REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :.....
Série :.....

Thèse

Présentée en vue de l'obtention du diplôme de docteur en sciences

Option : **Informatique**

Technique basée puzzle/SVM pour l'amélioration de la reconnaissance du texte arabe manuscrit

Par :

M. ZAIZ Faouzi

Soutenue le : 01/07/2017

Devant le jury :

DJEDI NourEddine	Professeur	Université de Biskra	Président
BABAHENINI Med Chaouki	Professeur	Université de Biskra	Rapporteur
BALLA Ammar	Professeur	ESI Oued Smar - Alger	Examinateur
ADLA Abdelkader	Professeur	Université d'Oran1	Examinateur
SERIDI Hamid	Professeur	Université de Guelma	Examinateur
MELKEMI KamalEddine	Professeur	Université de Biskra	Examinateur
DJEFFAL Abdelhamid	MCA	Université de Biskra	Invité

Dédicaces

Tout d'abord, je veux rendre grâce à Dieu, le Clément et le Très Miséricordieux pour son amour éternel. C'est ainsi que je dédie cette thèse à:

ma mère pour sa tendresse et mon père pour sa patience et encouragement
mes très chers frères et ma chère soeur pour leurs conseils,
mes cousins et cousines,
tous ceux que j'aime,
tous mes amis.

Remerciements

A Dieu, le tout puissant, nous rendons grâce pour nous avoir donné santé, patience, volonté et surtout raison.

En premier lieu, je tiens à remercier mon encadreur Mr. BABAHENINI Mohamed Chaouki qui m'a aidé et conseillé durant ce travail.

Je remercie également Mr. DJEFFAL Abdelhamid et Mr. MERAGHNI Djamel pour leur aide et encouragement.

Mes remerciements vont également aux membres de jury : Pr. DJEDI NourEddine, Pr. BALLA Ammar, Pr. ADLA Abdelkader, Pr. SERIDI Hamid et Pr. MELKEMI KamalEddine pour m'avoir honoré par leur évaluation de ce travail.

Enfin, je remercie tous ceux qui m'ont soutenu, encouragé et m'ont donné l'envie de mener à terme ce travail.

Résumé

Jusqu'aujourd'hui, la reconnaissance de caractères joue un rôle important dans plusieurs domaines tels que l'authentification de chèques bancaires, l'échange à distance des fichiers informatiques pour les télécommunications et l'authentification et l'identification des manuscrits. Elle permet de convertir une image de texte imprimée ou manuscrit en un texte codé par machine.

Nombreuses recherches ont été faites durant les dernières années afin d'améliorer le taux de reconnaissance des systèmes de reconnaissance du texte manuscrit Arabe. Plusieurs systèmes ont essayé d'utiliser divers techniques de post-traitement pour la sélection de mot telles que des techniques de votes et d'information contextuelles, ... etc.

Dans nos travaux précédents, nous avons proposé une technique basée sur le classificateur SVM pour la reconnaissance du manuscrit Arabe en se basant sur une méthode de segmentation à deux passes horizontale et verticale.

Dans ce travail, nous améliorons la technique de segmentation en intégrant une technique multi-scans (N-Scans) qui consiste à remplacer la segmentation en deux passes. En plus, nous ajoutons un algorithme du puzzle qui améliore le taux de reconnaissance surtout pour les caractères ambigus.

L'approche est testée sur la base du manuscrit Arabe IFN-ENIT. Elle donne des résultats encourageants et ouvre des perspectives dans le domaine de la reconnaissance du manuscrit Arabe.

Mots Clés: OCR, Manuscrit Arabe, PAW, SVMs, Puzzle.

Abstract

Until today, OCR keeps an important role demonstrated by various fields such as checks authentication, the remote exchange of computer files for telecommunications and authentication and identification of manuscripts. It converts printed and handwritten text images into a text encoded by machine.

Several researches have been done through the last years to improve the recognition rate of Arabic handwritten recognition systems. Many of these systems attempt to use different word selection techniques such as voting, contextual information, ... etc.

In our previous works, we proposed a technique based on SVM classifier to recognize Arabic handwritten based on two passes horizontal and vertical segmentation method.

In this work, we improve the segmentation technique to a technique based multi-scans (N-Scans) instead of two passes. Also, we add a Puzzle algorithm to improve the recognition rate especially for ambiguous characters.

The approach is tested on Tunisian database IFN/ENIT for handwritten Arabic. It gives encouraging results and opens other perspectives in the domain of Arabic handwritten recognition.

Keywords: OCR, Arabic handwritten, PAW, SVMs, Puzzle.

الملخص

حتى يومنا هذا، لا يزال التعرف على الكتابة (OCR: Optic Character Recognition) يلعب دور هام جدا يتبين من مختلف المجالات مثل مصادقة الشيكات المصرفية، وتبادل ملفات الكمبيوتر عن بعد لتوثيق و التعرف على المستندات المكتوبة بخط اليد.

العديد من البحوث تم إنجازها خلال السنوات الأخيرة بهدف تحسين مستوى التصنيف لأنظمة التعرف على النصوص المكتوبة باليدمن بينها الكثير من الأنظمة إعتمدت على إستخدام تقنيات مختلفة لإختيار الكلمات مثل الإنتخاب، المعلومات السياقية، . . . إلخ.

في أعمال سابقة، إقترحنا تقنية تعتمد على المصنف SVM لتصنيف خط اليد العربي إعتمادا على طريقة تجزئة بمرحلتين مرحلة أفقية وأخرى عمودية.

في هذا العمل، قمنا بتحسين أداء عملية التجزئة بإعتماد طريقة متعددة المسح (N-Scans) عوض السابقة التي تعتمد على مرحلتين . كما أضفنا خوارزمية الأحجية (puzzle) لتحسين مقدار التصنيف خاصة الأحرف غير الواضحة.

طربقتنا تستخدم مجموعة من المراحل(تنقية، تجزئة، إستخراج الخواص، التصنيف و التصنيف الإضافي) و تسمح بالحصول على نتائج أفضل.

تم تجربة هذه الطريقة على قاعدة خط اليد العربي التونسية IFN-ENIT. وهي تقدم نتائج مشجعة و تفتح أبعاد أخرى في مجال التعرف على خط اليد العربي.

كلمات مفتاحية: التعرف على الكتابة OCR، التجزئة، الحروف العربية، جزء كلمة، المعالجة، خوارزمية الأحجية Puzzle، المصنف SVM .

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction

Dans le domaine de l'intