴰⴷ ⵢⴰⵜ ⵜⵏⴰⵟⵟⵓⴼⵜ ⵉⴼⵓⵍⴽⵉⵏ ⵖ ⵜⵙⵉⵍⴰ ⵏ ⵜⵎⵓⵛⴰ ⵏⵏⵖ ⵉⵥⵍⵉⵏ ⵙ ⵉⵙⴽⴽⵉⵍⵏ ⵙⵉⴳⴳⵥⵏⵉⵏ ⴷ ⵥⵍⵉⵏⵉⵏ. ⵓⵔ ⵉⵍⵍⵉ ⴷ ⵓⵙⵎⵉⴷ ⴰⵙⵎⴽⵍ ⵏ ⵉⵙⴽⴽⵉⵍⵏ ⵓⵔ ⵉⴱⴹⵉⵏ. ⵃⵎⴰ ⴰⴷ ⵏⵙⵓⵖⴷ ⵜⵎⵓⴽⵔⵉⵙⵜ ⴰ, ⵏⵙⵖⵉⵡⵙ ⵉⵊⵊ ⵏ ⵓⵏⴳⵔⴰⵡ ⵏ ⵜⵓⴽⵣⴰ ⵏ ⵜⵉⵔⵔⴰ ⵏ ⵜⵎⴰⵣⵉⵖⵜ ⵉⵏⵜⴰⵏ ⴳⴼ ⵢⵉⴼⵉⵍⵉ ⴰⵏⴰⵎⵎⴰⵙ ⴰⵎⵏⵉⴷⴰⵏ ⵏ ⵓⵙⴽⴽⵉⵍ. ⵉⵙⵙⵎⵔⵙ ⵓⵏⴳⵔⴰⵡ ⴰ ⵡⵉⵙⵙ ⵙⵉⵏ ⴰⵥⵟⵟⴰ ⵏ ⵉⵙⴽⵉⴽⵉⵢⵏ ⵉⴳⵉⵙⵡⵉⵔⵏ ⴰⵎ ⵓⵙⵎⵉⵍⴰⵏ. ⵉⵙⵙⴽⴰⵏ ⵉⵛⵜ ⵏ ⵜⵡⵓⵔⵉ ⵜⵛⵏⴰ ⴷⵉ ⵜⵙⵉⵍⴰ ⵏ ⵜⵎⵓⵛⴰ ⵏ ⵓⵙⵎⵎⵣⴷⴰⵢ ⵏ ⵜⵡⵏⵏⵖⴰ ⵏ ⵜⵎⴰⵣⵉⵖⵜ ⴷ ⵜⵙⵉⵍⴰ ⵏⵏⵖ ⵏ ⵉⵙⴽⴽⵉⵍⵏ ⵜⵜⵢⴰⵔⴰⵏⵉⵏ ⵙ ⵓⴼⵓⵙ ⵉ ⵏⵚⴽⵓ ⴰⴷⵖⴰⵔⴰⵏ. ⵉⵜⵜⵡⴰⵙⵙⵓⵎⵔ ⵓⵄⴷⴰⵍ ⵏ ⵓⵥⵟⵟⴰ ⴰ, ⵜⵜⵡⴰⴷⴼⵏⵜ ⴷⴰⴳⵙ ⵜⵉⴼⵔⴰⵙ ⵉⵏⵜⴰⵏ ⴳⴼ ⵢⵉⴼⵉⵍⵉ ⴰⵏⴰⵎⵎⴰⵙ ⴰⵎⵖⵜⴰⵡ ⵏ ⵓⵙⴽⴽⵉⵍ. ⵉⵡⵛⴰ ⵓⵄⴷⴰⵍ ⴰ ⵜⴰⵏⴰⵟⵟⵓⴼⵜ ⵜⵚⴱⵃ.

ⵜⵉⴳⵓⵔⵉⵡⵉⵏ ⵜⵉⴷⵙⵍⴰⵏⵉⵏ

ⵜⴰⴽⵜⵜⴰⵡⵜ ⴱⵕⵕⴰ ⵏ ⵢⵉⴼⵉⵍⵉ ⵏ ⵜⵉⵔⵔⴰ, ⵜⵉⵔⵔⴰ ⵏ ⵜⵎⴰⵣⵉⵖⵜ, ⵜⵉⴼⵉⵏⴰⵖ, ⵜⴰⵎⵢⴰⴷⴰⵙⵜ ⵜⴰⵎⵙⵏⵉⵢⵜ, ⵉⵡⵓⵔⵎⴰⵏ ⵍⵍⴰⵏⵉⵏ ⴳ ⵡⴰⴷⴷⴰⴷ ⵉⵙⴰⵍⴰⵏ, ⴰⵥⵟⵟⴰ ⵏ ⵉⵙⴽⵉⴽⵉⵢⵏ.

Remerciements

Je voudrais remercier, M. Driss Mammass, Directeur de l'EST d'Agadir et responsable du Laboratoire IRF–SIC à la faculté des sciences d'Agadir, pour m'avoir accepté au sein de son équipe dynamique. C'est grâce à son soutien et ses encouragements que ce travail a vu le jour. J'aimerais lui témoigner ici toute ma reconnaissance pour ses suggestions pertinentes ainsi que pour la confiance permanente qu'il m'a accordée.

Je souhaite témoigner de toute ma reconnaissance à mes directeurs de thèse :

- M. Ali Rachidi, Professeur à l'ENCG d'Agadir, avec qui j'ai eu la chance et le plaisir de travailler pendant ces années. Ses conseils aux niveaux scientifiques et personnels ont toujours été clairs et m'ont permis d'aboutir à la production de ce travail. Je le remercie aussi pour sa grande disponibilité, ses encouragements, sa générosité et pour tout le temps précieux que nous avons passé ensemble.
- M. Mostafa El Yassa, Professeur à la faculté des sciences d'Agadir, pour m'avoir fait confiance pour conduire ces travaux. Les discussions avec lui me permettent de mieux comprendre la recherche et d'avoir plus de confiance en moi. Je le remercie aussi pour sa rigueur scientifique, son aide et ses encouragements durant mes études supérieures.

Je remercie M. El Houssine Bouyakhf, Professeur à la faculté des sciences de Rabat, M. Mohamed Fakir, Professeur à la faculté des sciences et techniques de Beni Mellal et M. Hassan Douzi, Professeur à la faculté des sciences d'Agadir pour avoir accepté de rapporter ma thèse, pour les efforts que leur a demandés ce travail et pour leurs conseils.

Je remercie également M. Youssef Ait Ouguengay, Directeur de centre CEISIC de l'IRCAM, pour l'intérêt qu'il a porté à mon travail et d'avoir accepté de participer au jury en tant que membre invité. Je le remercie aussi pour son accueil, son encadrement tout au long du mon stage effectué à l'IRCAM et pour tous les services rendus.

Un grand merci à l'ensemble des membres du Laboratoire IRF-SIC avec qui j'ai partagé, pendant plusieurs années, leur collaboration et leurs encouragements. Je remercie en particulier El Hajji, Amrouch et Bakkass de leurs précieux conseils, de leur aide et de leur collaboration.

Merci beaucoup à tous mes amis qui, de près ou de loin m'ont aidé et encouragé aux moments opportuns. Je les remercie pour tout le temps précieux que nous avons passé ensemble.

J'adresse enfin mes profonds remerciements à toute ma famille, particulièrement à mes parents et à ma femme qui sont toujours à côté de moi et prêts à me soutenir.

A mes parents
A ma femme Fatima et mon fils Ayoub
A mes frères et soeurs

Table des matières

Liste des tableaux

Liste des figures

Liste des Abréviations

AB : Académie Berbère

ACM : Automate Canonique Maximale

AMHCD : Amazigh Handwritten Character Database

ANN : Artificial Neural Network

ARFF : Attribute-Relation File Format

ASCII : American Standard Code for Information Interchange

CAL : Centre de l'Aménagement Linguistique

CEI : International Electrotechnical Commission

CEISIC : Centre des Études Informatiques et des Systèmes d'Information et de Communication

GCM : Grammaire Canonique Maximale

HMM : Hidden Markov Model

IRCAM : Institut Royale de la Culture Amazighe

ISO : International Organization for Standardization

JavaNNS : Java Neural Networks Simulator

MLP : Multi-Layer Perceptron

OCR : Optical Character Recognition

PC : Personal Computer

PDA : Personal Digital Assistant

PNG : Portable Network Graphics

RLSA : Run Length Smoothing Algorithm

RNA : Réseau de Neurones Artificiels

SOM : Self Organizing Map

SVM : Support Vector Machine

UNL : Universal Networking Language

WEKA : Waikato Environment for Knowledge Analysis

Introduction générale

La reconnaissance automatique de l'écriture est attachée à la reconnaissance de caractères manuscrits et imprimés, la reconnaissance de mots et de texte, la reconnaissance du scripteur et la reconnaissance de documents. Elle relève du domaine de la reconnaissance des formes qui s'intéresse aux formes de l'écriture en plus des autres formes telles que: signatures, objets, etc. Le but est de transformer un texte écrit en une représentation compréhensible par une machine et facilement reproductible par un système informatique. La reconnaissance de l'écriture concerne plus précisément toutes les tâches en relation avec la lecture et la rétro-conversion de documents papiers tels que le traitement des formulaires, l'indexation et l'archivage pour le catalogage numérique, le tri automatique du courrier, la lecture de chèques, des interfaces sans clavier, analyse du geste écrit, interaction avec le stylo électronique, etc. L'automatisation de tout ou partie de ces tâches n'est pas triviale, car les écritures possèdent une infinité de représentations. En effet, il existe de nombreuses fontes pour l'imprimé avec de nombreux styles et des mises en page différentes, et aussi chaque scripteur produit une écriture variée qui lui est propre. Les systèmes de reconnaissance de l'écriture sont différents suivant le type d'écriture à traiter (manuscrit, cursif ou imprimé), d'où les opérations à effectuer et les résultats peuvent varier notablement.

Grâce aux progrès récents de la puissance informatique, de nombreuses techniques de reconnaissance de l'écriture ont été également améliorées et perfectionnées, notamment pour les écritures latines et arabes. Cependant, la grande variabilité inhérente à la nature de l'écriture manuscrite a rendu ce domaine de recherche très actif. Ainsi, ces dernières années, avec la croissance des moyens de communication, d'autres alphabets, tels que l'alphabet Tifinagh de la langue Amazighe ont intégré les systèmes d'informations. Ce qui a entraîné l'apparition d'autres types de documents où l'écriture n'est pas encore traitée et donc plus délicate à reconnaître. La reconnaissance de texte de tels documents nécessite des techniques de traitement plus spécifiques.

Cette thèse a pour objectifs de développer des méthodes de reconnaissance de caractères manuscrits et imprimés, de textes et de documents. La langue d'application de nos travaux est l'Amazighe dans toutes ces composantes telles que le caractère, le mot et le texte.

Dans nos travaux de recherches, nous intéressons à développer des systèmes de reconnaissance automatique de l'écriture amazighe (Tifinagh) qui se basent sur deux approches différentes.

En effet et en premier lieu, nous avons développé une base de données d'images de caractères amazighes manuscrits pour pouvoir expérimenter nos contributions. Cette base de données sera destinée aussi à servir d'autres chercheurs dans le domaine afin de standardiser la recherche sur la reconnaissance de l'écriture amazighe manuscrite.

Dans le second lieu, nous avons conçus et met en place un système de reconnaissance de caractères amazighes imprimés. Nous proposons dans cette contribution une approche syntaxique de reconnaissance de caractères amazighes segmentés. L'approche proposée se base sur les automates à états finis avec des primitives structurelles. Elle s'intéresse à la forme du caractère amazighe qui' est composée de primitives structurelles telles que des segments, des points et/ou des petits cercles. D'autre part, les automates finis conviennent bien à modéliser les caractères amazighes du fait qu'ils permettent de modéliser des structures non récursives et ils sont particulièrement adaptés pour modéliser et contrôler des systèmes à nombre d'états finis. Nous évaluons cette approche sur la base de caractères amazighes imprimés développée localement.

Parmi les limites de cette approche est qu'elle n'est pas applicable à tous les caractères amazighes. Et pour remédier à ces limites, nous proposons une nouvelle approche qui tient compte de tous les caractères amazighes. L'efficacité et l'apport scientifiques de l'approche proposée résident dans l'extraction des primitives de densités des pixels basées sur la position des lignes centrales horizontale et verticale du caractère. Ces primitives alimenteront un réseau de neurones multicouches dans les phases d'apprentissage et de reconnaissance. Nous évaluons ce système sur deux bases, une de patterns de la graphie amazighe et une autre base de caractères amazighes manuscrits. Nous montrons les résultats de reconnaissance obtenus en fonction de l'intégration des caractéristiques dépendantes et indépendantes à la ligne centrale horizontale du caractère. Ensuite, nous comparons ces résultats avec ceux obtenus en intégrant des caractéristiques basées sur la ligne centrale verticale du caractère.

Ce mémoire est composé de deux grandes parties. La première présente l'état de l'art ainsi que les concepts et notations requis pour la compréhension des systèmes développés. Cette partie est divisée en trois chapitres. La deuxième partie de ce rapport est constituée de trois chapitres qui présentent l'ensemble de nos contributions.

Chapitre 1 : Il présente l'état de l'art sur la reconnaissance automatique de l'écriture. En effet, ce chapitre décrit le principe, l'architecture d'un système de reconnaissance de l'écriture et les différentes techniques utilisées dans ce domaine.

Chapitre 2 : Il est consacré aux théories des approches syntaxiques basées sur les automates à états finis et des approches connexionnistes. Nous y présentons une brève synthèse des concepts de base de la théorie des langages formels et les formalismes des grammaires régulières et d'automates finis. Ensuite, nous y exposons l'architecture générale des réseaux de neurones artificiels, en particulier les perceptrons multicouches. Nous y dressons, enfin, l'application de ces formalismes sur la reconnaissance de l'écriture, ainsi que les algorithmes d'apprentissage de ces modèles.

Chapitre 3 : Il s'intéresse plus particulièrement à l'écriture amazighe et aux travaux effectués dans le domaine de la reconnaissance automatique de l'écriture amazighe. En effet, nous y présentons tout d'abord un aperçu général de la langue Amazighe, en s'intéressant aux efforts d'informatisation et la promotion de cette langue ainsi aux caractéristiques morphologiques de son écriture. Par la suite nous décrirons les différents travaux antérieurement effectués dans le domaine de la reconnaissance automatique de l'écriture amazighe avec les principales bases de données de caractères amazighes existantes.

Chapitre 4 : Il présente la base de données d'images de caractères amazighes manuscrits que nous avons construit dans le cadre de cette thèse. En effet, nous décrivons le processus de création de cette base, la collecte des données, la numérisation, l'extraction des caractères isolés dans le formulaire utilisé, ainsi que les opérations de prétraitements effectuées sur les images des caractères. Nous présentons ensuite le mode de stockage et d'étiquetage des images de la base, en terminant par quelques caractéristiques et détails statistiques de la base.

Chapitre 5 : Il introduit un système de reconnaissance automatique de caractères amazighes imprimés en utilisant une approche syntaxique basée sur les automates finis. En effet, nous décrivons les différentes étapes de ce système telles que les prétraitements effectués sur les caractères et la procédure d'extraction de la chaine représentative du caractère. Par la suite, nous modélisons chaque caractère amazighe par une grammaire régulière et puis par un automate fini et nous construisons l'automate global qui reconnait tous les caractères amazighes segmentés. Enfin, nous présentons les résultats obtenus sur la base de caractères imprimés construite localement en proposant quelques perspectives du travail.

Chapitre 6 : Il décrit notre troisième contribution à la reconnaissance automatique de texte amazighe. Nous présentons l'architecture du nouveau système de reconnaissance de l'écriture à travers ses différentes étapes de prétraitement, d'extraction des caractéristiques et de classification. L'étape de prétraitement comprend la binarisation et la réduction du bruit, la détection et la correction de l'inclinaison et la segmentation du texte en lignes et puis en caractères isolés. Dans l'étape de l'extraction des caractéristiques, et dans un

premier temps, nous utilisons la ligne centrale horizontale du caractère pour extraire un ensemble de caractéristiques de densités basées sur cette ligne. Ensuite, nous présentons le processus d'apprentissage et de reconnaissance en utilisant un réseau de neurones multicouches. Nous montrons également les résultats expérimentaux obtenus suite aux tests effectués sur deux bases de données (base des patterns de la graphie amazighe et une autre de caractères amazighes manuscrits). Dans un second temps, nous proposons une amélioration à ce système de reconnaissance en ajoutant d'autres caractéristiques de densités basées sur la ligne centrale verticale du caractère. Enfin, nous présentons les résultats obtenus par le système amélioré et leur comparaison avec ceux obtenus par le premier système.

Comme conclusion de ce mémoire, nous présentons un bref aperçu sur nos contributions tout en proposant de multiples perspectives de recherche.

PREMIÈRE PARTIE :

ÉTUDE BIBLIOGRAPHIQUE

Chapitre 1 : La reconnaissance automatique de l'écriture

1. Introduction

La reconnaissance automatique de l'écriture s'intéresse à la conception et à la réalisation de systèmes capables d'interpréter et de transformer un texte sous forme d'image en un fichier texte. Grâce aux progrès récents de la puissance informatique, de nombreuses techniques de reconnaissance de l'écriture ont été également développées et perfectionnées, notamment pour les écritures latines et arabes. Cependant, la grande variabilité inhérente à la nature de l'écriture manuscrite a rendu ce domaine de recherche très actif. Nous présentons dans ce chapitre, un aperçu de l'état de l'art des techniques dans les principaux modules de reconnaissance automatique de l'écriture. Tout d'abord, nous présentons quelques applications et aspects de la reconnaissance automatiques de l'écriture. Après cela, nous décrivons les différentes étapes d'un système de reconnaissance de l'écriture hors ligne, à savoir les prétraitements, l'extraction des caractéristiques et les méthodes classification, usuellement utilisées dans la reconnaissance de l'écriture. Dans la section consacrée aux prétraitements, nous mettons l'accent sur les méthodes de binarisation, lissage, détection et correction d'inclinaison, segmentation, localisation de ligne de base, squelettisation et normalisation. Dans la section de l'extraction de caractéristiques, nous citons les différents types de caractéristiques appliquées pour la reconnaissance de l'écriture en présentant leurs méthodes d'extraction couramment utilisées. Dans la section classification, nous décrivons et analysons les principales approches de reconnaissance utilisées dans le domaine de reconnaissance de l'écriture. Enfin, nous présentons le principe de la reconnaissance de mots et de document en citant quelques techniques couramment utilisées.

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

2. Applications et aspects de la reconnaissance de l'écriture

2.1. Applications de la reconnaissance de l'écriture

La reconnaissance de l'écriture est mieux connue sous le nom d'OCR (Optical Character Recognition), du fait de l'emploi de procédés d'acquisitions optiques. Ce domaine a connu ces dernières années de grands progrès. Les succès des travaux de recherches ont donné lieu à de nombreuses applications dans plusieurs domaines d'activité parmi lesquelles on peut citer :

- la lecture automatique de formulaires et de documents administratifs ([Coua02], [Mile06]) ;
- la lecture des adresses postales et le tri automatique du courrier ([ElYa02], [Gran03]) ;
- l'authentification et la lecture de chèques bancaires par la reconnaissance des montants littéraux et des montants numériques manuscrits ([Leth96], [Cher00], [Soui06]) ;
- l'échange de fichiers informatisés à distance dans le domaine de télécommunications [Smit85] ;
- l'indexation et l'archivage automatique de documents [Ouss08] ;
- la reconnaissance de documents techniques (schémas électronique, dessins techniques, plans architecturaux, plans cartographiques, etc.) [Naka87] ;
- la transcription assistée par ordinateur appliqué à la reconnaissance du manuscrit mono-scripteur qui permettra à la rétro-conversion de manuscrits anciens [Bouc00] ;
- la reconnaissance de numéros minéralogiques pour le contrôle routier, l'authentification et l'identification de manuscrits et l'identification du scripteur ;
- la recherche d'information dans une base de documents manuscrits telle que l'identification du scripteur [Bene03] ;
- les applications de reconnaissance de l'écriture en ligne [Conn02] à travers les PDA, Tablet- PC, Ordinateurs sans clavier, ou stylo caméra.

2.2. Reconnaissance en ligne et hors-ligne

Suivant la nature des informations disponibles pour la reconnaissance, les systèmes de reconnaissance automatique de l'écriture peuvent être divisés en deux catégories principales : les systèmes de reconnaissance en ligne et les systèmes de reconnaissance hors ligne.

Reconnaissance en ligne : Dans ce type de reconnaissance, l'utilisateur écrit sur une table spéciale, le système va reconnaître l'écriture et envoyer le résultat à l'ordinateur. Ces systèmes sont utilisés dans plusieurs équipements électroniques comme PDA, Pocket PC ou Tablet PC. La reconnaissance d'écriture en ligne présente des avantages par rapport à celle hors-ligne. Par exemple, du fait que l'utilisateur écrit sur une table spéciale, il y a

moins de bruit. De plus, on peut déterminer comment un caractère est écrit, c'est à dire, l'ordre de traits constituants ce caractère. D'ailleurs, la contrainte du temps de reconnaissance n'est pas stricte, on peut utiliser des algorithmes complexes. C'est pourquoi le taux de reconnaissance de ces systèmes est assez élevé.

Reconnaissance hors ligne : Dans le cas de reconnaissance hors ligne, l'écriture de l'utilisateur est acquise par un numériseur. L'entrée du système de ce type de reconnaissance est une image numérisée d'un document préalablement rédigé, d'où la difficulté du problème. Par exemple, comment on peut enlever le bruit comme élément du fond d'image, comment on peut traiter le manque des traits, etc. De plus, il n'y a pas d'information supplémentaire comme le cas d'en ligne. D'ailleurs, pour être utilisés largement, ces systèmes doivent avoir un temps de traitement rapide et un taux de reconnaissance élevé.

2.3. Reconnaissance de l'imprimé ou du manuscrit

L'OCR peut être subdivisé en reconnaissance de l'écriture manuscrite et de reconnaissance de l'écriture imprimée. L'approche n'est pas la même selon qu'il s'agisse de reconnaître de l'imprimé ou du manuscrit. La reconnaissance de caractères manuscrits est plus difficile à appliquer que la reconnaissance de caractères imprimés. En effet, dans le cas de l'imprimé, les images des caractères à traiter sont mise en forme par des polices standard qui sont bien alignés et souvent bien séparés verticalement, ce qui simplifie la phase de lecture. De plus, le graphisme des caractères est conforme à un style calligraphique (fonte) qui constitue un modèle pour l'identification. Dans le cas du manuscrit, il existe divers styles d'écriture humaine dont les caractères sont souvent ligaturés et leur graphisme est inégalement proportionné. Cela nécessite l'emploi de techniques de délimitation très spécifiques et souvent des connaissances contextuelles pour guider la lecture.

3. Architecture d'un système de reconnaissance de caractères et de mots

Les tâches essentielles de reconnaissance d'écriture hors-ligne sont la reconnaissance de caractères et la reconnaissance de mots. L'architecture d'un système de reconnaissance de caractères ou de mots isolés est composée de trois grandes étapes : le prétraitement, l'extraction des primitives et la classification [Jain00] (cf. Figure 1.1 ci-dessous).

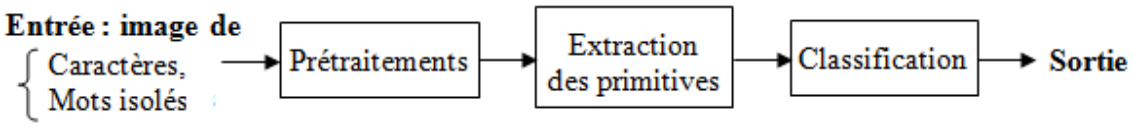


Figure 1.1 : Système de reconnaissance de caractères ou de mots isolés

La phase des prétraitements a pour but la réduction du nombre de données pour n'en garder que les informations utiles. Il s'agit de traitements dits de bas niveau (binarisation, réduction du bruit, segmentation… etc.).

La phase d'extraction des primitives permet de faire à partir de la représentation de l'image une description synthétique de la forme à reconnaître dans un espace à plusieurs dimensions.

La phase de classification est la phase de reconnaissance du caractère ou du mot. Cette étape implique l'existence d'une base de données d'apprentissage. La description du caractère à reconnaître est comparée aux descriptions des caractères de la base.

Les deux premières phases peuvent se résumer en une étape dite d'analyse durant laquelle des caractéristiques décrivant les formes de l'image sont extraites. En plus de ces trois phases, on pourra trouver une phase de post-traitement qu'est une opération facultative et dont le rôle consiste à vérifier et éventuellement à corriger l'hypothèse de reconnaissance générée par le module de classification en utilisant des informations contextuelles telles que: lexicales, syntaxiques, sémantiques, pragmatiques, etc.

Dans un premier temps, nous allons faire un état de l'art des techniques de prétraitement ensuite, des techniques d'extraction des primitives puis celui des différentes méthodes de reconnaissance qui existent.

3.1. Prétraitements

Il est nécessaire d'effectuer une série de prétraitements avant de reconnaître du texte dans des documents numérisés. Ces prétraitements préparent l'image d'entrée afin de faciliter l'étape d'extraction des caractéristiques. Certaines des opérations courantes réalisées préalablement à la reconnaissance de l'écriture sont les suivantes :

- La binarisation / le seuillage : il s'agit de convertir l'image en noir et blanc ou en niveaux de gris ;
- La réduction du bruit : elle vise à réduire le bruit et les imperfections de l'image ;
- Le redressement des écritures penchées : il corrige l'inclinaison de l'écriture ;
- La segmentation en lignes : elle sépare les lignes individuelles de texte ;
- La segmentation en mots et en caractères: elle sépare les lignes de texte en mots, puis en caractères ;
- La squelettisation : elle vise à obtenir une épaisseur du trait d'écriture égale à 1 pixel ;

- La normalisation : elle ramène les images de caractères ou de mots à des tailles standards. Certaines méthodes cherchent même à normaliser localement les différentes parties d'un mot.

3.1.1. Binarisation

La binarisation c'est le passage d'une image en couleur ou définie par plusieurs niveaux de gris en image bitonale (composée de deux valeurs 0 et 1) qui permet une classification entre le fond (image du support papier en blanc) et la forme (traits des gravures et des caractères en noir).

La binarisation des documents bruités reste un domaine de recherche actif. Plusieurs méthodes de binarisation d'images de documents ont été appliquées et étudiées dans la littérature. Une étude comparative de différentes méthodes a été donnée par Trier et al. ([Trie95a], [Trie95b]). Aussi, Sezgin et al. [Sezg04] proposent un tour d'horizon complet des différentes méthodes de binarisation, en les décrivant et en les évaluant. La plupart des méthodes sont basées sur un calcul de seuil afin de délimiter les deux classes. Celui-ci peut être global ou local.

Dans les approches globales ([Ostu79], [Tao02], [Tabb03]), ce seuil est calculé sur l'image entière et est ensuite uniformément appliqué à tous les pixels de l'image pour diviser l'histogramme des niveaux de gris en deux classes correspondant au fond et à l'objet. L'analyse de l'histogramme des niveaux de gris permet d'identifier ce seuil. La méthode d'Otsu [Ostu79] se base sur le même principe. Elle effectue une analyse statistique sur les histogrammes (variance intraclasse et variance interclasses) pour définir une fonction à maximiser qui permet d'estimer le seuil. Les méthodes de seuillage global donnent donc de bons résultats lorsque l'image contient des classes qui vérifient une certaine homogénéité, traduite par un histogramme multimodal. Ces méthodes ne sont pas efficaces pour des sources trop bruitées [Gaba08]. Il devient alors nécessaire d'employer des techniques avec un seuillage adaptatif.

Dans les approches locales ([Eikv91], [Kame93], [OGor94], [Yang00]), le seuil n'est plus unique, mais est déterminé pour chaque pixel ou pour des régions de pixels aux caractéristiques voisines. Ces approches sont plus précises, [Sauv00] mais elles utilisent différents seuils ou une actualisation selon la région considérée. Ils sont plus sensibles au bruit, mais donnent de meilleurs résultats dans la séparation des composantes.

Enfin, d'autres travaux permettent de combiner des approches globales et locales pour faire la binarisation en utilisant des techniques qui consistent à déterminer localement une région modèle dont les caractéristiques sont ensuite utilisées pour traiter l'image entière ([Chan99], [Sauv00], [Gaba05], [Yoko06]). Ces approches sont mieux appropriées au traitement des documents graphiques [Tabb06].

3.1.2. Réduction du bruit

Le bruit, une erreur aléatoire dans la valeur de pixel, est une valeur découlant habituellement de la reproduction, de la numérisation et de la transmission de l'image originale. Le bruit peut être réparti en trois catégories : bruit dépendant du signal, bruit non dépendant du signal et bruit noir et blanc. La réduction du bruit consiste à détecter et à éliminer les pixels qui le représentent. Le lissage et la suppression de bruit peuvent être obtenus par un filtrage. Le filtrage est une opération de voisinage, la valeur d'un pixel est remplacée par la valeur d'une fonction appliquée à ce pixel et à ses voisins. Dans le domaine de reconnaissance de l'écriture, plusieurs méthodes ont été utilisées pour éliminer le bruit [Khar99a].

Les opérateurs de morphologie mathématique [Mamm99] sont souvent utilisés pour le lissage ([Bena99], [Dehg01], [Khor03]). L'opérateur de fermeture (c.-à-d. une dilatation suivie d'une érosion) permet d'éliminer les petits trous et d'éliminer le bruit sur le contour, tandis que l'opérateur d'ouverture (c.-à-d. une érosion suivie d'une dilatation) permet de découper les isthmes étroits et d'éliminer les petits îlots, les pics anguleux et les capes dans l'écriture arabe [Alba95].

3.1.3. Détection et correction d'inclinaison

Les méthodes de correction d'inclinaison des lignes de texte (également appelée correction de "skew") sont utilisées pour redresser horizontalement les lignes d'écriture obliques. L'inclinaison peut provenir de la saisie, si le document a été placé en biais, de la numérisation, si la personne qui s'occupe de la numérisation a tourné un peu la page avant de la scanner. Il convient alors de redresser le document afin de retrouver la structure de lignes horizontales d'une image texte. A cet effet, deux étapes sont appliquées. Dans un premier temps, l'angle d'inclinaison θs est estimé avec un algorithme de détection de l'angle d'inclinaison. Plusieurs méthodes sont disponibles pour la détection d'inclinaison des lignes de texte. Les deux plus populaires sont la transformée de Hough (appliquée sur les centres de gravité des composantes connexes), et les histogrammes de projection.

La méthode de projection ([Pavl04], [Bagd97], [Kava02]) est basée sur le calcul des histogrammes. Elle est facile à implémenter, appropriée pour des documents à structure simple, mais non appropriée pour des documents complexes.

La transformée de Hough, ([Le94], [Berg98], [Jain96], [Pera99], [Amin96a]) permet de détecter l'angle d'inclinaison avec un intervalle de détection est compris entre 0° et 180°. Cette méthode est exacte, robuste et adéquate pour des documents multicolonnes, mais elle nécessite un espace mémoire important et le temps de traitement est très considérable.

Dans un deuxième temps, une fois l'angle d'inclinaison θs est estimé, l'image est tournée par l'angle estimé θs dans la direction opposée pour faire la correction. Pour ce faire, une

transformation de rotation de coordonnées peut être utilisée pour corriger l'inclinaison de l'image du document. Étant donné un point dans l'image, ses nouvelles coordonnées après la rotation de l'image entière autour de son origine par l'angle θs peuvent être calculées par l'équation :

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

3.1.4. Segmentation

La segmentation est une étape critique et décisive dans plusieurs systèmes de reconnaissance. En d'autres termes, l'efficacité d'un système de reconnaissance en dépend fortement. Elle a pour objectif d'isoler les différentes composantes d'un bloc de texte en lignes, en mots, et en caractères, avant la phase de reconnaissance de caractères. Le résultat de cette opération est une forme isolée à partir d'une image et qui pourrait être un caractère ou non. Les méthodes de segmentation dépendent en particulier de la langue traitée, des fontes utilisées, de la justification, de la variation de l'écriture manuscrite et aussi de leur interaction avec l'étape de reconnaissance. Cela peut conduire, selon les cas, à des opérations de séparation d'une lettre en graphèmes ou à des regroupements de quelques caractères en un seul mot.

Tout système de reconnaissance d'écriture doit impérativement localiser l'information avant de la reconnaître. Nous présentons dans ce qui suit le principe de la segmentation du texte en lignes, segmentation des lignes en mots, segmentation des mots en lettres.

3.1.4.1. Segmentation du texte en lignes

De nombreuses méthodes ont été développées pour réaliser une segmentation d'un document en lignes d'écriture. Une des plus faciles et des plus anciennes méthodes de segmentation en lignes est basée sur les histogrammes de projection horizontaux des pixels noirs, les minima locaux de l'histogramme étant considérés comme un espace interlignes. Cette méthode a été appliquée pour le latin ([Mart01], [Vinc04], [Nico09]) et pour l'arabe ([Rome95], [Syia06]), son principal inconvénient est qu'elle ne convient pas aux documents dont les lignes sont trop fortement inclinées. Surtout l'apparition fréquente des points diacritiques pour des écritures cursives de nature comme l'écriture l'arabe génère des fausses lignes en créant des faux minima qui correspondent aux espaces entre les caractères et leurs marques diacritiques.

D'autres méthodes ont été proposées dans la littérature, telles que les techniques par accroissement de groupes d'entités connexes ([Srih05], [Faro05], [Ball06]) et les méthodes par transformée de Hough ([Likf95], [Loul09]). On peut se référer à l'état de l'art de

Likforman-Sulem [Likf07] pour une revue des algorithmes de segmentation de documents en lignes de textes.

3.1.4.2. Segmentation en mots et en caractères

Une fois les lignes de textes extraites, il faut isoler les mots entre eux afin de les reconnaître. Il s'agit d'un problème de classification à deux classes dans lequel on essaie de différencier les espaces inter-mots et intra-mots. Ceci est plus compliqué dans le cas de l'écriture cursive et surtout l'existence des pseudos mots comme pour le cas de l'écriture arabe. Les méthodes de segmentation des lignes de textes en mots les plus utilisées sont celles à base de métrique, elles emploient l'histogramme de projections verticales [Bedd95] pour déterminer les espaces inter-mots. Ces méthodes peuvent échouer quand les mots se chevauchent horizontalement. Pour résoudre ce problème, d'autres méthodes procèdent par étiquetage des composantes connexes ([Mota97], [Mile98]) et regroupement de ces composantes [Faro05], ou par squelettisation [Abuh98].

Le caractère est le plus petit élément dans l'écriture d'une langue. La segmentation en caractères (ou en partie de caractères) constitue l'un des problèmes les plus durs liés à la reconnaissance de l'écriture. Il existe plusieurs approches pour la mise en œuvre d'une segmentation en caractères : les approches basées sur des analyses par morphologie mathématiques [Chen95], [Mota97]); les approches basées sur l'analyse des contours ([Alpe97], [Ball06]); les approches basées sur l'analyse du squelette ([Abuh98], [Zerm07]) et les approches basées sur l'analyse du profil d'histogramme de projection verticale ([Ameu92], [Syia06]). Ces dernières sont les plus simples à appliquer, mais elles sont assez peu utilisées pour segmenter les mots manuscrits à cause de la présence fréquente de chevauchements inter-pseudo-mots, des marques diacritiques et de ligatures verticales et à la variabilité de l'épaisseur de trait d'écriture.

3.1.5. Localisation des lignes d'écriture de base

La localisation des lignes d'écriture de base (ou lignes de référence) est une phase de prétraitement assez répandue dans les systèmes de reconnaissance hors ligne. Les lignes d'écriture de base comportent des informations importantes pour les systèmes de reconnaissance des manuscrits. Ces informations sont utilisées dans les phases de normalisation de l'inclinaison, de segmentation et d'extraction des caractéristiques liées à la position ([Blum02], [Pech03], [Blum03], [Lori05], [Elha07]).

Dans la littérature, différentes approches, basées sur les positions des lignes d'écriture, ont été proposées pour la reconnaissance de l'écriture. Dans le cas de l'écriture latine et arabe, plusieurs positions de lignes de base ont été utilisées pour extraire des caractéristiques qui dépendent de ces lignes ([Elha05], [Shan07], [Alsh08], [Aida09], [Razz10]). La Figure 1.2 ci-dessous illustre des exemples de lignes de base utilisées pour ces écritures [Elha05]. En

outre, plusieurs techniques ont été élaborées pour détecter les lignes de référence des mots manuscrits. Parmi ces techniques, on peut citer les méthodes suivantes : le profil vertical de projection horizontale des pixels ([Blum03], [Elha05], [Razz10]) ; la transformation de Hough [Pech03] ; les minima des contours inférieurs ([Faro05], [Beno08b]) et l'approche par voisinage et composantes connexes [Blum02]. On peut se référer à la thèse de Ramy El-Hajji [Elha07] pour une revue de ces techniques utilisées pour l'estimation des lignes d'écriture de base.

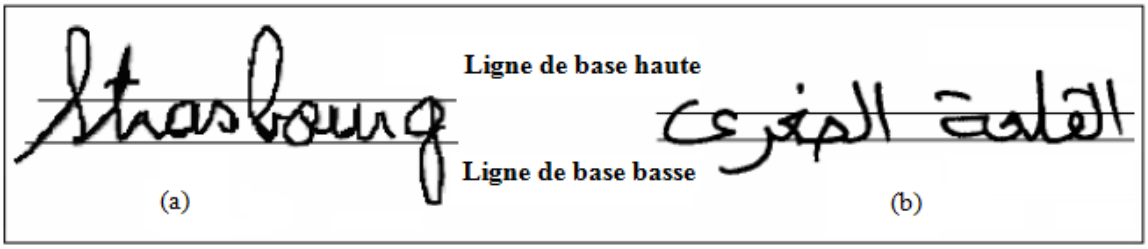


Figure 1.2 : Exemple de lignes de base d'écriture. (a) cas de l'écriture latine, (b) cas de l'écriture arabe [Elha05]

3.1.6. Squelettisation

Le squelette est une représentation d'une forme très utilisée, car il conserve les propriétés topologiques de la forme qu'il représente. La notion de squelette est apparue pour l'étude des objets minces. La squelettisation est une opération souvent utilisée en modélisation ou en représentation de l'écriture, car l'écriture est assimilable à un long ruban replié sur lui-même et d'épaisseur constante. Le modèle à l'origine de la production de l'écriture est un modèle linéaire sans épaisseur, de sorte que la représentation par le squelette est naturelle et d'aspect proche de l'écriture. L'opération de squelettisation consiste donc, dans le cas particulier de l'écriture, à éliminer l'épaisseur du trait ou plutôt à l'amincir jusqu'à l'épaisseur minimale d'un pixel.

De nombreux algorithmes ont été conçus pour la squelettisation de l'écriture ([Ruto66], [Hild69], [Zhan84], [Rose84], [Chin87], [Guo89], [Hall89], [Wu92]). Dans [Arri06], Denis Arrivault a fait une étude comparative de ces algorithmes sur différents caractères plus ou moins bruités en présentant le nombre d'itérations et le temps de calcul pour chaque algorithme. Nous décrivons ici l'algorithme de Zhang-Suen [Zhan84] que nous avons utilisé pour la squelettisation de caractères amazighes imprimés qui sera présenté dans le chapitre 5. Cet algorithme est très facile à mettre en œuvre. Afin de présenter cet algorithme, nous noterons le voisinage N(P) d'un point P de l'image comme explicité sur la Tableau 1.1 ci-dessous. De plus, nous conviendrons, naturellement, que les objets d'intérêts d'une image binaire sont représentés par les pixels de valeur 1 et que, par conséquent, les

pixels de valeur 0 représentent le fond ou les trous. Nous représenterons le nombre de transitions 0→1 qui se produisent tout en traversant les huit voisins du point P par X (P). Nous noterons, enfin, b(P) le nombre de pixels noirs, donc égaux à 1, dans N(P).

P1	P2	P3
P8	**P**	P4
P7	P6	P5

Tableau 1.1 : Pixels de N(P)

L'algorithme de squelettisation de Zhang-Suen utilise deux sous-itérations pour repérer les pixels à effacer selon les critères suivants :

– Première sous-itération :

- Z1 : $2 \leq b(P) \leq 6$
- Z2 : X (P) = 2
- Z3 : $p2 \times p4 \times p6 = 0$
- Z4 : $p4 \times p6 \times p8 = 0$

– Deuxième sous-itération :

- Z1b : idem Z1
- Z2b : idem Z2
- Z3b : rotationde180° par rapport à Z3 c.-à-d. $p2 \times p6 \times p8 = 0$
- Z4b : rotation de180° par rapport à Z4 c.-à-d. $p2 \times p4 \times p8 = 0$

Cet algorithme est relativement efficace et robuste au bruit de contour. Néanmoins certaines parties du squelette peuvent être décalées par rapport au centre.

Récemment, une étude comparative de différents algorithmes parallèles de squelettisation appliqués sur les caractères amazighes a été présentée dans [Faki09] avec le nombre d'itérations et le temps de calcul pour chaque algorithme.

Dans les méthodes de squelettisation, nous pouvons noter deux problèmes particuliers: l'apparition de barbules, et le comportement au niveau des intersections.

Après la squelettisation, un grand nombre de petits segments du squelette peuvent s'avérer non pertinents, comme le montre la Figure 1.3 ci-dessous. Toutefois, la squelettisation conserve un ensemble de petits traits, appelés barbules, sans importance dont la suppression ne modifie pas le nombre de composantes connexes. Il convient alors d'appliquer des post-traitements afin d'obtenir un squelette vectorisé en supprimant ces barbules. Une solution

adoptée à ce problème de barbules consiste à nettoyer le squelette en éliminant les traits dont la taille est inférieure à un certain seuil déterminé expérimentalement [Khor03].

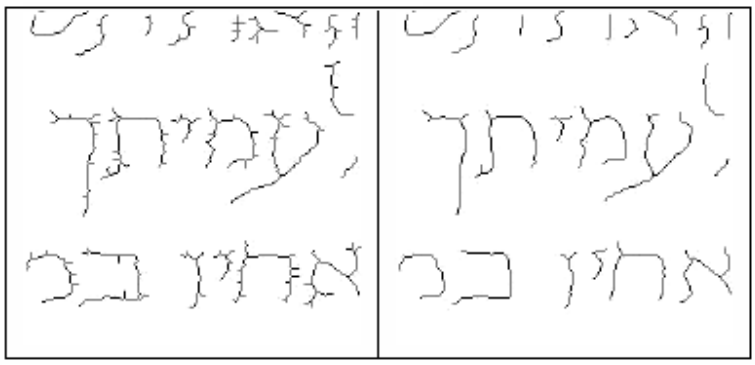
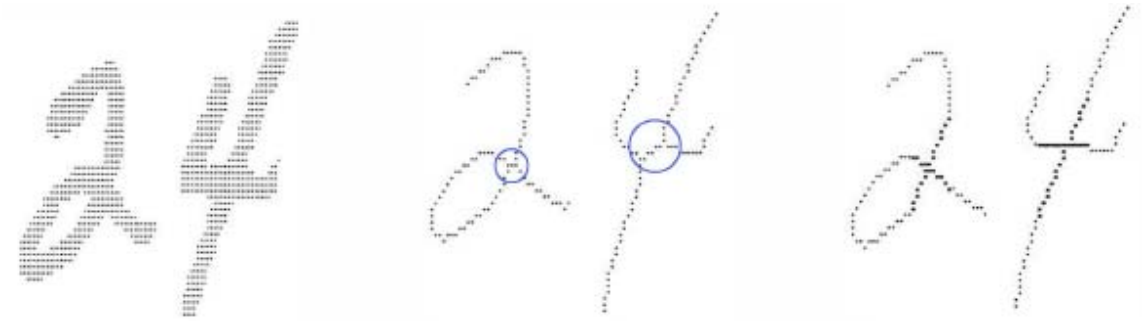


Figure 1.3 : Barbules (figure extraite de [Dupr03]). Image de gauche : squelette brut. Image de droite : squelette nettoyé de ses barbules

Plusieurs critères géométriques ont été proposés pour supprimer ces barbules ([Jang92], [Sann94], [Huan03]), mais il n'y a pas de règles pour ce nettoyage, il dépend à la fois de l'algorithme de squelettisation employé et de la précision désirée.



(a) (b) (c)

Figure 1.4 : Correction du squelette afin de mieux représenter les intersections (figures extraites de [Zhon99]), (a) image initiale, (b) squelette initial obtenu avec algorithme de Zhang-Suen [Zhan84], (c) squelette corrigé

Pour le problème des intersections entre deux segments de droite, qui se retrouvent parfois scindées comme le montre la Figure 1.4 ci-dessus, la méthode développée dans [Zhon99] propose d'améliorer la représentation des intersections dans le squelette. Cette méthode corrige le squelette obtenu en Figure 1.4-b pour aboutir à celui de la Figure 1.4-c. Elle est appliquée sur des caractères chinois qui présentent souvent des intersections croisant un trait horizontal et un trait vertical en parcourant l'image selon plusieurs directions, pour

déterminer les zones d'embranchements et marquer les points caractéristiques d'une intersection. En fin, le squelette est corrigé en effaçant la partie incluse entre ces points puis en faisant converger vers un point unique tous les arcs de squelette qui sont connectés dans cette zone (Figure 1.4-c).

3.1.7. Normalisation

La taille d'un caractère peut varier d'une écriture à l'autre, d'une fonte à l'autre et au sein d'une même fonte après agrandissement ou réduction, ce qui peut causer une instabilité des paramètres. La normalisation des caractères est considérée comme l'opération de prétraitement la plus importante pour la reconnaissance de caractères [Lee93]. L'objectif de la normalisation des caractères est de réduire la variation intra-classe de la forme des caractères, afin de faciliter le processus d'extraction de caractéristiques et d'améliorer leur précision pendant la classification. Une technique naturelle de normalisation consiste à ramener les caractères à la même taille. Plusieurs méthodes de normalisation ont été rencontrées dans la littérature. La plupart des méthodes ont été utilisées pour la normalisation de l'écriture chinoise manuscrite qui renferme une large variété de symboles. Elles sont aussi utilisées pour la normalisation des chiffres manuscrits. Il existe trois méthodes différentes : la normalisation linéaire, la normalisation à base des moments [Case70] et la normalisation non linéaire basée sur l'égalisation de densité des lignes ([Tsuk88], [Yama90], [Lee93]). On peut se référer au livre de Cheriet et al. [Cher07] pour une revue des algorithmes de normalisation de caractères.

D'autres méthodes basées sur la transformation de la fonction du contour par génération des caractéristiques normales (coefficients de Fourier) sont utilisées pour la normalisation de caractères imprimés [Szmu97]. Aussi certaines méthodes cherchent même à normaliser localement les différentes parties d'un mot : par exemple, on veut que la partie centrale, les hampes et les jambes occupent chacun un tiers de la hauteur [Saon97].

3.2. Extraction de caractéristiques

La phase d'extraction des primitives (encore appelées caractéristiques) est une étape très importante pour un système de reconnaissance de l'écriture. L'objectif de cette étape est la sélection des informations les plus pertinentes pour une tâche de classification donnée. Cependant, la problématique de cette étape a pour origine le risque de perte d'informations significatives.

Dans la littérature, plusieurs méthodes d'extraction de primitives pertinentes ont été appliquées pour la reconnaissance de l'écriture. Aussi, les primitives utilisées sont généralement classées selon trois façons différentes [Lema07].

- Une première distinction est faite entre les primitives selon qu'elles sont extraites d'une image en niveaux de gris, d'une image binarisée, du contour ou du squelette de la forme résultant de la phase de prétraitement ([Trie95], [Zhan04]). En effet :
  - dans le cas d'une image en niveaux de gris, les primitives extraites sont : des primitives du type zoning, moments géométriques ou de Zernike, des primitives discrètes : largeur, hauteur, épaisseur moyenne des traits, etc ;
  - dans le cas d'une image binarisée, les primitives extraites sont : des composantes connexes [Dela03], des projections, des histogrammes, des moments géométriques ou de Zernike, des primitives du type zoning et des primitives discrètes ;
  - dans le cas du contour, les primitives extraites sont : des profils [Shri84], des descripteurs de Fourier ou d'ondelettes, des primitives du type zoning, code de Freeman [Dupr03], contour code [Verm03] et des contours actifs tels que les snakes [Seni94] ainsi que des primitives discrètes [Zhan04] (périmètre, compacité, excentricité, etc.) ;
  - dans le cas du squelette, les primitives extraites sont : des descripteurs de Fourier, des descriptions de graphe ou des primitives discrètes (nombres de boucles, de croisements, rapport largeur sur hauteur, présence d'un point isolé, etc.) ;
- Une deuxième distinction est effectuée entre les primitives topologiques, structurelles ou statistiques [Trie95] ;
- Une troisième distinction est effectuée entre les primitives globales et les primitives locales [Darg94].

Dans la suite, nous présentons les principales caractéristiques structurelles et topologiques, les caractéristiques statistiques ainsi que les caractéristiques globales et locales.

Les primitives structurelles et topologiques ([Yama83], [Liu97], [Alpe97], [Kapo03], [Rach06a]) : décrivent les propriétés topologiques et géométriques de l'écriture. Il s'agit de compter dans l'image brute, ou à partir du squelette ou du contour de la forme :

- les segments de droite et leurs attributs (position, orientation, ...) ;
- la hauteur et la largeur du caractère ;
- les positions relatives entre segments ;
- les arcs, boucles et concavités, mesures de pentes et autres paramètres de courbures pour évaluer des orientations principales ;
- les points d'extrémités, de croisements et de jonctions des traits ;
- les angularités, points extremums et points terminaux ;

- le nombre de points diacritiques et leur position par rapport à la ligne de base de l'écriture ;
- autres paramètres tels que: la longueur et l'épaisseur des traits, le nombre de trous, les surfaces et les périmètres.

Les primitives structurelles constituent des caractéristiques très informatives et discriminantes. Elles permettent de prendre des décisions rapides dans la reconnaissance de l'écriture avec une complexité de calcul modérée pendant l'extraction, en comparaison aux caractéristiques statistiques.

Les primitives statistiques décrivent une forme en un ensemble de mesures statistiques extraites à partir de cette forme ([Rath03], [Elha05]). La raison principale de l'utilisation des primitives statistiques est de donner des informations locales sur le contenu de l'écriture. Il s'agit par exemple de :

- le profil de projection des densités de pixels noir/blanc qui représente le nombre de pixels sur chaque ligne ou chaque colonne de l'image ;
- l'histogramme directionnel permettant de compter le nombre de pixels sur une ligne dans une direction quelconque de l'image ;
- l'histogramme des transitions qui permet de retenir le nombre des transitions 0-1 et 1-0 entre pixels ;
- les moyennes locales de pixels de l'image situés à l'intérieur d'un masque rectangulaire (Zoning) ;
- les directions des contours dans une fenêtre locale ;
- les moments invariants qui sont des caractéristiques intéressantes, car elles sont invariantes en translation, taille et rotation. Ce sont des mesures statistiques de la distribution des pixels autour du centre de gravité du caractère. Ils ont été initialement appliqués à la reconnaissance de l'écriture latine et arabe, puis récemment ont été appliqués aux caractères amazighes [Elay10].

Avec ce type de caractéristiques, aucune interprétation directe n'est faite sur le contenu de l'écriture. Ainsi, ces caractéristiques se trouvent moins fortes que les primitives structurelles. Du fait que ces caractéristiques portent peu d'information sur l'écriture, un système de reconnaissance doit les gérer efficacement [Elha07].

Les primitives globales cherchent à représenter au mieux la forme générale d'un caractère et sont donc calculées sur des images relativement grandes. Citons par exemple la transformée de Fourier et la transformée de Hough qui détecte les lignes dans les images [Touj03].

Les primitives locales : sont calculées lors d'un parcours des pixels de l'image avec un pas d'analyse qui dépend de la modélisation, du type de primitive et de la taille de l'image. Parmi ces primitives, citons par exemple la transformée de Fourier fenêtrée et les dérivées qui extraient une information sur la direction des traits et les moments invariants [Schm96].

Finalement, plusieurs méthodes en littérature proposent la combinaison de différentes familles de caractéristiques afin d'obtenir plusieurs représentations d'une même forme et d'améliorer la discrimination ([Heut98], [Brit04], [Xue06]). Lorsque le nombre de caractéristiques devient trop élevé, des méthodes de sélection de caractéristiques peuvent être mises en œuvre ([Somo99], [Guyo03], [Oliv06]).

3.3. Classification

Dans le processus complet d'un système de reconnaissance de formes, la classification joue un rôle important en se prononçant sur l'appartenance d'une forme à une classe. L'idée principale de la classification est d'attribuer un exemple (une forme) non connu à une classe prédéfinie à partir de la description en paramètres de la forme. Plusieurs approches de classification sont utilisées dans le domaine de reconnaissance de formes qui sont plus ou moins bien adaptées à la reconnaissance de l'écriture. D'après [Jain00], les techniques de reconnaissance et de classification de textes sont regroupées en quatre catégories principales : méthodes par appariement de formes (pattern matching), méthodes statistiques, méthodes structurelles ou syntaxiques et méthodes neuronales. À cette catégorisation, nous pouvons ajouter les approches stochastiques. Ces deux dernières classes peuvent être considérées comme des sous-familles des méthodes statistiques. Ci-dessous, nous décrivons ces principales approches de classification utilisées dans le domaine de reconnaissance de l'écriture.

3.3.1. Approches par appariement de formes

L'appariement de formes (Template Matching en anglais) est l'une des méthodes de classification les plus courantes. La classification est effectuée en comparant une forme d'entrée avec un ensemble de modèles (ou prototypes) de chaque classe de caractère via une mesure de similarité. Il y a trois types de mesures qui sont couramment utilisées pour juger de la qualité de mise en correspondance (appariement) [Heut94] : les mesure de ressemblance du type intercorrélation ou intercorrélation normée; les mesures de dissemblance telles que les distance de Hamming, Chebychev ou euclidienne et les mesures de similarité du type Jaccard, Yule. Bien que ces approches soient parfaitement adaptées à la reconnaissance de l'écriture imprimée, elles sont peu adaptées à la reconnaissance de l'écriture manuscrite du fait de sa grande variabilité qui impliquerait un grand nombre de représentants pour chaque classe. Seules des méthodes de comparaison par programmation

dynamique ont été utilisées dans le cas de la comparaison de signaux pour la reconnaissance en ligne de l'écriture manuscrite [Tapp91].

3.3.2. Approches statistiques

L'approche statistique est une approche qui repose essentiellement sur des fondements mathématiques (probabilité et statistique). L'objet des méthodes statistiques est de décrire les formes à partir d'un modèle probabiliste simple à utiliser et de regrouper les formes dans des classes. Ces méthodes s'appuient en général sur des hypothèses concernant la description statistique des familles d'objets analogues dans l'espace de représentation. De plus, cette approche bénéficie des méthodes d'apprentissage automatique qui s'appuient sur des bases théoriques connues telles que la théorie de la décision bayésienne. La forme est décrite à l'aide d'un vecteur de caractéristiques $x = x_1, x_2, \cdots, x_n$ où les x_i représentent les n mesures caractéristiques adéquates et significatives. Chaque forme x appartenant à la classe u_i est vue comme une observation générée aléatoirement par la distribution de probabilité de la classe u_i: $p(x/u_i)$. Ces méthodes de classification sont dites statistiques, car elles font intervenir des fonctions de décisions statistiques. Pour un ensemble d'apprentissages donné, on peut construire les frontières de décision de deux manières différentes. En effet, on distingue deux grandes familles de méthodes statistiques : les méthodes dites non paramétriques où l'on cherche à définir les frontières des classes dans l'espace de représentation, de façon à pouvoir classer le point inconnu par une série de tests simples ; les méthodes dites paramétriques ou bayésiennes, où l'on se donne un modèle de la distribution de chaque classe (en général gaussien), et où l'on cherche la classe à laquelle le point a la probabilité la plus grande d'appartenir. Ces méthodes sont dites de séparation linéaire. Deux autres méthodes non paramétriques utilisées sont celles de la décision par plus proche voisin ([Cove67], [Amin80]) : on attribue au point inconnu la classe de son plus proche voisin de l'ensemble d'apprentissages et la méthode des fenêtres de Parzen.

Les méthodes statistiques sont relativement simples et peu coûteuses en temps de calcul surtout pour l'approche paramétrique et moyennement sensible au bruit [Arri04]. Pour plus de détails sur les approches statistiques, nous suggérons au lecteur de se référer aux travaux ([Bela92], [Goss96], [Webb02], [Bish06]).

3.3.3. Approches Markoviennes

Les Modèles de Markov Cachés (HMM) ont été tout d'abord introduits principalement en reconnaissance vocale ([Juan 91], [Saon 95]). Ces travaux ont été ensuite adaptés à l'écriture manuscrite qui peut être modélisée par une séquence d'observations. Car tout comme la parole, l'écriture se prête bien à une modélisation stochastique à tous les niveaux de la chaîne de reconnaissance : morphologique, lexical, syntaxique [Belaid 97]. Les HMM

sont considérés comme les méthodes stochastiques les plus utilisées dans le domaine de la reconnaissance de l'écriture cursive. Ils utilisent le processus de Markov qui a une capacité de modéliser statistiquement la variabilité de l'écriture. Un modèle de Markov caché est un processus doublement stochastique, constitué d'un processus sous-jacent non observable, qui peut être déduit au travers d'un second processus stochastique qui produit des séquences d'observations. Ce processus est représenté comme une machine d'état pour modéliser l'évolution temporelle de l'écriture. La distribution de la probabilité associée à chaque état dans un HMM, modélise la variabilité de l'écriture [Bena99]. Dans les méthodes de reconnaissance d'écriture par HMM, on cherche à modéliser la séquence de caractères par des modèles de Markov cachés. La couche cachée du modèle est illustrée par la séquence d'étiquettes, et la couche observable correspond à une séquence d'observations que l'extraction de caractéristiques fournira à partir d'un balayage séquentiel de l'image ou signal temporel.

De nombreux travaux ont été proposés dans la littérature utilisant des HMM pour la reconnaissance de l'écriture ([Alpe97], [Cher98], [Bena99], [ElYa99], [Pech03], [Chev04], [Elha05], [Elha06], [Elha07], [Beno08a], [Beno08b], [Kess09], [Kess10]). Récemment, une application des HMM pour les caractères amazighes a été présentée dans ([Amro09], [Amro10]). Nous y reviendrons largement au chapitre 3.

3.3.4. Machines à vecteurs de support

Les machines à vecteurs de support (SVM) appelées aussi séparateurs à vaste marge ont été introduites par le mathématicien Vladimir Vapnik [Vapn95]. L'idée principale est que deux classes peuvent être linéairement séparées dans un espace de grande dimension (cf. Figure 1.5 ci-dessous). Si les points d'apprentissage sont séparables, il existe une infinité d'hyperplans séparateurs. L'objectif est donc de maximiser la marge entre les classes, qu'est la distance entre la surface de décision et les exemples d'apprentissage. Pour cela, il faut trouver le meilleur hyperplan séparateur maximisant cette marge. Il s'agit d'un problème d'optimisation quadratique sous contraintes, ce qui est assez complexe en termes d'algorithmiques si l'espace est de très grande dimension, mais qui donne des garanties sur le temps de convergence. On veut maximiser la marge sous la contrainte dont les exemples sont correctement appris. Comme on cherche à maximiser cette marge, on parlera de séparateurs à vaste marge.

Les SVM offrent des performances intéressantes pour la reconnaissance de caractères manuscrits ([LeCu98], [Liu02]), mais ils sont peu applicables à la reconnaissance de mots (sauf éventuellement avec une segmentation explicite des mots en lettres). En effet, ils travaillent avec des données en dimension fixe et ne permettent donc pas d'introduire la

variabilité de longueur des mots. De plus, ils ont l'inconvénient d'être assez lents en phase d'apprentissage comme en phase de reconnaissance.

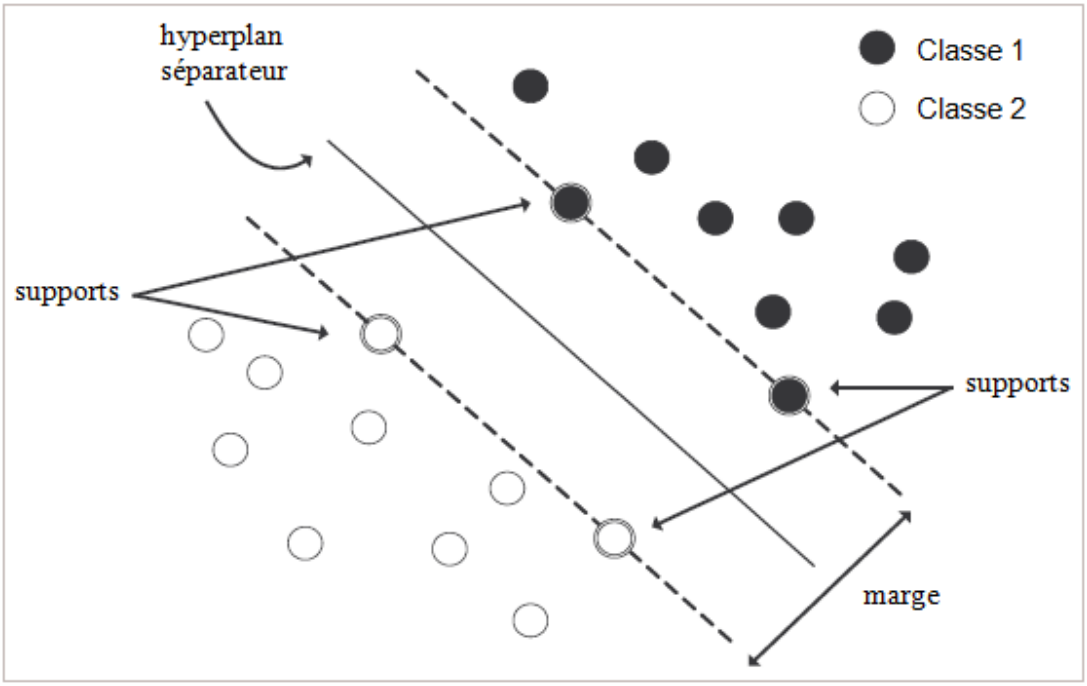


Figure 1.5 : Séparation souple par SVM : marge et hyperplan séparateur (figure extraite de [Pois05])

3.3.5. Réseaux de neurones

Les Réseaux de Neurones Artificiels (RNA), appelés aussi réseaux connexionnistes, sont des processus distribués parallèles qui permettent l'apprentissage et la reconnaissance. Ils ont connu un grand succès à partir des années 90, notamment grâce à la mise au point d'un algorithme d'apprentissage efficace et facile à mettre en œuvre : la rétropropagation du gradient ([LeCu98], [Bish06]). L'idée principale est qu'un neurone formel est capable de réaliser des calculs élémentaires comme la séparation d'un vecteur en deux classes, chaque classe étant déterminée par le poids du neurone. Le problème est alors de choisir les coefficients à affecter aux poids pour réaliser une séparation optimale. La multiplication des neurones permet de séparer plusieurs classes : il faut donc réaliser un choix sur la topologie du réseau en fonction du problème en adéquation avec le nombre de données en apprentissage.

Il existe de nombreux types de réseaux de neurones. Récemment, deux types sont plus utilisés dans les systèmes de reconnaissance de l'écriture, à savoir, les perceptrons multicouches (MLP) à propagation directe ([LeCu94], [Verm98], [Amin97]), et les cartes

organisatrices de Kohonen "Self Organizing Map" (SOM) [Koho01] qui sont des RNA récurrents. Ces derniers permettent de détecter automatiquement les prototypes de caractères dans un ensemble d'exemples d'apprentissage.

Les réseaux multicouches MLP se caractérisent par l'existence d'une ou de plusieurs couches cachées. Ces couches sont formées par des neurones cachés. Le rôle de ces neurones cachés est d'intervenir entre les entrées externes et les sorties du réseau, afin de connaître et mémoriser plus d'informations. Ces réseaux ont été récemment plus utilisés pour la reconnaissance de l'écriture imprimée et manuscrite ([Zhan00], [Wong02], [Basu10]).

Dans le chapitre suivant, nous détaillerons quelques topologies de réseaux de neurones et notamment les perceptrons multicouches que nous allons exploiter dans notre étude.

3.3.6. Approches structurelles et syntaxiques

Si les approches statistiques permettent de se placer dans un cadre mathématique solide et général, elles présentent cependant le défaut d'oublier la nature des mesures qui sont faites sur les formes et de les traiter de façon abstraite. En effet, les méthodes structurelles se basent sur la structure physique des caractères. Elles commencent soit par une squelettisation de la forme, soit par une détection des contours intérieurs et extérieurs soit par une détection des points singuliers, etc. Ensuite, elles cherchent à décomposer le caractère en primitives et à décrire leurs relations. Dans ces approches, les caractères sont généralement représentés par un ensemble de primitives reliées par des relations. Cette représentation permet de représenter l'organisation spatiale des différentes parties de la forme. Les primitives sont de type topologique comme un arc, un point, une boule, un coin, un segment, une courbe, une région, ou une forme simple et les relations peuvent être la position relative d'une primitive par rapport à une autre. Ces relations sont fréquemment représentées par des graphes, des chaînes ou des grammaires respectant des règles spécifiques de syntaxe [Micl84]. Dans ces approches on distingue plusieurs méthodes :

- *Comparaison des graphes* : cette méthode consiste à construire un graphe où les nœuds contiennent les primitives et les liens entre ces primitives. Ainsi la reconnaissance consiste à faire une mise en correspondance entre ce graphe et d'autres graphes représentant des caractères de référence construits lors la phase d'apprentissage. Cette méthode a été utilisée par ([Bair88], [Lebo91], [Darg94]) et elle donne des résultats acceptables;
- *Comparaison de chaines* : dans ce cas, les caractères sont représentés par des chaînes de primitives. La méthode consiste à mesurer la similitude entre les chaînes du caractère à reconnaitre et un modèle de référence par calcul de distance [Bela92]. Les

méthodes couramment utilisées pour comparer deux chaînes sont la distance d'édition et la comparaison dynamique [Alsa06] ;

– *Méthodes syntaxiques* : ces méthodes sont directement issues de la théorie des langages formels [Fu74] dont les modèles remontent aux années 50 et les développements de Chomsky [Chom56]. Elles se basent sur une grammaire formelle. L'idée générale est qu'une forme peut être décomposée en une séquence de primitives comme une phrase en une suite de mots. Chaque caractère est alors représenté par une phrase dans un langage où le vocabulaire est constitué de primitives. La reconnaissance consiste à déterminer si la phrase de la description du caractère peut être générée par la grammaire. Cette méthode a été utilisée par Ramesh [Rame89] et Baptista [Bapt88]. La limitation de ces méthodes réside dans l'absence d'algorithmes efficaces pour l'inférence grammaticale directe.

Les règles syntaxiques (grammaire) doivent être inférées à partir des exemples d'une base d'apprentissage. Contrairement aux méthodes statistiques, nécessitant un grand nombre d'échantillons dans l'espace de représentation, les méthodes structurelles et syntaxiques n'utilisent qu'un nombre réduit de prototypes pour représenter une classe. Cela permet d'une part de réduire le coût global de mise en correspondance entre les représentations inconnues et les représentations prototypes et d'autre part, d'inclure des prototypes représentant des formes très particulières afin de prendre en compte la variabilité de ces formes à l'intérieur d'une même classe [Heut94]. Malgré ces avantages, ces approches sont très sensibles aux problèmes de segmentation qui modifient la structure des formes, ainsi qu'au bruit.

Des résultats intéressants sont obtenus par l'application de ces approches sur les caractères bâton majuscules et les chiffres isolés [Suen92]. Les approches syntaxiques ont été utilisées aussi pour la reconnaissance de l'écriture arabe dans [Amin03] et pour la reconnaissance syntaxique des caractères arabes manuscrits [Bagh05].

Récemment, ces approches ont été utilisées aussi pour la reconnaissance d'expressions mathématiques [Rhee09], ainsi des règles en une seule dimension, soit verticales soit horizontales, sont combinées par programmation dynamique dans [Awal09]. Ils se retrouvent aussi dans des travaux d'analyse de documents structurés [Macé09], où l'analyse des tracés est guidée par le contexte, en suivant des règles formalisées par une grammaire.

Autrement, les règles grammaticales sont évoluées vers des formes floues et probabilistes pour donner naissance aux grammaires floues et stochastiques [Higu05] et notamment des outils de reconnaissance fondés sur les modèles de Markov cachés. L'étape de reconnaissance consiste alors à chercher la séquence d'états possédant la plus forte probabilité de générer l'image en entrée [Chan01]. Ces méthodes ont été utilisées aussi pour

la reconnaissance de caractères chinois. Certains travaux mettent à profit la nature hiérarchique de ces caractères pour développer des approches structurelles ([Zhen97], [Ota07]). Par exemple, dans [Ota07] les auteurs décrivent les caractères par une grammaire stochastique couplant des terminaux symboliques avec des relations de positionnement.

Des méthodes grammaticales sont aussi souvent utilisées dans les phases de post-traitement afin de corriger les erreurs de reconnaissance de manuscrits [Bouc99].

Dans cette étude, nous nous concentrons sur une méthode syntaxique en utilisant les automates finis pour reconnaître les caractères amazighes imprimés. Dans le chapitre suivant, nous reviendrons sur les approches syntaxiques en présentant les outils nécessaires et le principe d'utilisation des automates finis pour la reconnaissance de forme.

3.3.7. Combinaison des classifieurs

Nous avons décrit ci dessus plusieurs approches de classification utilisées en reconnaissance de l'écriture. Par ailleurs, ces approches peuvent être combinées pour améliorer les performances des classifieurs individuels. L'idée principale est d'exploiter au maximum la complémentarité des modélisations afin d'obtenir des décisions plus robustes, et compenser les faiblesses de chaque classifieur. La combinaison de classifieurs a été utilisée avec succès en reconnaissance de formes et en particulier en reconnaissance de caractères manuscrits ([Xu92], [Huan95], [Kitt98], [Rahm03], [Mena08], [Elha09]). Une étude de synthèse de méthodes hybrides et leur application à la reconnaissance de l'écriture cursive est présentée dans [Xu92]. Il existe trois architectures de combinaison de classifieurs [Zoua04] (voir Figure 1.6 ci-dessous) :

- La combinaison parallèle dans laquelle le caractère à reconnaître est présenté à plusieurs classifieurs indépendants dont les sorties sont combinées pour donner la décision finale ;
- La combinaison séquentielle où les classifieurs sont disposés en niveaux successifs de décision permettant de réduire progressivement le nombre de classes possibles ;
- Les approches hybrides consistent à combiner les architectures séquentielles et parallèles. Ce type d'approches est généralement dédié à un problème précis et qui est difficilement généralisable.

Pour plus d'informations sur les avantages et les inconvénients de chaque de type de ces combinaisons, nous suggérons au lecteur de se référer aux travaux ([Zoua04], [Elha07]).

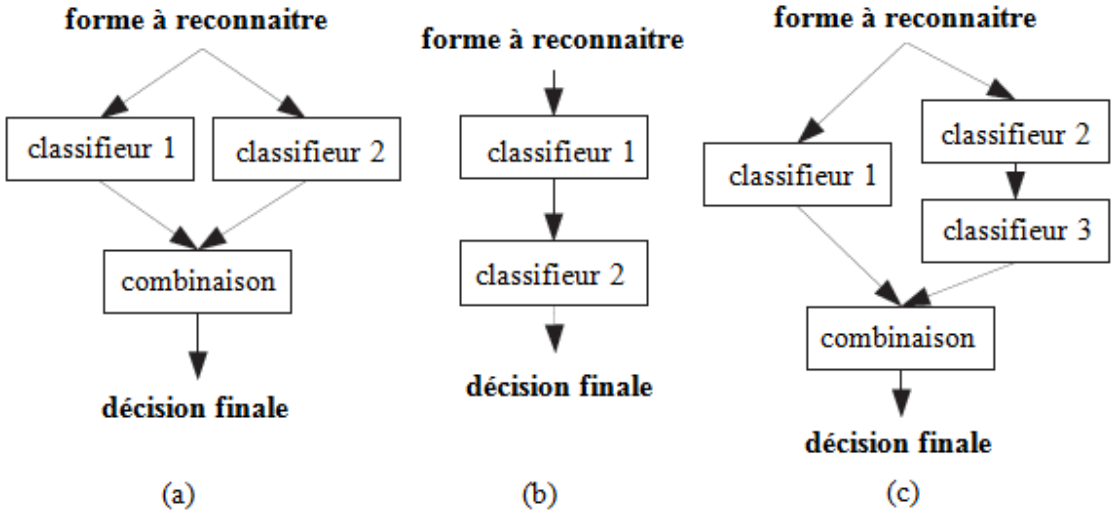


Figure 1.6 : Les 3 architectures de combinaison de classifieurs : (a) approche parallèle, (b) approche séquentielle, (c) approche hybride

4. Reconnaissance de mots

Les systèmes de reconnaissance hors ligne de mots isolés peuvent être classifiés selon deux grandes familles, les approches globales et les approches analytiques ([Kner97], [Vinc02]). La première approche consiste en une reconnaissance globale du mot qui considère le mot dans son ensemble sans chercher à localiser chacune des lettres qui le composent, elle est davantage adaptée aux lexiques de taille restreinte. La deuxième approche cherche à segmenter le mot en lettres puis de le reconnaitre via une identification de chacune de ses lettres, elle est beaucoup plus adaptée pour une application multi et omni scripteurs et dans le cadre de vocabulaires étendus. Dans les deux sous-sections suivantes, nous passerons en revue ces deux approches.

4.1. L'approche globale

L'approche globale, dite aussi holistique, a une vision générale du mot ; elle considère l'image du mot dans sa globalité comme une entité indivisible et comme étant la forme à reconnaître. Ainsi un ensemble de caractéristiques de haut niveau est utilisé afin de représenter de manière grossière l'apparence d'un mot [Kner97]. Une fois extraites, les caractéristiques sont regroupées dans un vecteur et soumises à un classifieur qui se prononcera alors en faveur d'une hypothèse de reconnaissance de mot. Ce type d'approche est plus largement utilisé dans le domaine hors ligne que dans le domaine en ligne, notamment pour la robustesse de l'approche face aux bruits et aux déformations de l'écriture. Elle présente l'avantage d'éviter la segmentation du mot en lettres, qu'est une

étape extrêmement délicate surtout pour l'écriture manuscrite cursive. Cependant, ces approches sont limitées à des applications avec un lexique de petit taille [Dehg01] telles que : le cas des montants numériques de chèques ([Kner97], [Lero97], [Nama05]), ou à des étapes de pré ou post-traitement visant soit à filtrer une partie du lexique [Anni94], soit à vérifier les solutions d'une approche analytique [Powa94].

4.2. L'approche analytique

L'approche analytique vise à reconnaître le mot en identifiant les lettres qui le composent. Une étape de segmentation est donc nécessaire afin de localiser les lettres du mot. Cette tâche est particulièrement délicate du fait de l'absence de segmentation idéale : les limites entre caractères sont parfois difficiles à déterminer même pour un être humain. Il existe deux types d'approches analytiques suivant que l'on effectue une segmentation explicite ou implicite. Les approches à segmentation implicite ([Vinc04], [Elha05], [Mori06], [Scha08]) considèrent tous les points du mot comme des points de segmentation potentiels à l'aide des techniques de fenêtre glissante. Des caractéristiques de bas niveau sont extraites de chaque fenêtre et sont soumises à un classifieur qui prend une décision globale de segmentation et de reconnaissance sur l'ensemble du mot. Les approches à segmentation explicite ([Kimu94], [ElYa99], [Koch04], [Lema07], [Mena08], [Kess10]) qui segmentent l'image du mot en des formes élémentaires appelées graphèmes après une analyse de celui-ci. Cette approche analytique est applicable dans le cas de grands vocabulaires, son inconvénient principal demeure la nécessité de l'étape de segmentation.

Certaines des approches actuelles, dites hybrides [Madd02], qui sont le fruit de la combinaison des approches globales et analytiques. Elles consistent soit à combiner les listes de mots fournis par différents classifieurs le plus souvent basés sur des primitives et des stratégies différentes, soit à se baser sur les résultats d'une méthode pour en guider une autre.

Pour le cas de l'écriture amazighe, l'approche analytique sera la meilleure stratégie a adoptée pour la reconnaissance de mots amazighes. En effet, l'écriture amazighe n'est pas cursive, ce qui facilite l'opération de segmentation du mot en lettres. En fait il est souhaitable de segmenter le mot amazighe en lettres et de le reconnaitre via une identification de chacune des lettres. Nous reviendrons sur les caractéristiques de l'écriture amazighe dans le chapitre 3.

5. Analyse et reconnaissance de documents

La reconnaissance de document consiste à convertir un document papier en document électronique, qui peut ensuite être traité par des ordinateurs, en se basant sur l'analyse et l'interprétation du document. Le domaine de l'analyse et la reconnaissance de document

favorise le développement de projet très ambitieux concernant la conversion du document papier en document électronique et son archivage. Le processus de reconnaissance de documents est relatif à toutes les questions autour du langage écrit et sa transformation numérique : reconnaissance de caractères, formatage du texte, structuration du contenu et accès à l'information pour des applications d'indexation. En outre, les documents existants présentent de grandes variabilités à différentes mesures : catégorie, type, structure, langue, fonte, taille de l'écriture, couleur, etc. Tous ces points doivent être pris en compte par le système de reconnaissance, ce qui nécessite des approches et des techniques de reconnaissance diverses bien adaptées au type de documents traités.

5.1. Catégories et structures de documents

Les documents structurés couvrent deux catégories : les documents imprimés et les documents manuscrits. Dans le cas des documents imprimés, ils diffèrent du point de vue du contenu et de l'organisation. En effet, trois structures sont possibles : structures linéaires, structures hiérarchiques simples et structures complexes suivant l'organisation spatiale des différentes zones [Hadj06]. Les premières sont représentées par exemple par les œuvres littéraires telles que les romans. Les deuxièmes sont représentées par les articles scientifiques ou les livres. Elles possèdent une organisation en chapitres, sections, articles et paragraphes que l'on peut représenter sous forme d'arbres. Les troisièmes sont représentées par les journaux, les magazines et les dépliants publicitaires. Elles possèdent une typographie riche et ne sont pas composées uniquement de texte, mais d'une combinaison, selon une disposition variable, de textes, de graphiques et d'images.

En cas des documents manuscrits, ils sont caractérisés par des lignes de longueur différente, plus ou moins variable. Les difficultés majeures sont l'imbrication des lignes, le chevauchement de composantes (composantes appartenant à plusieurs lignes de texte du fait de la présence de hampes et de jambages) et la fragmentation des caractères [Likf03].

Une autre catégorie de documents a été récemment étudiée ; elle concerne les documents anciens qui se caractérisent par des présentations et des écritures très variées et différentes de celles appliquées sur les autres types de documents et ils se caractérisent aussi par des variabilités de styles d'impression non utilisés de nos jours [Keta10].

Parallèlement, un document dispose de deux types de structures : structure physique et structure logique ([Bela92], [Hadj06]).

- La structure physique permet de représenter la structuration du document en vue de son impression. Il s'agit de la description du support physique et de son apparence visuelle et matérielle. Dans la forme papier, les entités physiques sont de plusieurs niveaux : le document lui-même dans sa globalité, l'ensemble de pages, la page, le cadre, le bloc, la

ligne, jusqu'à l'entité graphique élémentaire. La structure physique est alors souvent représentée comme une hiérarchie de ces entités physiques ;

- La structure logique est une représentation abstraite du document ne tenant compte ni du support ni de sa présentation. Elle reflète le point de vue de l'auteur et permet de représenter l'organisation du document en entités telles que chapitres, sections, paragraphes, etc. Il est à noter que le niveau de structuration utilisé est en fonction de l'application visée. La Figure 1.7 ci-dessous illustre un exemple de structure physique et de structure logique.

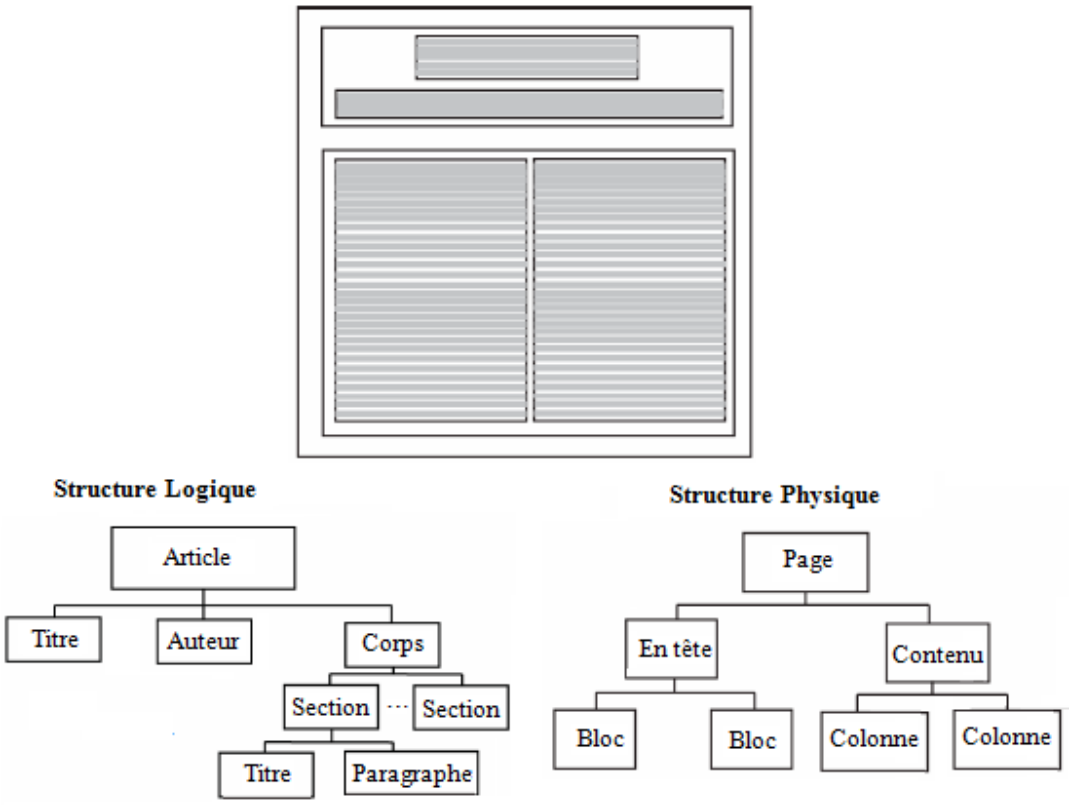


Figure 1.7 : Exemple de structures (figure extraite de [Hadj06])

5.2. Étapes du processus de reconnaissance de document

La reconnaissance de documents aura comme entrée une image numérisée épurée ou une image synthétique et elle est composée de deux étapes successives, l'une pour la reconnaissance de structures physiques (ou segmentation) et l'autre pour la reconnaissance de structures logiques [Keta10] (voir Figure 1.8 ci-dessous).

Le prétraitement consiste à éliminer les défauts liés à l'image numérisée afin de faciliter l'étape de reconnaissance. Il correspond à toutes les opérations de prétraitements (binarisation, seuillage, correction d'inclinaison, normalisation) que nous avons déjà citées précédemment pour un système de reconnaissance de caractères ou de mots.

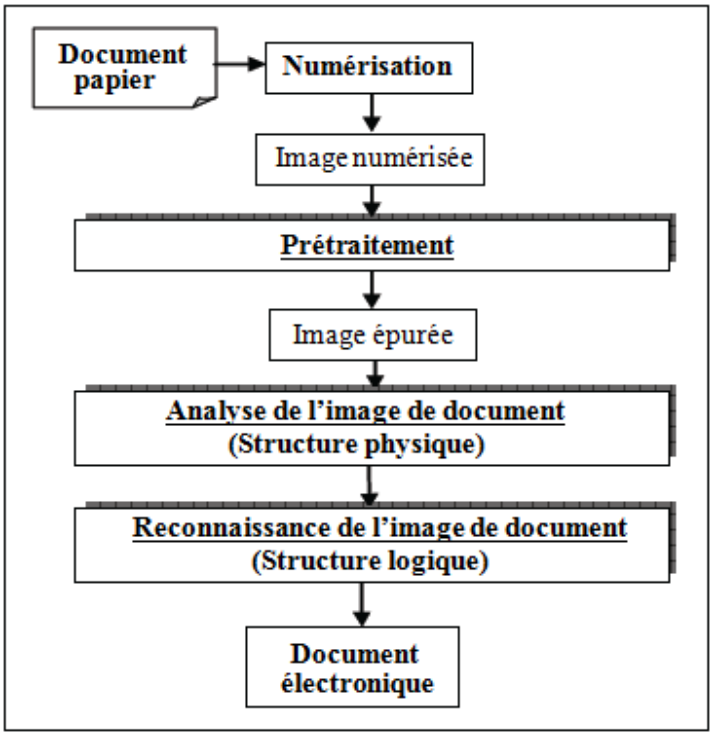


Figure 1.8 : Les étapes du processus de reconnaissance de document (figure extraite de [Keta10])

5.2.1. Reconnaissance de structures physiques

La reconnaissance de structures physiques comprend la détection et la classification des différentes zones de l'image. Il s'agit de segmenter l'image de document en composantes homogènes et de classifier chaque zone en texte, image, graphique, etc. La finesse de cette segmentation dépend du type de document et aussi de l'application visée. En effet, pour le cas du texte simple, la segmentation permet d'obtenir les lignes de texte, les mots et éventuellement les caractères. Pour le cas d'un document composé d'une simple colonne, la reconnaissance de structures physiques comprend la segmentation en lignes de texte, en mots, et la fusion des lignes pour former des blocs. Si par contre le document est multi colonnes, une étape supplémentaire est nécessaire à savoir la segmentation du texte en colonnes et aussi la détection des filets.

Dans la littérature, de nombreuses techniques classiques de segmentation de documents ont été utilisées. Elles peuvent être classées en trois approches ([Hadj06], [Keta10]) : approche descendante, approche ascendante et approche mixte.

L'approche descendante est guidée par un modèle. Elle commence par le niveau le plus élevé à savoir la page et descend d'un niveau à un autre jusqu'à arriver au niveau des composantes connexes ou au niveau pixel. Plusieurs algorithmes se basent sur l'approche descendante, telles que l'algorithme RLSA introduit par Wong [Wong82], l'algorithme X-Y introduit par Nagy utilisant les profils de projection [Nagy84] et la méthode se basant sur l'analyse du fond blanc de l'image introduite par Pavlidis et Zhou [Pavl91].

L'approche ascendante est guidée par les données. Elle fusionne les composantes connexes du document du plus bas niveau vers le niveau le plus élevé jusqu'à l'assemblage complet de la page. Plusieurs méthodes dans la littérature utilisent l'approche ascendante, telles que les composantes connexes ([Fish90], [Driv95]), le filtrage à base de fenêtres [Lebo92], la technique docstrum [OGor93] et les diagrammes de Voronoi [Kise98].

L'approche mixte ou encore hybride consiste à combiner le principe des deux approches descendantes et ascendantes pour la reconnaissance de structures physiques. Parmi les méthodes qui se basent sur l'approche mixte, on cite celles qui utilisent l'analyse syntaxique de documents ([Nagy92], [Kris93]) et d'autres comprennent une combinaison de méthodes ([Bair90], [Espo95], [Hadj01]).

D'autres méthodes ont été proposées dans la littérature pour les documents complexes et anciens, telles que les techniques qu'utilisent les grammaires [Coua03], des méthodes qui introduisent l'utilisateur dans le processus [Rame05] et d'autres méthodes de segmentation à base de texture [Jour06].

5.2.2. Reconnaissance de structures logiques

Une fois l'étape de la reconnaissance de structures physiques effectuée, l'étape qui s'ensuit est celle de la reconnaissance de structures logiques. Son objectif est de déterminer l'organisation logique des entités retrouvées au niveau de la reconnaissance de structures physiques en effectuant un étiquetage [Hadj06]. Elle repose essentiellement sur l'élaboration d'une correspondance entre les entités physiques, extraites lors de l'étape de l'extraction de structures physiques d'un document, et un ensemble d'entités logiques qui expriment généralement la sémantique du document. Cette mise en correspondance entre entités physiques et logiques est communément appelée étiquetage logique. Les étiquettes utilisées dans cette étape sont dépendantes de l'application visée et peuvent correspondre au titre, à auteur, au paragraphe et à l'article. La reconnaissance de structures logiques comprend aussi le recouvrement de l'ordre de lecture. Pour un document composé d'une

seule colonne, cet ordre est de haut en bas et de gauche à droite. Cependant, cet ordre est fortement dépendant de la langue dans laquelle le document est écrit.

Plusieurs techniques ont été utilisées pour l'extraction de structures logiques ([Duon05], [Hadj06]). La majorité des travaux se basent sur la structure physique pour aboutir à la structure logique. Il y a des méthodes d'extraction de structures logiques à base de règles ou de connaissances ([Ingo91], [Niyo95]) et d'autres qui reposent sur les modèles stochastiques [Brug97] et sur les modèles syntaxiques [Hu93]. Pour les documents à structure plus complexe des approches hybrides ([Héro01], [Soua02]) et des approches perceptuelles [Rang06], peuvent être appliquées. On peut se référer à la thèse de Karim Hadjar [Hadj06] et les références ([Jain98], [Mao03], [Duon05], [Mari08], [Keta10]) pour une revue des algorithmes des méthodes de la reconnaissance de structures physiques et aussi celles relatives aux structures logiques des documents.

6. Conclusion

Dans ce chapitre, après avoir introduit les applications et quelques aspects de la reconnaissance automatique de l'écriture, nous avons présenté un aperçu des différentes techniques employées dans le domaine. Nous avons dressé les différentes étapes d'un système de reconnaissance de caractères et de mot à savoir les prétraitements, l'extraction des caractéristiques et les méthodes de classification. Nous avons abordé chaque étape du système avec l'analyse et la catégorisation des différentes méthodes et techniques qui y sont développées et utilisées dans le cadre de la reconnaissance de l'écriture latine et arabe. Le choix d'une approche plutôt qu'une autre est lié à certaines contraintes comme les caractéristiques de l'écriture, la représentation d'entrée, la taille de la base d'apprentissage disponible et le temps de calcul requis. En fin, nous avons présenté la problématique de la reconnaissance automatique de mots et de documents en citant quelques techniques récemment employées pour le latin et l'arabe. Concernant la reconnaissance de mots, les performances varient beaucoup suivant le type de l'écriture et la taille du vocabulaire, et les performances sont acceptables pour les lexiques de taille raisonnable. Concernant les documents, on peut dire que la reconnaissance des documents imprimés surtout les latins est largement traitée. Cependant, la reconnaissance de certains types de document comme les documents arabes, les documents multilingues, les documents manuscrits anciens, est plutôt assez récent, les travaux sont en pleine expansion et plusieurs problèmes restent encore ouverts.

Avant d'aborder plus précisément les principes liés à nos contributions, nous nous focaliserons dans le chapitre suivant sur le formalisme des automates finis et les réseaux de neurones multicouches.

Chapitre 2 : Approches syntaxiques et Réseaux de neurones artificiels

1. Introduction

Parmi les approches utilisées dans le domaine de la reconnaissance de formes, on trouve les approches syntaxiques et les réseaux de neurones. Dans ce chapitre , nous décrivons, tout d'abord, les approches syntaxiques en rappelant les bases de la théorie des langages formels qui serviront pour la reconnaissance syntaxique de forme en s'intéressant aux formalismes des grammaires régulières et d'automates finis. Nous présentons, ensuite, l'application de ces formalismes dans la reconnaissance de l'écriture, ainsi que le principe d'apprentissage par l'inférence grammaticale, en particulier l'inférence régulière.

Dans la deuxième partie de ce chapitre, nous présentons les réseaux de neurones artificiels RNA en rappelant le modèle biologique qui inspira le modèle formel. Ensuite, nous donnons une description de l'architecture générale d'un réseau de neurones et de son mode de fonctionnement, et particulièrement pour un perceptron et un réseau multicouches. Nous terminons cette partie par une revue des exemples d'utilisation des perceptrons multicouches dans la reconnaissance automatique de l'écriture.

2. Les approches syntaxiques

L'idée générale des approches syntaxiques est qu'une forme peut être décomposée en une séquence de primitives comme une phrase peut l'être sous forme d'une suite de mots. Chaque forme est alors représentée par une phrase dans un langage dans lequel le vocabulaire est constitué de primitives qui représentent la forme. Avant de présenter le principe des approches syntaxiques en reconnaissance de formes, nous allons étudier rapidement les bases fondamentales de ces approches. Dans les sections suivantes, nous introduisons les notions de la théorie des langages nécessaires à la compréhension de ces approches. Nous nous sommes essentiellement basés sur l'ouvrage de L. Miclet [Micl84] que le lecteur pourra consulter pour plus de détails.

2.1. Langages et grammaires

2.1.1. Alphabet, mot

Un alphabet est un ensemble fini de symboles, noté *X*. Ses éléments appelés des lettres, seront notés : *a, b, c*.... La taille $|X|$ d'un alphabet *X* est le nombre de ses éléments.

Un mot ou une phrase sur un alphabet *X* est une suite de lettres de *X* représentée par une simple juxtaposition (concaténation) de ces éléments. On définit par convention le mot vide comme le mot de longueur nulle. On le note ici par ε.

L'ensemble de tous les mots que l'on peut écrire sur l'alphabet *X* se note : *X**. On note : $X+ = X*\text{-}\{\varepsilon\}$, l'ensemble des mots non vides.

Par exemple dans l'alphabet : $X = \{a,\ b,\ c\}$. Un mot sur l'alphabet *X* est $x = bcaab$. D'où, la longueur de x est 5 on écrit : |*x*|=5.

2.1.2. Langages

Un langage *L* sur l'alphabet *X* est un ensemble de mots de *X**. Par exemple :

$L1 = \{x \in X* / \textit{nombre de lettres de x est pair}\}$, $L2 = \{a,\ aa,\ aaa,\ ...an\ \ pour\ n > 0 \cdots\}$, $L3 = \emptyset$ et $L4 = \{\varepsilon\}$

On peut définir plusieurs opérations sur les langages. Les opérations booléennes sont l'union, l'intersection, la complémentation et la différence qui s'en déduit.

Si *A* et *B* deux langages construits sur un même alphabet *X*, alors :

$A \cup B = \{z \in X^* \mid z \in A \ \ ou \ \ z \in B\}$; $A \cap B = \{z \in X^* \mid z \in A \ et \ z \in B\}$

$A^c = X^* \setminus A = \{z \in X^* \mid z \notin A\}$; $A \setminus B = A \cap B^c = \{z \in X^* \mid z \in A \ et \ z \notin B\}$

2.1.2.1. Produit de deux langages

Soient L_1 et L_2 deux langages définis sur deux alphabets quelconques X_1 et X_2, alors le produit de L_1 et L_2 est défini sur l'alphabet $X_1 \cup X_2$ par : $L = \{xy \ / \ x \in L_1 \ \ et\ y \in L2\}$.

Par exemple, si $L_1 = \{a,\ abc\}$ et $L_2 = \{bcd,\ d\}$ Alors $L_1L_2 = \{abcd,\ ad,\ abcbcd,\ abcd\}$.

2.1.2.2. Fermeture d'un langage

La fermeture d'un langage *L*, notée *L** est définie de la façon suivante : $L^* = \bigcup_{n \geq 0} L^n \ \ avec \ \ L^0 = \{\lambda\},\ \lambda \ est\ le\ mot\ vide \ \ et \ \ L^n = LL^{n-1} \ \ pour\ n \geq 1$

Enfin, on note L^+ l'étoile propre de *L* : $L^+ = \bigcup_{n \geq 1} L^n = L.L* = L*.L$

Pour représenter commodément certains langages infinis par un nombre fini de symboles mathématiques, on dispose de grammaires formelles qui hiérarchisent l'ensemble des langages par la complexité de leur structure syntaxique.

Nous présentons par la suite la définition d'une grammaire ainsi que le procédé de génération d'une phrase par une grammaire.

2.1.3. Grammaire et langage généré

Une grammaire permet de représenter des langages infinis (à nombre infini de mots) par un nombre fini de symboles et de règles. Chomsky [Chom56] a défini une grammaire *G* comme un quadruplet $G = (X,\ V,\ S,\ P)$ où :

- ***X*** est l'alphabet où sont écrites les phrases du langage engendré, appelé alphabet terminal ;
- ***V*** est un alphabet auxiliaire ou non terminal ;
- ***S*** est un élément particulier de *V*, appelé axiome (symbole de départ) ;
- ***P*** est un ensemble de règles de production qui permettront le déroulement du processus de génération de mots.

Une règle de production peut être définie par le passage d'une chaîne α non vide formée de lettres de l'alphabet terminal et d'éléments de l'alphabet auxiliaire à une nouvelle chaîne β du même type pouvant être vide. Avec $\alpha \in (V \cup X)^* \times V \times (V \cup X)^*$ *et* $\beta \in (V \cup X)^*$. Cela signifie que α et β sont des chaînes formées en concaténant un nombre quelconque d'éléments de *V* ou *X* mais α contient au moins une lettre de l'alphabet et β peut être le mot vide.

2.1.3.1. Génération d'une phrase par une grammaire

Soit $\alpha \in (V \cup X)^+$ *et* $\beta \in (V \cup X)^*$. On dit que α se récrit en β selon la grammaire *G*, ou que la grammaire G dérive β de α en une étape (ce qui se note : $\alpha \Rightarrow \beta$, ou si besoin $\alpha \Rightarrow_G \beta$) si et seulement si on peut écrire α et β sous la forme :

- $\alpha = x\alpha' y$ (avec éventuellement $x = \varepsilon$ *ou* $y = \varepsilon$) ;
- $\beta = \mathrm{x}\,\beta'\,\mathrm{y}$;
- avec $\alpha' \to \beta' \in P$;

La grammaire *G* dérive en *k* étapes ß de α (ce qui se note : $\alpha \Rightarrow^k \beta$) si et seulement si $\exists k \geq 1$ et une suite $(\beta_0 \cdots \beta_k)$ de mots de V+ tels que :

- $\alpha = \beta_0$;
- $\beta = \beta_k$;
- $\beta_i \Rightarrow \beta_{i+1}$ pour $0 \leq i \leq$ k-1.

La grammaire *G* dérive β de α (ce qui se note : $\alpha \Rightarrow^* \beta$) s'il existe un entier *k* tel que : $\alpha \Rightarrow^k \beta$. Pour $k \geq 1$ la séquence $\alpha ... \Rightarrow \beta_i ... \Rightarrow \beta$ s'appelle une dérivation de β par α. De plus, une dérivation est dite gauche (respectivement droite) si, à chaque étape, on remplace le non-terminal le plus à gauche (respectivement le plus à droite).

Processus génératif associé à la grammaire G : On part de l'axiome *S* qui peut être considéré comme une règle de production particulière permettant d'initialiser le processus des réécritures successives puis on applique d'une manière successive des règles de production éligibles en procédant à des réécritures successives $a_1\ a_2 \cdots\cdots$ Si a_n n'est composé que de symboles du vocabulaire terminal (alphabet *X*) alors on ne peut plus faire de réécritures et a_n est donc un des mots générés par la grammaire *G*. On dit qu'un mot $v \in X$ est engendré par la grammaire *G* quand il peut se dériver en un nombre quelconque d'étapes à partir de l'axiome de *G*, ce qui s'écrit : $S \Rightarrow^* v$. L'ensemble des mots qui peuvent être générés par la grammaire forme le langage généré par *G* que nous noterons *L(G)*: $L(G) = \left\{ x \mid S \Rightarrow^* x \right\}$.

2.1.3.2. Exemple de grammaires et langages générés correspondants

Considérons la grammaire G_1 définit comme suit: $X = \{a, b\}$, $V = \{S, A\}$, avec S est l'axiome:

$$P : \begin{cases} S \rightarrow aS \\ S \rightarrow bA \\ bA \rightarrow bbA \\ A \rightarrow a \end{cases}$$

Exemples de réécritures : $S \Rightarrow aS \Rightarrow aaS \Rightarrow aabA \Rightarrow aaba$. On ne déduit que :
$aaba \in L(G1)$ et $L(G1) = \left\{ a^n b^m a \ avec\ n \geq 0 \ et\ m \geq 1 \right\}$.

2.1.3.3. Hiérarchie de Chomsky

Chomsky [Chom56] a construit à partir de la représentation par grammaire une hiérarchie de classes de représentation connue sous le terme de hiérarchie de Chomsky. Les classes de

la hiérarchie de Chomsky sont définies par des contraintes syntaxiques sur la forme des règles de production :

- Les grammaires de type 0 n'ont aucune contrainte sur les règles de production ;
- Les grammaires de type 1, ou grammaires sensibles au contexte, ont des règles de production qui ne contiennent qu'un seul non-terminal en partie gauche et une partie droite différente du mot vide ε ;
- Les grammaires de type 2, ou grammaires hors-contextes, ou encore grammaires algébriques, ont des règles de production dont les parties gauches sont formées d'un unique non terminal ;
- Les grammaires de type 3, ou grammaires régulières, ou encore grammaires rationnelles, ont des règles de production formées d'un non terminal en partie gauche et en partie droite soit d'une unique lettre, soit d'une lettre puis d'un non terminal.

La relation d'inclusion entre ces quatre types de grammaire est bien sur la suivante : $Type3 \subset Type2 \subset Type1 \subset Type0$.

Dans ce mémoire, nous nous intéressons uniquement aux grammaires régulières. Bien que cette classe soit la plus simple, donc la moins expressive, elle a été très étudiée dans le cadre de l'inférence grammaticale, avec de nombreux résultats positifs d'apprentissage, notamment dans le cadre de l'identification à la limite ([Gold67], [Higu97], [Higu05]).

2.2. Grammaires régulières

Une grammaire régulière est une grammaire dont les règles de production ne peuvent prendre que deux formes très simples: $A \to aB \ \ avec \ \ A \ et \ B \in V$ ou $A \to a \ \ avec \ \ a \in X$. Par exemple, la grammaire *Gr* ci-dessous est dite régulière :

$$Gr: \quad X = \{a,b\},\ V = \{S,A\}$$

$$P: \begin{cases} S \to aS \\ S \to bA \\ A \to bA \\ A \to a \end{cases}$$

Une phrase engendrée par Gr est : $S \Rightarrow aS \Rightarrow aaS \Rightarrow aabA \Rightarrow aaba$. On constate aisément que : $L(Gr) = \{a^n b^m a \ avec \ n \geq 0 \ et \ m \geq 1\}$.

Un langage engendré par une grammaire régulière est dit régulier. Les grammaires (resp. les langages) régulières sont aussi appelées grammaires (resp. langages) rationnelles, de Kleene, d'états finis. Les propriétés des langages réguliers sont plus faciles à décrire en

utilisant un formalisme équivalent aux grammaires régulières qu'est l'automate fini. Nous allons maintenant donner la définition d'un automate fini et nous verrons que toute grammaire régulière peut être représentée par un automate fini et vice versa.

2.3. Automates finis

Les automates à états finis constituent la représentation des langages réguliers la plus couramment utilisée en inférence grammaticale. Formellement, un automate à états fini est défini par le formalisme $A = (X,\ Q,\ \delta,\ q_0,\ Q')$ Avec X est un alphabet ; Q est un ensemble fini d'états ; $q_0 \in Q$ est état initial ; $Q' \subset Q$ est l'ensemble des états finals et $\delta : Q \times X \to Q$ est une application de l'ensemble $Q \times X$ dans Q, appelée fonction de transition.

Les automates sont des machines qui prennent en entrée une chaîne de symboles et qui effectuent un algorithme de reconnaissance de la chaîne. La chaîne est reconnue par l'automate si et seulement si celui-ci part d'un état initial, lit tous les symboles de la chaîne, puis s'arrête dans un état final. Le langage accepté par un automate à états fini est l'ensemble des chaînes dont les symboles font passer de l'état initial de l'automate jusqu'à un de ses états terminaux par une succession de transition utilisant tous ces symboles dans l'ordre. Un automate peut être représenté par son graphe de transitions d'états. Les cercles de cet automate sont appelés états, l'enchaînement des mots étant représenté à l'aide d'arcs appelés transitions. Un double cercle caractérise la fin d'une phrase, c'est-à-dire un état fini. A titre d'illustration, nous présentons sur la Figure 2.1 une représentation graphique de l'automate fini qui correspond à la grammaire régulière *Gr* vue précédemment.

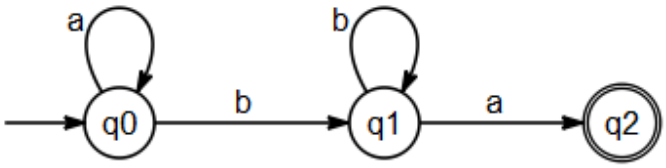


Figure 2.1 : Exemple d'automate fini

Pour cet exemple, l'alphabet est $X = \{a,\ b\}$. L'ensemble des états est $Q = \{q_0,\ q_1,\ q_2\}$; parmi ceux-ci, il y a un état initial q_0 et un état final q_2, donc $Q' = \{q_2\}$. Les transitions sont décrites dans l'ensemble $T = \{(q0,\ a,\ q0),\ (q0,\ b,\ q1),\ (q1,\ b,\ q1),\ (q1,\ a,\ q2)\}$.

2.3.1. Langage accepté par un automate fini

Un automate fini sert à définir un langage sur *X*. L'ensemble des phrases acceptées par un automate *A* définit le langage reconnu par cet automate, on le note par *L(A)*. Une phrase

$x = a_1 a_2 \cdots a_n$ est donc acceptée par l'automate A lorsque, partant de l'état initial, on peut trouver dans le graphe de A une suite d'arcs portant successivement les lettres $a_1, a_2, \cdots, a_n$ et menant un état final. Dans la suite, on ne considérera que des automates dont tous les états sont accessibles, c'est à dire : $\forall q \in Q, \exists x \in X^* \text{ tel que } \delta(q_0, x) = q$.

2.3.2. Automate non complètement spécifié

L'automate A est dit incomplètement spécifié, ou incomplet lorsque la fonction δ n'est pas définie sur tout $Q \times X$. On peut alors le compléter, sans changer le langage qu'il accepte, en ajoutant un état q_p non final. Cet état est souvent appelé état-puits, ou état-poubelle. Pour toute transition $\delta(q,a)$ non définie, on impose alors $\delta(q,a) = q_p$ et de plus : $\forall a \in X, \ \delta(q_p, a) = q_p \ et \ q_p \notin Q'$. Ce qui se représente graphiquement par la Figure 2.2 suivante:

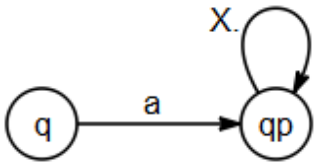


Figure 2.2 : Représentation graphique d'un automate incomplètement spécifié

2.3.3. Automate non déterministe

Dans le cas où l'application δ prend ses valeurs dans $\wp(Q)$, l'ensemble des parties de Q, c'est-à-dire ou cas où la fonction de transition est une application multivoque. Alors on dit que l'automate est non déterministe : à un couple $(a, q) \in X \times Q$ peut correspondre à un ensemble d'états tel que $\delta(q, a) = \delta(q_1, q_2)$ qui se représente graphiquement par la Figure 2.3 suivante :

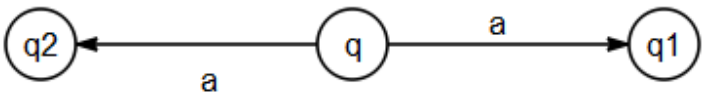


Figure 2.3 : Automate non déterministe

Un automate déterministe accepte un langage de façon analogue à un automate non déterministe. La Figure 2.4 ci-dessous présente un exemple d'automate fini non déterministe :

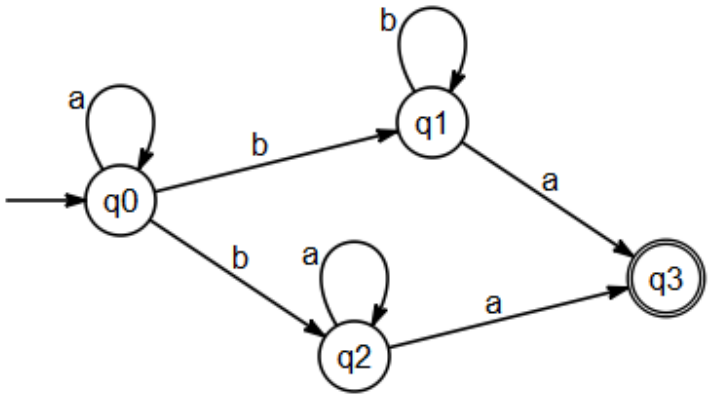


Figure 2.4 : Exemple d'un automate fini non déterministe

2.3.4. Grammaire régulière et automate fini correspondant

La correspondance entre grammaire régulière et automate fini est directe : l'état initial de l'automate correspond à l'axiome de la grammaire; les deux alphabets sont identiques; les règles de production correspondent à la fonction de transition de la façon représentée dans la Figure 2.5 suivante :

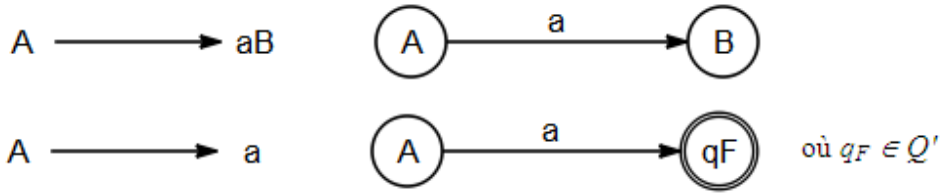


Figure 2.5 : Correspondance entre grammaire régulière et automate fini

La famille des langages sur un alphabet *X*, reconnus par un automate fini est la même que celle des langages (sur *X*) engendrés par une grammaire régulière. Par exemple la grammaire régulière suivante :

$G : \begin{cases} S \to aS \mid bA \\ A \to bA \mid a \end{cases}$ Correspond à l'automate fini représenté à la Figure 2.4 ci-dessous :

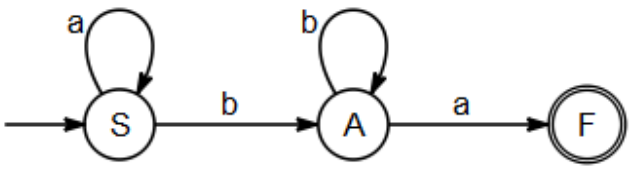


Figure 2.6 : Automate fini correspond à la grammaire régulière G précédente

2.4. Décision en structures syntaxiques régulières

2.4.1. Préliminaires

Les méthodes syntaxiques reposent sur une hypothèse forte : les formes qui appartiennent à la même classe présentent une structure commune qui peut être représentée par un modèle grammatical. En effet les formes (représentées par des phrases) qui appartiennent à cette classe peuvent être engendrées par le modèle grammatical indiqué. Afin d'opérer une décision de reconnaissance, nous devons pouvoir décider si une forme donnée appartient à une classe donnée. Autrement dit, dans la représentation grammaticale, on doit vérifier si une phrase donnée peut être engendrée par une grammaire donnée. Ceci est le problème de l'analyse syntaxique qui a été étudié à fond dans le domaine de traitement automatique de langage naturel ([Roch97]).

Idéalement, si chaque classe est définie par une grammaire connue et si une forme inconnue arrive et que l'on veut savoir à quelle classe l'affecter, alors on regarde quelle grammaire peut générer cette forme pour décider à quelle classe elle sera affectée.

Formalisons un système de reconnaissance syntaxique de la façon suivante : Supposons qu'on a n classes $C_1, C_2..., C_n$ de formes différentes représentées par des grammaires $G_1, G_2..., G_n$. Soit un mot x inconnu à classer dans une des classes. On regarde pour chaque grammaire G_i si x peut être engendré par G_i, autrement dit est-ce que $x \in L(Gi)$. Il s'agit là d'un problème d'analyse syntaxique :

- Si on a aucune solution alors x n'appartient à aucune des classes définies ;
- Si on a une réponse alors x est mis dans la classe correspondante ;
- Si on a plus d'une réponse positive alors il y a indécision (recouvrement de classes). Dans ce cas, une généralisation de la distance d'édition entre phrases peut être faite en traduisant une adéquation non binaire, da phrase à la grammaire [Micl84].

Regardons maintenant les problèmes posés par cette approche :

Si les classes sont exhaustives et que x n'est généré par aucune grammaire, on doit alors chercher à estimer le degré de ressemblance du mot avec chaque langage $L(Gi)$ afin de lui affecter une classe. Si x peut être généré par plusieurs grammaires il est alors probable que les langages générés par les grammaires correspondantes ont trop d'intersections, c'est-à-dire que chacune de ces grammaires n'est pas assez restrictive. Il est alors peut-être possible de revoir le processus qui a généré ces grammaires.

Dans la suite, nous abordons ce problème de reconnaissance dans le cas particulier des grammaires régulières, donc des automates finis appliqués à la reconnaissance automatique de caractères.

2.4.2. Analyse syntaxique et grammaires régulières

Le problème de l'analyse syntaxique, c'est-à-dire de décider si une phrase donnée appartient ou non à un langage défini par une grammaire donnée *G*, est particulièrement simple à résoudre dans le cas où *G* est régulière. Nous allons traiter ce problème avec l'automate fini correspondant à la grammaire régulière. Le problème est défini comme suit: Soit $x \in X^*$ et soit un automate $A = (X, Q, \delta, q_0, Q')$ complètement spécifié. Comment décider si *x* est un mot de *L(A) ?*

Notons $x = a_1 a_2 \cdots a_n$ avec $a_i \in X$. L'algorithme général d'analyse syntaxique est le suivant [Micl84] :

```
Début
        Pour i=0 A n-1 Faire
            q_{i+1}=δ(q_i, a_{i+1})
        Fin pour
        SI
             q_n ∈ Q'
        Alors
            x ∈ L(A)
        Sinon
            x ∉ L(A)
        Fin si
Fin
```

Cet algorithme revient simplement à tester si, dans l'automate fini, il existe un chemin allant de l'état initial à un état final, et portant comme étiquettes sur ses arcs la suite exacte des lettres de la phrase à reconnaître. Dans le cas où on trouve qu'une donnée *x* ne fait pas partie des mots générés par l'automate *A*, on essaye de voir si en modifiant le mot *x* par suppression, ou ajouts ou substitutions de lettres, on peut arriver à trouver un mot qui pourrait être généré par *A*. La complexité plus ou moins importante des modifications donnera une distance de *x* à *L(A).* Le calcul de la distance d'édition entre un mot et un langage régulier est effectué avec un algorithme qui généralise l'algorithme standard du calcul de la distance d'édition entre deux mots. L'algorithme de Wagner [Wagn74] consiste à calculer cette distance en s'appuyant sur les transformations nécessaires pour passer de *x* à un mot de *L(A)*. Cette distance s'exprime donc en un nombre minimal d'opérations à faire

subir à un mot *x* (insertion, substitution, élimination de lettre) pour arriver à un mot généré par l'automate et de ce fait pour atteindre un état final de l'automate.

2.5. Apprentissage : inférence grammaticale

2.5.1. Préliminaires

Nous nous intéressons ici à l'apprentissage de modèles structurels pouvant être décrits par le biais d'une grammaire formelle. Ce problème relève de l'inférence grammaticale dans le sens classique du terme [Gold67]. Elle consiste en l'apprentissage d'une grammaire représentant un langage à partir d'un ensemble d'exemples. Cet ensemble est formé au moins d'un échantillon positif, c'est-à-dire un sous-ensemble fini d'un langage. Nous pouvons également disposer d'un échantillon négatif, c'est-à-dire un ensemble fini de chaînes n'appartenant pas au langage.

L'inférence grammaticale est un domaine de recherche né dans les années 1960, motivé par l'analyse et l'acquisition de la langue naturelle. Les travaux fondateurs sont apparus dans l'article d'E.M. Gold [Gold67] après une formalisation de la notion de langage par N.Chomsky [Chom56]. Outre son intérêt théorique, elle offre un ensemble d'applications potentielles, telles que le traitement automatique de la langue naturelle [Roch97], l'annotation automatique des séquences de type biologique ([Fred03], [Cost04], [Lero05], [Kerb08]) et aussi dans les domaines de la reconnaissance syntaxiques de formes (reconnaissance de la parole ([Dupo96], [Mohr07]), de l'écriture ([Ron98], [Amin03], [Ota07], [Abdu09]) et de documents ([Ahon94], [Soua02])).

Les méthodes d'inférence grammaticale développées dans le domaine des langages formels sont souvent inadéquates pour apprendre des structures de formes complexes, mais elles peuvent être utiles pour des formes très complexes, comme les figures géométriques, ou des parties de formes dont la structure est plus simple que celle de la forme globale. Il existe plusieurs algorithmes pour la solution de l'inférence grammaticale. Dans ce mémoire, nous nous limiterons à l'inférence régulière ([Dupo94], [Dupo98]), c'est-à-dire l'apprentissage d'une grammaire représentant un langage supposé régulier. Nous donnons dans la section suivante le principe de l'algorithme d'inférence régulière. Pour une vue globale du domaine de l'inférence grammaticale, le lecteur pourra se référer aux références ([Micl84], [Higu97], [Saka97], [Dupo98], [Cost00], [Higu05]).

2.5.2. L'inférence régulière

Lorsque l'apprentissage concerne plus spécifiquement des grammaires régulières pouvant être représentées sous la forme d'un automate fini, on parle alors d'inférence grammaticale régulière [Dupo98]. Le problème posé consiste à faire l'apprentissage d'un automate fini à partir d'un multi-ensemble d'exemples. On se place ici dans le cadre de l'apprentissage

supervisé où les exemples peuvent être étiquetés positivement, c'est-à-dire appartenant au langage à apprendre et éventuellement négativement ou bien ne faisant pas partie de ce langage (on parle alors de contre-exemples). D'un point de vue d'apprentissage automatique, l'inférence consiste en la recherche d'un modèle de généralisation à partir d'un échantillon d'apprentissage positif, voire d'un échantillon négatif. On considère que l'échantillon positif a été généré par un automate particulier appelé automate cible. L'objectif de l'inférence est alors d'identifier cet automate qui a généré les données d'apprentissage.

En reconnaissance syntaxique de formes, nous allons supposer que les formes que l'on gère sont en fait des mots d'un langage particulier et nous supposons que les classes sont en fait définies par des échantillons de mots. Chaque classe est donc définie par un petit paquet de mots. C'est à partir de ce paquet de mots qu'il faut trouver une technique de discrimination. Autrement dit, il s'agit de découvrir quelle est la grammaire régulière (ou l'automate fini) qui génère la classe des échantillons. Notons *I* l'ensemble d'échantillons définissant une classe, alors il faut trouver l'automate ou la grammaire qui génère ces mots. Ce problème d'inférence admet une infinité de solutions : l'une d'entre elles est simplement la grammaire universelle *Gu* dont le langage accepté est *X**. Si *X* est l'alphabet sur lequel l'ensemble *I* est écrit, alors l'automate fini correspondant à cette grammaire est présenté à la Figure 2.7 ci-dessous :

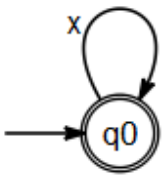


Figure 2.7 : Automate universel sur l'alphabet X= {a, b}

Remarquons que l'on peut ajouter à *X* autant de lettres que l'on veut, tout en conservant la grammaire universelle ainsi engendrée dans les solutions au problème d'inférence.

Une autre solution, nommée grammaire canonique maximale notée *GCM(I)* qui ne génère que les échantillons de *I*. Donnons sa construction sur un exemple :

Soit $I = \{a,\ bc,\ abc,\ ababc\}$. La grammaire canonique maximale *GCM(I)* serait représentée par l'automate de la Figure 2.8. Cet automate est appelé automate canonique maximale de *I ACM(I).*

De la même façon, on peut ajouter à GCM(I) des transitions quelconques portant des lettres de *X*, des lettres extérieures à *X* sans sortir de l'ensemble des solutions. Il existe donc une infinité de solutions au niveau des grammaires. Pour éviter cette infinité de solutions sans

intérêt, une condition a été imposée sur l'ensemble des échantillons *I*, pour qu'il soit structurellement complet par rapport à la grammaire proposée.

Nous donnons la définition d'un échantillon structurellement complet, correspondant à celle introduite dans [Dupo94], et qui représente la structure que doit avoir un échantillon caractéristique pour permettre l'inférence d'un automate déterministe.

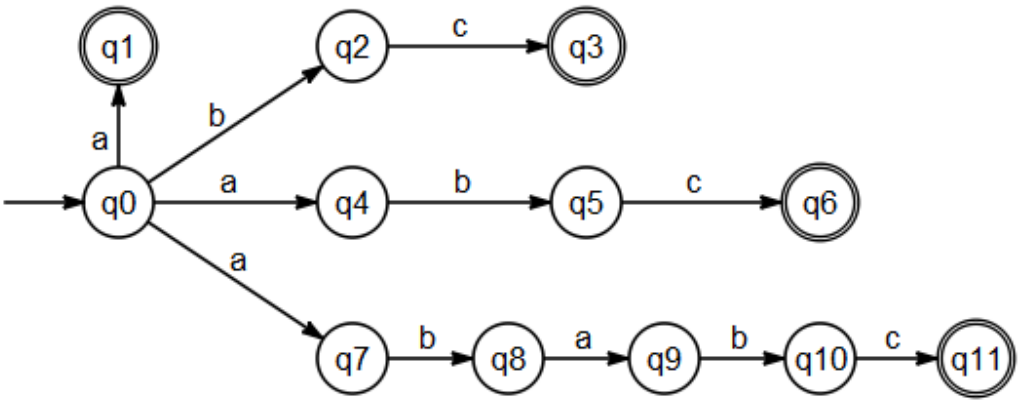


Figure 2.8 : L'automate ACM(I)

Un ensemble d'échantillons *I* est complet par rapport à la grammaire $G = (X, V, P, S)$ si et seulement si :

1. $I \subset L(G)$;
2. L'alphabet sur lequel est écrit l'ensemble *I* est égal à *X* ;
3. Toutes les règles de *P* sont utilisées au moins une fois dans une génération des mots de *I*.

Moyennant cette condition, le problème de l'inférence grammaticale devient alors :
Pour un échantillon *I*, il faut trouver une grammaire (ou toutes les grammaires) *G* telle que,

1. $I \subset L(G)$;
2. I est complet par rapport à *G*.

Cette restriction naturelle permettra de trouver des algorithmes d'énumération des solutions, en particulier dans le cas des grammaires régulières. Cependant, leur nombre est toujours très grand par rapport à la taille de l'échantillon.

Plusieurs algorithmes ont été proposés dans le cadre de l'inférence d'automates. L'approche la plus utilisée afin de réaliser l'inférence d'automates est celle dite par fusion d'états à partir d'un automate canonique maximal (MCA) [Dupo98]. Le MCA des échantillons positifs représente exactement le langage des exemples donnés. L'idée est de

procéder à l'apprentissage par fusion d'états et de généraliser le langage (augmenter le nombre des échantillons acceptés) en appliquant des fusions entre les états. En effet, celles-ci génèrent de nouveaux automates introduisant de nouveaux chemins et donc de nouveaux échantillons dans le langage, à partir du MCA. Les différents algorithmes diffèrent ensuite par leur stratégie de choix d'états à fusionner et leur critère d'arrêt [Dupo98]. Pour une revue des algorithmes de l'inférence régulière, le lecteur pourra se référer aux références ([Dupo98], [Cost00], [Fred03], [Higu05]).

Récemment, d'autres approches utilisent des automates à états finis pondérés pour la reconnaissance syntaxique de formes. Y. LeCun et al. proposent d'étendre ce formalisme à la reconnaissance de l'écriture [LeCu98]. Cette idée est également reprise par X. Dupré [Dupr03], et ensuite par F. Menasri [Mena08] sur une tâche plus simple de reconnaissance de noms de famille français écrits en lettres capitales.

3. Les réseaux de neurones artificiels

Depuis une dizaine d'années, l'utilisation des réseaux de neurones artificiels (RNA) s'est développée dans de nombreuses disciplines (sciences économiques, écologie et environnement, biologie et médecine…). Ils sont notamment appliqués pour résoudre des problèmes de classification, de prédiction, de catégorisation, d'optimisation et de reconnaissance des formes [Drew00].

Les réseaux de neurones artificiels peuvent être considérés comme des processeurs distribués, massivement parallèles qui possèdent la propriété de mémoriser des expériences passées en vue de les utiliser dans des procédés non connus à l'avance [Lipp87]. Ils ressemblent au cerveau par deux aspects :

- La connaissance est acquise par le réseau au moyen d'un processus d'apprentissage ;
- Les intensités des connexions entre les neurones connues par les "poids synaptiques" sont utilisées pour mémoriser la connaissance.

La procédure utilisée pour effectuer le processus d'apprentissage est appelée algorithme d'apprentissage. Cet algorithme a pour rôle de modifier les poids synaptiques du réseau afin d'avoir les résultats désirés.

Dans la suite, nous présentons les principales propriétés des réseaux de neurones. L'accent est surtout mis sur les perceptrons multicouches. Une présentation plus générale des réseaux de neurones et de leurs applications à la reconnaissance de texte peut être trouvée dans ([Goss96], [Drew00], [Duda00]).

3.1. Du neurone biologique au neurone formel

Le neurone biologique est une cellule vivante, qui est composée de quatre parties distinctes (cf. Figure 2.9 ci-dessous) :

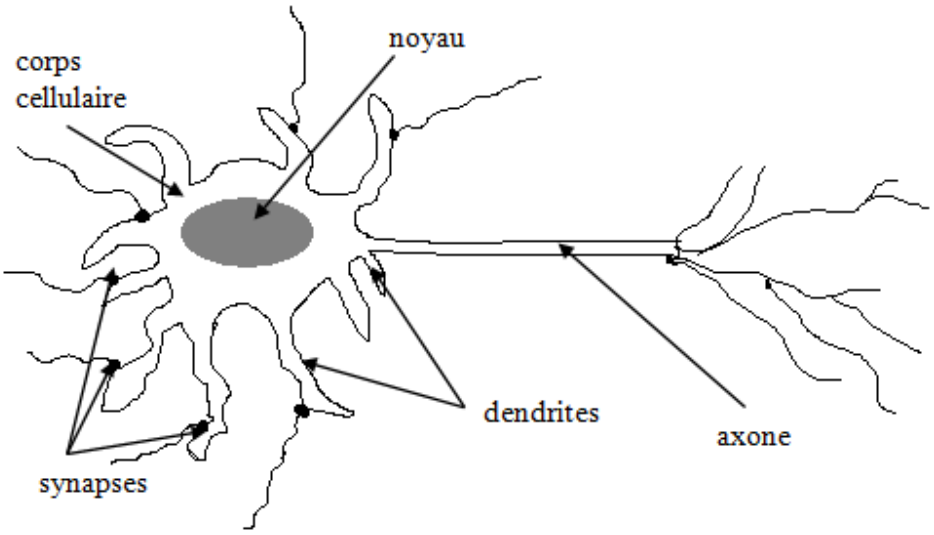


Figure 2.9 : Schéma d'un neurone biologique

- un corps cellulaire, qui contient le noyau de la cellule nerveuse; joue le rôle d'un sommateur à seuil. Il effectue une sommation des influx nerveux par ses dendrites; si la somme est supérieure à un seuil donné, le neurone répond par un flux nerveux ou potentiel d'action qui se propage le long de son axone; si la somme est inférieure au seuil, il reste inactif ;
- des dendrites, ramifications tubulaires courtes formant une espèce d'arborescence autour du corps cellulaire; ce sont les entrées principales du neurone, qui captent l'information venant d'autres neurones ;
- un axone, longue fibre nerveuse qui se ramifie à son extrémité ; c'est la sortie du neurone et le support de l'information vers les autres neurones ;
- une synapse, qu'est un élément de jonction. Les synapses permettent aux cellules de communiquer entre elles, de plus il joue un rôle dans la modulation des signaux qui transitent le système nerveux.

Les réseaux de neurones formels sont à l'origine une tentative de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'œuvre de MM. McCulloch et Pitts [Lipp87] qui s'inspirent de leurs travaux sur les neurones

biologiques. Ils présentent un modèle assez simple pour les neurones et explorent les possibilités de ce modèle (cf. Figure 2.10 ci-dessous).

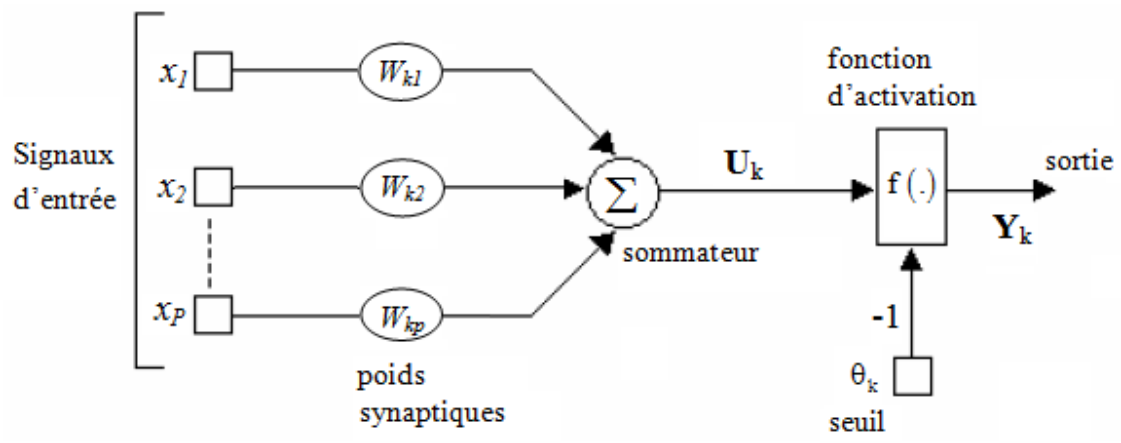


Figure 2.10 : Le modèle de neurone formel

Le neurone formel rappelle beaucoup le neurone biologique. Pour cette raison, le vocabulaire utilisé dans la littérature pour décrire un neurone formel est largement emprunté à la biologie. Nous constatons trois éléments de base (cf. Figure 2.10 ci-dessus) :

- un ensemble de synapses ou connexions, chaque connexion est caractérisée par un poids (intensité) W_{ij} ;
- un sommateur réalisant une sommation des signaux d'entrée pondérés par les poids synaptiques relatifs. L'opération constitue une combinaison linéaire des signaux d'entrée : $U_k = \sum_{j=1}^{p} W_{kj} x_j$;
- une fonction d'activation limitant l'amplitude de la sortie du neurone, de cette façon l'amplitude de la sortie sera normalisée. On peut citer plusieurs types de base des fonctions d'activations telles que la fonction seuil et la fonction sigmoïde définie par : $f(x) = \frac{1}{1+e^{-ax}}$ où a représente le paramètre de pente de la fonction.

Il y a bien sûr beaucoup de fonctions d'activation possibles telles que la sigmoïde, la tangente hyperbolique et la fonction de Heaviside. La fonction sigmoïde est de loin la plus utilisée comme fonction d'activation dans la construction des réseaux de neurones artificiels.

Le modèle du neurone possède aussi un seuil extérieur θ_k dont le rôle est d'agir sur la fonction d'activation. Du point de vue mathématique, un neurone *k* peut être décrit de la façon suivante : $U_k = \sum_{j=1}^{p} W_{kj} x_j$ et $Y_k = f\left(U_k - \theta_k\right)$ où Y_k est la sortie du neurone *k*.

3.2. Les réseaux de neurones multicouches

3.2.1. Le perceptron

Présenté originellement par Rosenblatt, en 1958 [Rose58], le perceptron est la plus simple forme d'un réseau de neurones. Il permet de classifier correctement des objets appartenant à deux classes linéairement séparables (cf. Figure 2.11 ci-dessous). Il consiste en un seul neurone qui possède un seuil ainsi qu'un vecteur de poids synaptiques ajustable.

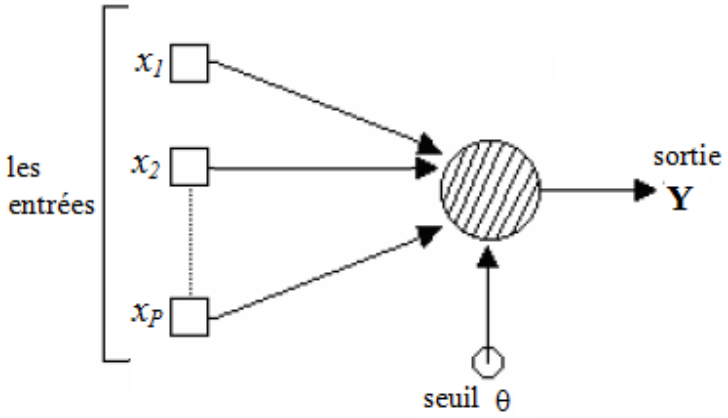


Figure 2.11 : Le perceptron

Ce type de perceptron est limité à effectuer la classification dans un problème à deux classes seulement. La reconnaissance de plusieurs classes est cependant rendue possible par la mise en parallèle de plusieurs perceptrons. Le perceptron ainsi obtenu comporte un neurone par classe, chacun de ceux-ci réalisant une fonction discriminante linéaire de la classe à laquelle il est associé. Ainsi, les frontières de décision pour la classification offertes par le perceptron sont linéaires. Néanmoins, de nombreux problèmes de classification nécessitent une partition non linéaire de l'espace d'entrée. Ces problèmes peuvent être résolus à l'aide des perceptrons multicouches, qui consistent à mettre en cascade deux couches ou plus de perceptrons.

3.2.2. Les perceptrons multicouches

Les réseaux de neurones multicouches, désignés par les perceptrons multicouches ("multi-layer perceptrons" MLP), sont des réseaux composés de couches successives. La première

couche d'un MLP est appelée la couche d'entrée, la dernière couche est appelée la couche de sortie, et les autres couches sont appelées les couches cachées (cf. Figure 2.12 ci-dessous). Ces couches de neurones cachées permettent au réseau d'apprendre des tâches très complexes. Le vecteur d'entrée se propage à travers le réseau dans le sens entrée-sortie d'une couche à l'autre. En effet, les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée (c'est en fait l'entrée du réseau). Ils calculent alors leur sorties qui sont transmises aux neurones de la deuxième couche, etc. Les sorties des neurones de la dernière couche forment la sortie du réseau.

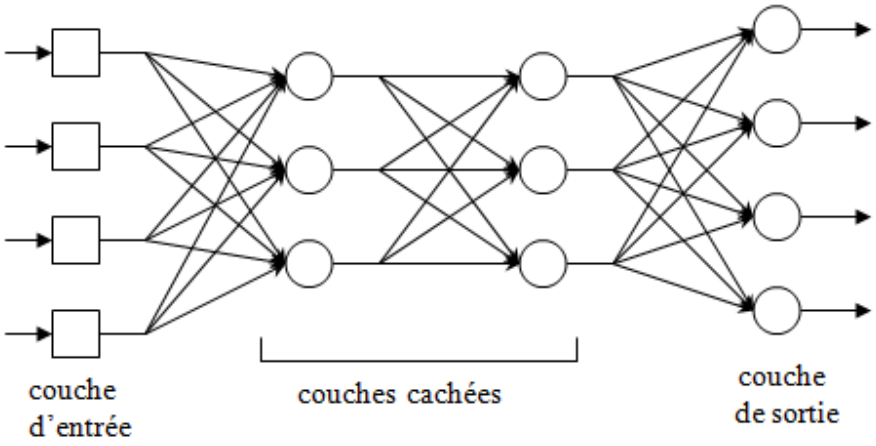


Figure 2.12 : Perceptron multicouche fortement connecté aux couches cachées

Les perceptrons multicouches ont été utilisés pour résoudre des problèmes difficiles de classification par apprentissage supervisé à partir d'un algorithme connu sous le nom d'algorithme de rétropropagation du gradient qui se base sur la règle d'apprentissage par correction d'erreur.

3.3. Apprentissage et classification

La propriété intéressante des réseaux de neurones est la faculté que possède un réseau à apprendre à partir de son environnement et à améliorer ses performances par l'apprentissage ([Lipp87], [Hayk99]). L'amélioration s'obtient avec le temps à partir de mesures réelles effectuées. Un réseau de neurones apprend de son environnement à partir de processus itératifs en ajustant les poids synaptiques et les seuils. Théoriquement, le réseau s'améliore après chaque itération d'apprentissage.

L'utilisation d'un réseau de neurones artificiels se fait en deux temps. Tout d'abord une phase d'apprentissage qui est chargée d'établir des valeurs pour chacune des connexions du réseau, puis une phase d'utilisation proprement dite, où l'on présente au réseau une entrée et où il nous indique en retour sa sortie calculée.

3.3.1. Apprentissage

L'apprentissage est l'une des propriétés intéressantes des réseaux de neurones. Par définition l'apprentissage est un processus de simulation continu à partir duquel les paramètres libres du réseau de neurones s'adaptent à l'environnement de ce réseau.

On se limite ici à l'apprentissage supervisé qui utilise un superviseur ou enseignant extérieur. Conceptuellement, le superviseur est l'élément qui connaît l'environnement en le représentant par un ensemble d'exemples entrée-sortie. Cet environnement est inconnu pour le réseau de neurones en question. La forme de l'apprentissage supervisé est de type apprentissage par correction d'erreur décrit au-dessous.

Entraîner un réseau, consiste à présenter des entrées et à comparer la sortie obtenue avec la sortie désirée. La différence est appelée l'erreur. Les différentes règles d'apprentissage définissent comment ajuster les poids synaptiques pour réduire cette erreur. On dit que l'apprentissage a convergé lorsque l'erreur a atteint une petite valeur acceptable. Il faut donc corriger les poids à chaque nœud pour réduire l'erreur de la fonction *f* à chaque nœud :

- On le fait petit à petit en rentrant de nouvelles données ;
- On le fait dans un souci de convergence (stabilité du réseau).

On considère une paire de signaux x_j et v_k reliés par un poids synaptique w_{kj}. Le signal x_j représente la sortie du neurone *j* et le signal v_k représente l'activité du neurone *k*. (cf. Figure 2.13 ci-dessous).

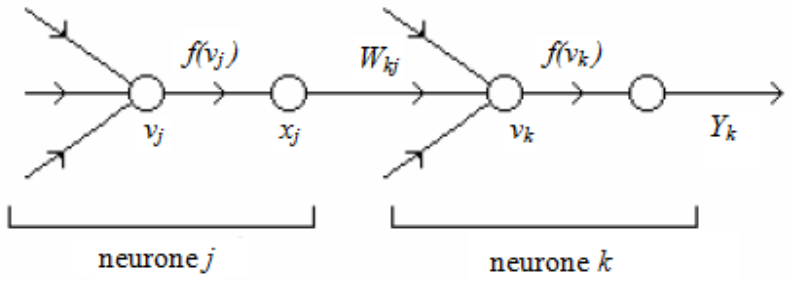


Figure 2.13 : Sorties, activités et poids synaptiques dans un réseau de neurones (figure extraite de [Elha05])

On suppose que $w_{kj}(n)$ est le poids synaptique w_{kj} à l'instant *n*. On suppose aussi qu'un ajustement $\Delta w_{kj}(n)$ est réalisé sur le poids à l'instant *n*, alors on obtient une nouvelle valeur du poids notée $w_{kj}(n+1)$ à l'instant *n+1*, par la formule suivante :

$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$.

Cette règle est utilisée dans les algorithmes d'apprentissage. Il existe plusieurs solutions dépendant de la formulation de l'ajustement $\Delta w_{kj}(n)$ au poids synaptique $w_{kj}(n)$. Ces solutions dépendent aussi de la manière dont le réseau s'adapte à son environnement.

3.3.2. Apprentissage par correction d'erreur

Le but de l'apprentissage par correction d'erreur est de minimiser une fonction coût basée sur le signal d'erreur $e_k(n)$ obtenu par les différences entre les réponses désirées des neurones et les réponses actuelles de ces mêmes neurones. Soit $d_k(n)$ la réponse désirée d'un neurone *k* à l'instant *n*, et $y_k(n)$ la réponse à cet instant de ce même neurone, donc le signal d'erreur de ce neurone peut être définit par la relation : $e_k(n) = d_k(n) - y_k(n)$.

L'apprentissage par correction d'erreur devient un problème d'optimisation classique. Il consiste à minimiser un indice de performance *E* basé sur les signaux d'erreur $e_k(n)$, dans le but de faire converger les sorties du réseau avec ce qu'on voudrait qu'elles soient. Un critère très populaire est la somme des erreurs quadratiques dont la valeur instantanée est définie de la manière suivante :

$E(n) = \frac{1}{2} \sum_{k=1}^{n} e_k^{\,2}(n)$ où la sommation s'effectue sur tous les neurones de la couche de sortie.

La minimisation de la fonction *E(n)* par rapport aux paramètres du réseau mène à la méthode du "gradient descendant".

Le réseau sera alors optimisé en minimisant *E(n)* par rapport aux poids synaptiques du réseau. Si l'on utilise la méthode de descente du gradient ou gradient stochastique, l'ajustement $\Delta w_{kj}(n)$ du poids synaptique $w_{kj}(n)$ à l'instant *n* sera donné par l'expression : $\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$ où η est une constante positive qui détermine le pas d'apprentissage.

Le choix de pas d'apprentissage η est un facteur important pour assurer une stabilité dans le processus d'apprentissage par correction d'erreur.

3.3.3. Classification

Dans une phase de classification, il existe un nombre déterminé de classes. Les poids du réseau de neurones ont été appris pendant la phase d'apprentissage en lui présentant un ensemble d'échantillons d'entrée avec les classes correspondantes. Lors de la classification, un nouvel ensemble d'échantillons est présenté appartenant aux mêmes classes utilisées dans l'apprentissage, mais qui n'a pas été présenté auparavant au réseau. Le réseau sera chargé alors de classifier correctement ce nouvel ensemble. Les sorties du réseau permettent de déterminer la classe reconnue.

L'avantage de l'utilisation des réseaux de neurones dans le cas de la classification c'est qu'ils peuvent traiter des problèmes avec des paramètres très nombreux, et ils sont aussi capables de construire des frontières de décision non linéaires entre les différentes classes d'une manière non paramétrique. Par conséquent, ils offrent une méthode pratique pour la résolution de problèmes complexes de classification.

3.3.4. Processus de la rétropropagation du gradient

La rétropropagation du gradient est une technique permettant de modifier un réseau de neurones afin de lui faire apprendre la réaction désirée correspondante à un stimulus précis. C'est donc une technique indispensable dans le cadre de l'apprentissage. On la retrouve dans la grande majorité des simulateurs de réseaux de neurones existants.

Le processus de la rétropropagation d'erreur consiste dans un premier temps à propager vers l'avant les entrées jusqu'à obtenir une sortie calculée par le réseau. La seconde étape compare la sortie calculée à la sortie réelle connue. Les poids sont alors modifiés de telle sorte qu'à la prochaine itération, l'erreur commise entre la sortie calculée et connue soit minimisée. L'erreur commise est ensuite propagée vers l'arrière jusqu'à la couche d'entrée tout en modifiant la pondération. Ce processus est répété sur tous les exemples jusqu'à ce que l'erreur de sortie soit négligeable.

Dans l'application pratique de l'algorithme de retro-propagation, l'apprentissage résulte de la présentation d'un ensemble d'exemples d'entraînement au perceptron multicouches. Une présentation complète de tout l'ensemble d'entraînement au cours du processus d'apprentissage s'appelle époque (Epoch). Le processus d'entraînement est itéré d'époque en époque jusqu'à ce que les poids synaptiques et les niveaux des seuils du réseau se stabilisent et la moyenne quadratique de l'erreur sur tout l'ensemble d'entraînement converge vers une valeur minimale.

L'algorithme d'apprentissage par rétropropagation du gradient de l'erreur, qui présente l'avantage d'exister, reste discutable dans la mesure où sa convergence n'est pas prouvée. Son utilisation peut conduire à des blocages dans un minimum local de la surface d'erreur. Son efficacité dépend, en effet, d'un grand nombre de paramètres que doit fixer l'utilisateur: le pas du gradient, les paramètres des fonctions sigmoïdes, l'architecture du réseau ; le nombre de couches, le nombre de neurones par couche, l'initialisation des poids, etc.

3.4. Des exemples d'utilisation des MLP en reconnaissance de l'écriture

Du fait de ses nombreux avantages, les perceptrons multicouches MLP sont parmi les RNA fréquemment utilisés dans les systèmes de reconnaissance automatique des textes. En effet, plusieurs systèmes à base des MLP sont proposés pour la reconnaissance de l'écriture des textes imprimés et des textes manuscrits [Basu10], en particulier pour la reconnaissance

caractères manuscrits ([LeCu94], [Verm98], [Zhan00]), de chiffres isolés ([LeCu89], [Gade96], [LeCu98], [Liu99], [Wong02], [Mori06]) et pour la lecture des montants numériques de chèques ([Lero97], [Kner97], [Fara06]).

Les réseaux de neurones sont également utilisés pour la reconnaissance de l'écriture arabe. Dans ce sens, il existe de nombreux type de réseaux de neurones. Mais parmi les plus utilisés en reconnaissance d'écriture arabe manuscrite sont les perceptrons multicouches ([Amin96b], [Amin97], [Aloh02], [Klas02], [Alma06], [Rach06a]).

Autrement, dans certaines méthodes de reconnaissance de mots, les auteurs proposent des techniques de segmentation explicite des mots en caractères en utilisant des RNA ([East97], [Blum99]). Dans ce cas, les RNA sont premièrement entraînés pour détecter les hypothèses de segmentation correctes. En fin, du fait de leur rapidité, les MLP ont été largement utilisés en premier dans des combinaisons séquentielles de classifieurs, en particulier avec les SVM [Bell01] et aussi avec les HMM [Zerm07]).

4. Conclusion

L'objectif de ce chapitre est de donner un aperçu général sur les deux approches de reconnaissance adoptées dans ce travail. Dans un premier temps, nous sommes intéressés aux approches syntaxiques pour la reconnaissance de formes en rappelant les notions de la théorie des langages formels nécessaires à la compréhension de ces approches. Ces approches consistent à mettre en relation la structure des formes analysées et la syntaxe d'un langage formel. La description des formes est réalisée par l'intermédiaire de phrases et le problème de classification est ramené à un problème d'analyse de grammaire. Aussi, nous avons présenté par la suite une synthèse des concepts de base de la théorie des réseaux de neurones, en introduisant les éléments de base de ces modèles, ainsi que le processus d'apprentissage communément utilisé pour les entraîner. En fin, nous avons cité des exemples d'utilisation des perceptrons multicouches dans la reconnaissance automatique de l'écriture.

Dans le chapitre suivant, nous présentons un aperçu de la langue Amazighe avec une revue des approches utilisées pour la reconnaissance automatique de l'écriture amazighe.

Chapitre 3 : État de l'art sur la reconnaissance de l'écriture amazighe

1. Introduction

La langue amazighe possède sa propre écriture depuis l'Antiquité. Cette écriture est de nature alphabétique consonantique. Elle est encore utilisée de nos jours chez les Amazighes des zones sahariennes, les Touarègues, qui l'appellent « Tifinagh ». C'est dans cet alphabet que sont rédigées les inscriptions anciennes dites « libyco-berbères » relevées partout en Afrique du Nord et au Sahara, de la méditerranée au sud du Niger et des iles Canaries à la frontière ouest de l'Égypte [Ameu06].

Les premières tentatives de normalisation pour l'apprentissage de cette langue au Maroc étaient amorcées par l'Institut Royal de la Culture (IRCAM), depuis 2002. En effet, l'IRCAM a fait une action majeure pour la standardisation de la langue amazighe. Dans la même foulée, et depuis 2003, l'enseignement de l'Amazighe couvre tous les niveaux du primaire et les études amazighes font leur entrée à l'université [Boum09]. Les supports pédagogiques sont élaborés par les équipes pédagogiques et linguistiques de l'IRCAM. En outre, l'adoption de la graphie Tifinagh pour l'écriture amazighe [Iazz04] a permis d'avoir une graphie et une orthographe officielles[Ameu06], un codage propre dans le standard Unicode et dans la norme ISO/ CEI 10646 [Zenk04], des normes appropriées pour la disposition d'un clavier amazighe [Zenk08], des supports médiatiques et pédagogiques et des structures linguistiques qui sont en phase d'élaboration avec une démarche progressive. Cette démarche a été initiée par la construction des lexiques ([Kame06], [Ameu09]), l'homogénéisation de l'orthographe et la mise en place des règles de segmentation de la chaîne parlée [Ameu06] et par l'élaboration des règles de grammaire [Bouk08], d'un manuel de conjugaison et d'un vocabulaire grammatical [Boum09].

D'autre part, avec la diffusion de la langue amazighe sur le Web et la disponibilité des moyens de manipulation de texte amazighe, les travaux de recherche ont abordé des problématiques plus variées pour que l'Amazighe puisse rejoindre ses consœurs dans le domaine des nouvelles technologies de l'information et de la communication. Dans ce contexte, de nombreuses recherches scientifiques sont lancées au niveau national pour

améliorer la situation actuelle. Principalement, elles se focalisent sur l'élaboration des ressources et outils linguistiques ([Iazz08], [Boula09], [Ataa10], [Outa11]), la traduction automatique [Rach07], la correction orthographique [Essa09], l'éditeur de texte ([Rach05a], [Essa08b]), et la reconnaissance optique de l'écriture ([Essa08a], [Aito09], [Amro09], [Essa10], [Amro10], [Elay10], [Elay11], [Essa11a], [Essa11b], [Essa11c]).

Dans ce chapitre nous présenterons la langue amazighe, l'histoire de cette langue, son alphabet, son utilisation informatique et les caractéristiques morphologiques de son écriture. Le but de présenter ces caractéristiques est de donner une idée sur la façon de mettre en ouvre un système de reconnaissance de texte amazighe. Par la suite nous décrirons les différents travaux effectués dans le domaine de la reconnaissance automatique de l'écriture amazighe avec les principales bases de données de caractères amazighes existantes.

2. Présentation de la langue Amazighe

2.1. Historique de l'écriture tifinagh

Le tifinagh est le système d'écriture de la langue amazighe. Il tire son origine du vieil alphabet libyque et saharien, déjà utilisé depuis le VIe siècle avant l'ère chrétienne par les populations de l'Afrique du Nord, du Sahel et des Iles Canaries. Le tifinagh a subi des modifications et des variations depuis son origine jusqu'à nos jours, et ce du libyque jusqu'au néotifinaghe en passant par le tifinagh saharien et le tifinagh touareg. Nous retraçons ci-dessous les aspects les plus importants de chacune de ces variations ([Ameu04], [Ameu06]).

Le libyque : Il s'agit des variétés de tifinaghs les plus anciennes. Il existe deux formes du libyque, l'occidental utilisé le long de la côte méditerranéenne de la Kabylie jusqu'au Maroc et sans doute aux Îles Canaries. Et la forme orientale a été utilisée dans le Constantinois, en Aurès et en Tunisie.

Le Tifinaghe saharien : Cette variété est également appelée libyco-berbère ou touareg ancien. Elle contient des signes supplémentaires par rapport au libyque, plus particulièrement un trait vertical pour noter la voyelle finale /a/. Cette variété fut utilisée pour transcrire le touareg ancien, mais ses inscriptions sont incomprises. L'âge des inscriptions les plus récentes remonte probablement à quelque 200 ans.

Le Tifinaghe touareg : Il existe au sein du tifinagh touareg quelques divergences dans la valeur attribuée aux signes qui correspondent aux variations dialectales touarègues. Si d'une région à une autre, la forme et le nombre des signes peuvent changer, les textes restent en général mutuellement compréhensibles.

Le néotifinaghe : il désigne les systèmes d'écriture développés pour représenter les parlers amazighes du Maghreb. La première variante fut celle proposée à la fin des années 60 par l'Académie berbère (AB) sur la base de lettres tifinaghs touarègues. Elle est largement diffusée au Maroc et en Algérie. Ce vocable comprend aussi d'autres variantes venues développer ou pour certaines corriger les quelques imperfections du système de l'Académie berbère.

Pour une étude relativement complète à ces variations, le lecteur peut se référer au livre de Meftaha Ameur et al. [Ameu06] publié par l'IRCAM, qui a fourni une étude d'ensemble sur la graphie tifinagh moderne. Il livre aussi un historique de l'alphabet, son origine, ses différentes variantes et leur déchiffrement.

La renaissance de l'alphabet amazighe en Afrique du Nord est incontestablement due au travail énorme accompli par l'AB. Cette association formée par de jeunes militants amazighes (Kabyles en grande partie) installés à Paris a largement diffusé l'alphabet tifinagh en Algérie et au Maroc.

Au Maroc, l'amazighe se répartit en trois grandes variétés régionales qui couvrent l'ensemble des régions montagneuses : le Tarifite au nord-est ; le Tamazighte au centre, le Moyen-Atlas et une partie du Haut-Atlas; et le Tachelhite au sud et sud-ouest, le Haut-Atlas, l'Anti-Atlas et Souss. 50% de la population marocaine est berbérophone [Bouk95]. Toutefois, tous les Marocains sont concernés par cet alphabet. L'IRCAM a fait une action majeure pour la standardisation de la langue amazighe. La langue amazighe a intégré le système éducatif marocain, depuis 2003. Elle est enseignée dans les classes du primaire des différentes écoles marocaines, en perspective d'une généralisation graduelle aux niveaux scolaires et de l'extension à de nouvelles écoles [Bouk08].

L'écriture tifinagh ancienne se trouve gravée dans les pierres et les tombes de certains sites historiques dans le nord de l'Algérie, au Maroc, en Tunisie, et dans les régions touarègues du Sahara. La Figure 3.1 ci-dessous présente une image d'une écriture tifinagh ancienne trouvée dans le site de gravures rupestres d'Intedeni près de la ville Essouk au Mali [Wiki06].

Un autre exemple d'une écriture tifinagh ancienne trouvée sur une dalle gisant sur le bord de l'oued I-n-Ana, dans l'adrar Tekembaret en Immidir (Algérie) [Yves03]. Un extrait de cette écriture est présenté dans la Figure 3.2 ci-dessous.

Récemment, Skounti et al. [Skou03] ont constitué un corpus des inscriptions amazighes des sites rupestres au Maroc. Ils ont développé une analyse détaillée des données du corpus constitué d'une soixantaine d'inscriptions issues de treize sites répartis sur la moitié méridionale du pays, du Haut-Atlas jusqu'à la frontière mauritanienne. Ce corpus se

focalise sur les inscriptions amazighes du Maroc, et il permet de faire la lumière sur l'écriture de l'amazighe dans cette partie de l'Afrique du Nord.

Figure 3.1 : Écriture tifinagh ancienne, site des gravures rupestres d'Intédeni près d'Essouk au Mali (figure extraite de [Wiki06])

Figure 3.2 : Inscriptions sur dalle horizontale, Oued I-n-Ana (figure extraite de [Yves03])

2.2. Alphabet Tifinaghe-IRCAM

L'IRCAM a développé un système Tifinaghe-IRCAM ayant pour objectif la normalisation de la graphie amazighe tout en s'inscrivant dans la continuité historique de l'alphabet tifinagh. Le Centre de l'Aménagement Linguistique (CAL) de l'IRCAM a préconisé les

caractères tifinaghs, présentés dans le Tableau 3.1 ci-dessous, comme alphabet de la langue amazighe [Ameu04]. Ce tableau présente le répertoire officiel de l'alphabet Tifinaghe-IRCAM avec leurs correspondants en arabe et en caractères latins. Cet alphabet est formé de 33 graphèmes correspondant aux 33 phonèmes de l'Amazighe standard.

	TIFINAGHE	Correspondance latine	Correspondance arabe	Exemples
ya	ⴰ	a	ا	ⴰⵏⴰⴰ
yab	ⴱ	b	ب	ⴰⴱⴰⵔⵏ
yag	ⴳ	g	گ	ⵜⵉⵅⵍⵍⵉ
yagʷ	ⴳⵯ	gʷ	گ،	ⴰⵣⵅⵅᵘⴰⵢ
yad	ⴷ	d	د	ⴰⵀⵙⵏ
yaḍ	ⴹ	ḍ	ض	ⴰⴹⴰⵇ
yey	ⵄ	e		ⴰⵀⵇⴻⵙⴻⴻⵙ
yaf	ⴼ	f	ف	ⴰⵀⵙⵙ
yak	ⴽ	k	ک	ⵜⵉⵙⴽⴰⵜ
yakʷ	ⴽⵯ	kʷ	ک،	ⴰⵍⵏⵏⴰⴽⴽᵘⵀ
yah	ⵀ	h	ه	ⴰⵀⵏⵏⵙⵍ
yaḥ	ⵃ	ḥ	ح	ⴰⵃⵍⴰⵛ
yaɛ	ⵄ	ɛ	ع	ⴰⵄⵜⴰⵍ
yax	ⵅ	x	خ	ⵜⵉⵅⵅⵙⵉ
yaq	ⵇ	q	ق	ⴰⵇⴰⴰⵜ
yi	ⵉ	i	ي	ⵉⵍⵉ
yaj	ⵊ	j	ج	ⴰⵢⵍⵊⴰ
yal	ⵍ	l	ل	ⴰⵍⵀⴰⵀ
yam	ⵎ	m	م	ⴰⵍⴰⵍ
yan	ⵏ	n	ن	ⵉⵔⵏⵍ
yu	ⵓ	u	و	ⵙⵏⵍ
yar	ⵔ	r	ر	ⵙⵔⴰⵔ
yaṛ	ⵕ	ṛ	ڕ	ⵜⵕⵕⴰ
yaɣ	ⵖ	ɣ	غ	ⴰⵢⵔⵙⵍ
yas	ⵙ	s	س	ⵉⵀⵙ
yaṣ	ⵚ	ṣ	ص	ⵜⴰⵍⵚⵕⵇⵉⵖⵜ
yac	ⵛ	c	ش	ⴰⵛⵍⵍⴰⵀ
yat	ⵜ	t	ت	ⵜⴰⵀⵙⵍⴰⵔⵜ
yaṭ	ⵟ	ṭ	ط	ⵜⵉⵟⵟ
yaw	ⵡ	w	وْ	ⴰⵡⴰⵀ
yay	ⵢ	y	يْ	ⵜⴰⵢⵔⴰ
yaz	ⵣ	z	ز	ⴰⵍⴰⵣⵉⵢ
yaẓ	ⵥ	ẓ	ژ	ⴰⵣⴰⵀⵉⵍ

Tableau 3.1 : Le répertoire officiel de l'alphabet Tifinaghe-IRCAM avec leurs correspondants en arabe et en caractères latins [Ameu04]

L'alphabet Tifinaghe-IRCAM est réparti de la manière suivante :

- Quatre graphèmes vocaliques : ⵄ «ya», ⵉ «yi » et ⵓ «you», qui notent les trois voyelles de base de l'amazighe standard, et ⴻ «yey» qui traduit la voyelle neutre ;
- Deux semi-consonnes : ⵢ «yay» et ⵡ «yaw» ;
- Vingt consonnes simples : ⴱ «yab », ⵎ «yam », ⴼ «yaf», ⵜ «yat», ⴷ «yad», ⵏ «yan», ⵙ «yas», ⵣ «yaz», ⵍ «yal», ⵔ «yar», �right «yash», ⵊ «yaj», ⴽ «yak», ⵅ «yag», ⵖ «yagh», ⵀ «yaa», ⵇ «yaq», ⵃ «yah», ⵅ «yakh», ⵄ «yah» ;
- Cinq emphatiques : ⴹ «yadd», ⵟ «yatt», ⵥ «yazz», ⵚ «yass», ⵕ «yarr» ;
- Deux labiovélaires : ⴽⵯ «yakw», ⵅⵯ «yagw» ;
- 27 consonnes dont les labiales (ⴼ, ⴱ, ⵎ), les dentales (ⵜ, ⴷ, ⵟ, ⴹ, ⵏ, ⵔ, ⵕ, ⵍ), les alvéolaires (ⵙ, ⵣ, ⵚ, ⵥ), les palatales (�right, ⵊ), les vélaires (ⴽ, ⵅ), les labiovélaires (ⴽⵯ, ⵅⵯ), les uvulaires (ⵇ, ⵅ, ⵖ), les pharyngales (ⵃ, ⵀ) et la laryngale (ⵄ) ;
- Deux semi-consonnes : ⵢ et ⵡ ;
- Quatre voyelles : trois voyelles pleines ⵄ, ⵉ, ⵓ et la voyelle neutre ⴻ qui a un statut assez particulier en phonologie amazighe.

Pour la ponctuation, nous ne connaissons pas de signes de ponctuation particuliers au tifinagh. L'IRCAM a préconisé l'emploi des signes conventionnels qu'on retrouve dans les écritures latines : « » (espace), «. », « , », « ; », « : », « ? », « ! », « … », etc. En conséquence, cette proposition ne présente aucun signe de ponctuation tifinagh.

L'IRCAM a retenu aussi les chiffres arabes occidentaux (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) et tous les signes logiques conventionnels (+, -, *, /, =, %, α, β, ∑, Ø, π, etc.) pour l'écriture amazighe.

Pour le tri des lettres tifinaghs, dans les différentes variations du tifinagh, aussi bien anciennes que modernes, il n'ya pas d'ordre pour énoncer les lettres de l'alphabet. Seul l'Ircam a défini un ordre précis décrit par l'expression présentée dans la Figure 3.3 ci-dessous (a < b, signifie que a est trié avant b) :

Figure 3.3 : Le tri des caractères amazighes

Historiquement, l'Amazighe des anciennes inscriptions s'écrivait horizontalement de gauche à droite ou de droite à gauche, ou bien verticalement de bas en haut ou de haut en bas [Ircam04c]. L'orientation la plus souvent adoptée dans l'écriture amazighe moderne est horizontale et de gauche à droite, et c'est l'orientation adoptée pour le Tifinaghe-IRCAM [Ameu04].

2.3. Utilisation informatique du tifinagh

La langue Amazighe a connu un processus de standardisation et d'intégration dans les nouvelles technologies de l'information et de la communication, qui est passé par plusieurs étapes : le codage spécifié par l'ASCII étendu, la création des polices de caractères tifinaghs, le codage propre dans le standard Unicode et l'élaboration des normes appropriées concernant la disposition du clavier amazighe [Zenk08], ainsi le développement des applications de l'informatisation de l'Amazighe.

L'IRCAM a proposé à l'Organisation de Standarisation Internationale (21/06/2004) l'Alphabet tifinagh et ce dernier a été confirmé le 05/07/2004 [Ircam04c]. Cette proposition comprend quatre sous-ensembles de caractères tifinaghs :

1. le jeu de base de l'IRCAM ;
2. le jeu étendu de l'IRCAM ;
3. d'autres lettres néotifinaghes en usage ;
4. des lettres touarègues modernes dont l'usage est attesté.

La Figure 3.4 ci-dessous illustre le bloc Unicode réservé aux caractères tifinaghs, qui précise les codes réservés à chacun de ces quatre sous-ensembles de caractères tifinaghs. Le premier sous-ensemble représente les lettres alphabétiques de base préconisées par l'IRCAM dont le nombre est de 33. L'Unicode ne code directement que 31 caractères et le caractère modificatif « ⵯ », qui permet de former les deux unités labiovélaires «ⵅⵯ» et « ⴽⵯ». Le deuxième sous-ensemble contient les 8 caractères de la liste étendue, qui a été définie par l'IRCAM pour l'intérêt historique et scientifique. Le troisième sous-ensemble est formé de 4 lettres néo-tifinaghs utilisés fréquemment dans le reste du Maghreb. Et le quatrième sous-ensemble contient 11 lettres touarègues modernes dont l'usage est attesté [Andr08].

L'IRCAM a développé également plusieurs pilotes de clavier tifinagh ainsi qu'un assortiment de polices [Zenk03]. Grâce à ce clavier et ces polices, des manuels scolaires en tifinagh et des articles en tifinagh dans des journaux ont pu être produits. Le répertoire de l'IRCAM [Ircam04a] a été déposé auprès du service responsable de la normalisation industrielle du Maroc [Snima04].

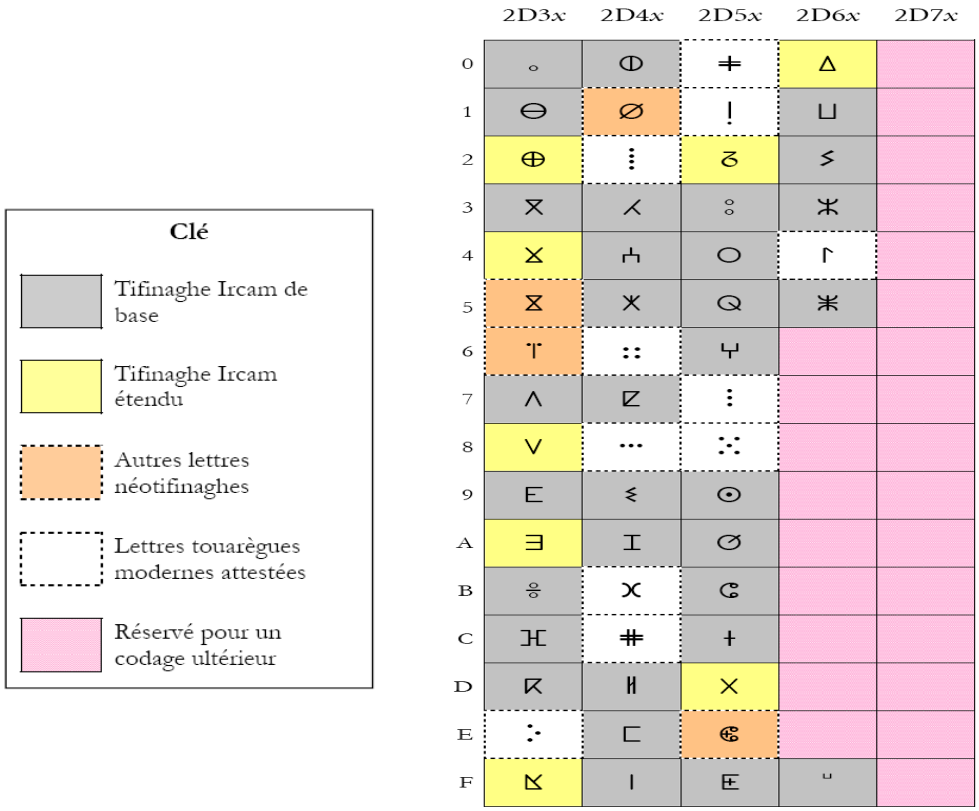


Figure 3.4 : L'alphabet Tifinagh et leur code Hexadécimal dans le format Unicode

Le centre CEISIC de L'IRCAM a proposé un clavier sous format ASCII (police, pilote) comme l'illustre la Figure 3.5 ci-dessous **(**[Zenk03], [Ircam03]). Les 26 premiers caractères sont accessibles directement. Les caractères emphases s'obtiennent en utilisant la case Noir (le « ^ » en clavier latin) de la même façon qu'on utilise le « ^ » en français (pour taper le « â »).

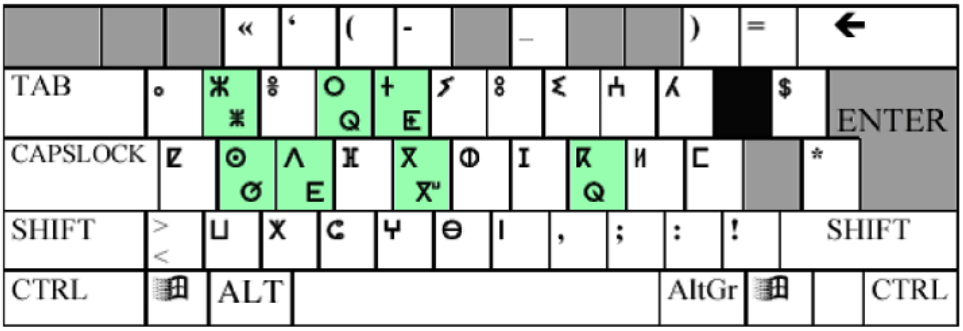


Figure 3.5 : Clavier Tifinaghe-IRCAM

Après l'intégration de l'Amazighe dans le standard Unicode, L'IRCAM a proposé un projet qui normalise deux groupes de claviers en précisant deux niveaux de conformité, l'un pour la saisie stricte des trente-trois lettres de l'alphabet tifinagh de base tel qu'enseigné dans les écoles marocaines, l'autre pour la saisie de l'alphabet de base, plus les vingt-deux lettres de l'alphabet tifinagh étendu et des ligatures. Ce projet à été confirmé, actuellement on dispose des claviers et polices Unicode, clavier amazighe de base (cf. Figure 3.6 ci-dessous) [Ircam03] et clavier amazighe de base étendue (cf. Figure 3.7 ci-dessous) [Ircam04b] qui permet de saisir des caractères amazighes on format Unicode. Ce clavier est conforme à la norme internationale ISO/CEI 14651. Dans ce clavier l'accès aux autres lettres ce fait en utilisant la touche MAJ, au contraire au clavier Tifinaghe-IRCAM (Figure 3.6 et Figure 3.7 ci-dessous).

Figure 3.6 : Clavier Tifinagh de base

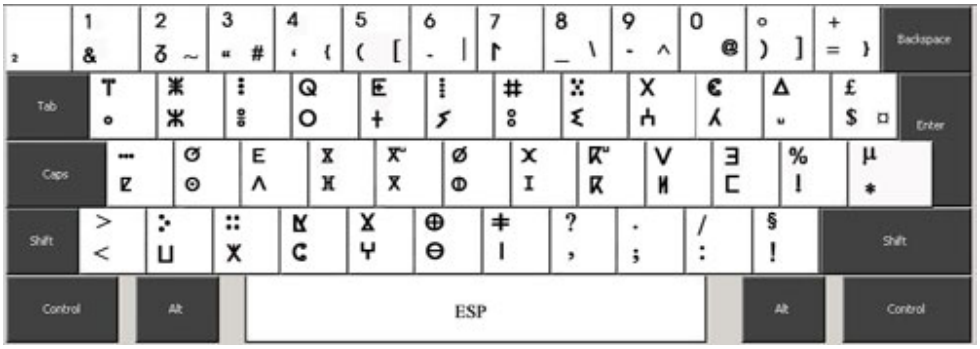


Figure 3.7 : Clavier Tifinagh étendue

Pour une revue sur le processus de standardisation informatique de la langue Amazighe par L'IRCAM, le lecteur peut se référer aux articles de L. Zenkouar ([Zenk04], [Zenk08]).

En plus des travaux réalisés par les différents centres de recherche de l'IRCAM sur la l'informatisation de la langue Amazighe, on trouve aussi des travaux de recherches scientifiques effectués par les universitaires permettent d'intégrer cette langue dans les nouvelles technologies d'informations et de communication. L'un des volets prioritaires de ces recherches est de concevoir et réaliser des applications capables de traiter de façon automatique des données linguistiques de l'Amazighe.

Parmi ces travaux, Rachidi et Mammass proposent dans ([Rach05a], [Rach05b]) une réflexion sur les stratégies et les méthodes d'informatisation de l'Amazighe et leurs mises en œuvre en décrivant les ressources et les logiciels retenus dans le cadre d'informatisation de la langue, ainsi les éléments de base de système d'écriture informatique amazighe. Ils décrivent aussi la mise en œuvre de certaines méthodes et les outils impliqués, tels que la réalisation d'un éditeur de traitement de texte pour l'Amazighe et la création de fenêtres d'édition fortement multilingues. Ils ont proposé aussi dans [Rach07] une réflexion pour développer une plate forme logicielle qu'intègre l'Amazighe dans les documents multilingues en construisant un serveur amazighe qu'ajoute l'enconvertisseur et le déconvertisseur UNL en se basant sur les bases de connaissances construites sur la base des objets typés. Ces objets sont décrits dans le langage UNL qui est un système de traduction automatique universel soutenu par la communauté internationale. L'objectif c'est d'avoir un moyen de diffusion du savoir amazighe afin de valoriser cette culture sur le Web et dans le monde informatisé.

Dans le contexte de traitement automatique de la langue amazighe, nous avons développé un système automatique d'écriture qui permet l'édition du texte amazighe ([Essa08b], [Bakk08]). Ce système, dans sa première version, permet d'effectuer les opérations de base d'un éditeur de texte. Ainsi, il est bien adapté à l'écriture amazighe en offrant des fonctionnalités spécifiques de l'écriture amazighe.

Autrement, le texte amazighe peut être édité et mis en forme dans des éditeurs de texte existants en utilisant des polices tifinaghs de l'IRCAM, mais la correction orthographique du texte amazighe n'existe pas encore dans ces éditeurs. Dans ce cadre, nous avons commencé un travail de création d'un outil de correction orthographique libre pour l'Amazighe [Essa09] pouvant être intégré à l'environnement bureautique et dont les ressources langagières seront accessibles et modifiables. Ce travail est basé sur l'algorithme de correction orthographique du programme libre Hunspell [Hans], qui est le correcteur orthographique directement intégré à 'OpenOffice.org' et que nous avons trouvé mieux adapté au cas de l'Amazighe. D'où, le plus important du travail c'est la création d'un fichier des affixes contenant les règles d'affixation à partir des lemmes figurant dans un dictionnaire de mots amazighes autorisés. Ce fichier sera utilisé par le programme Hunspell pour vérifier l'existence d'un mot soit dans le dictionnaire soit en composant de nouveaux

mots à l'aide de ce fichier. Nous avons trouvé des difficultés au niveau de la morphologie amazighe pour créer ce fichier qui englobe toutes les règles d'affixation de cette langue. Il est souvent créé manuellement et c'est un travail très coûteux, qui consomme beaucoup de temps, mais qui ne demande pas des compétences en informatique ce qui en fait un travail des linguistes connaissant les règles orthographiques et grammaticales de la langue Amazighe. En perspective, nous essayons de continuer sur ce projet en espérant d'avoir des contributions avec des linguistes pour créer le fichier des affixes de la langue Amazighe.

2.4. Caractéristiques de l'écriture amazighe

Nous citons ci-dessous quelques caractéristiques morphologiques de l'écriture amazighe adoptée par l'IRCAM. Ces caractéristiques seront exploitées pendant la mise en ouvre d'un système de reconnaissance de texte amazighe.

- À la différence des caractères latins et arabes, l'écriture amazighe n'est pas cursive, ce qui facilite toute opération de segmentation. De plus, l'écriture amazighe est écrite de gauche à droite, elle utilise des signes de ponctuation classique acceptés en alphabet latin ;
- La majorité des modèles graphiques des caractères est composée de points, de petits cercles, et/ou de segments. Nous pouvons faire une première classification des caractères amazighes suivant la forme de la graphie :
    - Lettres circulaires : ⵢ, ⵙ, ⵔ, ⴰ, ⵇ, ⵕ, ⵚ, ⴱ, ⵀ ;
    - Lettres linaires : ⵏ, ⵍ, ⵜ, ⵃ, ⵖ, ⵄ, ⵅ, ⴽ, ⵟ, ⵛ, ⵡ, ⵎ, ⴹ, ⴷ, ⵓ, ⴳ, ⵊ, ⴻ, ⵉ ;
    - Autres : ⴼ, ⵣ, ⵥ, ⴽⵯ, ⴳⵯ.
- L'alphabet Tifinaghe-IRCAM ne comporte pas de forme majuscule, ainsi des glyphes marquant le début de phrases ; en d'autres termes par ce que les écritures latines dénomment majuscules ;
- Par contre aux caractères arabes et latins qui ne possédant pas une taille fixe (hauteur et largeur), leur taille varie d'un caractère à un autre, tous les caractères amazighes s'écrivent de mêmes tailles verticales sauf le seul caractère ya (ⵢ), qu'est plut petit par rapport aux autres caractères. D'où, en écriture amazighe, il n'y a pas d'ascendants et descendants ;
- Pour les signes diacritiques, l'alphabet tifinagh comporte seulement deux caractères affectés d'un signe diacritique : ⴽⵯ «yakw» et ⴳⵯ «yagw». Ces derniers servent à noter les labiovélaires qui sont des articulations complexes [Ameu06]. Le signe diacritique en question a aussi pour fonction de noter l'arrondissement des lèvres qui accompagne la réalisation des consonnes ⴽ et ⴳ, d'où ⴽⵯ et ⴳⵯ ;
- En écriture amazighe, il n'y a pas de notion de pseudo-mot ;

– Les caractères amazighes sont formés généralement par une seule composante connexe. Néanmoins, des cas particuliers existent illustrés dans la Figure 3.8 ci-dessous, qui sont formés de deux ou de trois composantes connexes.

Caractère amazighe	ⵓ	ⴻ	ⴽⵯ	ⴳⵯ
Correspondance latine	u	e	k^w	g^w

Figure 3.8 : Caractères amazighes à plusieurs composantes connexes

– Une autre caractéristique importante de caractères amazighes est que la majorité de ces caractères possèdent la ligne centrale horizontale ou verticale comme un axe de symétrie. En effet, la ligne centrale du caractère peut être considérée comme une ligne de base de l'écriture amazighe en exploitant les caractéristiques basées sur cette ligne pendant la reconnaissance. Cette propriété sera exploitée pour l'extraction des caractéristiques du système de reconnaissance de l'écriture amazighe présenté dans le chapitre 6. Nous citons ci-dessous les caractères amazighes qui possèdent les lignes centrales (horizontale et verticale) comme axes de symétries.

- Les caractères qui possèdent une symétrie orthogonale par rapport à la ligne centrale horizontale du caractère :

  ⵓ, ⵔ, ⴱ, ⴻ, ⵟ, ⵀ, ⵣ, ⵥ, ⵊ, ⵏ, ⵜ, ⵛ, ⵃ, ⵎ, ⵉ, ⵞ, ⵄ, ⵚ ;

- Les caractères qui possèdent une symétrie orthogonale par rapport à la ligne centrale verticale du caractère :

  ⵓ, ⵔ, ⴱ, ⴻ, ⵟ, ⵀ, ⵣ, ⵥ, ⵊ, ⵏ, ⵜ, ⵛ, ⵃ, ⵡ, ⵖ, ⵕ, ⴷ, ⴳ, ⵅ, ⵙ ;

- Autres caractères qui ne possèdent ni l'axe verticale ni l'axe horizontale comme axe de symétrie :

  ⵇ, ⵚ, ⴹ, ⵢ, ⴽ, ⵤ, ⴽⵯ, ⴳⵯ.

La Figure 3.9 ci-dessous montre certaines de ces caractéristiques morphologiques dans un texte amazighe extrait dans un manuel scolaire.

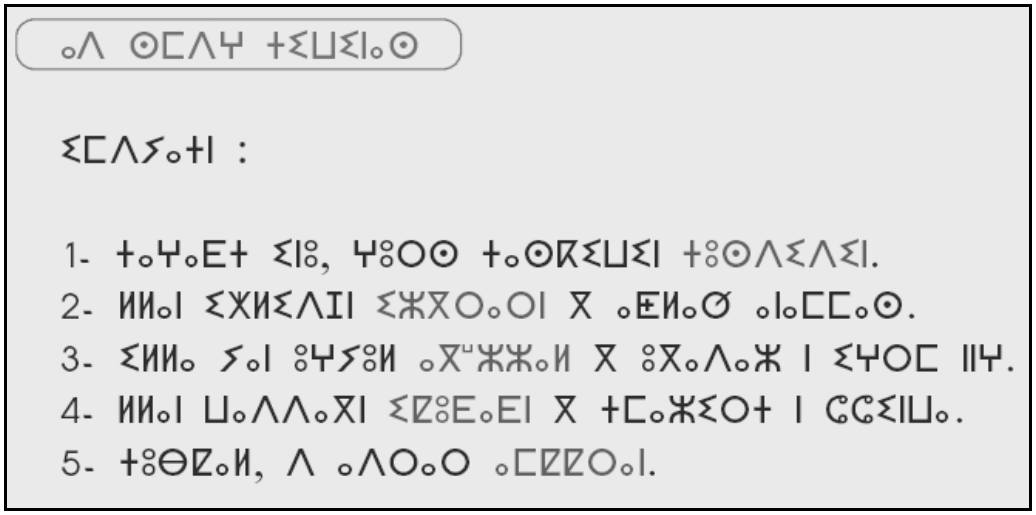


Figure 3.9 : Exemple du texte amazighe dans un manuel scolaire

3. Travaux antérieurs sur la reconnaissance de l'écriture amazighe

3.1. Principales bases de données existantes

Dans le domaine de la reconnaissance automatique de l'écriture amazighe, les bases de données d'images annotées sont inexistantes. Ce domaine a ainsi péché de l'absence d'une base de données de référence, qui permet des comparaisons objectives entre les différents systèmes de reconnaissance. Tous les travaux publiés dans ce domaine, cités précédemment, ont été expérimentés sur des bases de données locales, qui contiennent un nombre restreint de l'alphabet amazighe.

Parmi ces bases de caractères on cite, en premier lieu, celle élaborée par Ait Ouguengay [Aito09] qui contient à peu près vingt mille de patterns de la graphie amazighe. Cette base sera utilisée pour tester notre approche qui sera présentée dans le chapitre 6. En second lieu, nous avons crée une base de caractères amazighes manuscrits qui a été publié dans [Essa11b]. Dans les deux sous sections suivantes, nous décrivons ces deux bases de caractères.

3.1.1. Base des patterns de la graphie amazighe

C'est une base de patterns de différentes fontes amazighes et de tailles variées. Elle contient au total 12 polices de caractères et les tailles du 10 points au 28 points pour chaque modèle. Les patterns sont fournis sous forme d'images bitonales de tailles variables. La taille maximale est de 102×129 pixels, tandis que la taille minimale est de 19×2 pixels. Une telle disparité s'explique par le fait que le caractère ya (ⴰ) est un petit cercle, et est donc beaucoup plus petit que les autres caractères. Outre le cas particulier du caractère ya (ⴰ), la

base est constituée des patterns de différentes fontes amazighes et de tailles variées, qui ne sont pas normalisées. Le Tableau 3.2 ci-dessous donne une liste de quelques patterns dans cette base.

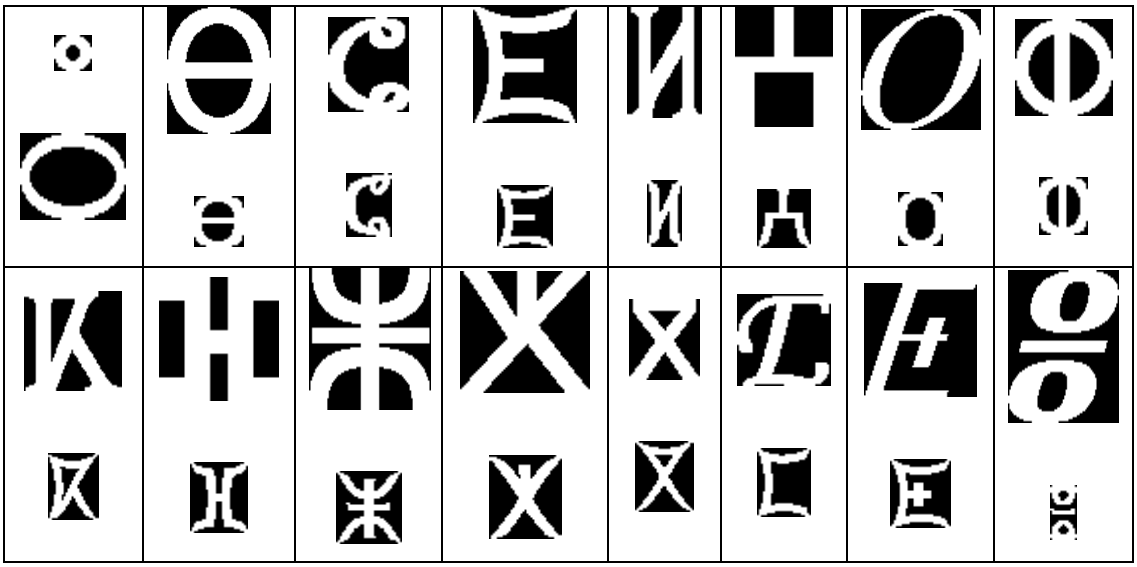


Tableau 3.2 : Exemples de quelques caractères dans la base des patterns de la graphie amazighe

Dans cette base, la manière dont sont créées les images des patterns ne permet pas la possibilité de renormaliser leur taille en une taille moyenne fixe. En effet, ceci peut être gênant en particulier à cause de la ressemblance des caractères ya (ⴰ) et yar (ⵔ) qui ne se différencient que par la taille : le caractère ya (ⴰ) est un petit cercle, tandis que le caractère yar (ⵔ) est un grand cercle. Dans certains cas, on aura une confusion réelle entre des images de ces deux classes. Ce problème aura une influence sur les résultats des tests et sera discuté dans le chapitre 6.

3.1.2. Base de caractères manuscrits

Dans le cadre de cette thèse, nous avons construit une base de données de caractères amazighes manuscrits. Actuellement, la base contient 25740 caractères isolés produits par 60 scripteurs. Elle est conçue pour l'apprentissage et les tests des systèmes de reconnaissance de l'écriture amazighe manuscrite et elle est disponible pour la recherche. Cette base de données sera décrite en détail dans le chapitre suivant.

3.2. Différentes approches existantes

En comparant à l'anglais, l'arabe ou le chinois, les recherches sur la reconnaissance d'écriture amazighe n'ont pas atteint la perfection. Autant que nous le sachions, peu de

tentatives ont été menées sur la reconnaissance d'écriture amazighe. Dans cette section, nous citons des travaux publiés qui touchent à la reconnaissance de l'écriture amazighe.

Parmi les anciennes études qui portent sur la reconnaissance de caractères tifinaghs, on cite, en premier lieu, les travaux d'Oulamara qui ont été publiés dans [Oula88]. La méthode proposée dans cette référence est une méthode statistique basée sur l'extraction de segments de droite par la transformée de Hough. L'analyse du caractère dans l'espace paramétrique, obtenu par la transformation de Hough, permet d'extraire les caractéristiques spécifiques en association avec un modèle de référence générateur de l'ensemble des caractères de l'alphabet. Un codage original est déduit puis utilisé comme base de construction de la matrice de lecture représentant une forme codée de l'alphabet. L'auteur a obtenu des résultats qui semblent intéressants sur les caractères amazighs imprimés d'une base locale.

Djematene et al. [Djem98] considèrent que la méthode publiée par Oulamara [Oula88] n'est pas une technique appropriée pour les caractères amazighes manuscrits puisqu'elle produit des segmentations incorrectes. Pour surmonter la difficulté des caractères présentant des traits inclinés, les auteurs de ([Djem97], [Djem98]) proposent une méthode statistique de reconnaissance de caractères berbères manuscrits basée sur la position des points caractéristiques dans le rectangle-enveloppe de l'image du caractère. Après des prétraitements (normalisation bidirectionnelle, le lissage, l'extraction des composantes connexes) sur le caractère, des primitives sont extraites sur chaque squelette, comme les extrémités, les points, les sommets (points de changements de direction) et les nœuds à 3 et 4 branches. En fin, la représentation du caractère fournit une description sous forme de lettres utilisant un codage prédéfini. Cette description code les positions des points caractéristiques du caractère dans le rectangle-enveloppe. La reconnaissance consiste à mesurer le degré de ressemblance entre le code élaboré et les codes de référence en utilisant la distance métrique. Les résultats obtenus sont encourageants sur une base de caractères localement définie malgré quelques erreurs qui viennent du module de prétraitement.

Dans [Aito09], Ait Ouguengay a proposé un réseau de neurones artificiels (RNA) pour la reconnaissance de caractères amazighes. Le réseau de neurones utilisé est un perceptron multicouche á une seule couche cachée. Ce dernier a été entrainé sur une base de données, qui contient des patterns de la graphie amazighe de différentes fontes et de tailles, crée localement. La simulation du réseau de neurones a été réalisée par le logiciel libre JavaNNS (java neural networks simulator). Les caractéristiques géométriques utilisées sont : les projections horizontales et verticales, les centres de gravité en x et en y, le périmètre, l'aire, la compacité et les moments centraux d'ordre 2. D'après l'auteur, cette approche a donné de bons résultats sur l'ensemble des patterns d'entraînement. Cependant, les résultats de test sont encore loin d'être satisfaisants à cause de la base de test qui est très faible par rapport aux poids de RNA à déterminer.

Amrouch et al. ([Amro09], [Amro10]) présentent un système automatique de reconnaissance de caractères amazighes basés sur le Modèle de Markov cachée (HMM). Après des prétraitements sur l'image du caractère, la chaîne représentative du caractère a été construite à partir de la transformation de Hough. La chaîne obtenue est traduite en séquence d'observations qui est utilisée, lors de la phase d'apprentissage, par le HMM. Le classifieur Forword a été utilisé pour reconnaître le caractère. Les résultats obtenus sont prometteurs sur une base localement définie. Cependant, la discrimination de ces modèles n'est pas très bonne parce que chaque modèle de Markov caché utilise l'apprentissage d'un seul caractère. Le taux d'erreur a été enregistré principalement en raison de la mauvaise écriture et les données d'apprentissage.

Dans [Elay10], El Ayachi et al. Les auteurs proposent un système de reconnaissance de l'écriture tifinagh basé sur les moments invariants et la transformée de Walsh utilisant la programmation dynamique. Le système proposé contient trois parties principales : le prétraitement, l'extraction de caractéristiques et la reconnaissance. Dans le processus de prétraitement l'image du document numérisé est nettoyée puis elle est segmentée en caractères isolés à l'aide des techniques l'histogramme. Dans le processus d'extraction de caractéristiques, les moments invariants et les coefficients de Walsh sont calculés sur les caractères segmentés. La programmation dynamique est adoptée dans l'étape de reconnaissance. Les tests ont été faits sur plusieurs images d'écriture amazighe. D'après les auteurs, les résultats expérimentaux montrent que la méthode de la reconnaissance utilisant des moments invariants donne de meilleurs résultats par rapport à la méthode fondée sur la transformée de Walsh en termes de taux de reconnaissance, de taux d'erreur et de temps de calcul. Plus récemment, les auteurs ont proposé dans [Elay11] un réseau de neurones multi couches avec les mêmes caractéristiques utilisées précédemment. Les résultats trouvés avec un réseau de neurones d'une seule couche cachée sont meilleurs que ceux obtenus avec la programmation dynamique. De plus, le taux de reconnaissance obtenu en utilisant une seule couche cachée est plus élevé que celui obtenu avec deux et trois couches cachées.

Dans le cadre de cette thèse, nous avons proposé deux approches qu'ont contribué à augmenter les performances de la reconnaissance d'écriture amazighe. En premier temps, nous avons proposé une approche syntaxique en utilisant des automates à états finis pour reconnaitre les caractères amazighes imprimés ([Essa08a], [Essa10]). Ce travail sera décrit en détail dans le chapitre 5. En deuxième temps, nous avons développé un système de reconnaissance d'écriture amazighe basée sur la ligne centrale horizontale de l'écriture [Essa11a]. Ce système a montré de bonnes performances sur la base de données de paternes de la graphie amazighe et la base de caractères manuscrits construit localement. Une amélioration de ce système a été publiée dans [Essa11c] en intégrant d'autres

caractéristiques basées sur la ligne centrale verticale du caractère. Ce travail sera décrit en détail dans le chapitre 6.

4. Conclusion

Dans ce chapitre, nous avons présenté un aperçu général sur la langue amazighe ainsi que son utilisation informatique et les principales propriétés morphologiques de son écriture. Aussi, nous avons cité les différents travaux qui touchent la reconnaissance de l'écriture amazighe. L'objectif est de faire une synthèse des travaux effectués sur ce sujet. Ces travaux présentent un certain nombre de limites qui proviennent à la fois de module de prétraitement et des caractéristiques prises dans la phase d'apprentissage. En plus, les bases de caractères utilisées dans les tests restent très faibles et non standards. Par conséquent, des bases de données de référence doivent être créées, ainsi que des travaux de recherches doivent apporter des améliorations d'un côté sur ces approches et d'un autre côté de développer d'autres systèmes complets qui répondent aux attentes.

Dans les chapitres qui suivent, nous présentons nos contributions dans le domaine de la reconnaissance de l'écriture amazighe en commençant par la création d'une base de caractères amazighes manuscrits.

DEUXIÈME PARTIE :

CONTRIBUTIONS

Chapitre 4 : Conception de la base de caractères amazighes manuscrits

1. Introduction

La recherche en Reconnaissance Optique de Caractères (OCR) a commencé dans le début des années 1960. Un problème très crucial et encore ouverte est l'évaluation des systèmes proposés sur la base des ressources communes. En effet, la majorité des chercheurs utilisent leurs propres données en étape d'apprentissage et de test. Par conséquent, l'extraction des conclusions utiles est une tâche très difficile quant à la contribution des systèmes proposés. De plus, la construction d'une base de données est un processus coûteux en temps, mais la disponibilité des bases de données publiques permet aux chercheurs d'accorder plus d'attention afin d'améliorer leurs systèmes plutôt que de collecter de grandes données pour les tests. Actuellement, de nombreuses bases de données ont été recueillies et utilisées dans diverses langues pour les applications de reconnaissance hors-ligne de l'écriture manuscrite. Il ya des bases de données pour le latin ([Hull94], [Kava01], [Mart02]), chinois [Su07], indiens [Bhat09], coréen [Kim93], arabe ([Alma04], [Khor03], [Pech02]) et pour farsi [Ziar09]. Par ailleurs, à ce jour, il n'y a aucune base de données standard pour les caractères amazighes, qui permet des comparaisons objectives entre les différents systèmes. Tous les travaux publiés dans ([Oula88], [Djem97], [Aito09], [Essa10], [Amro10], [Elay10], [Essa11a]) ont été expérimentés sur des bases de données locales, qui contiennent un nombre restreint de caractères amazighes. À l'exception de la base des patterns de la graphique amazighe développée dans [Aito09], elle contient à peut près vingt mille patterns. Nous utilisons cette base de données pour tester notre approche qui sera présentée dans le chapitre 6.

En fait, la construction d'une base de données d'images de caractères amazighes manuscrits a été nécessaire pour cette étude. Dans ce cadre, nous avons développé une base de données de caractères amazighes manuscrits. Cette base de données sera destinée à servir d'autres chercheurs dans le domaine et de standardiser la recherche sur la reconnaissance de l'écriture amazighe manuscrite. Elle contient plus de 25000 images de caractères amazighes manuscrites qui composent 33 classes. Elle sera désignée par le nom "AMHCD" (Amazigh

Handwritten Character Database) dans le reste du manuscrit. Le travail de la création de cette base a été publié dans [Essa11b].

Dans ce chapitre, nous présentons dans un premier temps le processus de la collecte des données de la base. Puis, nous décrivons la phase d'extraction des caractères isolés dans le formulaire utilisé, ainsi les opérations de prétraitements effectuées sur les images des caractères. Nous présentons ensuite le mode de stockage et d'étiquetage des images de la base en montrant sa structure. Enfin, nous exposons les caractéristiques et les détails statistiques de la base.

2. Collecte des données

Notre base de données de caractères amazighes isolés manuscrite a été recueillie auprès de 60 scripteurs de différents âges, différents sexes, différentes fonctions, différents niveau d'étude, différents moments et sur différents supports. Chaque scripteur nous a donné 13 × 33 caractères. Les échantillons ont été collectés en demandant aux participants d'écrire sur un formulaire 13 exemples pour chaque caractère amazighe. Nous avons recueilli 420 formulaires. La Figure 4.1 illustre un exemple de formulaire rempli que nous avons utilisé pour la collecte des données.

2.1. Modèle du formulaire utilisé

Pour éviter le problème complexe de la segmentation du document en caractères, nous avons choisi un formulaire d'une mise en page simple qui permet une segmentation relativement facile. Nous avons conçu un formulaire pré-imprimé constitué de sept pages. La première page contient un entête qui est composé des informations du scripteur telles que, l'âge, la profession, le sexe et la langue maternelle. Elle contient aussi les 5 premiers caractères amazighes à remplir. Les six autres pages contiennent les autres caractères amazighes à remplir. Chaque scripteur écrit chaque caractère 13 fois. La Figure 4.1 ci-dessous illustre un exemple d'un formulaire utilisé pour la collecte des données.

2.2. Numérisation des formulaires

Les documents collectés sont numérisés à l'aide du scanner HP-scan jet 5550c à 2000 ppp, ce qui est généralement un bruit faible et une image de bonne qualité. Les images numérisées sont stockées comme des images couleur en format JPG. La Figure 4.1 ci-dessus présente un exemple d'un formulaire numérisé.

3. Extraction de données et prétraitements

Après la numérisation des formulaires collectés. Nous avons développé un système automatique qui traite et segmente le document numérisé en caractères isolés. En effet, une

correction de pente de page du document a été réalisée en utilisant la transformée de Hough [Le94]. Cette méthode est utilisée pour estimer l'angle d'inclinaison et corriger l'inclinaison des images numérisées. Puis, nous avons développé une méthode avancée de projections horizontale / verticale pour l'extraction automatique des caractères isolés à partir du formulaire numérisé. Cette méthode localise les caractères dans les cellules du tableau en utilisant les techniques d'analyse d'histogramme de projections des pixels. Nous avons vérifié chaque exemple manuellement pour quelques erreurs de segmentation. Par conséquence, tous les caractères de la base de données étaient bien segmentés.

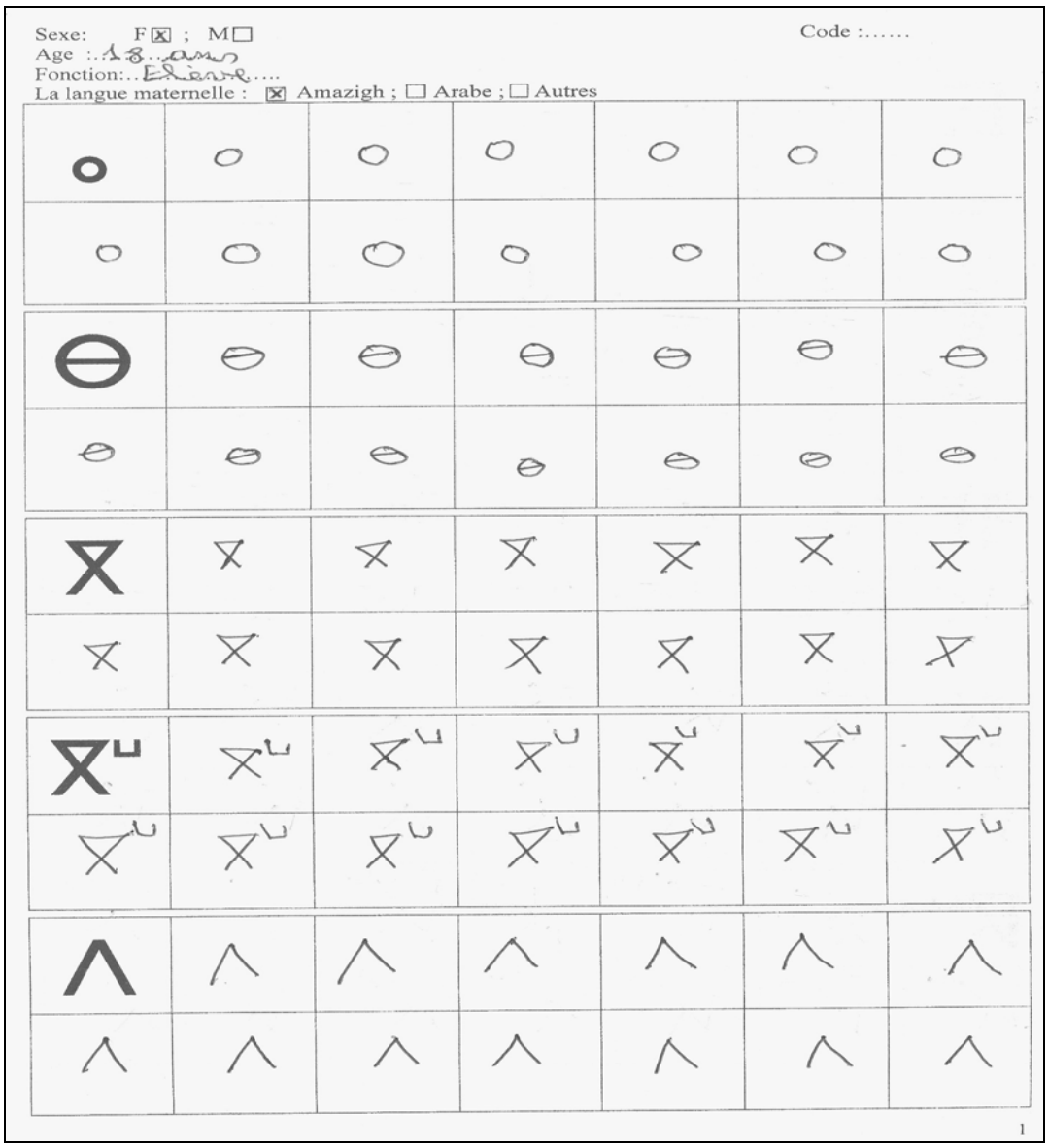


Figure 4.1 : Exemple du formulaire utilisé pour la collecte des données

Ensuite, une normalisation de l'image des caractères est appliquée pour éliminer les zones indésirables en utilisant les techniques de projections comme le montre la Figure 4.2 suivante.

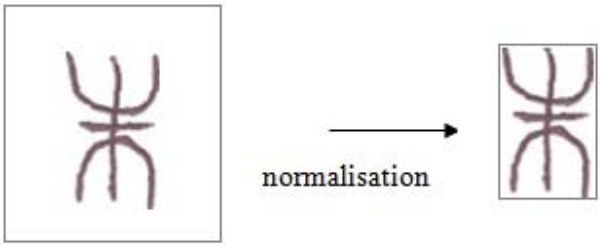


Figure 4.2 : Normalisation

L'une des caractéristiques de l'écriture amazighe est que tous les caractères s'écrivent comme des lettres majuscules sauf la lettre ya (ⴰ), qu'est un petit cercle et beaucoup plus petit que les autres caractères. De plus, le caractère ya (ⴰ) est très semblable au caractère yar (ⵔ), qui ne se différencient que par la taille. Dans certains cas, on aura une confusion réelle entre des images de ces deux caractères. Par conséquence, la normalisation précédente appliquée au caractère ya (ⴰ) va générer des problèmes. Pour surmonter ce problème, nous avons appliqué une normalisation particulière à cette lettre. En effet, nous avons ajouté une zone vide au sommet du caractère ya (ⴰ) après sa normalisation comme la montre la Figure 4.3 ci-dessous. La taille de cette zone ajoutée est égale à la taille du caractère ya (ⴰ) normalisée.

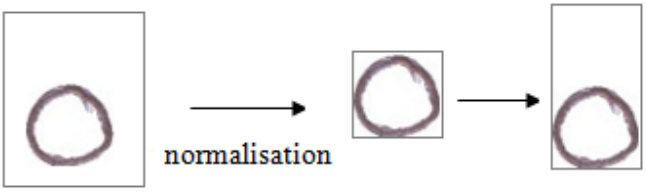


Figure 4.3 : La normalisation spécifiée au caractère ya (ⴰ)

4. Stockage et étiquetage des données

Après l'extraction et la normalisation du caractère, l'image extraite est enregistrée sous le format PNG, qu'est un format très pratique puisqu'il compresse l'image sans perte d'information. L'image du caractère est nommée en utilisant la structure suivante : la base

d'images contient une liste de répertoires. Chaque répertoire correspond à une lettre de l'alphabet. La Figure 4.4 ci-dessous montre la structure de quelques dossiers dans base.

Figure 4.4 : Structure des dossiers dans la base

Chaque répertoire est nommé par la prononciation de la lettre tifinagh qu'il contient. Par conséquent, chaque image nouvellement extraite devrait être ajouté à un de ces répertoires. Cette image est sauvegardée dans le dossier correspondant sous un nom explicite (PTL*_NS_NE.png)*, avec PTL est la prononciation de la lettre tifinagh ; NS est le numéro du scripteur ; NE est le numéro de l'échantillon écrit par le scripteur et png est l'extension du fichier.

5. Caractéristiques et statistiques de la base de données

Cette section décrit plus de détails concernant les statistiques utiles sur le nombre de caractères et de scripteurs. Pour cette base de données nous avons essayé de recueillir des données auprès des scripteurs avec les différents âges, différents sexes, différentes fonctions, et de différents milieux éducatifs. Environ 60% des scripteurs étaient des hommes et le reste étaient des femmes. Le Tableau 4.1 ci-dessous montre la répartition des scripteurs de la base de données AMHCD selon l'âge.

La répartition des scripteurs par âge était la suivante : 11% entre 6 - 12 ans, 22% entre 12-18, 38% entre 18-30, 22% entre 30-50 et 07% plus de 50 ans (cf. Tableau 4.1). Ainsi, la majorité des formulaires ont été remplis dans les établissements scolaires. De plus 60% des scripteurs ont la langue Amazighe comme langue maternelle. Enfin la base AMHCD est

recensée auprès de 60 scripteurs, chaque scripteur nous a donné 13 × 33 caractères. Au total, nous avons obtenu 780 modèles pour chaque caractère. Par conséquent, la base contient 25740 caractères (780× 33) amazighes manuscrits. Le Tableau 4.2 ci-dessous montre quelques exemples de caractères dans la base AMHCD collectés auprès de quatre scripteurs.

Age (ans)	**Pourcentage (%)**
Entre 06 et 12	11
Entre 12 et 18	22
Entre 18 et 30	38
Entre 30 et 50	22
Plus de 50	07

Tableau 4.1 : Répartition des scripteurs selon l'âge

6. Conclusion

Une première version de la base de données AMHCD a été présentée dans ce chapitre. Cette base de données contient plus de 25.000 caractères amazighes manuscrits isolés écrits par 60 scripteurs. Jusqu'à présent, l'AMHCD est la seule base de données qui contient les caractères amazighes manuscrits. Cette base de données a pour objectif de fournir les données d'apprentissage et de test pour l'expérimentation des approches sur la reconnaissance d'écriture amazighe afin de permettre d'obtenir des comparaisons objectives entre les différents systèmes. Une grande partie de cette base est utilisée pour expérimenter notre approche qui sera présentée dans le chapitre 6. Actuellement, nous faisons nos efforts pour élargir encore la base de données, en ajoutant d'autres échantillons de caractères extraits dans des documents amazighes.

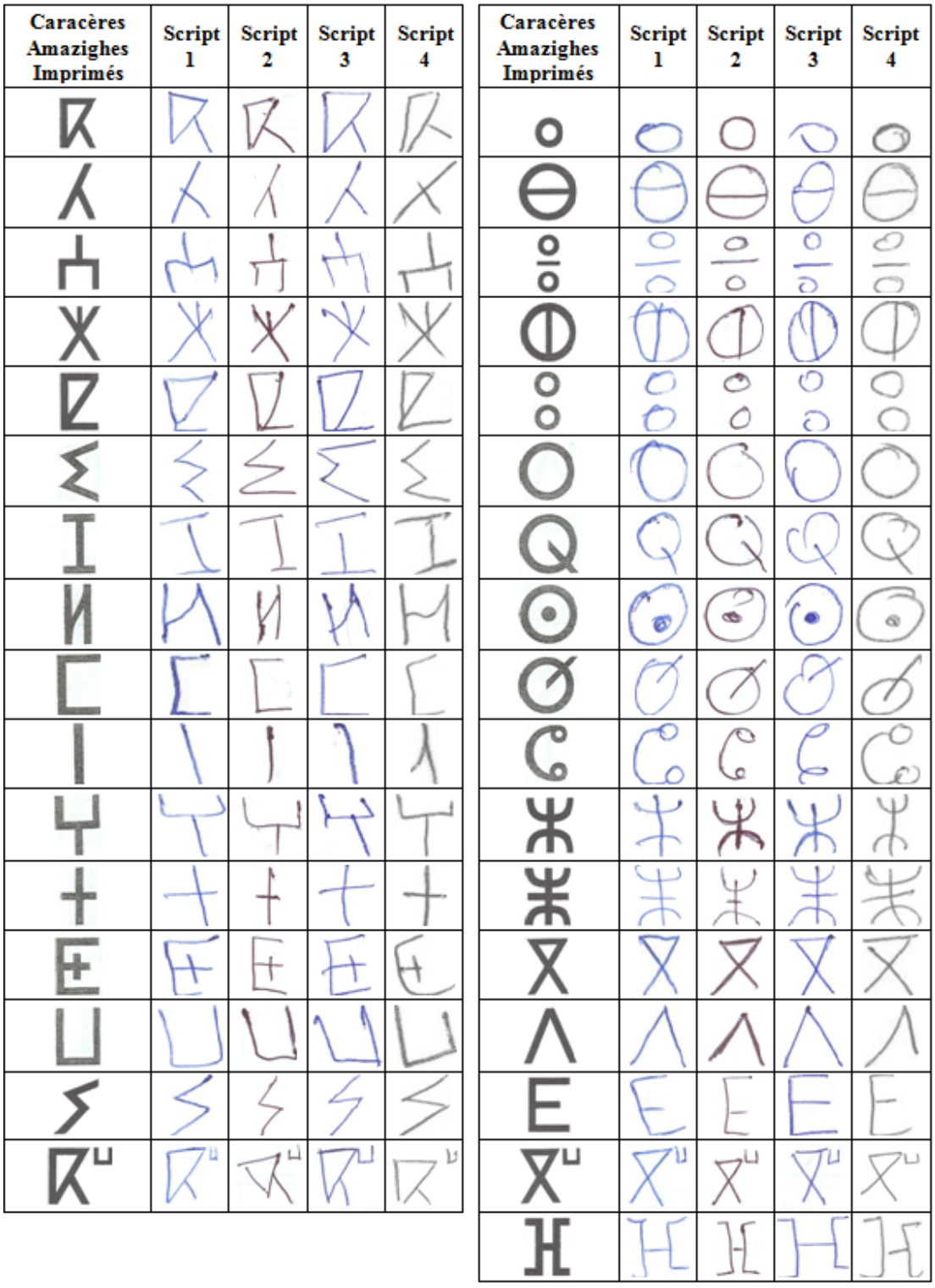


Tableau 4.2 : Certains échantillons de caractères amazighes manuscrits dans la base AMHCD

Chapitre 5 : Reconnaissance de caractères amazighes imprimés par les automates finis

1. Introduction

Dans ce chapitre, nous proposons un système de reconnaissance automatique de caractères amazighes imprimés et segmentés en utilisant une approche syntaxique basée sur les automates finis. Cette approche s'intéresse à la forme du caractère amazighe qui' est composée de primitives structurelles telles que des segments, des points et/ou des petits cercles. Nous décrivons dans un premier temps les prétraitements effectuées sur les caractères amazighes. Ensuite nous présentons la procédure de construction de la chaine représentative du caractère à partir de son squelette en utilisant le codage de Freeman. A partir de cette chaine construite, nous modélisons chaque caractère amazighe segmenté par une grammaire régulière et puis par un automate fini. Après, nous construisons l'automate globale qui reconnait tous les caractères amazighes étudiés à partir des automates spécifiques de chaque caractère. En fin nous présentons les résultats obtenus sur une base de caractères imprimés construite localement en proposons quelques perspectives du travail.

2. Prétraitements sur les caractères amazighes

Le prétraitement inclut toutes les fonctions effectuées avant l'extraction des primitives pour produire une version nettoyée de l'image d'origine. Ainsi, les prétraitements que nous avons appliqués sur les caractères amazighes imprimés comprennent : la binarisation, la réduction du bruit et la squelettisation.

2.1. Réduction du bruit

Après une binarisation simple effectuée sur le caractère, nous avons utilisé le lissage pour remplacer la valeur d'un pixel par la moyenne des valeurs des pixels entourant (et incluant) le pixel d'origine [Khar99a]. La Figure 5.1 ci-dessous présente l'image du caractère yaf (ⵅ) initiale et l'image obtenue après la binarisation et la réduction du bruit.

Figure 5.1 : Image d'un caractère avant et après la binarisation et la réduction du bruit

2.2. Squelettisation

Le but de la squelettisation est de simplifier l'image du caractère en une image plus facile à traiter en la réduisant à une dimension. Une fois la binarisation effectuée, on extrait le squelette de l'image. Dans ce travail, nous avons appliqué l'algorithme de Zhang-Suen [Zhan84] pour faire la squelettisation des caractères amazighes (cf. chapitre 1).

Après la squelettisation, un nombre de petits segments du squelette, appelés barbules, a été remarqué dans certaines caractères. Pour corriger la forme du squelette, nous avons appliqué un post-traitement qui consiste à nettoyer le squelette en éliminant les traits dont la taille est inférieure à un certain seuil [Khor03]. Nous avons aussi utilisé la méthode développée dans [Zhon99] pour améliorer la représentation des intersections dans le squelette de certains caractères. La Figure 5.2 ci-dessous présente quelques exemples de caractères amazighes et leurs squelettes obtenus par l'algorithme de Zhang-Suen [Zhan84], ainsi les résultats obtenus après les post-traitements ajoutés sur le squelette.

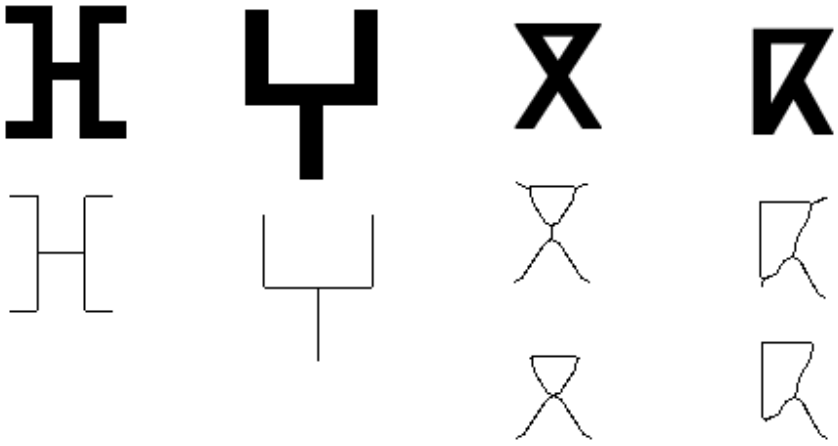


Figure 5.2 : Exemples de caractères amazighes imprimés et leurs squelettes obtenus, ainsi l'amélioration obtenue de quelques squelettes

3. Construction de la chaine de Freeman du caractère

Une fois le squelette obtenu, nous allons chercher à le décomposer en un ensemble de segments élémentaires qui seront la base de construction de la chaine représentative du caractère. Tout d'abord, trois types de points caractéristiques seront extraits du squelette du tracé [Elba06].

- Le point de fin de trait (extrémités) : c'est le pixel noir qui possède un seul voisin de même type ;
- Le point de croisement : c'est le pixel noir qui possède plus de trois points voisins de même type ;
- Le point d'inflexion : c'est le pixel noir qui possède deux voisins de même type.

La Figure 5.3 ci-dessous illustre les points caractéristiques extraits du squelette de quelques caractères amazighes imprimés. Les points caractéristiques détectés sont en noir sur le squelette.

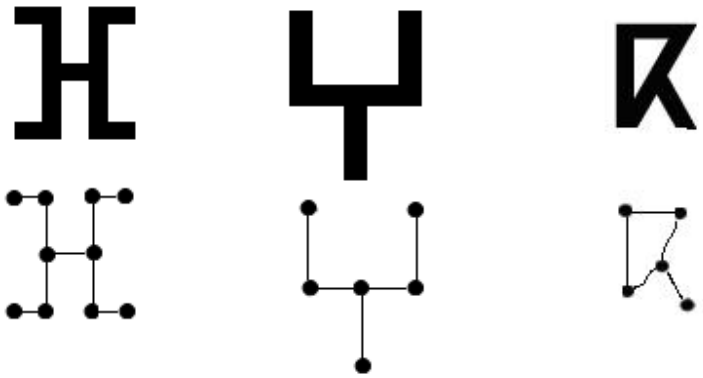


Figure 5.3 : Exemples de caractères amazighes imprimés, ainsi les points caractéristiques extraits dans leurs squelettes

Un algorithme de suivi de squelette permet de construire la chaine de Freeman d'un caractère. En effet, pour construire cette chaine, on recherche uniquement les "extrémités" et les "croisements" et on les relie en passant par les "inflexions". Pour cela, on analyse les 8 voisins de chaque pixel du squelette et on compte le nombre de transitions $0 \rightarrow 1$, selon le sens horaire (cf. Tableau 5.1).

a	b	c
h	**X**	d
g	f	e

Tableau 5.1 : Voisins de X (sens horaire de parcours) : a, b, c, d, e, f, g, h

- Si le nombre de transitions est 1, alors on est à une extrémité ;

0	1	0
0	X	0
0	0	0

Voisins de X (sens horaire) : 0, 1, 0, 0, 0, 0, 0, 0
1 transition

- Si le nombre de transitions est ≥3, alors on est à un croisement ;

0	1	0
0	X	1
1	0	0

Voisins de X (sens horaire) : 0, 1, 0, 1, 0, 0, 1, 0
3 transitions

- Si le nombre de transitions est 2, alors on est au milieu d'une continuité ou d'inflexion. Dans ce cas, on regarde la position des pixels de transition pour déterminer l'angle d'inflexion. Cet angle se calcule à partir de l'écart entre les positions des transitions $0 \rightarrow 1$ (cf. Tableau 5.2 ci-dessous).

1	0	0
0	X	0
0	1	0

Voisins de X (sens horaire) : 1, 0, 0, 0, 0, 1, 0, 0
2 transitions et les positions des transitions sont : 1 et 6, d'où l'écart entre les positions est : 5 (5 = 6-1)

Écart	**Angle**
1,7	45°
2,6	90° (angle droit)
3,5	135°
4	180° (angle plat)

Tableau 5.2 : Correspondance en angle

Le point de départ du parcours est la première extrémité, appartenant au squelette, qui est trouvée en effectuant un balayage de l'image de haut en bas et de gauche à droite. Le deuxième point du parcours est ensuite recherché dans le voisinage immédiat du premier.

Cette recherche débute selon une direction verticale, et se poursuit en éventail, selon des directions qui s'écartent progressivement de la verticale (cf. Figure 5.4 ci-dessous). La recherche orientée permet d'estimer, dans une certaine mesure, l'historique du tracé qui a été suivie lors du processus d'écriture du caractère amazighe.

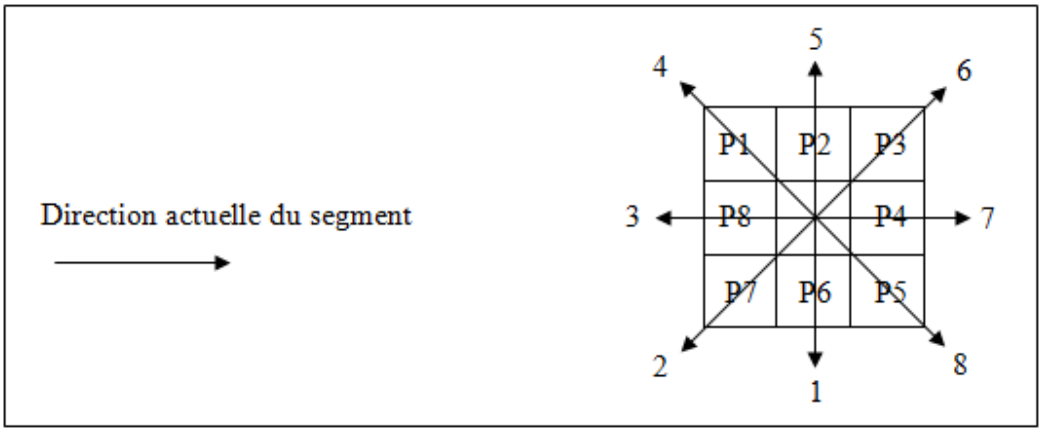


Figure 5.4 : Ordre préférentiel des directions de recherche de points adjacents

Dès qu'un pixel du squelette a été traité, une intensité de valeur nulle lui est assignée, de manière à éviter qu'il soit à nouveau pris en considération ultérieurement. La procédure de recherche continue jusqu'à ce que toutes les extrémités du squelette soient traitées. En plus, comme un pixel de croisement doit rester présent dans l'image jusqu'à ce que tous les segments qui y aboutissent aient été construits, l'intensité de ce pixel ne peut pas être mise à une valeur nulle dès qu'il aura été incorporé à l'un des segments.

Pendant le parcours, nous avons utilisé les huit directions de Freeman [Free74], illustrées sur la Figure 5.5 ci-dessous, pour coder le passage d'un point au suivant. En fait le chemin extrait sera conservé sous forme de liste de numéros associés au codage de Freeman.

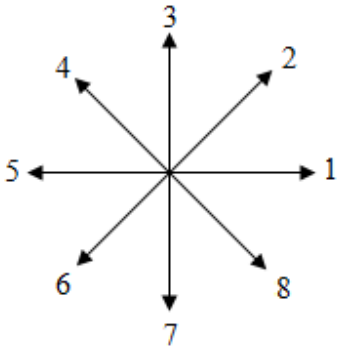


Figure 5.5 : Les 8 directions de Freeman

Un exemple d'illustration du processus de construction de la chaine représentative du caractère est présenté dans la Figure 5.6 ci-dessous. Cet exemple montre la procédure de construction de la chaine du caractère yaf (ⴼ), qui' est '1775131715331'.

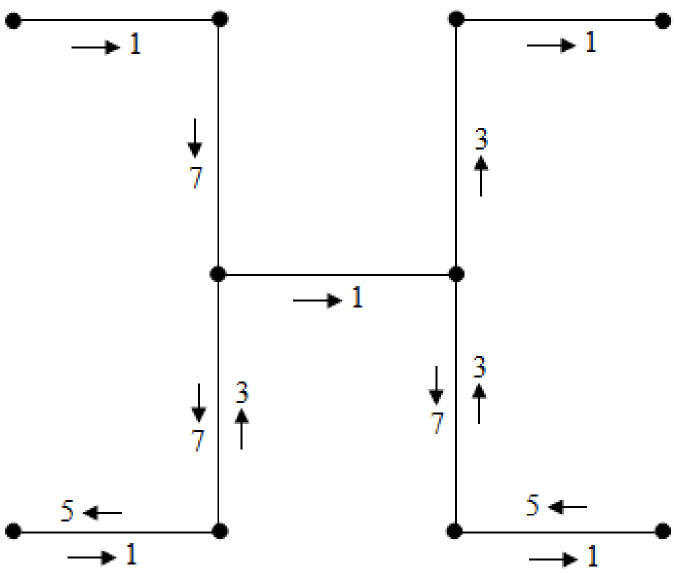


Figure 5.6 : Illustration de l'algorithme de construction de la chaine du caractère yaf (ⴼ)

Pour un autre exemple, le code relatif de la chaîne de Freeman du caractère yak (ⴽ) serait : '46316' et celui du caractère yagh (ⵖ) serait : '717313', (les caractères illustrés dans la Figure 5.3 ci-dessus).

Chaque caractère amazighe est représenté, dans cette étape, par une chaine de Freeman. Nous allons maintenant étudier dans la section suivante comment exploiter ces primitives extraites pour la reconnaissance de caractères en utilisant les automates finis.

4. Modélisation des caractères amazighes imprimés : Automates finis

4.1. Vocabulaire retenu

La grammaire doit permettre de décrire un caractère et de définir un contexte utilisable dans le processus de reconnaissance. Pour ce faire, les éléments du vocabulaire terminal doivent être les entités de base de la description du caractère. D'autre part le codage de Freeman permet de coder un contour chaîné en ne stockant que l'ensemble des déplacements relatifs. Pour cela, nous avons retenues les huit directions de Freeman pour coder tous les caractères amazighes segmentés (ⵊ, ⵏ, ⵜ, ⵖ, ⵀ, ⴼ, ⴽ, ⵍ, ⵇ, ⵎ, ⵓ, ⵛ, ⵞ, ⴷ, ⴹ, ⴳ, ⴳⵯ, ⵢ, ⵉ, ⵣ, ⵥ). Ce codage nous a permet de retenir l'ensemble {1, 2, 3, 4, 5, 6, 7, 8} comme vocabulaire

terminal des grammaires régulières, qui prmettent de modéliser les caractères amazighes imprimés et segmentés.

4.2. Représentation des caractères par les grammaires régulières

A partir de la chaine représentative du caractère on déduit la grammaire régulière qui représente ce caractère. Par exemple, la chaine relative au caractère yaf (ⵀ) est '1775131715331'. Donc le caractère yaf (ⵀ) se décrit par la grammaire suivante :

$G = (X, V, S, P)$ où : $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$ est le vocabulaire terminal et $V = \{A, B, C, D, E, F, G, H, I, J\}$ est le vocabulaire non terminal. Les règles de production se décrivent dans la Figure 5.7 ci-dessous :

P : S → 1A
A → 7B|7A
B → 5C
C → 1D
D → 3E
E → 1F
F → 7G
G → 1H
H → 5I
I → 3J|3I
J → 1

Figure 5.7 : Règles de production de la grammaire régulière qui représente la lettre yaf (ⵀ)

Pour le caractère yagh (ⵖ), sa représentation par la grammaire régulière serait : $G = (X, V, S, P)$ où : $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $V = \{A, B, C, D, E\}$. Les règles de production se décrivent dans la Figure 5.8 ci-dessous :

P : S → 7A
A → 1B
B → 7C
C → 3D
D → 1E
E → 3

Figure 5.8 : Règles de production de la grammaire régulière qui représente la lettre yagh (ⵖ)

4.3. Représentation des caractères par les automates finis

Une fois que tous les caractères sont représentés par des grammaires régulières, on peut les représenter par des automates finis en utilisant les règles de correspondance entre les grammaires régulières et les automates finis vues dans le chapitre 2. A titre d'exemple, pour les grammaires précédentes, nous obtenons les automates finis de la Figure 5.9 et la Figure 5.10 ci-dessous.

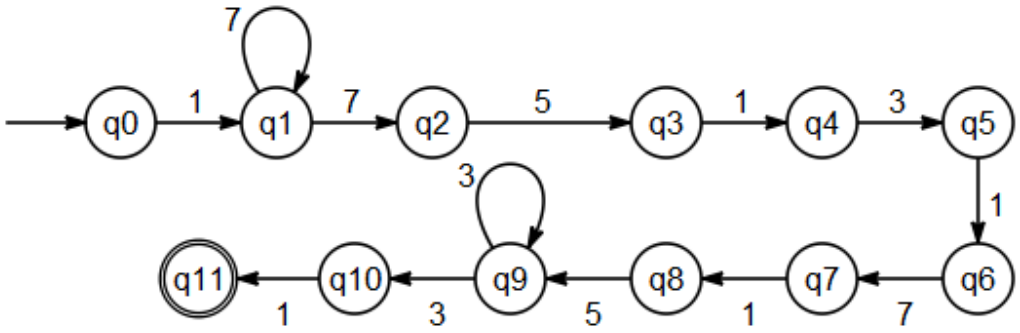


Figure 5.9 : L'automate fini qui reconnaît le caractère yaf (ﻉ)

Figure 5.10 : L'automate fini qui reconnaît le caractère yagh (ﻍ)

5. Apprentissage et reconnaissance par un automate à état fini

La phase de reconnaissance consiste à interpréter les données obtenues dans les étapes précédentes et à décider à quelle classe appartient le caractère analysé. Cette décision d'appartenance du caractère à une classe va être effectuée par un automate fini chargé de faire la mise en adéquation des propriétés de l'objet aux propriétés de la classe. Cet automate fini sera obtenu à la suite d'une phase d'apprentissage.

Le problème classique de l'apprentissage d'automates à partir d'un modèle consiste à retrouver un automate à partir d'un échantillon d'apprentissage (que nous noterons *I*). Cet échantillon est composé d'exemples de modèle appartenant au langage de l'automate à inférer (ensemble des exemples positifs, noté *I*+) et éventuellement de modèle n'appartenant pas au langage (ensemble des exemples négatifs ou contre-exemples, noté *I*). Une des approches les plus utilisées pour l'apprentissage d'automates est celle dite par fusions d'états. Elle consiste à construire l'ACM (Automate Canonique Maximal)

représentant exactement les exemples de l'échantillon d'apprentissage, et à généraliser cet automate par une succession de fusions d'états [Dupo98]. Des techniques d'inférence grammaticale peuvent être utilisées pour construire automatiquement un automate à partir d'exemples, mais ces méthodes peuvent échouer dans les cas les plus généraux [Olsz01]. Par conséquent, les systèmes existants de reconnaissance syntaxique de formes sont essentiellement appliqués à des domaines où les grammaires syntaxiques requises pour la classification peuvent être construites à la main.

Dans ce travail, nous avons construit un automate canonique maximal à partir des automates spécifiques de chacun des caractères imprimés. Cet automate reconnaît les caractères amazighes (ⵊ, ⵏ, ⵜ, ⵖ, ⵃ, ⵌ, ⴽ, ⵍ, ⵇ, ⵎ, ⵡ, ⵄ, ⵟ, ⴷ, ⴹ, ⵅ, ⵆ, ⵙ, ⵢ, ⵣ, ⵥ).

Soit L le langage régulier qui reconnaît tous les caractères amazighes imprimés et segmentés. Etant donné un échantillon positif I_+ du langage L. Le plus grand automate pour l'échantillon d'apprentissage I_+, sous l'hypothèse de complétude structurelle [Dupo98], s'appelle l'automate canonique maximal (ACM (I_+)). Il est construit en réalisant l'union de l'ensemble des automates acceptant chacun un mot de l'échantillon d'apprentissage. Cet automate réalise un apprentissage par cœur de l'échantillon d'apprentissage. Nous avons construit cet automate à la main.

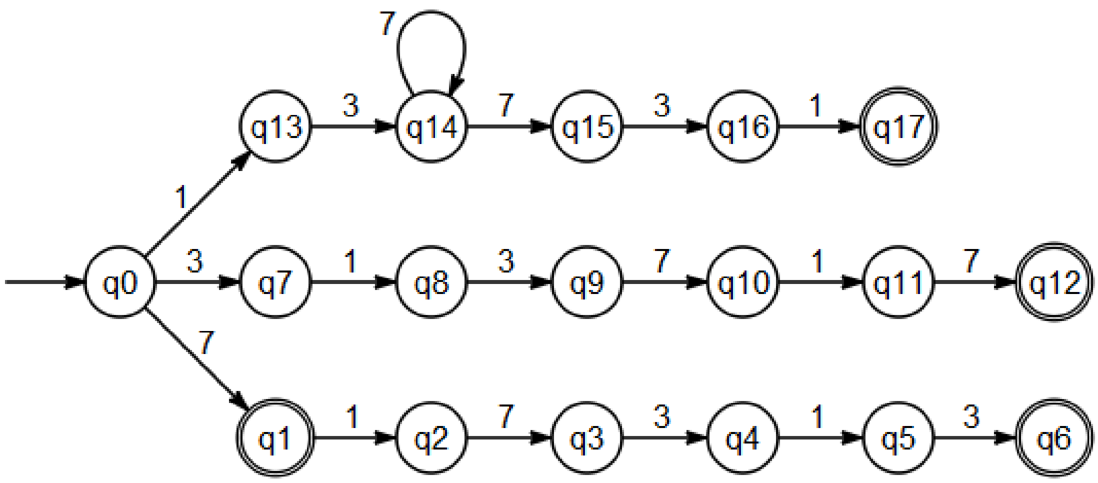


Figure 5.11 : ACM relatif à l'échantillon I+ qui représente les caractères (ⵖ, ⵃ, ⵜ, ⵏ)

A titre d'exemple, l'automate représenté à la Figure 5.11 est l'automate maximal canonique relatif à l'échantillon I_+ qui représente les quatre caractères amazighes (ⵖ, ⵃ, ⵜ, ⵏ).

6. Expérimentations et résultats

6.1. Base de donnée utilisée

Pour pouvoir expérimenter notre approche, nous avons construit une base de caractères amazighes imprimés isolés, de différentes tailles, et sous forme d'images brutes. Elle est composée de 630 caractères (30 caractères pour chaque classe). Un aperçu de certains de ces caractères est présenté dans la Figure 5.12 ci-dessous.

Figure 5.12 : Exemple de modèles de quelques caractères amazighes imprimés de la base

6.2. Tests et résultats

Pour valider notre approche, et dans un premier temps, nous l'avons testé sur les caractères amazighes imprimés (ⵊ, ⵏ, ⵜ, ⵖ, ⵀ, ⵃ, ⴽ, ⵍ, ⵇ, ⵛ, ⵎ, ⴻ, ⵟ, ⴷ, ⴹ, ⴳ, ⵅ, ⵉ, ⵄ, ⵣ, ⵥ) extraits de la base de caractères, déjà décrite précédemment. Ces caractères portent des caractéristiques tenues en compte par notre approche. Nous avons obtenu des résultats encourageants. En effet, sur les 630 caractères lus, 589 ont été reconnus, soit un taux de reconnaissance de 93,49%. Le Tableau 5.3 ci-dessous présente les taux des mauvaises affectations et de mauvais rejets. Ces erreurs proviennent de la forme de certains caractères non reconnus dont le squelette comporte plus des segments non orthogonaux. En effet, la méthode de reconnaissance est basée sur une vectorisation du squelette du caractère à reconnaître. Donc une erreur de vectorisation va forcément entraîner une erreur dans la description du caractère. En fait, le principal inconvénient de cette méthode est la sensibilité du squelette au bruit.

La complexité algorithmique de notre approche est polynomiale. En effet, la chaine de codage d'un caractère amazighe est composée au maximum de 13 symboles. Chaque

chiffre doit être composé à chaque arc de l'automate global qui reconnait l'ensemble de vingt et un caractères amazighes segmentés. D'où la complexité est de l'ordre (13^{21}).

Taux d'affectations à tort	2,28 %
Taux de rejets à tort	4,23 %

Tableau 5.3 : Pourcentages des erreurs de reconnaissance

7. Conclusion et perspectives

Dans ce chapitre, nous avons présenté une approche pour la reconnaissance de caractères amazighes imprimés, en utilisant une approche syntaxique basée sur les automates finis. Après les prétraitements, des algorithmes appropriés permettent de construire la chaîne du codage de Freeman représentant le caractère en entrée. La chaîne est utilisée dans l'entrée de l'automate maximal canonique, qui reconnaît tous les caractères amazighes segmentés pour décider la classe d'appartenance du caractère. Cet automate est construit à partir des automates spécifiques de chacun de caractères amazighes imprimés. Sur une base de données de caractères amazighes imprimés isolés, les résultats expérimentaux montrent la robustesse de l'approche. Les résultats de ce travail ont été publiés dans le journal international ICGST-GVIP [Essa10]. De nombreuses perspectives sont ouvertes sur différents aspects de ce travail, aussi bien au niveau de l'étape d'apprentissage que de l'étape d'extraction de caractéristiques. Tout d'abord, nous pensons qu'il est nécessaire de construire un réseau d'automates qui reconnaît l'ensemble de l'alphabet tifinagh en intégrant d'autres vocabulaires qui permettra de coder les caractères amazighes circulaires.

En outre, dans ce travail nous avons utilisé le squelette du caractère, mais on peut se baser sur la détection de contours : elle se base sur l'analyse des contours des caractères pour extraire des chaines du codage du Freeman. De plus, on pourra exploiter la technique des contours actifs qui consiste à modéliser un contour par une courbe fermée. Cette technique a été utilisée dans différents domaines, notamment en traitement d'images et vision par ordinateur. Autrement, dans ce travail, nous avons utilisé des grammaires et les automates finis simples. En perspectives, l'ajout des probabilités affectées aux règles de production avec une grammaire contenant des valeurs statistiques attachées aux caractéristiques peut être mis en ouvre. Cette modélisation permet d'avoir des grammaires et des automates finis stochastiques. Enfin, nous comptons d'appliquer des algorithmes d'apprentissage automatique pour générer l'automate final.

Dans le chapitre suivant, nous allons remédier à l'une des limites de cette méthode syntaxique en proposant une nouvelle approche qui tient compte de tous les caractères amazighs.

Chapitre 6 : Reconnaissance de l'écriture amazighe à base des lignes centrales du caractère

1. Introduction

Nous avons mis en œuvre dans le chapitre précédent un système automatique de reconnaissance de caractères amazighes imprimés isolés, basé sur une approche syntaxique en utilisant les automates à états finis. La limite de cette approche est qu'elle ne traite pas les caractères circulaires. De plus, les caractères amazighes manuscrits ne peuvent être pris en compte par cette approche. Pour remédier à ces limites, nous proposons dans ce chapitre une nouvelle approche qui tient compte de tous les caractères amazighes. En effet, dans la phase d'extraction des primitives, notre approche est basée sur la position des lignes centrales du caractère. Ces primitives alimenteront un réseau de neurones multicouches dans les phases d'apprentissage et de reconnaissance.

Dans la littérature, différentes approches basées sur les positions des lignes d'écriture ont été proposées pour les écritures des autres langues. Dans le cas de l'écriture latine et arabe, plusieurs positions de lignes de base on été utilisées pour extraire des caractéristiques qui dépendent de ces lignes ([Elha05], [Shan07], [Alsh08], [Aida09], [Razz10]). La Figure 1.2 ci-dessus illustre des exemples de lignes de base utilisées pour ces écritures [Elha05].

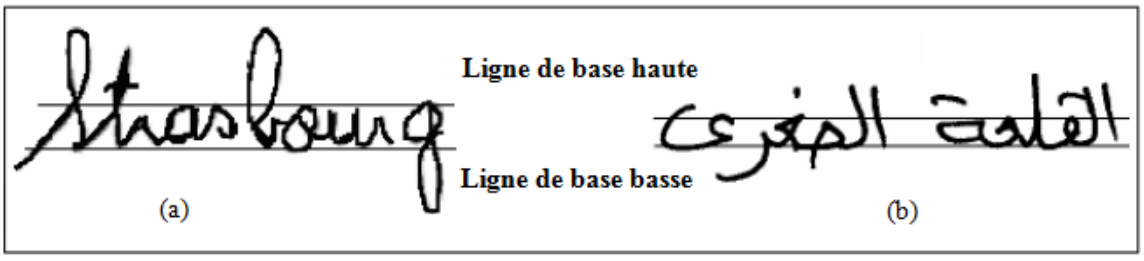


Figure 6.1 : Exemple de lignes de base d'écriture. (a) cas de l'écriture latine, (b) cas de l'écriture arabe [Elha05]

Pour l'écriture amazighe, on propose d'utiliser une ligne centrale, ligne supérieure et inférieure de l'écriture pour dériver un ensemble de caractéristiques dépendantes et indépendantes à ces lignes. La Figure 6.2 ci-dessous montre les positions des lignes d'écriture sur quelques caractères amazighes. Les caractéristiques extraites sont de types

statistiques extraits au niveau pixels en se basant sur l'utilisation de la technique des fenêtres glissantes.

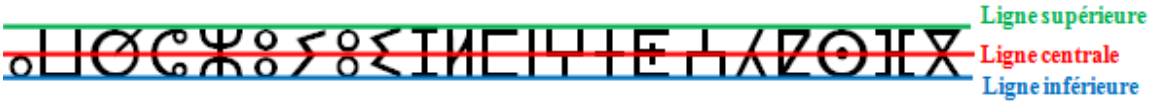


Figure 6.2 : Les positions des lignes d'écriture sur quelques caractères amazighes

Dans notre système et dans un premier temps, on utilise la ligne centrale horizontale du caractère pour extraire un ensemble de caractéristiques de densités basées sur cette ligne. Nous montrons les résultats de reconnaissance obtenus en fonction de l'intégration des caractéristiques dépendantes et indépendantes à la ligne centrale horizontale du caractère. Part la suite, on compare ces résultats avec celles obtenues en ajoutant des caractéristiques basées sur la ligne centrale verticale du caractère.

Dans ce qui suit, nous décrirons notre approche. Nous commençons par une description générale sur la structure du système proposé. Puis, nous décrivons les deux premières étapes du système : l'étape des prétraitements comprenant la segmentation du texte en lignes et puis en caractères isolés, et l'étape d'extraction des caractéristiques. Dans cette dernière étape, nous utilisons les caractéristiques dépendantes et indépendantes de la ligne centrale horizontale. Ensuite, nous présentons le processus d'apprentissage et de reconnaissance en utilisant un réseau de neurones multicouches. Juste après, nous donnons les résultats des expérimentations menées sur les deux bases utilisées : une base des patterns de la graphie amazighe et une autre de caractères amazighes manuscrits. Nous proposons, ensuite, une amélioration du système de reconnaissance en ajoutant des caractéristiques basées sur la ligne centrale verticale du caractère. Enfin, nous présentons les résultats obtenus par le système amélioré en les comparants avec ceux obtenus par le système de base.

2. Description générale du système proposé

Comme nous l'avons présenté dans le chapitre 1, un système de reconnaissance hors ligne de l'écriture manuscrite comprend en générale trois étapes : les prétraitements, l'extraction de caractéristiques et la classification. Le processus de reconnaissance d'écriture implique l'extraction de certaines caractéristiques définies permettant de classer un caractère inconnu dans l'une des classes connues. Les prétraitements sont principalement utilisés pour réduire les variations de l'écriture manuscrite, de corriger l'inclinaison des lignes de texte et de segmenter le texte en caractères isolés. L'étape de l'extraction de caractéristiques est essentielle pour la représentation efficace des données et l'extraction de primitives

significatives pour un traitement ultérieur. La classification se faite par un réseau de neurones multicouches, en utilisant l'algorithme de rétropropagation d'erreur. L'architecture générale de notre système est illustrée dans la Figure 6.3 ci-dessous.

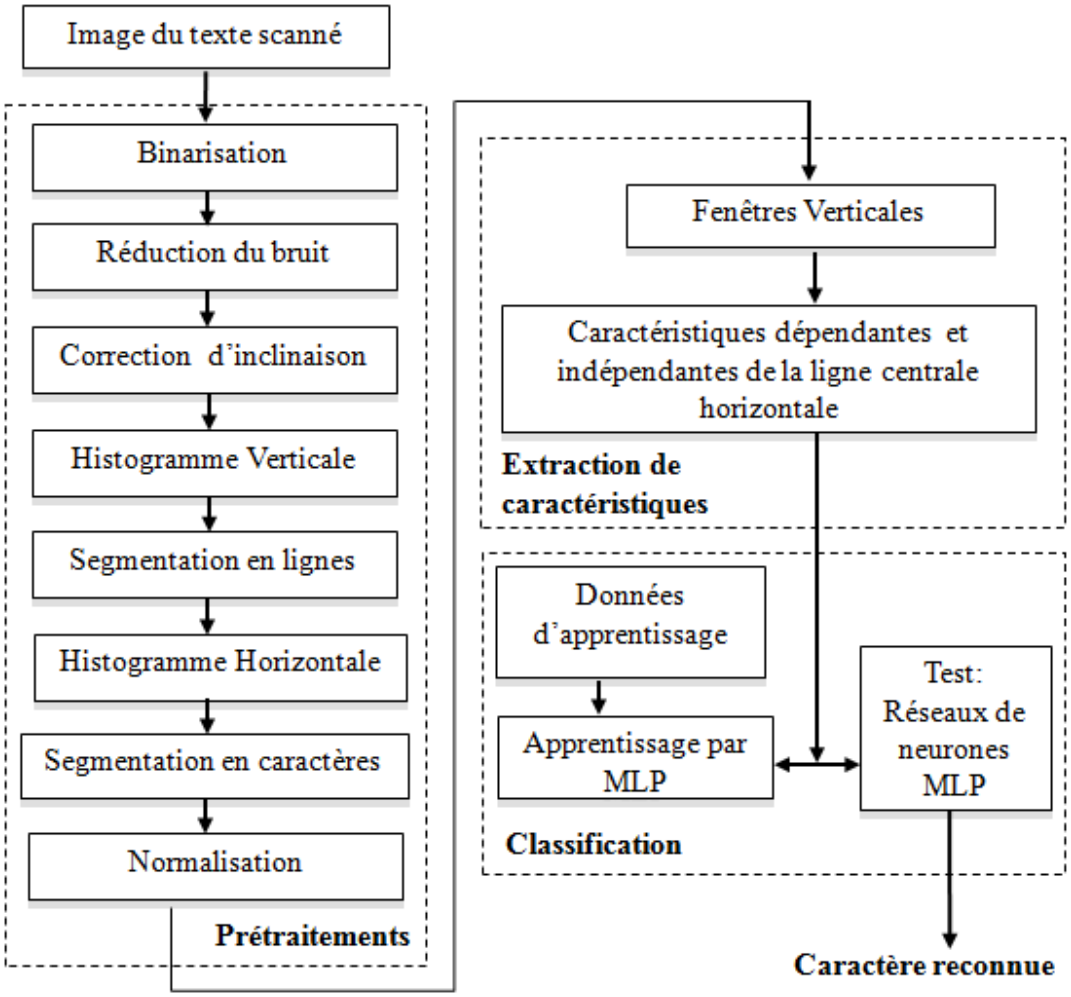


Figure 6.3 : Architecture générale du système

3. Prétraitements

Une fois l'image est numérisée, une série de prétraitements est appliquée. Ces prétraitements préparent l'image d'entrée afin de faciliter l'étape d'extraction des caractéristiques. Il s'agit essentiellement de réduire le bruit superposé aux textes et essayer de ne garder que l'information significative de la forme représentée. Nous avons utilisé le seuillage, la réduction du bruit, la détection et la correction d'inclinaison des lignes de texte

du document, la segmentation en lignes puis en caractères, et enfin la normalisation en taille.

3.1. Binarisation

La séparation Avant/Arrière plan est réalisée avec une binarisation. Il s'agit de passer d'une image en niveau de gris ou en couleurs à une image bitonale (composée de deux valeurs 0 et 1, noir et blanc) en se basant sur un seuil global. Nous avons utilisé la méthode d'Otsu pour faire la binarisation [Ostu79]. La Figure 6.4 ci-dessous présente le résultat obtenu avec la méthode d'Otsu. Cette méthode effectue une analyse statistique sur les histogrammes (variance intra-classe et variance inter-classe) pour définir une fonction à maximiser qui permet d'estimer le seuil de binarisation.

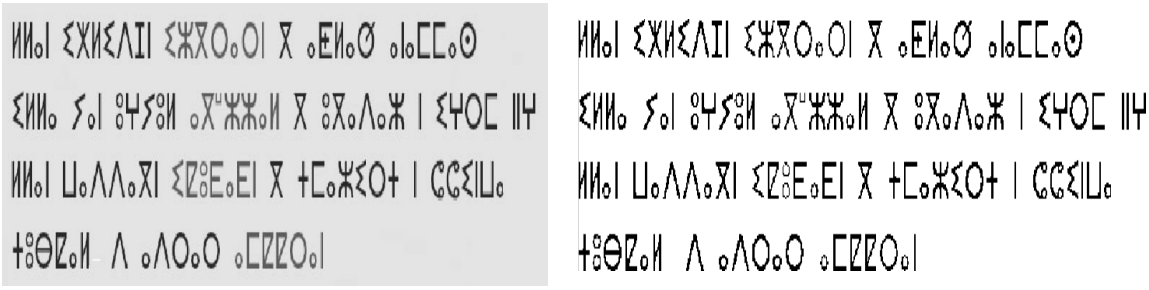


Figure 6.4 : l'image du texte et sa binarisation avec la méthode d'Otsu

3.2. Réduction du bruit

La réduction du bruit consiste à détecter et à éliminer les pixels qui représentent des bruits. Dans le domaine de reconnaissance de l'écriture, plusieurs méthodes ont été utilisées pour éliminer le bruit. Dans ce travail, nous avons utilisé une technique simple du lissage qui consiste à remplacer la valeur d'un pixel par la moyenne des valeurs des pixels entourant (et incluant) le pixel d'origine [Khar99a].

3.3. Détection et correction d'inclinaison des lignes de texte

Les méthodes de correction d'inclinaison des lignes de texte (également appelée correction de "skew") sont utilisées pour redresser horizontalement les lignes d'écriture obliques. A cet effet, deux étapes sont appliquées. Premièrement, l'angle d'inclinaison est estimé, deuxièmement, l'image d'entrée est tournée par l'angle estimé. Plusieurs méthodes sont disponibles pour la correction d'inclinaison des lignes de texte. Les deux plus populaires sont la projection des profils [Pavl04] et la transformée de Hough [Le94]. Dans ce travail, nous utilisons la transformée de Hough pour estimer la ligne de plus grande pente et par la suite la direction saillante de l'écriture. Une fois l'angle d'inclinaison θs est estimé, l'image

est pivotée par l'angle θs dans la direction opposée pour faire la correction. La Figure 6.5 ci-dessous montre une page de texte contenant une écriture inclinée ainsi que la même page après la correction de l'inclinaison.

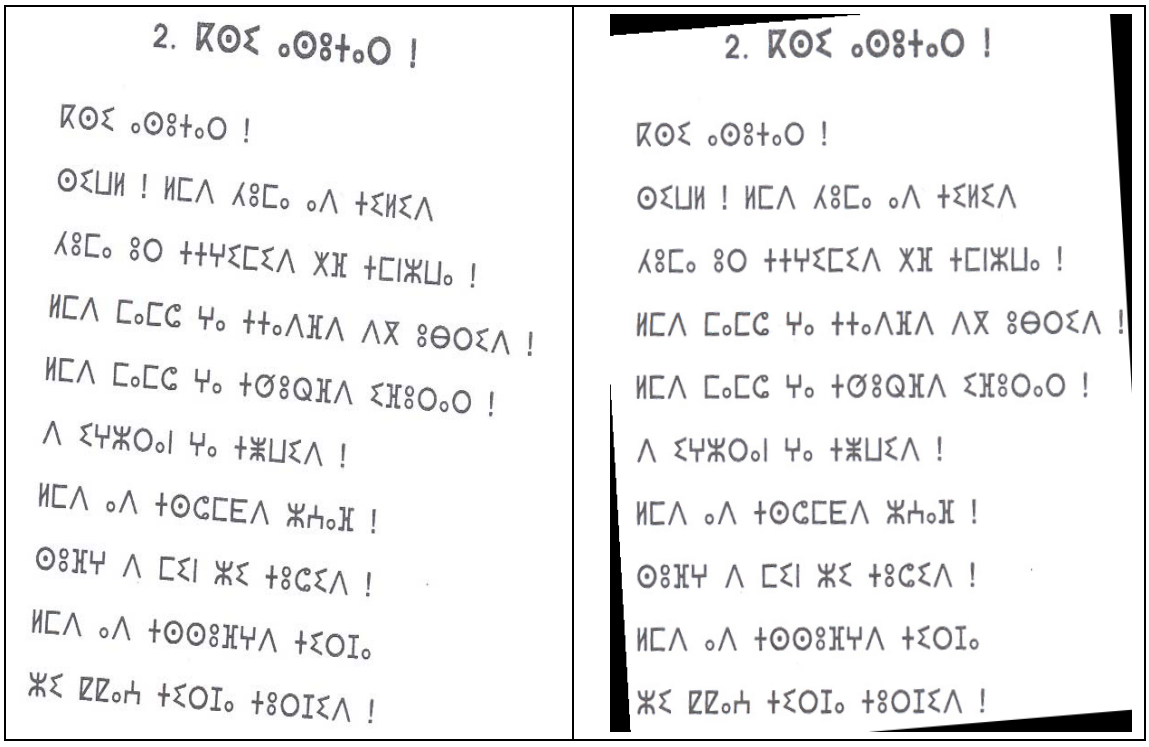


Figure 6.5 : Correction de l'inclinaison des lignes dans une page de texte contenant une écriture amazighe inclinée

3.4. Segmentation du texte en lignes

Une fois l'image du texte est nettoyée, le texte sera segmenté en lignes. Nous avons utilisé les techniques d'analyse d'histogramme de projections horizontales des pixels de manière à distinguer les régions de forte densité (les lignes) des régions de faible densité (les espaces interlignes) (cf. Figure 6.6). Ces techniques ont été utilisées souvent pour extraire des lignes dans les textes imprimés, qui ne présentent pas autant de variabilité au niveau de la disposition spatiale des entités connexes comme le cas de l'écriture amazighe imprimée.

3.5. Segmentation en caractères

L'écriture amazighe n'est pas cursive. Cela facilite l'opération de segmentation d'une ligne de texte en caractères. Nous avons utilisé les techniques d'analyse d'histogramme de projections verticales pour segmenter chaque ligne de texte en caractères. La Figure 6.7 ci-

dessous présente une ligne de texte, son histogramme vertical et le résultat obtenu de la segmentation en caractères.

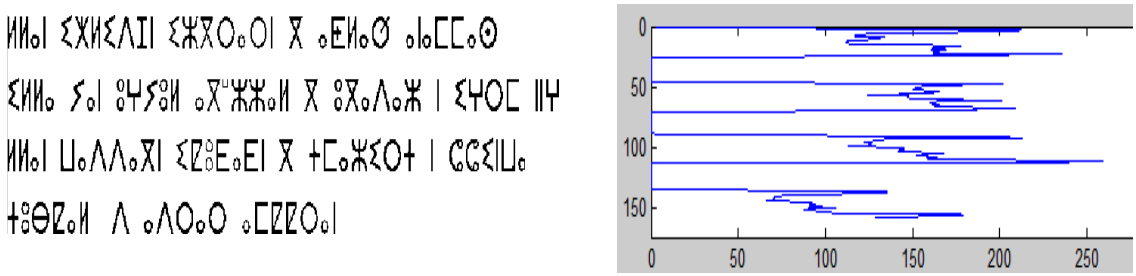


Figure 6.6 : Histogramme de projections horizontales

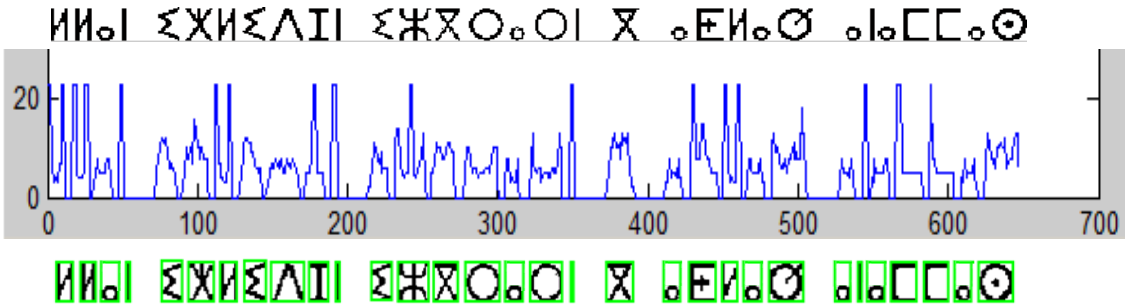


Figure 6.7 : Histogramme de projections verticales d'une ligne de texte et le résultat de la segmentation en caractères

3.6. Normalisation de la taille

La taille d'un caractère peut varier d'une écriture à l'autre, ce qui peut causer une instabilité des paramètres. Une technique naturelle de prétraitement consiste à ramener les caractères à la même taille. De plus, notre système oblige que les images des caractères en entrée soient d'une taille normalisée. Pour cela, nous avons normalisé ces caractères en une taille moyenne 60 × 50. C'est la taille moyenne des images de caractères dans les bases de données utilisées. Ces images de taille normalisée et en format prétraitée seront directement soumises en entrée au module d'extraction des caractéristiques.

4. Extraction des caractéristiques

Après les prétraitements, une méthode d'extraction de caractéristiques est appliquée pour extraire les primitives les plus pertinentes du caractère à reconnaître. Dans ce travail, les caractéristiques de densités des pixels d'écriture extraites se basent sur la position de la ligne de base, qui est la ligne centrale horizontale du caractère [Elha05].

L'étape d'extraction des caractéristiques précède d'une étape de prétraitement qui permet d'extraire la ligne de base du caractère. Cette étape permet de séparer l'image du caractère en 2 zones : une zone supérieure qui correspond à la zone en dessus de la ligne centrale, et une zone inférieure qui correspond à la zone en dessous de la ligne centrale (cf. Figure 6.2).

L'image du caractère est ensuite balayée de gauche à droite et de haut en bas par une fenêtre glissante qui s'adapte en hauteur à celle du caractère (cf. Figure 6.8). La hauteur et la largeur des fenêtres sont constantes et sont considérés comme des paramètres du système [Elha05].

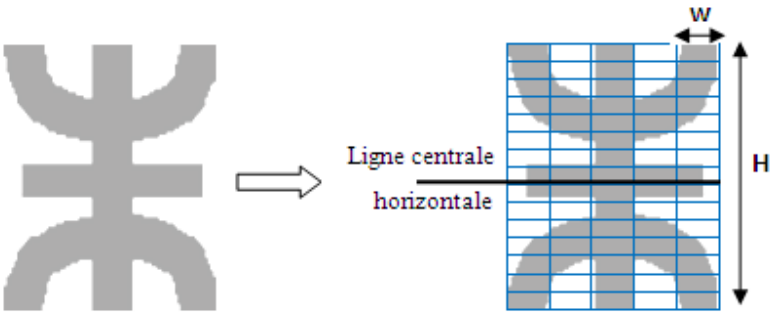


Figure 6.8 : L'image du caractère est divisée en fenêtres verticales

Dans chaque fenêtre, on génère un ensemble de 19 caractéristiques. Celles ci sont représentatives de densités des pixels d'écriture. Pour cela, chaque fenêtre est divisée en un nombre de cellules fixe.

Supposons que H est la hauteur en pixels de la fenêtre dans chaque image, h est la hauteur de chaque cellule et w est la largeur de chaque fenêtre. La fenêtre étant divisée verticalement en n_c cellules, donc n_c=H/h. Soit :

- n(i) : Le nombre de pixels d'écriture (pixels noirs) dans la cellule i ;
- r(j) : Le nombre de pixels d'écriture dans la $j^{ème}$ rangée de pixels dans une fenêtre verticale (une fenêtre contient H rangées de pixels) ;
- b(i) : Le niveau d'intensité de la cellule i : b(i)=0 si n(i)=0, b(i)=1 sinon.

Pour chaque fenêtre, 19 caractéristiques sont extraites. Un sous ensemble de ces caractéristiques est lié à la position de la ligne de centrale horizontale pour prendre en compte la similarité de la majorité des caractères amazighes par rapport à cette ligne.

4.1. Les caractéristiques indépendantes de la ligne centrale

Pour chaque fenêtre, 13 caractéristiques de densités indépendantes de la ligne centrale sont extraites et sont les suivantes:

f_1 : densité des pixels noirs dans la fenêtre $f_1 = \frac{1}{H \times w} \sum_{i=1}^{n_c} n(i)$

f_2 : nombre de transitions Noir/Blanc entre cellules $f_2 = \sum_{i=2}^{n_c} \left| b(i) - b(i-1) \right|$

f_3 : différence de position entre les centres de gravité g des pixels d'écriture dans deux fenêtres consécutives (l'indice *t* est omis) : $f_3 = g(t) - g(t-1)$

Où la position g est calculée comme suit : $g = \frac{\sum_{j=1}^{H} j.r(j)}{\sum_{j=1}^{H} r(j)}$

f_4 à f_{13} : sont les densités de pixels d'écriture dans chaque colonne de la fenêtre.

4.2. Les caractéristiques dépendantes de la ligne centrale

Soit LH la position (ordonnée y) de la ligne de base, qui est la ligne centrale horizontale du caractère. Les caractéristiques suivantes dépendent de la position de la ligne centrale horizontale.

f_{14} : position verticale normalisée du centre de gravité des pixels d'écriture, par rapport à la ligne centrale.

$f_{14} = \frac{g - LH}{H}$ Avec *LH* est la position de la ligne centrale horizontale.

f_{15}, f_{16} : deux primitives qui représentent les densités des pixels d'écriture au dessus et au dessous de la ligne centrale.

$$f_{15} = \frac{1}{H \times w} \sum_{j=LH+1}^{H} r(j) \qquad f_{16} = \frac{1}{H \times w} \sum_{j=1}^{LH-1} r(j)$$

f_{17}, f_{18} : nombre de transitions Noir/Blanc entre les cellules situées au dessus et au dessous de la ligne centrale.

$$f_{17} = \sum_{i=2}^{k} \left| b(i) - b(i-1) \right| \qquad f_{18} = \sum_{i=k}^{n_c} \left| b(i) - b(i-1) \right|$$

où k est la cellule contenant la ligne centrale.

f_{19} : densité des pixels noirs dans la ligne centrale. Cette caractéristique a été ajoutée pour distinguer entre certains caractères amazighes qui se ressemblent entres eux et qui se diffèrent seulement par un point ou par un trait sur la ligne centrale horizontale du caractère comme le montre la Figure 6.9 ci-dessous .

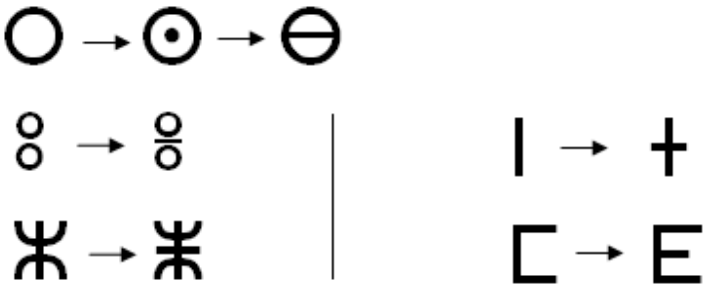


Figure 6.9 : Certains caractères amazighes qui se ressemblent entrent eux

L'ensemble des 19 caractéristiques extraites comporte 6 caractéristiques qui dépendent de la position de la ligne centrale horizontale, et 13 qui n'en dépendent pas. Toutes ces caractéristiques sont normalisées entre 0 et 1 avant de les exploiter par un réseau de neurones multicouches. En effet, la normalisation est essentielle pour les RNA multicouches car elle permet de maintenir les poids du réseau dans des intervalles relativement restreints, d'optimiser les conditions d'apprentissage et par la même occasion d'améliorer la convergence.

5. Apprentissage et reconnaissance

Avant l'extraction de caractéristiques, chaque lettre est normalisée à 60 × 50 pixels. Ensuite, un ensemble de caractéristiques dépendantes et indépendantes de la ligne centrale horizontale (95 caractéristiques) sont utilisées dans les modules d'apprentissage et de reconnaissance. Nous utilisons une architecture perceptron multicouches en utilisant la rétropropagation du gradient. L'architecture du perceptron multicouche que nous avons utilisé se compose d'une couche d'entrée, une couche cachée et une couche de sortie. La couche d'entrée reçoit un vecteur d'entrée représentant l'entité à reconnaître, la couche cachée apprend à recoder les entrées et la couche de sortie fournit le résultat de reconnaissance. Les neurones de la couche de sortie représentent les classes de

reconnaissance. L'architecture générale du réseau de neurones artificiels (RNA) utilisé est représentée dans la Figure 6.10 ci-dessous.

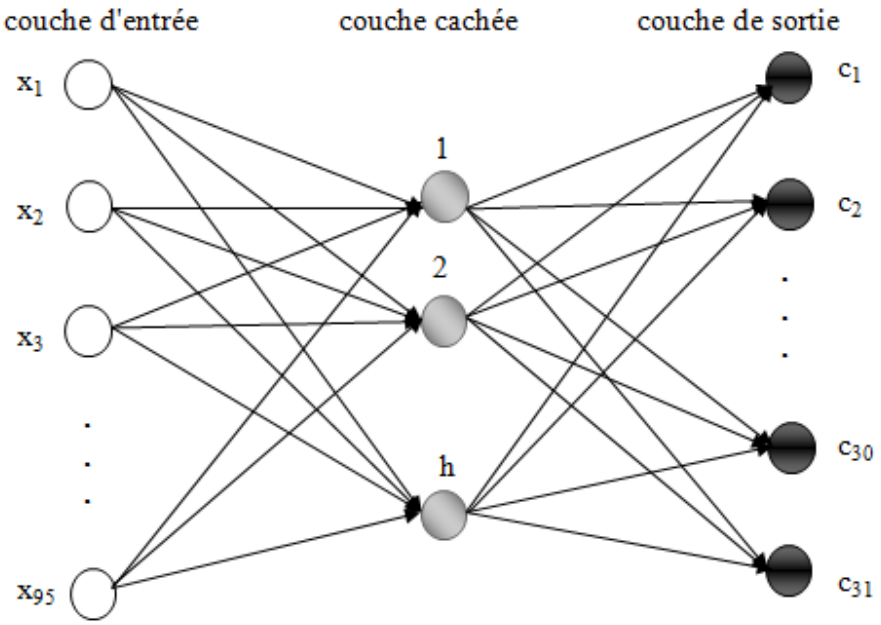


Figure 6.10 : Architecture du réseau de neurones utilisé

La topologie des RNAs utilisée est la suivante :

- la couche d'entrée possède 95 neurones qui correspondent aux l'ensemble des caractéristiques extraites ;
- la couche de sortie est composée de 31 neurones qui correspondent aux nombre de classes ;
- Pour la couche cachée, Il n'y a pas de règle absolue qui permet de déterminer avec exactitude le nombre de neurones à utiliser dans cette couche [Zhan00]. En effet, le nombre de noeuds cachés varie à travers différentes applications, et ce nombre doit être déterminé expérimentalement. Certaines heuristiques communément admises avancent les chiffres de (nb d'entrées + nb de sorties)/2 ou $\sqrt{\text{nb d'entrées} \times \text{nb de sorties}}$ sans toutefois prendre en compte la difficulté du problème [Chat06]. Dans notre cas, nous avons adopté le choix qui recommande de prendre un nombre de neurone de la couche cachée égale au moyen des nombres des neurones des couches d'entrée et de sortie (nb d'entrées + nb de sorties)/2.

De nos jours, de nombreux algorithmes de classification automatique existent et de nombreuses implémentations de ces algorithmes sont disponibles au téléchargement. Plusieurs boites à outils d'apprentissage regroupent de telles implémentations, ce qui en fait des outils idéaux pour lancer des expériences systématiques. Nous avons fait le choix de la plate-forme Weka [Witt05] pour réaliser l'apprentissage et testons la méthode proposée.

WEKA est un projet open source de l'Université de Waikato. Il a été largement utilisé dans les universités et par plusieurs chercheurs du monde dans le domaine d'exploration de données. Cet outil public propose un ensemble varié d'algorithmes d'apprentissage prêts à l'emploi pour la fouille de donnée. Nous utilisons la méthode de classification : réseaux de neurones perceptrons multicouches (Multi Layer Perception (MLP)). Le perceptron multicouches de WEKA a été mis en œuvre par Malcolm Ware en 2000 [Ware00]. Son utilisation a été documentée dans un certain nombre de publications de recherche [Klau02].

Les paramètres d'apprentissage pour les RNAs que nous avons utilisé sont les suivants :

- L'initialisation du RNA se fait avec des poids aléatoires ayant des valeurs comprises entre -1.0 et +1.0. La fonction de transfert que nous avons utilisée est la fonction logistique standard (la sigmoïde) ;
- La valeur du pas η de l'algorithme de la descente du gradient définie dans le chapitre 2 a été fixé à 0.2. Il est à noter que le choix d'une valeur trop grande du pas risque de faire diverger le processus de détection du minimum optimal, et inversement, une valeur du pas trop petite augmente les temps de calcul ;
- Le réseau est formé par le taux d'apprentissage 0,3 ;
- Le nombre d'itérations a été fixé à 1000.

De plus, Weka spécifie un format standard aux fichiers d'entraînement et de test (ce sont des fichiers texte avec une extension *.arff) format attribute-relation file format (ARFF). Pour cela nous avons généré deux fichiers, un pour l'apprentissage et l'autre pour le test. La Figure 6.11 ci-dessous illustre un extrait d'un exemple de fichier d'entraînement. Ce fichier est généré automatiqument avec un programme Matlab à partir de la base des images en utilsant le module d'extraction des caractéristiques.

```
@attribute f13C real
@attribute f14C real
@attribute f15C real
@attribute f16C real
@attribute f17C real
@attribute f18C real
@attribute f19C real
@attribute Letter
{ya,yab,ych,yad,yadd,yae,yaf,yag,yagh,yah,yahh,yaj,yak,yal,yam,
yan,yaq,yar,yarr,yas,yass,yat,yatt,yaw,yax,yay,yaz,yazz,yey,yi,yu}

@data
0.161667,0.071429,0.273883,0.323333,0.125000,0.125000,0.116667,0.1
0.168333,0.071429,0.300825,0.336667,0.000000,0.125000,0.000000,0.1
0.136667,0.071429,0.278252,0.273333,0.125000,0.125000,0.000000,0.1
0.176667,0.071429,0.240881,0.353333,0.125000,0.125000,0.000000,0.1
0.145000,0.142857,0.225479,0.290000,0.125000,0.250000,0.100000,0.1
0.153333,0.071429,0.296739,0.306667,0.000000,0.125000,0.000000,0.1
0.105000,0.071429,0.343122,0.210000,0.000000,0.125000,0.083333,0.1
```

Figure 6.11 : Un extrait du fichier ARFF d'entraînement

6. Expérimentations et Résultats

Dans cette section, nous passons en revue la procédure d'expérimentation de la l'approche proposée et les résultats obtenus sur les bases de données que nous avons utilisées.

6.1. Bases de données utilisées

6.1.1. Base des patterns de la graphie amazighe

La base de caractères imprimés utilisée est présentée dans la section 3.2.1. Il s'agit d'une base des patterns de différentes fontes amazighes et de tailles variées [Aito09]. La base contient à peut prés vingt mille patterns. Elle contient au total 12 polices de caractères et les tailles du 10 points au 28 points pour chaque modèle. Les patterns sont fournis sous forme d'images bitonales de tailles variables.

6.1.2. Base de caractères manuscrits

La base de caractères manuscrits utilisée est présentée dans le chapitre 4. Il s'agit de la base AMHCD que nous avons créé dans le cadre de cette thèse [Essa11b]. Nous avons exprémenté notre approche sur une grande partie de la base, recensée auprès de 50 scripteurs différents. Chaque scripteur nous a donné 13 exemples de chaque caractères. Au total, nous avons obtenu 650 modèles pour chaque caractère.

6.2. Résultats d'apprentissage et de reconnaissance

Pour évaluer les performances de la méthode proposée, des expériences ont été réalisées sur les deux bases de caractères amazighes décrits précédemment. Les tests ont été effectués en fonction de l'intégration des caractéristiques, dépendantes et indépendantes de la ligne centrale. Ainsi, trois expériences ont été réalisées sur un ensemble de 19437 des patterns de la graphie amazighes (31 × 627) et un autre ensemble de 20150 caractères amazighes manuscrits (31 × 650).

6.2.1. Résultats sur la base des patterns de la graphie amazighe

Dans un premier test, nous avons dévisé la base en deux sous base : un sous ensemble de 12958 images (66,67%) pour l'apprentissage, et un sous ensemble de 6479 images (33,33%) pour le test. Les deux classes sont équiprobables.

Le Tableau 6.1 ci-dessous, présente les résultats du système proposé sur la base des patterns de la graphie amazighe. Le taux de reconnaissance croit à 98,25 % lorsqu'on intègre les caractéristiques basées sur la position de la ligne centrale horizontale. Ce qui montre que les caractéristiques basées sur la position de cette ligne offrent une amélioration significative aux performances de reconnaissance.

Les caractéristiques intégrées	**Apprentissage**		**Test**	
	Taille de la base	*Taux de Recon*	*Taille de la base*	*Taux de Recon*
f_1,…, f_{13} (indépendantes de la ligne centrale horizontale)	12958 caractères	**88,78 %**	6479 caractères	**85,38 %**
f_{14}, …, f_{19} (dépendantes de la ligne de centrale horizontale)	12958 caractères	**95,89 %**	6479 caractères	**94,67 %**
f_1, .., f_{19} (dépendantes et indépendantes de la ligne centrale horizontale)	12958 caractères	**98,98 %**	6479 caractères	**98,25 %**

Tableau 6.1 : Résultats de reconnaissance sur la base de la graphie amazighe en fonction des caractéristiques intégrées

6.2.2. Résultats sur la base de caractères manuscrits

De la même manière, nous avons dévisé la base en deux sous base : un sous ensemble de 13454 images (66,67%) pour l'apprentissage, et un sous ensemble de 6696 images (33,33%) pour le test. Les deux classes sont équiprobables. Le Tableau 6.2 ci-dessous, présente les résultats obtenus sur la base de caractères manuscrits. Le taux de reconnaissance croit à 92,06 % lorsqu'on intègre les caractéristiques basées sur la position de la ligne centrale horizontale. Ce qui montre que les caractéristiques basées sur la

position de cette ligne offrent une amélioration significative aux performances de reconnaissance.

Les caractéristiques intégrées	**Apprentissage**		**Test**	
	Taille de la base	*Taux de Recon*	*Taille de la base*	*Taux de Recon*
f_1,…, f_{13} (indépendantes de la ligne centrale horizontale)	13454 caractères	**90,83 %**	6696 caractères	**83,76 %**
f_{14}, …, f_{19} (dépendantes de la ligne de centrale horizontale)	13454 caractères	**92,95 %**	6696 caractères	**88,82 %**
f_1, .., f_{19} (dépendantes et indépendantes de la ligne centrale horizontale)	13454 caractères	**97,73 %**	6696 caractères	**92,06 %**

Tableau 6.2 : Résultats de reconnaissance sur la base de caractères manuscrits en fonction des caractéristiques intégrées

6.3. Validation croisée

Nous avons aussi utilisé des techniques de validation croisée pour l'évaluation des résultats de reconnaissance. La validation croisée est une méthode d'estimation de la fiabilité des résultats, fondée sur une technique d'échantillonnage. Dans un système de classification, cette technique permet de mesurer les taux d'erreur du système en utilisant toutes les données disponibles, à la fois en apprentissage et en test [Koha95].

La validation croisée originale, dénommée validation croisée K-blocs (K-fold Cross Validation), est utile quand nous n'avons pas beaucoup de données pour les expériences [Lyva05]. On partage les données disponibles en K blocs disjoints, d'à-peu-près la même taille. On entraîne le système de classification en utilisant les données de K-1 blocs, puis on teste le système obtenu en utilisant les données du bloc restant. L'entraînement et le test sont répétés K fois (K expériences), afin que tous les blocs servent en test. Le taux d'erreur final du système sera cumulé sur toutes les K expériences.

Le Tableau 6.3 ci-dessous, présente les résultats obtenus par le système proposé en utilisant la validation croisée 10 fois sur les deux bases utilisées.

Pour la base des patterns de la graphie amazighe, le taux de reconnaissance est 88,68% lors de l'utisation seulement des caractéristiques indépendantes de la ligne centrale horizontale et augmente à 98,49% lors de l'ajout des caractéristiques basées sur la position de la ligne centrale horizontale. Pour la base de caractères amazighes manuscrits, le taux augmente de 84.49% à 92,23% lors de l'ajout des caractéristiques basées sur la position de la ligne centrale horizontale. Cela confirme que les caractéristiques basées sur la position de la ligne

centrale horizontale offrent une amélioration significative aux performances de reconnaissance.

Les caractéristiques intégrées	Base des patterns de la graphie amazighe		Base de caractères amazighes manuscrits	
	Taille de la base	*Taux de Recon*	*Taille de la base*	*Taux de Recon*
f_1,…, f_{13} (indépendantes de la ligne centrale horizontale)	19437 caractères	**88.68 %**	20150 caractères	**84.49 %**
f_{14}, …, f_{19} (dépendantes de la ligne centrale horizontale)	19437 caractères	**97.28 %**	20150 caractères	**88,57 %**
f_1, .., f_{19} (dépendantes et indépendantes de la ligne centrale horizontale)	19437 caractères	**98.49 %**	20150 caractères	**92,23 %**

Tableau 6.3 : Résultats de reconnaissance en fonction des caractéristiques intégrées en utilisant la validation croisée 10 fois

Les causes d'erreurs sont principalement dues à une grande similarité morphologique entre certains caractères amazighes et parfois sur des fontes différentes. En effet, l'étude de la matrice de confusion[1] du système obtenue sur la base des patternes de la graphie amazighe, présentée sur le Tableau 6.4 ci-dessous, a mis en évidence que la majorité des erreurs étaient faites sur les caractères yan (ⵍ), yar (ⵔ), ya (ⵄ), yazz (ⵣ) et yab (ⵖ). A titre d'exemple, 26 images (4.14%) du caractère yan (ⵍ) ont été reconnus comme caractère yaj (ⵉ). Dalleurs, le format du caractère yan (ⵍ) sur la fonte 'tassafut' ressemble entièrement au caractère yaj (ⵉ), comme le montre la Figure 6.12 ci-dessous.

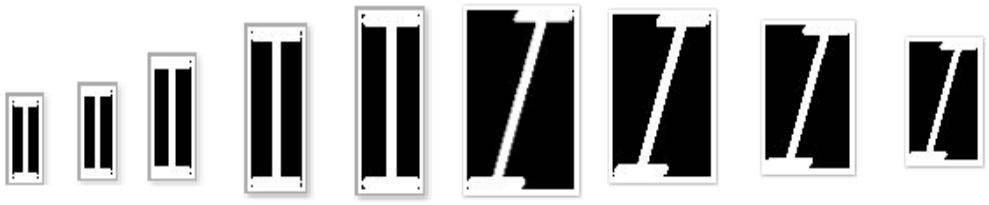


Figure 6.12 : Quelques exemples de caractère yan (ⵍ) dans la base, qui est de la police «tassafut»

[1]La matrice de confusion est la base de plusieurs évaluations statistiques des performances, elle nous renseigne sur la distribution de l'erreur de classification interclasses. Elle se présente sous forme d'un tableau qui met en relation le nombre d'échantillons bien classés et mal classés.

ⵄ	ⵖ	ⵛ	ⵜ	ⵟ	ⵀ	ⵃ	ⵅ	ⵇ	ⵍ	ⵎ	ⵏ	ⵔ	ⵕ	ⵙ	ⵚ	ⵣ	ⵡ	ⵥ	ⵢ	ⵣ	ⵤ	ⵧ	⵨	⵩	⵪	⵫	ⵯ		
ya	yab	yach	yad	yadd	yae	yaf	yag	yagh	yah	yahh	yaj	yak	yal	yam	yan	yax	yar	yarr	yas	yass	yat	yatt	yaw	yaz	yazz	yey	yi	yu	Taux d'err %	
609	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	**2.87**	ⵄ
2	610	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	2	1	0	0	0	0	0	0	10	0	**2.71**	ⵖ
0	0	626	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	**0.16**	ⵛ
0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵜ
0	0	0	0	620	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	3	0	0	0	0	0	0	**1.12**	ⵟ
0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵀ
0	0	0	0	0	0	622	0	0	0	0	0	1	0	0	3	0	0	0	0	0	0	0	1	0	0	0	0	0	**0.80**	ⵃ
0	0	0	1	0	0	0	606	0	0	5	0	0	0	0	1	8	0	0	0	3	2	0	0	0	0	1	0	0	**3.35**	ⵅ
0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵇ
0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵍ
0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵎ
0	0	0	0	0	0	0	0	0	0	0	623	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	1	0	**0.64**	ⵏ
0	0	0	0	0	0	0	0	0	0	0	0	626	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	**0.16**	ⵔ
0	0	0	0	0	0	2	0	0	0	4	0	0	612	0	3	3	0	0	0	0	0	0	0	3	0	0	0	0	**2.39**	ⵕ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵙ
0	0	0	0	0	0	1	0	0	0	0	26	1	0	0	585	1	0	0	0	0	10	1	0	0	0	0	2	0	0	0	**6.70**	ⵚ
0	0	0	0	0	0	0	8	0	1	0	1	0	0	0	6	605	0	0	0	0	3	0	0	0	1	2	0	0	0	0	**3.51**	ⵣ
15	1	0	0	0	0	0	0	0	6	0	0	0	0	0	5	0	593	0	0	3	0	1	2	0	0	0	0	0	0	1	**5.42**	ⵄ
1	0	0	0	0	0	0	0	5	8	0	0	0	0	0	1	0	0	607	0	0	0	1	1	1	1	1	0	0	0	0	**3.19**	ⵇ
1	5	0	1	1	0	0	0	0	6	0	0	0	0	0	0	0	1	0	610	0	0	0	0	0	0	0	0	2	0	0	**2.71**	ⵖ
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	624	0	0	0	0	0	0	0	0	0	0	**0.48**	ⵛ
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	623	0	0	0	0	0	2	0	0	0	**0.64**	ⵜ
0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	617	0	0	1	0	0	3	0	0	**1.59**	ⵟ
0	0	0	0	0	0	0	0	0	0	0	0	1	7	0	4	0	0	0	0	0	0	1	613	1	0	0	0	0	0	0	**2.23**	ⵀ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	626	0	0	0	0	0	0	**0.16**	ⵃ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	**0.00**	ⵅ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	626	1	0	0	0	**0.16**	ⵇ
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	613	0	0	0	**2.23**	ⵍ
1	4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	615	0	5	**1.91**	ⵎ
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	620	0	**1.12**	ⵏ
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	625	**0.32**	ⵔ

Tableau 6.4 : Matrice de confusion obtenue par le système de base, évalué sur la base des patterns de la graphie amazighe

Un autre exemple consiste à 18 remplacements de la lettre ya (ⵄ) par la lettre yar (ⵔ). Ce problème est dû à la conception des caractères ya (ⵄ) dans la base d'images utilisée. En outre, dans cette base de données, les images des caractères sont crées d'une manière qui ne permet pas de renormaliser leurs tailles en une taille moyenne fixe. En effet, cela peut être particulièrement gênant à cause de la ressemblance des caractères ya (ⵄ) et yar (ⵔ), qui ne

diffèrent que par sa taille: le caractère ya (ⵄ) est un petit cercle, tandis que le caractère yar (ⵔ) est un grand cercle. Dans certains cas, il y aura une réelle confusion entre les images de ces deux caractères. La Figure 6.13 ci-dessous ullistre quleques exemples de caractères ya (ⵄ) et de caractères yar (ⵔ) dans la base utilisée. Ce problème a une influence sur les résultats et il a été résolu dans notre base de caractères amazighes manuscrits.

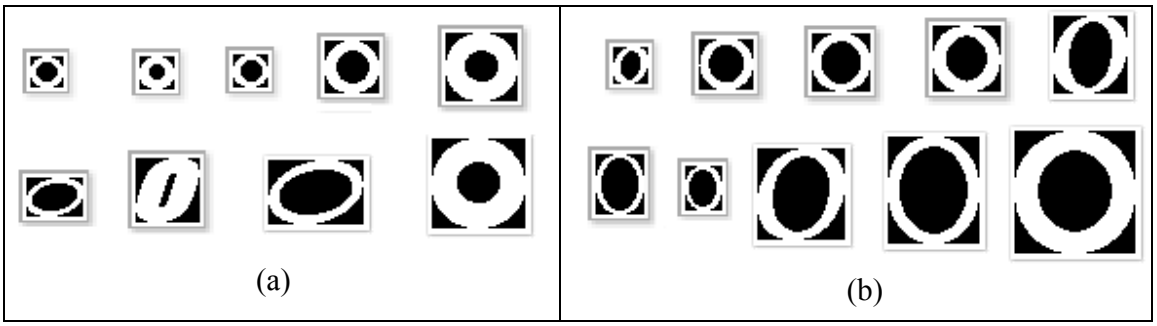


Figure 6.13 : (a) Exemples de caractères ya (ⵄ) dans la base des patterns de la graphie, (b) exemples de caractères yar (ⵔ) dans la même base

Pour le cas de la base de caractères manuscrits, une analyse de la matrice de confusion du système obtenue sur cette base, présentée sur le Tableau 6.5 ci-dessous, nous permet de faire trois remarques importantes. En premier lieu, on remarque que toutes les lettres ont été connues au moins 9 erreurs sauf la lettre ya (ⵄ), qui a seulement une seule erreur.

En second lieu, le taux d'erreur est très important pour certains caractères ayant une grande similarité morphologique. Par exemple, la lettre yaz (ⵣ) est confondue 41 fois avec la lettre yazz (ⵥ), 41 fois avec la lettre yaj (ⵊ) et 21 fois avec la lettre yaq (ⵇ). Un autre exemple, la lettre yatt (ⵟ) est confondue 47 fois avec la lettre yadd (ⴹ).

En troisième lieu, certaines confusions entre quelques lettres sont absurdes, du fait qu'il n'y ait aucune ressemblance entre ces lettres, par exemple celle entre la lettre yaf (ⵃ) et la lettre yak (ⴽ), celle entre la lattre yatt (ⵟ) et la lettre yak (ⴽ) et celle entre la lattre yi (ⵉ) et la lettre yey (ⵢ). Ces types d'erreurs expliquent que les caractéristiques utilisées sont insufisantes pour modéliser les caractères amazighes.

Dans la section suivante nous présentons une amélioration de ce système permettant de réduire les problèmes de classification liés à la grande similarité morphologique entre plusieurs caractères amazighes. Cette amélioration sera au nivau de module de l'extraction des caractéristiques en ajoutant des nouvelles prémitives basées sur la ligne centrale verticale du caractère.

ya	yab	yach	yad	yadd	yae	yaf	yag	yagh	yah	yahh	yai	yak	yal	yam	yan	yax	yar	yarr	yas	yass	yat	yatt	yaw	yaq	yay	yaz	yazz	yey	yi	yu	Taux d'err %	
649	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	**0,15**	ⵄ
0	612	1	0	7	1	0	0	0	0	0	0	0	1	0	1	0	3	1	10	6	0	5	0	0	0	0	0	1	0	1	**5,85**	ⵃ
0	3	610	0	2	0	0	0	0	0	0	0	3	0	4	1	0	1	6	0	4	0	1	0	3	0	0	1	1	2	8	**6,15**	ⵅ
0	0	0	639	0	3	0	0	0	0	2	0	0	5	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1,69**	ⵆ
0	5	2	1	566	0	0	1	0	0	0	0	1	0	6	1	0	0	2	1	1	0	30	0	12	2	0	1	4	6	8	**12,92**	ⵇ
0	0	1	6	0	628	0	0	1	2	7	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	**3,38**	ⵈ
2	0	0	0	1	1	589	6	1	0	2	9	7	7	2	3	6	0	1	0	1	2	1	1	0	0	0	5	0	1	2	**9,38**	ⵉ
0	0	0	1	0	0	2	624	1	0	1	0	0	0	0	3	10	0	1	0	0	0	0	0	0	3	2	0	0	2	0	**4,00**	ⵊ
0	0	0	0	0	0	1	0	631	0	0	0	1	5	1	1	0	0	5	1	1	0	0	2	0	0	1	0	0	0	0	**2,92**	ⵋ
0	0	0	0	0	1	1	0	0	620	0	2	0	0	0	0	0	3	6	8	1	1	1	0	1	0	3	1	0	1	0	**4,62**	ⵌ
1	0	0	6	2	0	3	0	0	0	621	0	0	0	4	0	1	0	0	0	0	0	1	5	0	1	3	1	0	1	0	**4,46**	ⵍ
1	0	0	0	0	0	3	0	0	0	0	605	0	0	2	8	2	1	1	0	0	5	1	0	0	0	17	4	0	0	0	**6,92**	ⵎ
0	1	1	2	1	0	11	7	2	1	1	4	556	7	4	7	2	0	3	2	4	1	2	0	24	3	2	1	0	0	1	**14,46**	ⵏ
0	1	0	2	1	1	8	0	3	2	1	0	8	608	2	0	1	1	0	0	2	1	2	2	3	0	1	0	0	0	0	**6,46**	ⵐ
1	0	0	0	3	0	0	0	1	0	3	0	3	1	622	4	0	1	1	0	0	1	0	2	0	2	3	1	0	0	1	**4,31**	ⵓ
0	0	0	2	0	1	0	1	0	0	3	2	0	1	5	619	0	0	2	1	1	2	4	1	0	2	0	1	1	0	1	**4,77**	ⵔ
0	0	0	4	0	0	2	11	2	1	3	7	0	0	2	3	593	0	0	0	2	2	0	0	1	2	8	4	0	3	0	**8,77**	ⵕ
0	1	10	0	1	2	0	0	2	2	0	0	0	0	5	0	1	614	2	0	7	0	0	1	0	0	0	0	0	0	2	**5,54**	ⵖ
0	5	11	1	1	0	1	1	5	4	0	2	3	1	0	0	1	6	586	5	3	3	1	1	0	5	1	0	1	2	0	**9,85**	ⵗ
0	16	1	0	2	1	0	0	2	2	0	0	1	1	1	0	1	4	5	599	10	0	2	0	0	0	0	0	1	0	1	**7,85**	ⵘ
0	4	6	0	1	1	1	0	0	5	4	0	1	3	1	0	0	8	0	3	596	0	3	0	7	3	0	1	0	2	0	**8,31**	ⵙ
0	0	0	1	3	2	1	1	0	0	1	4	0	0	1	5	3	0	0	0	1	619	0	1	0	1	1	4	1	0	0	**4,77**	ⵚ
1	5	0	0	47	0	7	1	4	0	3	0	3	0	2	5	0	0	2	5	1	0	537	0	11	1	1	3	4	6	1	**17,38**	ⵛ
1	0	0	0	0	0	0	0	1	0	1	0	0	5	0	0	0	1	0	0	0	0	0	641	0	0	0	0	0	0	0	**1,38**	ⵜ
0	1	0	1	8	0	3	3	0	0	2	2	20	1	6	2	1	0	2	0	5	0	7	1	576	0	2	0	0	6	1	**11,38**	ⵝ
1	0	1	2	0	0	0	5	1	0	2	0	0	0	0	3	0	0	0	0	1	0	0	0	0	615	0	0	5	12	2	**5,38**	ⵞ
0	0	1	0	0	3	3	4	5	2	4	41	4	0	1	3	21	0	2	3	2	2	6	1	0	1	491	41	0	3	6	**24,46**	ⵟ
0	0	0	0	1	0	5	1	4	1	2	4	1	1	0	3	14	0	1	1	1	15	4	0	0	4	54	515	10	6	2	**20,77**	ⵠ
1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	0	5	0	9	594	18	16	**8,62**	ⵡ
0	0	1	0	1	1	0	0	0	0	1	1	0	0	3	1	2	0	0	0	1	0	1	0	0	6	0	2	11	614	4	**5,54**	ⵢ
1	2	1	0	4	0	0	7	1	0	0	3	0	1	1	2	0	0	3	3	0	0	3	0	0	3	3	2	10	4	596	**8,31**	ⵣ

Tableau 6.5 : Matrice de confusion obtenue par le système de base, évalué sur la base de caractères amazighes manuscrits

7. Amélioration du système proposé par ajout d'autres caractéristiques

Nous avons mis en œuvre dans les sections précédentes un système de reconnaissance de l'écriture amazighe, basé sur la position de la ligne centrale horizontale, et qui donne des résultats encourageants. Nous proposons dans cette section une amélioration à ce système de reconnaissance. L'idée de base consiste à intégrer une information qui permet de remédier aux problèmes de classifications issues de la ressemblance entre plusieurs

caractères amazighes, en ajoutant de nouvelles caractéristiques. En effet, un des principaux problèmes avec la méthode ci-dessus est que la majorité des erreurs sont principalement dues à une grande similarité morphologique entre certains caractères amazighes. Il est à noter que ces problèmes se trouvent accentués dans le cas de l'ecriture imprimée sur des fontes différentes, ainsi que dans le cas de manuscrits à cause de la diversité des formes et de la variabilité des scripts. Pour éviter ce problème, il serait bon d'exploiter la similarité de plusieurs caractères amazighes par rapport à la ligne centrale verticale du caractère. La méthode présentée dans cette section consiste à améliorer le taux de reconnaissance obtenu par rapport au système de reconnaissance décrit précédemment en lui ajoutant des caractéristiques dépendantes et indépendantes de la ligne centrale verticale. La Figure 6.14 ci-dessous illustre l'architecture générale du système amélioré.

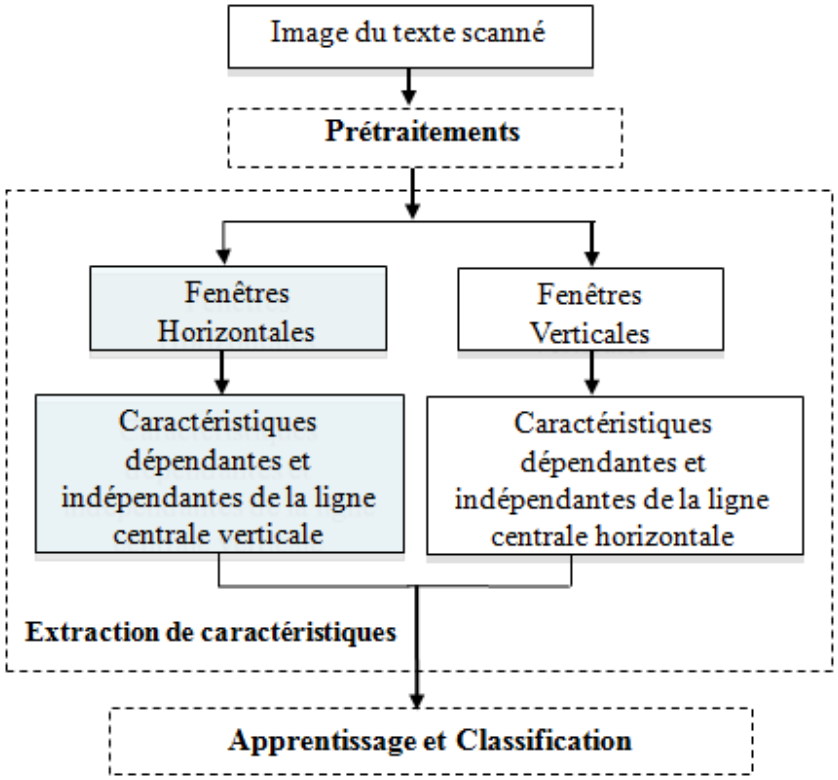


Figure 6.14 : Architecture générale du système amélioré

7.1. Extraction des caractéristiques basées sur la ligne centrale verticale

Pour exploiter les informations provenant de la ligne centrale verticale du caractère, nous avons généré un deuxième groupe de caractéristiques de densité dépandantes et indépendantes de cette ligne. Pour créer ce deuxième groupe de caractéristiques, l'image du

caractère est balayée de haut en bas et de gauche à droite avec une fenêtre glissante. L'image est divisée en fenêtres horizontaux (cf. Figure 6.15). Le nombre de fenêtre est constant et il est considéré comme l'un des paramètres du système (5 fenêtres dans nos expériences). Chaque fenêtre est divisée en cellules où la hauteur de cellule est fixe (dans nos expériences à 4 pixels).

Dans chaque fenêtre, nous générons un ensemble de 9 caractéristiques (g1, g2, g3, g14, g15, g16, g17, g18, g19). Ces caractéristiques sont similaires aux caractéristiques basées sur la ligne centrale horizontale (f1, f2, f3, f14, f15, f16, f17, f18, f19), respectivement. Nous appliquons les mêmes formules que celles utilisées pour extraire les caractéristiques basées sur la ligne centrale horizontale présentées précédemment (cf. section 6.4).

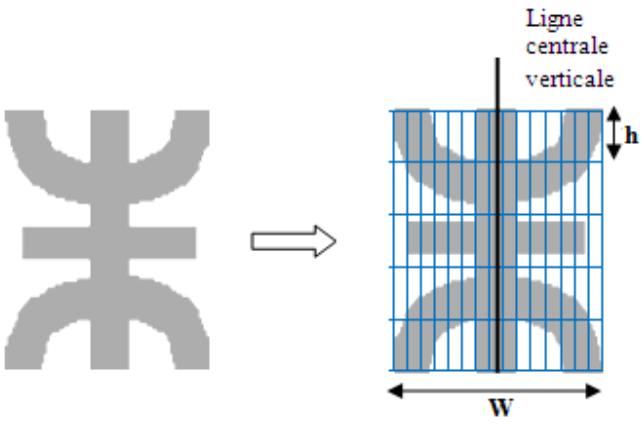


Figure 6.15 : Diviser les lettres en fenêtres horizontales

7.2. L'ensemble de caractéristiques retenues

Le résultat du module d'extraction de vecteurs de caractéristiques est maintenant deux groupes de vecteurs de primitives. Chaque groupe correspond au vecteur de caractéristiques extraites à base d'une ligne centrale de caractère.

Soient les deux ensembles suivants :

- F : l'ensemble de toutes les caractéristiques, extraites dans chaque fenêtre verticale, basées sur la ligne centrale horizontale éxaminées précedement (cf. section 6.4)

  $F= \{f_1, f_2, \ldots\ldots, f_{19}\}$

- G : l'ensemble de toutes les caractéristiques extraites dans chaque fenêtre horizontale, basées sur la ligne centrale verticale

  $G= \{g_1, g_2, \ldots\ldots, g_{19}\}$

Pour optimiser la taille du vecteur de caratéristiques, nous avons éliminé les 10 caractéristiques (f4, f5, ……, f13), associées aux densités de pixels d'écriture dans chaque colonne d'une fenêtre verticale et les 10 caractéristiques (g4, g5, ……, g13), associées aux densités de pixels d'écriture dans chaque ligne d'une fenêtre horizontale. La raison de diminuer la taille du vecteur de caratéristiques est pour rendre efficace le processus d'apprentissage du réseau de neuronnes. En fait, ces caractéristiques sont indépandantes des lignes de base utilisées et sont moins importantes par rapport aux nouvelles caractéristiques ajoutées. Enfin nous retenons l'union de deux ensembles F′ et G′ suivants :

$$F' = F - \{f_4, f_5, \ldots\ldots, f_{13}\} = \{f_1, f_2, f_3, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}\}$$

$$G' = G - \{g_4, g_5, \ldots\ldots, g_{13}\} = \{g_1, g_2, g_3, g_{14}, g_{15}, g_{16}, g_{17}, g_{18}, g_{19}\}$$

L'ensemble de 18 caractéristiques extraites comporte 6 caractéristiques qui dépendent de la position de la ligne centrale horizontale, 6 qui dépendent de la position de la ligne centrale verticale, et 6 qui n'en dépendent pas. Ces caractéristiques alimenteront un réseau de neurones multicouches dans les phases d'apprentissage et de reconnaissance.

7.3. Résultats et Discussion

Nous avaons utilisé seulement les techniques de validation croisée 10 fois pour tester et valider la perfermance du système amélioré.

Le Tableau 6.6 ci-dessous, présente les résultats du système proposé en utilisant la validation croisée 10 fois sur la base des patterns de la graphie amazighe et sur la base de caractères manuscrits.

Les caractéristiques intégrées	**Base des patterns de la graphie amazighe**		**Base de caractères amazighes manuscrits**	
	Taille de la base	*Taux de Recon*	*Taille de la base*	*Taux de Recon*
Caractéristiques indépendantes de la ligne de centrale	19437 caractères	**88.68 %**	20150 caractères	**84.49 %**
Caractéristiques dépendantes et indépendantes de la ligne centrale horizontale	19437 caractères	**98.49 %**	20150 caractères	**92.23 %**
Caractéristiques dépendantes et indépendantes de la ligne centrale horizontale et verticale	19437 caractères	**99.28 %**	20150 caractères	**96.32 %**

Tableau 6.6 : Résultats de reconnaissance du système amélioré en fonction des caractéristiques intégrées en utilisant la validation croisée 10 fois

Pour la base des patterns de la graphie amazighe, le taux de reconnaissance est 98,49% lors de l'intégration des caractéristiques basées sur la position de la ligne centrale horizontale et augmente à 99,28% lors de l'ajout des caractéristiques basées sur la position de la ligne centrale vertical.

En comparant le taux de reconnaissance obtenu par le système amélioré avec celui obtenu par le système de base, nous constatons une amélioration due à l'intégration des caractéristiques basées sur la position de la ligne centrale vertical. Cela démontre que les caractéristiques basées sur la position des lignes centraux (verticale et horizontale) offrent une amélioration significative à la performance de reconnaissance.

Le Tableau 6.7 ci-dessous illustre la matrice de confusion obtenue par le système amélioré sur la base des patterns de la graphie amazighe. En analysant cette matrice de confusion, nous constatons que seulement quatre lettres de l'ensemble de l'alphabet avaient peu d'erreurs.

La première erreur est due à la ressemblance entre certaines lettres dans différentes polices (le problème de resemblance entre les deux caractères yan (ⵍ) et yaj (ⵊ) discuté précedement). Comme expliqué précedement, ces erreurs ne peuvent pas réellement être corrigées car même un humain ne pourrait différencier certains de ces lettres.

La deuxième erreur consiste à 24 remplacements de la lettre ya (ⵄ) par la lettre yar (ⵔ) sur 627 examples. Ce problème est dû au création des caractères ya (ⵄ) dans la base de données utilisée (ce problème a été discuté dans la section précedente).

La troisième erreur consiste à 13 remplacements de la lettre yazz (ⵣ) par la lettre yaz (ⵥ). Ce problème du par la grande similarité morphologique entre ces deux lettres. La seule différence entre ces deux lettres est un trait dans le centre de la lettre. La lettre yaz (ⵥ) est une dérivation de la lettre yazz (ⵣ).

ⵄ	ⵖ	ⵛ	ⵜ	ⵟ	ⵀ	ⵃ	ⵅ	ⵇ	ⵈ	ⵉ	ⵊ	ⵍ	ⵎ	ⵏ	ⵔ	ⵕ	ⵙ	ⵚ	ⵜ	ⵝ	ⵞ	ⵟ	ⵠ	ⵡ	ⵢ	ⵣ	ⵤ	ⵥ	ⵦ	ⵧ	⵨		
ⵄa	ⵖab	ⵛach	ⵜad	ⵟadd	ⵀae	ⵃaf	ⵅag	ⵇagh	ⵈah	ⵉahh	ⵊaj	ⵍak	ⵎal	ⵏam	ⵔan	ⵕax	ⵙar	ⵚarr	ⵜas	ⵝass	ⵞat	ⵟatt	ⵠaw	ⵡab	ⵢay	ⵣaz	ⵤazz	ⵥey	ⵦi	ⵧu	**Taux d'err %**	
602	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2 4	0	0	0	0	0	0	0	0	0	0	0	0	**3.99**	ⵄ
0	616	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	3	0	5	0	0	0	0	0	0	0	1	0	0	**1.75**	ⵖ
0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵛ
0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵜ
0	0	0	0	625	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	**0.32**	ⵟ
0	0	0	0	0	626	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.16**	ⵀ
0	0	0	0	0	0	624	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.48**	ⵃ
0	0	0	0	0	0	0	624	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	**0.48**	ⵅ
0	0	0	0	0	0	0	0	626	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	**0.16**	ⵇ
0	0	0	0	0	0	0	0	0	626	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.16**	ⵈ
0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵉ
0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵊ
0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵋ
0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵍ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵎ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.00**	ⵏ
0	0	1	0	0	0	0	0	0	0	3 7	0	0	0	587	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	**6.38**	ⵔ
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	626	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.16**	ⵅ
17	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	603	0	0	1	0	1	0	0	0	0	0	3	0	0	**3.83**	ⵄ
0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	623	0	0	0	0	0	0	0	0	0	2	0	0	**0.64**	ⵇ
0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	625	0	0	0	0	0	0	0	0	0	0	0	**0.32**	ⵈ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	625	0	0	0	0	0	0	0	0	0	0	**0.32**	ⵉ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	0	**0.00**	ⵊ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	0	**0.00**	ⵋ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	0	**0.00**	ⵍ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	0	**0.00**	ⵎ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	0	**0.00**	ⵏ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	0	**0.00**	ⵔ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	0	0	**0.00**	ⵅ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1 3	614	0	0	0	**2.07**	ⵄ
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	621	0	5	**0.96**	ⵖ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	0	**0.00**	ⵛ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	627	**0.00**	ⵜ

Tableau 6.7 : Matrice de confusion obtenue par le système amélioré, évalué sur la base des patterns de la graphie amazighe

Dans le Tableau 6.8 ci-dessous nous dressons la matrice de confusion obtenue sur la base de caractères amazighes manuscrits.

L'étude de cette matrice de confusion a montré que la plupart des erreurs sont principalement dues à la ressemblance entre certains caractères amazighes. par exemples,

des confusions entre les deux lettres yaz (ⵣ) et yazz (ⵥ), entre les deux lettres yatt (ⵟ) et yadd (ⴹ), entre les deux lettres yaz (ⵣ) et yax (ⵅ) et entre les deux lettres yey (ⵢ) et yu (ⵓ).

ⴰ	ⴱ	ⴳ	ⴷ	ⴹ	ⴻ	ⴼ	ⴳ	ⴳ	ⴽ	ⵀ	ⵃ	ⵄ	ⵇ	ⵉ	ⵊ	ⵍ	ⵎ	ⵏ	ⵔ	ⵕ	ⵙ	ⵚ	ⵛ	ⵜ	ⵟ	ⵡ	ⵣ	ⵥ	ⵢ	ⵉ	ⵓ		
ya	yab	yach	yad	yadd	yae	yaf	yag	yagh	yah	yahh	yaj	yak	yal	yam	yan	yax	yar	yarr	yas	yass	yat	yatt	yaw	yaq	yay	yaz	yazz	yey	yi	yu	Taux d'err %	
649	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	**0,15**	ⴰ
0	626	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	7	3	0	6	0	1	1	0	0	1	0	1	**3,69**	ⴱ
0	2	625	0	0	0	0	1	0	0	0	1	1	0	2	0	0	4	7	1	2	0	1	0	0	0	0	1	0	0	2	**3,85**	ⴳ
0	0	0	643	0	1	0	0	0	0	2	0	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**1,08**	ⴷ
0	0	1	0	599	0	0	0	0	0	0	1	3	0	9	0	0	0	0	0	0	0	29	0	2	0	0	0	0	6	0	**7,85**	ⴹ
1	0	0	4	0	638	0	0	0	0	3	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	**1,85**	ⴻ
0	0	0	0	0	0	635	1	1	0	0	1	0	1	0	0	3	0	0	3	0	0	0	1	0	0	1	2	1	0	0	**2,31**	ⴼ
0	0	0	1	0	0	1	641	0	0	0	0	3	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	1	**1,38**	ⴳ
0	0	0	0	1	0	0	0	639	0	0	0	0	3	0	0	0	0	0	0	4	1	0	1	0	0	1	0	0	0	0	**1,69**	ⴳ
0	0	0	0	0	0	0	1	0	627	0	2	1	3	1	0	0	1	6	3	1	1	0	0	3	0	0	0	0	0	0	**3,54**	ⴽ
0	0	0	2	0	2	0	0	0	0	644	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0,92**	ⵀ
0	0	0	0	0	0	0	4	0	1	0	628	0	0	6	3	1	0	0	1	0	2	0	0	0	0	3	0	0	0	1	**3,38**	ⵃ
0	0	0	1	1	0	0	6	0	3	6	0	623	0	0	0	0	0	0	1	2	0	1	0	6	0	0	0	0	0	0	**4,15**	ⵄ
0	0	0	0	0	0	0	0	1	2	1	0	0	640	0	0	1	0	0	1	3	0	0	0	0	0	1	0	0	0	0	**1,54**	ⵇ
0	0	4	0	3	0	0	0	0	0	0	1	0	0	640	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	**1,54**	ⵉ
0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	644	0	0	0	0	0	2	1	1	0	0	0	0	0	0	0	**0,92**	ⵊ
0	0	0	0	0	0	1	2	2	0	5	0	0	1	0	0	622	0	0	0	0	2	0	0	0	0	13	2	0	0	0	**4,31**	ⵅ
0	2	5	0	2	0	1	0	0	1	0	0	0	0	2	1	0	620	2	2	9	0	0	2	0	0	0	0	0	0	1	**4,62**	ⵍ
0	1	4	0	0	2	2	0	1	4	0	0	3	0	0	0	0	5	617	6	1	1	0	0	0	2	0	1	0	0	0	**5,08**	ⵎ
0	1	2	0	2	0	0	0	1	2	0	0	1	0	0	0	0	6	7	611	6	0	4	0	4	0	0	2	0	0	1	**6,00**	ⵏ
0	2	1	0	0	0	2	0	0	3	2	1	0	1	0	0	2	8	0	9	614	0	1	1	3	0	0	0	0	0	0	**5,54**	ⵔ
0	0	0	0	3	2	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	634	1	0	0	0	0	3	4	0	0	**2,46**	ⵕ
1	9	0	0	23	0	1	0	0	0	0	2	3	0	1	2	0	0	1	1	1	0	589	1	7	0	0	1	0	4	3	**9,38**	ⵙ
0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	648	0	0	0	0	0	0	0	**0,31**	ⵚ
0	0	0	1	1	0	0	0	0	3	0	2	4	0	1	1	0	1	0	2	1	0	0	0	633	0	0	0	0	0	0	**2,62**	ⵛ
0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	644	0	0	0	1	0	**0,92**	ⵜ
0	0	0	0	0	3	1	1	0	0	1	4	0	3	0	0	18	0	0	3	3	2	0	0	0	0	577	33	1	0	0	**11,23**	ⵟ
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	6	0	0	0	2	2	0	0	1	1	40	593	1	0	1	**8,77**	ⵡ
0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	3	0	0	1	0	0	2	0	0	0	3	1	2	621	4	10	**4,46**	ⵣ
0	0	1	0	3	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	2	1	0	5	635	1	**2,31**	ⵥ
0	0	1	0	0	0	0	3	0	1	0	2	0	0	0	1	1	0	0	0	0	0	2	0	1	1	0	2	9	0	626	**3,69**	ⵢ

Tableau 6.8 : Matrice de confusion obtenue par le système amélioré, évalué sur la base de caractères amazighes manuscrits

8. Conclusion et perspectives

Dans ce chapitre, nous avons présenté un système pour la reconnaissance automatique de l'écriture amazighe à base de la position des lignes centrales de chaque caractère. Plusieurs caractéristiques ont été étudiées et comparées. L'importance de l'utilisation de la position

de la ligne de centrale horizontale et verticale dans l'image du caractère a été prouvée. Les caractéristiques extraites sont basées sur la densité des pixels dérivée dans une fenêtre glissante. Le système développé a été expérimenté sur deux base : une base des patterns de la graphie amazighe et une autre base de caractères amazighes manuscrits développée localement. L'analyse des résultats obtenus par le système montre une amélioration significative du taux de reconnaissance lorsqu'on intègre les caractéristiques dépendantes de la ligne centrale horizontale. La valeur du taux de reconnaissance croit encore lorsqu'on intègre les caractéristiques basées sur la ligne centrale verticale du caractère. Ce qui montre que l'amélioration proposée apporte de bons résultats. Les résultats de ce système ont été publiés dans le journal international IJAST [Essa11c].

Parmi les travaux futurs de ce travail, nous allons ajouter d'autres caractéristiques qui améliorent les résultats pour certains caractères dont le taux de reconnaissance est faible par apport aux restes, telles que les informations sur l'inclinaison possible de l'écriture manuscrite. En plus, nous allons appliquer notre approche sur des données des documents amazighes.

Conclusion générale et perspectives

Dans ce travail de thèse, nous nous sommes principalement intéressés à la reconnaissance automatique de l'écriture amazighe. Dans ce cadre, nous avons construit une base de données de caractères amazighes manuscrits et nous avons proposé deux approches de reconnaissance automatique de l'écriture amazighe qui' ont contribué à améliorer les performances. La première approche est syntaxique, elle utilise des automates finis pour reconnaitre les caractères amazighes imprimés. La deuxième approche est basée sur la ligne centrale horizontale du caractère. Elle a montré de bonnes performances sur la base de données de patterns de la graphie amazighe et notre base de caractères manuscrits. Une amélioration de cette approche a été proposée en intégrant d'autres caractéristiques basées sur la ligne centrale verticale du caractère.

En effet, nous avons présenté une première version de la base de données AMHCD, qui contient plus de 25.000 caractères amazighes manuscrits isolés écrits par 60 scripteurs différents. Cette base de données est destinée à la fois à tester et valider nos approches et à servir d'autres chercheurs dans le domaine. L'objectif c'est de préparer les données d'apprentissage et de test pour expérimenter des approches de reconnaissance d'écriture amazighe afin de faire des comparaisons objectives entre les différents systèmes.

Nous avons présenté ensuite, une approche syntaxique de reconnaissance de caractères amazighes imprimés, en se basant sur les automates à états finis. Nous avons utilisé le codage de Freeman pour construire la chaîne représentative du caractère à partir de son squelette. Cette chaîne est modélisée par une grammaire régulière puis par un automate fini. A partir de tous les automates finis modélisant de tous les caractères amazighes segmentés, nous avons construit l'automate maximal canonique qui reconnaît tous les caractères. En étape de reconnaissance, ce dernier automate permet de décider la classe d'appartenance d'un caractère en entrée. Le système proposé a de bonnes performances sur notre base de données. Mais, il présente des limites sur les caractères non segmentés. Pour remédier à ces limites, nous avons développé une deuxième approche qui consiste à exploiter l'information qui vient de la ligne centrale horizontale et verticale du caractère amazighe. En effet, ce système de reconnaissance automatique de texte amazighe est basé sur la position des lignes centrales de chaque caractère. Plusieurs caractéristiques de densités des pixels dérivées dans une fenêtre glissante ont été étudiées et comparées. L'importance de l'utilisation de cette position a été prouvée. L'apprentissage et la classification on été faites avec un perceptron multicouches en utilisant la rétropropagation du gradient. L'analyse des

résultats obtenus par le système sur les deux bases (la base des patterns de la graphie amazighe et notre base de caractères amazighes manuscrits) montre une amélioration significative du taux de reconnaissance lorsqu'on intègre les caractéristiques dépendantes de la ligne centrale horizontale. De plus, le taux de reconnaissance s'améliore encore suite à une amélioration qui intègre d'autres caractéristiques basées sur la ligne centrale verticale du caractère. Ce qui montre que l'amélioration proposée du système apporte de bons résultats.

Les perspectives envisageables pour l'amélioration et la poursuite de nos travaux sont nombreuses. Celles que nous avons jugées importantes sont les suivantes :

- Au sujet de la base de données de caractères manuscrits, nous comptons de rendre la base plus large en ajoutant d'autres exemples de caractères tifinaghs extraits dans des documents réels. Par ailleurs, nous préférons d'utiliser des formulaires qui contiennent des textes amazighes de plusieurs tailles et ensuite demander aux scripteurs de réécrire les textes en tailles variées.
- Au sujet de l'approche syntaxique qui se base sur les automates finis pour la reconnaissance de caractères amazighes, plusieurs améliorations pourraient être considérées. En premier lieu, des améliorations au niveau de la phase d'analyse peuvent être effectuées. En effet, dans ce travail nous avons basé sur le squelette du caractère afin d'extraire les chaines du codage du Freeman. Or on peut se baser sur l'analyse du contour du caractère au lieu de squelette. En second lieu, nous avons adopté les grammaires régulières et les automates finis simples. En perspectives, l'ajout des probabilités affectées aux règles de production avec une grammaire contenant des valeurs statistiques attachées aux caractéristiques peut être mis en ouvre. Cette modélisation permet d'avoir des grammaires et des automates finis stochastiques. De plus, un algorithme d'inférence d'automates peut être appliqué afin de générer l'automate global. Enfin, nous pensons qu'il est nécessaire de construire un réseau d'automates qui reconnaît l'ensemble de l'alphabet tifinagh, soit en intégrant d'autres codages qui permettra de coder les autres caractères amazighes non segmentés, soit en combinaison l'automate avec d'autres classifieurs supportant les autres caractères.
- Parmi les travaux futurs de la dernière contribution, nous allons ajouter d'autres caractéristiques qui améliorent les résultats pour certains caractères dont le taux de reconnaissance est faible par apport aux restes, telles que les informations sur l'inclinaison possible de l'écriture manuscrite. En plus, la combinaison des réseaux de neurones avec d'autres classifieurs tels que les Modèles de Markov Cachés peut aussi améliorer la reconnaissance. Enfin, nous allons appliquer notre approche sur des documents amazighes.

Bibliographie

[Abdu09] S. Abdul, S. Shams-ul Haque, M. Khan Pathan, "A Finite State Model for Urdu Nastalique Optical Character Recognition", IJCSNS International Journal of Computer Science and Network Security, vol.9 (9), Sept 2009.

[Abuh98] I. S. I. Abuhaiba, M. J. J. Holt, S. Datta, "Recognition of off-line cursive handwriting", Computer Vision and Image Understanding, vol.71 (1), pp.19-38, July 1998.

[Ahon94] H. Ahonen, "Generating grammars for structured documents using grammatical inference methods", These de PhD, University of Helsinki, Department of Computer Science, 1994.

[Aida09] R. Aida-zade and Z. Hasanov, "Word base line detection in handwritten text recognition systems", International Journal of Electrical and Computer Engineering, vol.4 (5), pp.310-314, 2009.

[Aito09] Y. Ait Ouguengay, M. Taalabi, "Elaboration d'un réseau de neurones artificiels pour la reconnaissance optique de la graphie amazighe: Phase d'apprentissage", Systèmes intelligents-Théories et applications, Paris : Europia, cop. 2009 (impr. au Maroc), ISBN-102909285553, 2009.

[Alba95] B. Al Badr, S. A. Mahmoud, "Survey and bibliography of Arabic optical text recognition", Signal processing, vol.41 (1), pp.49-77, 1995.

[Alma04] S. Alma'adeed, D. Elliman, and C.A. Higgins, "A Data Base for Arabic Handwritten Text Recognition Research", International Arab Journal of Information Technology, vol.1 (1), January 2004.

[Alma06] S. Al-Ma'adeed, "Recognition of off-line handwritten Arabic words using neural network", Proc. of GMAI'06, International Conference on Geometric Modeling and Imaging, pp.141-114, London, England, July 2006.

[Aloh02] Y. Al-Ohali, "Handwritten Word Recognition – Application to Arabic Cheque Processing", PhD Thesis, Concordia University, Montreal, Quebec, Canada, February 2002.

[Alpe97] A. Alper Atici, F. T. Yarman-Vural, "A heuristic algorithm for optical character recognition of Arabic script", Signal Processing, vol.62 (1), pp.87-99, October 1997.

[Alsa06] B. Alsallakh, H. Safadi, "AraPen : an Arabic online handwriting recognition system", Proc. of ICTTA'06, 2nd IEEE International Conference on Information & Communication Technologies : from Theory to Applications, vol.1, pp.1844-1849, Damascus, Syria, April 2006.

[Alsh08] A. AL-Shatnawi, O. Khairuddin, "Methods of Arabic Language Baseline Detection – The State of Art", IJCSNS International Journal, vol.8 (10), pp.137-143, 2008.

[Ameu04] M. Ameur, A. Bouhjar, F. Boukhris, A. Boukouss, A. Boumalk, M. Elmedlaoui, E. Iazzi, H. Souifi, "Initiation à la langue Amazighe", Publications de l'IRCAM, CAL, Rabat, 2004.

[Ameu06] M. Ameur, A. Bouhjar, F. Boukhris, A. Boukouss, A. Boumalk, M. Elmedlaoui, E. Iazzi, H. Souifi, "Graphie et orthographe de l'amazighe", Publications de l'IRCAM, CAL, Rabat, 2006.

[Ameu09] M. Ameur, A. Bouhjar, A. Boumalk, N. El Azrak, R. Laabdelaoui, "Vocabulaire de la langue Amazighe (amazighe-arabe)", IRCAM, 2009.

[Ameu92] A. Ameur, "Une méthode de reconnaissance de l'écriture manuscrite arabe", Thèse de Doctorat, Université de Rouen, Mont-Saint-Aignan, France, 1992.

[Amin03] A. Amin, "Recognition of hand-printed characters based on structural description and inductive logic programming", Pattern Recognition Letters, vol.24, pp.3187-3196, 2003.

[Amin80] A. Amin, A. Kaced, J. P. Haton, R. Mohr, "Hand written Arabic character recognition by the IRAC system", Proc. of 5th ICPR'80, , pp.729-731, Miami, Florida, USA, December 1980.

[Amin96a] A. Amin, S. Fischer, T. Parkinson, R. Shiu, "Fast algorithm for skew detection", SPIE Proceedings, 1996.

[Amin96b] A. Amin, H. Al-Sadoun, S. Fischer, "Hand-printed arabic character recognition system using an artificial network", Pattern Recognition, vol.29 (4), pp.663-675, April 1996.

[Amin97] A. Amin and W. Mansoon, "Recognition of printed Arabic text using Neural networks", Proc. 4th Int. ICDAR, Ulm, Germany, pp.612-615, 1997.

[Amro09] M. Amrouch, Y. Es Saady, A. Rachidi, M. El Yassa, D. Mammass, "Printed Amazigh Character Recognition by a Hybrid Approach Based on Hidden Markov Models and the Hough Transform", ICMCS'09, Ouarzazate, 2009.

[Amro10] M. Amrouch, A. Rachidi, M. El Yassa, D. Mammass, "Handwritten Amazigh Character Recognition Based On Hidden Markov Models", ICGST-GVIP Journal, vol.10, Issue 5, pp.11-18, 2010.

[Andr08] P. Andries, "Unicode 5.0 en pratique, Codage des caractères et internationalisation des logiciels et des documents", France, Dunod, Collection InfoPro, 2008.

[Anni94] L. Annick, "Lexicon reduction based on global features for on-line handwriting", IWFHR, 12 pp, 1994.

[Arri04] D. Arrivault, N. Richard, C. Fernandez-Maloigne, P. Bouyer, "Collaboration entre approche statistique et structurelle pour la reconnaissance de caractères anciens", CIFED'04, vol.1, pp.197-202, La Rochelle, Juin 2004.

[Arri06] D. Arrivault, "Apport des Graphes dans la Reconnaissance Non-Contrainte de Caractères Manuscrits Anciens", Rapport de thèse, Université de Poitiers, 2006.

[Ataa10] F. Ataa Allah, S. Boulaknadel, "Pseudo-racinisation de la langue amazighe", TALN 2010, Montréal, pp.19-23, juillet 2010.

[Awal09] A. Awal, H. Mouchère, C. Viard-Gaudin, "Towards handwritten mathematical expression recognition", Proc. of the 10th ICDAR, Barcelona, Spain, pp.1046-1050, 2009.

[Bagd97] A. Bagdanov, J. Kanai, "Projection Profile Based Skew Estimation Algorithm for JBIG Compressed Images", the 4th ICDAR, pp.401-405, 1997.

[Bagh05] M. S. Baghshah, S. B. Shouraki, S. Kasaei, "A novel fuzzy approach to recognition of online Persian handwriting", Proc. of the ISDA'05, 5th International Conference on Intelligent Systems Design and Applications, pp.268-273, Wroclaw, Poland, September 2005.

[Bair88] H. S. Baird, "Applications of Multi-dimensional Search to Structural Feature Identification", Syntactic & Structural Pattern Recognition, Springer-Verlag (Berlin), pp.137–149, 1988.

[Bair90] H. S. Baird, S. E. Jones and S. J. Fortune, "Image Segmentation by Shape-Directed Covers", 10th ICPR, Atlantic City, NJ, pp.820-825, June 1990.

[Bakk08] B. Bakkass, "Base lexicale et analyseur morphologie de l'amazighe", Rapport de DESA MIA, Faculté des sciences, Agadir, 2008.

[Ball06] G. Ball, S. N. Srihari and H. Srinivasan, "Segmentation free and segmentation dependent approaches to Arabic word spotting", IWFHR, pp.53–58, 2006.

[Bapt88] G. Baptista and K. M. Kulkarni, "A High Accuracy Algorithm for Recognition of hand-written numerals", Pattern Recognition, vol.4, pp.287-291, 1988.

[Basu10] J. K. Basu, D. Bhattacharyya, T. Kim, "Use of Artificial Neural Network in Pattern Recognition", International Journal of Software Engineering and Its Application, vol.4 (2), pp.23-34, April 2010.

[Bedd95] M. Bedda, M. Ramdani, N. Doghmane, "Classification des caractères arabes manuscrits par un perceptron multi-couches", Conférence méditerranéenne sur l'automatique MCEA, France, 1995.

[Bela92] A. Belaïd et Y. Belaïd, "Reconnaissance des formes, méthodes et applications", Interéditions, 1992.

[Bela97] A. Belaid et G. Saon, "Utilisation des processus markoviens en reconnaissance de l'écriture", Traitement du signal, vol.14 (2), pp.161–177, 1997.

[Bell01] A. Bellili, M. Gilloux and P. Gallinari, "An hybrid mlp-svm handwritten digit recognizer", In ICDAR'01, 2001.

[Bena99] N. Ben Amara, "Utilisation des modèles de Markov cachés planaires en reconnaissance de l'écriture arabe imprimée", Thèse de doctorat, Université des sciences, des Techniques et de médecine de Tunis II, 1999.

[Bene03] A. Bensefia, T. Paquet, L. Heutte, "Information retrieval based writer identification", 11th Conference of the International Graphonomics Society, IGS'2003, Scottsdale, Arizona, pp.320-323, 2003.

[Beno08a] A. Benouareth, A. Ennaji, M. Sellami, "Arabic handwritten word recognition using HMMs with explicit state duration", EURASIP Journal on Advances in Signal Processing, vol.2008, 13 pp, 2008.

[Beno08b] A. Benouareth, A. Ennaji, M. Sellami, "Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition", Pattern Recognition Letters, vol.29 (12), pp.742-752, September 2008.

[Berg98] S. Bergler, S. Khoury, B. C. Y. Suen, B. Waked, "Skew detection, page segmentation and script classification of printed document images", IEEE International Conference on Systems, Man, and Cybernetics, pp.4470-4475, 1998.

[Bhat09] U. Bhattacharya, B.B. Chaudhuri, "Handwritten Numeral Databases of Indian Scripts and Multistage Recognition of Mixed Numerals", IEEE Trans. on PAMI 31, 2009.

[Bish06] C. M. Bishop, "Pattern recognition and machine learning", Springer, 2006.

[Blum02] M. Blumenstein, C. K. Cheng, X. Y. Liu, "New Preprocessing techniques for Handwritten Word Recognition", Proc. of the 2nd IASTED Conf. on Visualization, Imaging and Image Processing , pp.480-484, 2002.

[Blum03] M. Blumenstein, B. Verma and H. Basli, "A novel feature extraction technique for the recognition of segmented handwritten characters", ICDAR '03, Edinburgh, Scotland, pp.137-141, 2003.

[Blum99] M. Blumenstein, and Verma B. "Neural Based Solutions for the Segmentation and recognition of difficult Handwritten Words from a Benchmark Database" Proc. of the Fifth ICDAR, pp.281-284, 1999.

[Bouc00] R. Bouché, H. Emptoz, L. Bourgeois, "DEBORA -Digital Acces to Books of Renaissance", Ouvrage en ligne, 167pp, http://rfv6.insa-lyon.fr/debora, Lyon, juin 2000.

[Bouc99] D. Bouchaffra, V. Govindaraju, and S. N. Srihari, "Postprocessing of recognized strings using nonstationary Markovian models", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.21, pp.990-999, 1999.

[Bouk08] F. Boukhris, A. Boumalk, E. El Moujahid, H. Souifi, "La Nouvelle Grammaire de l'Amazighe", IRCAM, CAL, Publications de l'IRCAM, Rabat, 2008.

[Bouk95] A. Boukous, "Société, langues et cultures au Maroc: Enjeux symboliques", Enjeux symboliques, Publications de la Faculté des Lettres de Rabat, Casablanca, Najah El Jadida, 1995.

[Boula09] S. Boulaknadel, "Amazigh ConCorde: an appropriate concordance for Amazigh", Actes du 1er symposium international sur le traitement automatique de la culture amazighe, SITACAM'09, pp.176-182, Agadir 2009.

[Boum09] A. Boumalk et K. Nait-Zerrad, "La Vocabulaire grammatical amazighe", IRCAM, CAL, Publications de l'IRCAM, Rabat, 2009.

[Brit04] A. S. Britto, R. Sabourin, F. Bortolozzi and C.Y. Suen, "Foreground and Background Information in an HMM-Based Method for Recognition of

Isolated Characters and Numeral Strings", 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9), pp 371-376, Tokyo, 2004.

[Brug97] R. Brugger, A. Zramdini and R. Ingold, "Modeling Documents for Structure Recognition Using Generalized n-grams", Proc. of the 4th ICDAR, pp.56-60, Ulm, Germany, August 1997.

[Case70] R. G. Casey, "Moment normalization of handprinted character", IBM Journal of Research and Development, Vol.14, pp.548–557, 1970.

[Chat06] C. Chatelain, "Extraction de séquences numériques dans des documents manuscrits quelconques", Thèse de doctorat, Université de Rouen 2006.

[Chan01] M.T. Chang, S.Y. Chen, "Deformed trademark retrieval based on 2D pseudo-hidden Markov model", Pattern Recognition, vol.34 (5), pp.953-967, Mai 2001.

[Chan99] F. Chang, K. H. Liang, T. M. Tan, W. L. Hwang, "Binarization of document images using Hadamard multiresolution analysis", Proc. of 5th ICDAR, pp.157–160, Bangalore, India, Sept 1999.

[Chen95] M. Y. Chen, A. Kundu, S. N. Srihari, "Variable duration hidden Markov model and morphological segmentation for handwritten word recognition", IEEE Transactions on Image Processing, vol.4 (12), pp.1675-1688, December 1995.

[Cher00] M. Cheriet N. E. Ayat and C. Y. Suen, "Un système neuro-flou pour la reconnaissance de montants numériques des chèques arabes", In CIFED'00, Lyon, France, juillet 2000.

[Cher07] M. Cheriet, N. Kharma, C. Liu, C. Y. Suen, "Character Recognition Systems A Guide for Students and Practioners", Published by John Wiley & Sons, Inc., Hoboken, New Jersey, 2007.

[Cher98] M. Cheriet, H. Miled, C. Olivier, "Visual aspect of cursive Arabic handwriting recognition", Proc. Vision Interface (VI'98), pp.262-270, 1998.

[Chev04] S. Chevalier, "Reconnaissance d'écriture manuscrite par des techniques markoviennes: une approche bidimensionnelle et générique", Thèse de Doctorat, Université René Descartes - Paris 5, 2004.

[Chin87] R.T. Chin, H. -K. Wan, D.L. Stover, and R.D. Iverson, "A one-pass thinning algorithm and its parallel implementation", Computer Vision, Graphics and Image Processing, vol.40, pp.30–40, 1987.

[Chom56] N. Chomsky, "Three models for the description of language", IRE Transactions on Information Theory, vol.2, pp.113–124, 1956.

[Conn02] S. D. Connell and A. K. Jain, "Writer Adaptation for Online Handwriting Recognition", IEEE Trans. on PAMI, vol.24 (3), pp.329–346, 2002.

[Cost00] F. Coste, "Apprentissage d'automates classifieurs en inférence grammaticale", Thèse de doctorat, Université de Rennes 1, 2000.

[Cost04] F. Coste, G. Kerbellec, B. Idmont, D. Fredouille, et C. Delamarche, "Apprentissage d'automates par fusions de paires de fragments significativement similaires et premières expérimentations sur les protéines MIP", In JOBIM. 32, 2004.

[Coua02] B. Coüasnon, J. Camillerapp, "DMOS: Une méthode générique de reconnaissance d'images de documents : évaluation sur 60 000 formulaire du XIXe siècle", CIFED'02, pp.225-234, Hammamet, Tunisie, Octobre 2002.

[Coua03] B. Couasnon, J. Camillerapp, "Accès par le contenu aux documents manuscrits d'archives numérisés", Document Numérique, vol.7 (3-4), pp.61-81, 2003.

[Cove67] T. M. Cover and P. E. Hart, "Nearest neigbor pattern classification", IEEE Transaction on Information Theory, IT- vol.13 (1), pp.123–140, janvier 1967.

[Darg94] P. Dargenton, "Contribution à la segmentation et à la reconnaissance de l'écriture manuscrite par l'ordinateur", Thèse de doctorat: INSA Lyon, 224pp, Décembre 1994.

[Dehg01] M. Dehghan, K. Faez, M. Ahmadi, M. Shridhar, "Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM", Pattern Recognition, vol.34 (5), pp.1057-1065, May 2001.

[Dela03] M. Delalandre, E. Trupin, and J.-M. Ogier, "Analyse structurelle en interprétation de documents: un bref survol", International Conference on Image and Signal Processing, pp.640-649, 2003.

[Djem97] A. Djematen, B. Taconet, A. Zahour, "A Geometrical Method for Printing and Handwritten Berber Character Recognition", ICDAR'97, pp.564, 1997.

[Djem98] A. Djematen, B. Taconet, A. Zahour: "Une méthode statistique pour la reconnaissance de caractères berbères manuscrits", CIFED'98, pp.170-178, 1998.

[Drew00] P.J. Drew, J.R.T. Monson, "Artificial neural networks", Surgery 127: pp.3-11, 2000.

[Driv95] D. Drivas and A. Amin, "Page Segmentation and Classification Utilizing Bottom-Up Approach", Proc 3rd ICDAR, Canada, pp.610-614, 1995.

[Duda00] Richard O. Duda, Peter E. Hart, and David G. Stork, "Pattern Classification (2nd Edition) ", Wiley-Interscience, 2 edition, November 2000.

[Duon05] Jean Duong, "Etude des Documents Imprimés: Approche Statistique et Contribution Méthodologique", Thèse de doctorat, INSA de Lyon, 2005.

[Dupo94] P. Dupont, "Regular grammatical inference from positive and negative samples by genetic search: the GIG method", In Grammatical Inference and Applications, ICGI'94, number 862 in Lecture Notes in Artificial Intelligence, pp.236–245. Springer Verlag, 1994.

[Dupo96] P. Dupont, "Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue", Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, 1996.

[Dupo98] P. Dupont, L. Miclet, "Inférence grammaticale régulière: fondements théoriques et principaux algorithmes", Technical report, INRIA N3449, 1998.

[Dupr03] 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.

[East97] B. Eastwood, A. Jennings, A. Harvey, "A Feature Based neural Network Segmenter for Handwritten Words", Int. Conf. Computational Intelligence and Multimedia Applications, Gold Coast, Australia, pp.286-290, 1997.

[Eikv91] L. Eikvil, T. Taxt, K. Moen, "A fast adaptative method for binarization of document images", Proc. of 1st International Conference on Document Analysis and Recognition ,Saint-Malo, France, vol.1, pp.435–443, 1991.

[Elay10] R. El Ayachi, K. Moro, M. Fakir, B. Bouikhalene, "On the Recognition of Tifinaghe Scripts", Journal of Theoretical and Applied Information Technology, vol.20 (2), pp.61-66, 2010.

[Elay11] R. EL Ayachi, M. Fakir and B. Bouikhalene, "Recognition of Tifinaghe Characters Using a Multilayer Neural Network", International Journal Of Image Processing (IJIP), vol. 5, Issue 2, 2011.

[Elba06] A. Elbaati, M. Kherallah, A. M. Alimi, A. Ennaji: "De l'Hors-Ligne Vers un Système de Reconnaissance En-Ligne: Application à la Modélisation de l'Écriture Arabe Manuscrite Ancienne", ANAGRAM'06, septembre 2006.

[Elha05] R. El-Hajj, L. Likforman-Sulem, C. Mokbel, "Arabic handwriting recognition using baseline dependent features and Hidden Markov Modeling", ICDAR 05, Seoul, Corée du Sud, 2005.

[Elha06] R. El-Hajj, C. Mokbel, L. Likeforman-Seulem, "Reconnaissance de l'écriture arabe cursive : combinaison de classifieurs MMCs à fenêtres orientées", Actes CIFED'06, pp.271-276, Fribourg, Suisse 2006.

[Elha07] R. El-Hajj, "Reconnaissance hors ligne de textes manuscrits cursifs par l'utilisation de systèmes hybrides et de techniques d'apprentissage automatique", Thèse de doctorat, ENST, Paris, Juillet 2007.

[Elha09] R. EL-Hajj, L. Likforman-Sulem, C. Mokbel, "Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition", IEEE PAMI, vol.31 (7), pp 1165-1177, 2009.

[ElYa02] A. El-Yacoubi, M. Gilloux and J.-M. Bertille, "A Statistical Approach for Phrase Location and Recognition within a Text Line: An Application to Street Name Recognition", IEEE Trans. on PAMI, vol.24, (2), pp.172–188, 2002.

[ElYa99] A. El-Yacoubi, M. Gilloux, R. Sabourin and C. Y.Suen, "An HMM Based Approach for Off-line Unconstrained Handwritten Word Modeling and Recognition", IEEE Trans. on PAMI, vol.21 (8), pp.752–760, 1999.

[Espo95] F. Esposito, D. Malerba and G. Semeraro, "A Knowledge-Based Approach to the Layout Analysis", Proc. of the 3rd International Conference on Document Analysis and Recognition, Montreal, Canada, pp.466-471, 1995.

[Essa08a] Y. Es Saady, A. Rachidi, M. El Yassa, D. Mammass, "Une méthode syntaxique pour la reconnaissance de caractères amazighs imprimés", CARI'08, Maroc, 27-31 Octobre 2008.

[Essa08b] Y. Es Saady, B. Bakkass, A. Rachidi, M. El Yassa, D. Mammass : "Développement d'un Éditeur de Texte Amazighe: État d'avancement et Perspectives", 3ème atelier international sur le thème "Les Technologies de l'information: statuts et opportunités pour l'amazighe", IRCAM, Rabat– Maroc, 24-25 novembre 2008.

[Essa09] Y. Es Saady, Y. Ait Ouguengay, A. Rachidi, M. Elyassa, D. Mammass, "Adaptation d'un correcteur orthographique existant à la langue Amazighe: cas du correcteur Hunspell", Actes du 1er symposium international sur le traitement automatique de la culture amazighe, SITACAM, Agadir, 2009.

[Essa10] Y. Es Saady, A. Rachidi, M. El Yassa, D. Mammass, "Printed Amazigh Character Recognition by a Syntactic Approach using Finite Automata", ICGST-GVIP Journal, vol.10, Issue 2, pp.1-8, 2010.

[Essa11a] Y. Es Saady, A. Rachidi, M. El Yassa, D. Mammass, "Reconnaissance Automatique de l'Ecriture Amazighe à base de Ligne Centrale de l'Écriture", 4ème Atelier international sur l'amazighe et les TIC, IRCAM, Maroc, 2011.

[Essa11b] Y. Es Saady, A. Rachidi, M. El Yassa and D. Mammass, "AMHCD: A Database for Amazigh Handwritten Character Recognition Research", International Journal of Computer Applications, vol.27 (4), pp.44-48, published by Foundation of Computer Science, New York, August 2011.

[Essa11c] Y. Es Saady, A. Rachidi, M. El Yassa, D. Mammass, "Amazigh Handwritten Character Recognition based on Horizontal and Vertical Centerline of Character", International Journal of Advanced Science and Technology, vol.33, pp.33-50, August, 2011.

[Faki09] M. Fakir, B. Bouikhalene, K. Moro, "Skeletonization Methods Evaluation for the Recognition of Printed Tifinaghe characters", SITACAM09, Agadir, 2009.

[Fara06] N. Farah, L. Souici, M. Sellami, "Classifiers combination and syntax analysis for Arabic literal amount recognition", Engineering Applications of Artificial Intelligence, vol.19 (1), pp.29-39, February 2006.

[Faro05] F. Farooq, V. Govindaraju and M. Perrone, "Preprocessing methods for handwritten Arabic documents", ICDAR'05, vol.1, pp.267–271, 2005.

[Fish90] J. Fisher, S. Hinds and K. D'Amato, "A Rule-Based System for Document Image Segmentation", Proc. of the 10th International Conference on Pattern Recognition, Atlantic City, USA, pp.113-122, 1990.

[Fred03] D. Fredouille, "Inférence d'automates finis non déterministes par gestion de l'ambiguïté, en vue d'applications en bioinformatique", Thèse de doctorat, Université de Rennes 1, 2003.

[Free74] H. Freeman, "Computer processing of line-drawing images", Computing Surveys, vol.6 (1), pp.57–97, March 1974.

[Fu74] K.-S. Fu, "Syntactic methods in pattern recognition", Academic Press, New York, 1974.

[Gaba05] E. Gabarra, A. Tabbone, "Combining global and local threshold to binarize document of images", Proc. of the 2d Iberian conference on pattern recognition and image analysis, pp.371–378, Juin 2005.

[Gaba08] É. Gabarra, "De la binarisation de documents vers la reconnaissance de symboles dans l'analyse de schémas électriques", rapport de thèse, Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour, 2008.

[Gade96] P.D. Gader and M.A. Khabou, "Automatic Feature Generation for Handwritten Digit Recognition", IEEE Trans. on PAMI, vol.18 (12), pp.1256–1261, 1996.

[Gold67] E.M. Gold, "Language identification in the limit", Information and Control, vol.10 (5), pp.447–474, 1967.

[Goss96] B. Gosselin, "Application De Réseaux De Neurones Artificiels a la Reconnaissance Automatique De Caractères Manuscrits", Thèse de doctorat de Faculté Polytechnique de Mons, 1996.

[Gran03] F. Grandidier, "Un nouvel algorithme de sélection des caractéristiques– Application à la lecture automatique de l'écriture manuscrite", Thèse de doctorat, École de Technologie Supérieure, Université du Québec, Jan 2003.

[Guo89] Z. Guo and R.W. Hall, "Parallel thinning with two-subiteration algorithms", Comm. ACM, vol.32 (3), pp.359–373, March 1989.

[Guyo03] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", JMLR, vol.3, pp.1157–1182, 2003.

[Hadj01] K. Hadjar, O. Hitz and R. Ingold, "Newspaper Page Decomposition using a Split and Merge Approach", Proc 6th ICDAR, USA, pp.1186-1189, 2001.

[Hadj06] K. HADJAR, "Une étude de l'évolutivité des modèles pour la reconnaissance de documents arabes dans un contexte interactif", Thèse de doctorat, Université de Fribourg (Suisse), 2006.

[Hall89] R.W. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation", Comm. ACM, vol.32 (1), pp.124–131, January 1989.

[Hans] Hunspell: http://hunspell.sourceforge.net/.

[Hayk99] S. Haykin, "Neural Networks: a Comprehensive Foundation", Prentice-Hall, New Jersey, USA, 1999.

[Héro01] P. Héroux, "Contribution au problème de la rétro-conversion des documents structurés", Thèse de doctorat, Université de Rouen, 2001.

[Heut94] L. Heutte. "Reconnaissance de caractères manuscrits: Application à la lecture automatique des chèques et enveloppes postales", Thèse de doctorat, Université de Rouen, 1994.

[Heut98] L. Heutte, T. Paquet, J. V. Moreau, Y. Lecourtier and C. Olivier, "A structural/statistical feature based vector for handwritten character recognition", Pattern Recognition Letters", vol.19, pp.629–641, 1998.

[Higu05] C. de la Higuera, "A bibliographical study of grammatical inference", Pattern Recognition, vol.38 (9), pp.1332–1348, septembre 2005.

[Higu97] C. de la Higuera, "Characteristic sets for polynomial grammatical inference", Machine Learning, vol.27 (2), pp.125–138, 1997.

[Hild69] J. Hilditch, "Linear skeletons from square cupboards", Machine Intelligence 4 (B. Meltzer and D. Michie, Eds.), pp.404–420, 1969.

[Hu93] T. Hu and R. Ingold, "A mixed approach toward an efficient logical structure recognition from document images", Electronic Publishing, vol.6 (4), pp.457-468, 1993.

[Huan03] L. Huang, G. Wan, and C. Liu, "An improved parallel thinning algorithm", ICDAR, vol.2, pp.780-783, Edinburgh, Scotland, August 2003.

[Huan95] Y. S. Huang and C. Y. Suen, "A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals", IEEE Trans on PAMI, vol.17 (1), pp.90–94, 1995.

[Hull94] J. Hull, "A database for handwritten text recognition research", IEEE Trans. on PAMI, vol.16 (5), pp.550–554, 1994.

[Iazz04] Institut Royal de la Culture Amazighe, Centre de l'Aménagement Linguistique, "Graphie de la langue Amazighe", Coordinateur El Mehdi Iazzi, Publications de l'IRCAM, Rabat, 2004.

[Iazz08] E. Iazzi, M. Outahajala, "Amazigh Data Base", Actes de L'atelier HLT & NLP within the Arabic world: Arabic language and local languages processing status updates and prospects, pp.36-39, 2008.

[Ingo91] R. Ingold and D. Armangil, "A Top-Down Document Analysis Method for Logical Structure Recognition", Proc 1st ICDAR, pp.41-49, France, 1991.

[Ircam03] Institut royal de la culture Amazighe, "Conception et mise au point des polices tifinaghes", CEISIC, cf. http://www.ircam.ma/fr/index.php?soc=telec&rd=1, plan d'action 2003.

[Ircam04a] Institut royal de la culture amazighe, "Proposition de codification du tifinaghe dans Unicode", Centre de l'aménagement linguistique, Rabat, avril 2004.

[Ircam04b] Institut royal de la culture Amazighe, "Polices et Claviers UNICODE", CEISIC, http://www.ircam.ma/fr/index.php?soc=telec&rd=3, 2004.

[Ircam04c] Institut royal de la culture Amazighe, "Proposition d'ajout de l'écriture Tifinaghe au répertoire de l'ISO/CEI 10646 (format Unicode) ", 21/06/2004, CEISIC, IRCAM, Rabat, 2004.

[Jain00] A. K. Jain, R. P.W. Duin and J. Mao, "Statistical pattern recognition: A review", IEEE Trans, on PAMI, vol.22 (1), pp.4–37, 2000.

[Jain96] A. Jain, B. Yu, "A robust and fast skew detection algorithm for generic documents", Pattern Recognition, pp.1599-1629, 1996.

[Jain98] A. K. Jain and B. Yu, "Document Representation and its Application to Page Decomposition", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.20 (3), pp.294-308, March 1998.

[Jang92] B. K. Jang and R. T. Chin, "One-pass parallel thinning: analysis, properties, and quantitative evaluation", Pattern Analysis and Machine Intelligence, IEEE Tran- sactions on, vol.14 (11), pp.1129–1140, Nov 1992.

[Jour06] N. Journet, "Analyse d'images de documents anciens: une approche texture", Thèse de doctorat, Université de La Rochelle, 2006.

[Juan91] B. H. Juang and L. R. Rabiner, "Hidden Markov models for speech recognition", Technometrics, vol.33 (3), pp.251–272, 1991.

[Kame06] S. KAMEL, "Lexique Amazighe de géologie", 2006, IRCAM, Rabat, Maroc.

[Kame93] M. Kamel, A. Zhao, "Extraction of binary character/graphics images from grayscale document images", CVGIP: Graphical Models and Image Processing, vol.55 (3), pp.203–217, 1993.

[Kapo03] R. Kapoor, D. Bagai and T. Kamal, "Representation and extraction of nodal features of DevNagri letters", ICVGIP, 2003.

[Kava01] E. Kavallieratou, N. Liolios, E. Koutsogeorgos, N. Fakotakis, G. Kokkinakis, "The GRUHD Database of Greek Unconstrained Handwriting", ICDAR, pp.561-565, 2001.

[Kava02] E. Kavallieratou, N. Fakotakis, G. Kokkinakis, "Skew angle estimation for printed and handwritten documents using the Wigner-Ville distribution", Image and Vision Computing, pp.813–824, 2002.

[Kerb08] G. Kerbellec, "Apprentissage d'automates modélisant des familles de séquences protéiques", Thèse de doctorat, Université Rennes 1, 2008.

[Kess09] Y. Kessentini, T. Paquet, A. Benhamadou, "A Multi-Lingual Recognition System for Arabic and Latin Handwriting", 10th ICDAR, pp.1196-1200, Espagne 2009.

[Kess10] Y. Kessentini, T. Paquet, A. Benhamadou, "Off-Line Handwritten Word Recognition Using Multi-Stream Hidden Markov Models", Pattern recognition Letters (PRL), vol 30, Issue 1, pp.60-70, January 2010.

[Keta10] D. Ketata, M. Khemakhem, "Un survol sur l'Analyse et la Reconnaissance de Documents: Imprimé, Ancien et Manuscrit", In CIFED 2010.

[Khar99a] N. Kharma and R. K. Ward, "Systèmes de reconnaissance de caractères pour les non-experts", IEEE Canadian Review – Summer ,1999.

[Khar99b] N. Kharma, M. Ahmed, R. Ward, "A New Comprehensive Database of Hand-written Arabic Words, Numbers, and Signatures used for OCR Testing", IEEE Canadian Conference on Electrical & Computer Engineering, pp.766-799, 1999.

[Khor03] M. S. Khorsheed, "Recognising handwritten Arabic manuscripts using a single hidden Markov model", Pattern Recognition Letters, vol.24 (14), pp.2235-2242, October 2003.

[Kim93] D. Kim, Y. Hwang, S. Park, E. Kim, S. Paek, S. Bang, "Handwritten korean character image database PE92", ICDAR, pp.470-473, 1993.

[Kimu94] F. Kimura, S. Tsuruoka, Y. Miyake and M. Shridhar, "A Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words", IEICE Trans. Inf. and Syst., vol.E77-D (7), pp.785–793, 1994.

[Kise98] K. Kise, S. Akinori, I. Motoi, "Segmentation of page images using the area Voronoi diagram", Computer Vision and Image Understanding archive, vol.70 (3), Special issue on document image understanding and retrieval, pp. 370-382, 1998.

[Kitt98] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On Combining Classifiers", IEEE Trans. Pattern Analysis and Machine Intelligence, vol.20 (3), pp.239-266, Mar. 1998.

[Klas02] T. J. Klassen, "Towards the on-line recognition of Arabic characters", Proc. of IJCNN'02, International Joint Conference on Neural Networks, pp.1900-1905, Honolulu, Hawaii, USA, May 2002.

[Klau02] A. Klautau, "Classification of peterson and barney's vowels using weka", Technical report, Universidade Federal do Par, 2002.

[Kner97] S. Knerr, V. Asimov, O. Baret, N. Gorsky and J.C. Simon D. Price, "The a2ia intercheque system: Courtesy amount and legal amount recognition for french checks", Automatic bankcheck processing, World Scientific, pp.43–86, 1997.

[Koch04] G. Koch, T. Paquet and L. Heutte, "Combination of contextual information for handwritten word recognition", IWFHR, pp.468–473, 2004.

[Koha95] R. Kohavi, "A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[Koho01] T. Kohonen, "Self-Organizing Maps", Springer Series in Information Science, 3rd edition, 2001.

[Kris93] M. Krishnamoorthy, G. Nagy, S. Seth and M. Viswanathan, "Syntactic Segmentation and Labeling of Digitized Pages from Technical Journals", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.15 (7), pp.737-747, July 1993.

[Le94] D. S. Le, G. R.Thoma and H. Wechsler, "Automatic page orientation and skew angle detection for binary document images", Pattern Recognition, vol.27, pp.1325-1344, 1994.

[Lebo91] F. Lebourgeois, "Approche mixte pour la reconnaissance des documents imprimés", Thèse de doctorat: Sciences, Institut National des Sciences Appliquées de LYON, LISPI, 174p, 1991.

[Lebo92] F. Lebourgeois, Z. Bublinski and H. Emptoz, "A Fast and Efficient Method for Extracting Text Paragraphs and Graphics From Unconstrained Documents", Proc. of the 11th ICPR, The Hague, pp.272-276, 1992.

[LeCu89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation, vol.1 (4), pp.541–551, 1989.

[LeCu94] Y. LeCun and Y. Benjio, "World-level training of a handwritten word recognizer based on convolutional neural netwoks", IAPR, editor, Proc. of the International Conference on Pattern Recognition, vol.2, pp.88-92, Jerusalem, October 1994.

[LeCu98] Y. LeCun, L. Bottou, Y. Bengio and P. Haffiner, "Gradient-based learning applied to document recognition", Proc. of the IEEE, vol.86 (11), pp.2278–2324, 1998.

[Lee93] S. W. Lee, J. S. Park, Y. Y. Tang, "Performance Evaluation of Nonlinear Shape Normalization Methods for the recognition of large-Sat handwritten Characters", ICDAR, IEEE, pp 402-407, 1993.

[Lema07] M. Lemaitre, "Approche markovienne bidimensionnelle d'analyse et de reconnaissance de documents manuscrits", thèse de doctorat, Université Paris 5 René Descartes, 2007.

[Lero05] A. Leroux, "Inférence grammaticale sur des alphabets ordonnés: application à la découverte de motifs dans des familles de protéines", Thèse de doctorat, Université de Rennes 1, 2005.

[Lero97] M. Leroux, E. Lethelier, M. Gilloux and B. Lemarie, "Automatic reading of handwritten amounts on french checks", Automatic bank check processing, World Scientific, pp.157–176, 1997.

[Lero97] M. Leroux, E. Lethelier, M. Gilloux and B. Lemarie, "Automatic reading of handwritten amounts on french checks", Automatic bank check processing. World Scientific, pp.157–176, 1997.

[Leth96] E. Lethelier, "Combinaison des concepts de segmentation et de reconnaissance pour l'écriture manuscrite hors ligne: application au traitement des montants numériques de chèques", Thèse de doctorat, 1996.

[Likf03] L. Likforman-Sulem, "Apport du traitement des images à la numérisation des documents manuscrits anciens", Document numérique, vol.7 (3-4), pp.13-26, 2003.

[Likf07] L. Likforman-Sulem, A. Zahour and B.Taconet, "Text line segmentation of historical documents: a survey", IJDAR, vol.9 (2), pp.123–138, 2007.

[Likf95] L. Likforman-Sulem, A. Hanimyan, C. Faure, "A Hough based algorithm for extracting text lines in handwritten documents", ICDAR, vol.2, pp.774–777, 1995.

[Lipp87] R. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, vol. 4 (2), pp.4-22, Avril 1987.

[Liu02] C. L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritting digit recognition using state-of-the-art techniques", International Workshop on Frontiers in Handwriting Recognition, pp.320–325, 2002.

[Liu97] C. -L. Liu, Y. -J. Liu, and R. -W. Dai, "Preprocessing and statistical/structural feature extraction for handwritten numeral recognition", In A. C. Downton and S. Impedovo, editors, Progress of Handwriting Recognition, World Scientific, Singapore, pp.161–168, 1997.

[Liu99] C.Liu, and M. Naagawa, "Handwritten numeral recognition using neural networks: improving the accuracy by discriminative training", Fifth ICDAR. pp.257-260, 1999.

[Lori05] L. Lorigo, V. Govindaraju, "Segmentation and Pre-Recognition of Arabic Handwriting", ICDAR 05, Corée du Sud, pp.605-609, 2005.

[Loul09] G. Louloudis, B. Gatos, I. Pratikakis et C. Halatsis, "Text line and word segmentation of handwritten documents", Pattern Recognition, vol.42 (12), pp.3169–3183, 2009.

[Lyva05] B. Ly Van, "Réalisation d'un Système de Vérification de Signature Manuscrite En-ligne Indépendant de la Plateforme d'Acquisition", Thèse de doctorat de l'Institut National des Télécommunications, Décembre 2005

[Macé09] S. Macé, E. Anquetil, "Eager interpretation of on-line hand-drawn structured documents: The DALI methodology", Pattern Recognition, vol.42, pp.3202-3214, 2009.

[Madd02] S. Snoussi Maddouri, H. Amiri, A. Belaid and C. Choisy, "Combination of Local and Global Vision Modeling for Arabic Handwritten Words Recognition", International Workshop on Frontiers in Handwriting Recognition, pp.128–135, 2002.

[Mamm99] D. Mammass, "Approches prétopologique et multi-échelles appliquées à l'extraction de l'écriture du fond de chèques bancaires, à la détection de contours et à la compression d'images", Thèse de Doctorat d'Etat, Univ Ibn Zohr, 1999.

[Mao03] S. Mao, A. Rosenfeld and T. Kanungo, "Document Structure Analysis Algorithms: A Literature Survey", Proc. SPIE Electronic Imaging, Santa Clara, California, USA, pp.197-207, January 2003.

[Mari08] S. Marinai, "Introduction to Document Analysis and Recognition", Studies in Computational Intelligence (SCI) 90, pp.1-20, Springer-Verlag Berlin Heidelberg, 2008.

[Mart01] U.V. Marti and H. Bunke, "Text Line Segmentation and Word Recognition in a System for General Writer Independent Handwriting Recognition", ICDAR, pp.159–163, 2001.

[Mart02] U.V. Marti, H. Bunke, "The IAM-database: an English sentence database for offline handwriting recognition", IJDAR, vol.5 (1), pp.39-46, 2002.

[Mena08] F. Menasri, "Contributions à la reconnaissance de l'écriture arabe manuscrite", Thèse de doctorat, Université paris Descartes, 2008.

[Micl79] L. Miclet, "Inférence des grammaires régulières", Thèse DI, ENST, 1979.

[Micl84] L. Miclet, "Méthodes structurelles pour la reconnaissance des formes", Ed, Eyrolles, collection technique et scientifique des télécommunications, 1984.

[Mile06] R. Milewski and V. Govindaraju, "Extraction of Handwritten Text from Carbon Copy Medical Form Images", DAS06, pp.106–116, 2006.

[Mile98] H. Miled, "Stratégies de résolution en reconnaissance de l'écriture semi-cursive: application aux mots manuscrits arabes", Thèse de doctorat, Université de Rouen, 1998.

[Mohr07] M. Mohri, F. C. N. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers", Handbook on Speech Processing and Speech Communication, Part E: Speech recognition, 2007.

[Mori06] M. Morita, R. Sabourin, F. Bortolozzi, C.Y. Suen, "Segmentation and Recognition of Handwritten Dates: An HMMMLP hybrid approach", IJDAR, 2006.

[Mota97] D. Motawa, A. Amin, R. Sabourin, "Segmentation of Arabic cursive script", Proc. of the ICDAR'97, vol.2, pp.625-628, Ulm, Germany, August 1997.

[Nagy84] G. Nagy, S. Seth, "Hierarchical representation of optically scanned documents", ICPR, pp.347-349, 1984.

[Nagy92] G. Nagy, S. Seth and M. Viswanathan, "A prototype document image analysis system for technical journals", Computer, vol.25 (7), pp.10-22, July 1992.

[Naka87] Y. Nakamura et al., "An optical character recognition system for industrial applications: TOSEYE-1000", Proc. Int. Workshop Industrial, Applications of Machine vision and Machine Intelligence, pp.364-368, Tokyo, 1987.

[Nama05] A. Namane, A. Guessoum and P. Meyrueis, "New Holistic Handwritten Word Recognition and Its Application to French Legal Amount", Springer-Verlag Berlin Heidelberg, pp.654-663, 2005.

[Nico09] A. Nicolaou et B. Gatos, "Handwritten text line segmentation by shredding text into its lines", Proc. ICDAR, pp.626–630, 2009.

[Niyo95] D. Niyogi and S. Srihari, "Knowledge-Based Derivation of Document Logical Structure", 3rd ICDAR, Montreal, Canada, pp.472-475, 1995.

[OGor93] L. O'Gorman, "The document spectrum for page layout analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.15, pp.1162-1173, 1993.

[OGor94] L. O'Gorman, "Binarization and multithresholding of document images using connectivity", CVGIP: Graphical Models and Image Processing, vol.56 (6), pp.494–506, 1994.

[Oliv06] L.S. Oliveira, M. Morita and R, Sabourin, "Feature Selection for Ensembles Applied to Handwriting Recognition", IJDAR, vol.18, pp.262–279, 2006.

[Olsz01] R. T. Olszewski, "Generalized Feature Extraction for Structural Pattern Recognition in Time Series Data", PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2001.

[Ostu79] N. Otsu, "A threshold selection method from gray-level histograms", IEEE Trans. Sys, Man., Cyber, vol.9, pp.62–66, 1979.

[Ota07] I. Ota, R. Yamamoto, S. Sako, S. Sagayama, "On-line Handwritten Kanji Recognition Based on Inter-stroke Grammar ", Proc. of the 9th ICDAR, vol.2, pp.1188-1192, 2007.

[Oula88] A. Oulamara, J Duvernoy, "An application of the Hough transform to automatic recognition of Berber characters", Signal Processing, vol.14, pp.79-90, 1988.

[Ouss08] M. Oussalah, "Content Based Image Retrieval: Review of State of Art and Future Directions", In IPTA' 08, pp.1–10, 2008.

[Outa11] M. Outahajala, Y. Benajiba, P. Rosso, L. Zenkouar, "POS Tagging in Amazighe Using Support Vector Machines and Conditional Random Fields ", Natural Language Processing and Information Systems, Lecture Notes in Computer Science, Vol.6716/2011, pp.238-241, 2011.

[Pavl04] T. Pavlidis and J. Zhou, "Page segmentation and classification", Comput. Vision Graphics Image Process, Vol.54, pp.484-496, 1992.

[Pavl91] T. Pavlidis, J. Zhou, "Page segmentation by white streams", ICDAR, Saint-Malo, vol.2, pp.945-953, 1991.

[Pech02] M. Pechwitz, S.S. Maddouri, V. Maergner, N. Ellouze, H. Amiri, "IFN/ENIT-database of handwritten Arabic words", CIFED'02, pp.129-136, 2002.

[Pech03] M. Pechwitz, V. Maegner, "HMM-based approach for handwritten Arabic Word Recognition Using the IFN/ENIT– DataBase", ICDAR'03, Edinburgh, pp.890-894, 2003.

[Pera99] S. J. Perantonis, B. Gatos, N. Papamarkos, "Block decomposition and segmentation for fast hough transform evaluation", Pattern Recognition, pp.811–824, 1999.

[Pois05] E. Poisson, "Architecture et Apprentissage d'un Système Hybride Neuro-Markovien pour la Reconnaissance de l'Écriture Manuscrite En-Ligne", Rapport de thèse, Univ de Nantes, 2005.

[Powa94] R. K. Powalka, N. Sherkat and R. J. Whitrow, "The Use of Word Shape Information for Cursive Script Recognition" IWFHR, pp.67–76, 1994.

[Rach05a] A. Rachidi, D. Mammass, "Informatisation de la Langue Amazighe: Méthodes et Mises En Œuvre", SETIT 2005 3rd International Conférence– Tunisia, March 2005.

[Rach05b] A. Rachidi, et D. Mammass, "Methods and developments in the creation of a computerised Amazigh script", European Language Resources Association letter's, the ELRA Newsletter, vol.10 N°1, January-March 2005.

[Rach06a] A. Rachidi, M. El Yassa and D. Mammass, "A pretopological approach for handwritten isolated Arabic characters recognition", Conférence ISCCSP'06, Marrakech, 13-15 Mars 2006.

[Rach06b] A. Rachidi, M. El Yassa et D. Mammass, "Vers un Système de reconnaissance automatique de caractères manuscrits isolés basé sur une approche prétopologique neuronale", Proc. MCSEAI, Agadir, Décembre 2006.

[Rach07] A. Rachidi, D. Mammass, "Vers un système de traduction automatique en ligne des documents amazighs fondé sur les graphes UNL", Revue E-TI, ISSN 1114-8802, vol.4, Juin 2007.

[Rahm03] A.F.R. Rahman, M.C. Fairhurst, "Multiple classifier decision combination strategies for character recognition: A review", IJDAR, vol.5, pp.166–194, 2003.

[Rame05] J.Y. Ramel, S. Leriche, "Segmentation et analyse interactives de documents anciens imprimés", Traitement du Signal, vol.22 (3), pp.209-222, 2005.

[Rame89] S.R. Ramesh, "A generalized character recognition algorithm: A graphical approach", In Proc. of Pattern Recognition, pp.347-350, 1989.

[Rang06] Y. Rangoni, A. Belaid, "document logical structure analysis based on perceptive cycles", 7th IAPR workshop on Document Analysis Systems, Springer verlag (Ed), pp.117-128, 2006.

[Rath03] T. Rath, V. Lavrenko and R. Manmatha, "A statistical approach to retrieving historical manuscript images without recognition", Tech. rep., Center for Intelligent Information Retrieval technical, 2003.

[Razz10] M. Razzak, M. Sher, S. Hussain, "Locally baseline detection for online Arabic script based languages character recognition", International Journal of the Physical Sciences, vol.5 (7), pp.955-959, 2010.

[Rhee09] T. Rhee, J. Kim, "Efficient search strategy in structural analysis for handwritten mathematical expression recognition", Pattern Recognition, vol.42, pp.3192-3201, 2009.

[Roch97] E. Roche, Y. Schabes, "Finite-State language processing", Cambridge, Massachusetts, The Mit Press, 1997.

[Rome95] K. Romeo-Pakker, A. Ameur, C. Olivier and Y. Lecourtier, "Structural analysis of Arabic handwriting: segmentation and recognition", Machine Vision and Applications, vol.8, pp.232–240, 1995.

[Ron98] D. Ron, Y. Singer and N. Tishby, "On the learnability and usage of acyclic probabilistic fnite automata", Journal of Computer and System Sciences, vol.56, n° 2, pp.133-152, 1998.

[Rose58] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain", Physical Review, vol.65, pp.386, 1958.

[Rose84] A. Rosenfeld, "The fuzzy geometry of image subsets", Pattern Recognition Letters, Vol.2, pp.311–317, 1984.

[Ruto66] D. Rutovitz, "Pattern recognition", Journal of Royal Statistical Society, 129, pp.504–530, 1966.

[Saka97] Y. Sakakibara, "Recent advances of grammatical inference", Theoretical Computer Science 185, pp.15-45, 1997.

[Sann94] G. Sanniti di Baja, "Well-shaper, stable and reversible skeletons from the (3, 4)-distance transform", Journal of Visual Communication and Image Representation, vol.5, pp.107-115, 1994.

[Saon95] G. Saon, A. Belaïd and Y. Gong, "Stochastic Trajectory Modeling for Recognition of Unconstrained Handwritten Words", Proc. ICDAR, 1, pp.508-511, Montréal (Canada), août 1995.

[Saon97] G. Saon, "Modèles markoviens uni- et bidimensionnels pour la reconnaissance de l'écriture manuscrite hors-ligne", Thèse de Doctorat, Université Henri Poincaré - Nancy1, 1997.

[Sauv00] J. Sauvola, M. Pietikainen, "Adaptative document image binarization", Pattern Recognition, vol.33 (2), pp.225-236, Février 2000.

[Scha08] M. P. Schambach, J. Rottland and T. Alary, "How to Convert a Latin Handwriting Recognition System to Arabic", In International Conference on Frontiers of Handwriting Recognition, 2008.

[Schm96] C. Schmid, "Appariement d'images par invarients locaux de niveaux de gris", PhD thesis, INPG, 1996.

[Seni94] A. Senior, "Offline handwriting recognition using recurrent neural networks", PhD thesis, Cambridge University, 1994.

[Sezg04] M. Sezgin, B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation", Journal of ELectronic imagins, vol.13 (1), pp.146-165, Janvier 2004.

[Shan07] J. Shanbehzadeh, H. Pezashki, A. Sarrafzadeh, "Features Extraction from Farsi Handwritten Letters", Proc. of Image and Vision Computing, Hamilton, New Zealand, pp.35–40, December 2007.

[Shri84] M. Shridhar and A. Badreldin, "Recognition of isolated and connected handwritten numerals", Proc, IEEE International Conference on Systems, Man and Cybernetics, pp.142-146, 1984.

[Skou03] A. Skounti, A. Lemjidi, E. Nami, "Tirra- Aux origines de l'écriture au Maroc", Publications de l'IRCAM, Centre des Etudes Artistiques, des Expressions Littéraires et de la Production Audiovisuelle, IRCAM, Rabat, 2003.

[Smit85] J. W. T. Smithand and Z. Merali, "Optical Character Recognition: The Technology and its Application in Information Units and Libraries", Report 33, British Library, Boston Spa Wetherby, West Yorks, England, 1985.

[Snima04] Service de la Normalisation Industrielle Marocaine (SNIMA), Alphabet tifinaghe, projet de norme marocaine PNM 17.1.100, Rabat 2004.

[Somo99] P. Somol, P. Pudil, J. Novovicová et P. Paclík, "Adaptive floating search methods in feature selection", Pattern Recognition Letters, vol.20 (11–13), pp.1157–1163, November 1999.

[Soua02] S. Souafi, "Contribution à la reconnaissance des structures des documents écrits: Approche probabiliste", Thèse de doctorat, Université Laval, Institut National des Sciences Appliquées de Lyon, 2002.

[Soui06] L. Souici N. Farah and M. Sellami, "Classifiers combination and syntax analysis for arabic literal amount recognition", Engineering Applications of Artificial Intelligence, vol.19 (1), pp.29–39, 2006.

[Srih05] S. N. Srihari, H. Srinivasan, P. Babu and C. Bhol, "Handwritten Arabic word spotting using CEDARABIC document analysis system", Proc. of SDIUT, pp.123–132, 2005.

[Su07] T.-H. Su, T.-W. Zhang and D.J. Guan, "Corpus-based HIT-MW database for offline recognition of general-purpose Chinese handwritten text", IJDAR, vol.10 (1), pp.27-38, 2007.

[Suen92] C. Y. Suen, Ch. Nadal, R. Legault, T.A. Mai et L. Lam, "Computer recognition of unconstrained Handwritten Numerals", Proc. of the IEEE, vol.80 (7), pp.1162-1180, juil. 1992.

[Syia06] M. Syiam, T. M. Nazmy, A. E. Fahmy, H. Fathi and K. Ali, "Histogram clustering and hybrid classifier for handwritten arabic characters recognition", Proc. of the 24th IASTED, pp.44–49, 2006.

[Szmu97] M. Szmulo, "Boundary normalization for recognition of non touching non-degraded characters", ICDAR, IEEE, pp 463-466, 1997.

[Tabb03] S. Tabbone, L. Wendling, "Multi-scale binarization of images, Pattern Recognition Letters", Elsevier Science Publishers B.V. North Holland, vol.24 (1–3), pp.403–411, 2003.

[Tabb06] S. Tabbone, T. Nguyen, G. Masini, "Une méthode de binarisation hiérarchique floue", CIFED'06, France, 2006.

[Tao02] W. B. Tao, J. W. Tian, J. Liu, "Image segmentation by three-level thresholding based on maximum fuzzy entropy and genetic algorithm", Pattern Recognition Letters, vol.24, no 16, pp.3069–3078, 2002.

[Tapp91] C. C. Tappert, "Speed, Accuracy, and Flexibility Trade-Offs in On-Line Character Recognition", IJPRAI, vol.5 (1-2), pp.79-95, 1991.

[Touj03] S. Touj, N. B. Amara and H. Amiri, "Generalized Hough transform for Arabic optical character recognition", ICDAR'03, pp.1242-1246, 2003.

[Trie95b] O. D. Trier and T. Taxt, "Evaluation of binarization methods for document images", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.17, pp.312–314, 1995.

[Trie95] I. Trier, A. Jain and T. Taxt, "Feature extraction methods for character recognition - A survey", Pattern Recognition, vol.29 (4), pp.641-662, 1995.

[Trie95a] O. D. Trier and A. K. Jain, "Goal-directed evaluation of binarization methods", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.17, pp.1191–1201, 1995.

[Tsuk88] J. Tsukumo and H. Tanaka, "Classification of handprinted Chinese characters using nonlinear normalization and correlation methods", In Proc. of the 9th ICPR, Italy, pp.168–171, 1988.

[Vapn95] V. Vapnik, "The Nature of Statistical Learning Theory", Springer, 1995.

[Verm03] B. Verma, "A contour code feature based segmentation for handwriting recognition", ICDAR'03, pp.1038-1042, 2003.

[Verm98] B. Verma, M. Blumenstein, and S. Kukarni, "Recent Achievements in Off-line Handwriting Recognition Systems", International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '98), Australia, pp.27-33, 1998.

[Vinc02] A. Vinciarelli, "A survey on off-line Cursive Word Recognition", Pattern Recognition, The journal off pattern recognition society, vol.35 (7), pp.1433-1446, 2002.

[Vinc04] A. Vinciarelli, S. Bengio and H. Bunke, "Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models", IEEE Trans. On PAMI, vol.26 (6), pp.709–720, 2004.

[Wagn74] R. A. Wagner, "Order-n correction for regular Languages", CACM, vol.17 (5), 1974.

[Ware00] M. Ware, "WEKA Documentation", University of Waikoto, 2000.

[Webb02] A. Webb, "Statistical Pattern Recognition", John Wiley & Sons Inc., 2002.

[Wiki06] http://fr.wikipedia.org/wiki/Fichier:Tinifagh_intedeni.jpg, 2006.

[Witt05] I. H. Witten, E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques", San Francisco: Morgan Kaufmann Publishers, sec edition, 2005.

[Wong02] K.W. Wong, C.S. Leung and S.J. Chang, "Handwritten digit recognition using multi-layer feedforward neural networks with periodic and monotonic activation functions", ICPR, vol.3, pp.106–109, 2002.

[Wong82] K. Y. Wong, R.G. Casey, F.M. Wahl, "Document Analysis System", IBM journal of Research Development, vol.26 (6), pp.647-656, 1982.

[Wu92] R.-Y. Wu and W. -H. Tsai, "A new one-pass parallel thinning algorithm for binary images", Pattern Recognition Letters, vol.13, pp.715–723, October 1992.

[Xu92] L. Xu, A. Krzyzak and C. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition", IEEE Trans. Systems Man Cybernet, vol.22 (3), pp.418–435, 1992.

[Xue06] H. Xue and V. Govindaraju, "Hidden Markov Models Combining Discrete Symbols and Continuous Attributes in Handwriting Recognition", IEEE Trans. On PAMI, vol.28 (3), pp.458–462, 2006.

[Yama83] Y. Yamashita, K. Higuchi, Y. Yamada, and Y. Haga, "Classification of handprinted Kanji characters by the structured segment matching method", Pattern Recognition Letters, Vol.1, pp.475–479, 1983.

[Yama90] H. Yamada, K. Yamamoto, and T. Saito, "A nonlinear normalization method for hanprinted Kanji character recognition—line density equalization", Pattern Recognition, vol.23 (9), pp.1023–1029, 1990.

[Yang00] Y. Yang, H. Yan, "An adaptive logical method for binarization of degraded document images", Pattern Recognition, vol.33 (5), pp.787–807, 2000.

[Yoko06] M. Yokobayashi and T. Wakahara, "Binarization and recognition of degraded characters using a maximum separability axis in color space and gat correlation", 18th ICPR 2006, vol.2, pp.885–888, August 2006.

[Yves03] Yves et C. Gauthier, "Quelques exemples récents d'utilisation des Tifinagh en Immidir (Algérie) et remarques sur des caractères dits 'sahariens", Cahiers de l'AARS No.8, pp.33-40, 2003.

[Zenk03] L. Zenkouar, Y. Belkasmi et Y. Aït Ouguengay, "Élaboration d'une première version du clavier Amazighe", IRCAM, CEISIC, Rabat, plan d'action 2003, http://www.IRCAM.ma/Telecharger/pilote.htm.

[Zenk04] L. Zenkouar, "L'écriture Amazighe Tifinaghe et Unicode, in Études et documents berbères", n° 22, pp.175-192, Paris (France), 2004.

[Zenk08] L. Zenkouar, "Normes des technologies de l'information pour l'ancrage de l'écriture Amazighe", Études et Documents Berbères, n°27, pp.159-172, 2008.

[Zerm07] N. Zermi, M. Ramdani, M. Bedda, "Arabic handwriting word recognition based on hybride HMM/ANN approach", International Journal of Soft Computing, vol.2 (1), pp.5-10, 2007.

[Zhan00] G.P. Zhang, "Neural Networks for Classification: A Survey", IEEE Trans on Systems, Man and Cybernetics - Part C, vol.30, no. 4, pp.641–662, 2000.

[Zhan04] D. Zhang and G. Lu, "Review of shape representation and descriptions techniques", Pattern recognition, vol.37 (1), pp.1-19, 2004.

[Zhan84] T.Y. Zhang and C.Y. Suen, "A fast parallel algorithm for thinning digital patterns", Communications of the ACM, vol.27 (3), pp.236–240, mars 1984.

[Zhen97] J. Zheng, X. Ding, Y. Wu, "Ecognizing On-line Handwritten Chinese Character via FARG Matching", Proc. of the 4th ICDAR, pp.621-624, 1997.

[Zhon99] D. X. Zhong and H. Yan, "Pattern skeletonization using run-length-wise processing for intersection distortion problem", Pattern Recognition Letter, vol.20 (8), pp.833–846, 1999.

[Ziar09] M. Ziaratban, K. Faez, F. Bagheri, "FHT: An Unconstraint Farsi Handwritten Text Database", ICDAR'09, pp.281-285, 2009.

[Zoua04] H. Zouari, "Contribution à l'évaluation des méthodes de combinaison parallèle de classifieurs par simulation", Thèse de Doctorat, Université de Rouen, 2004.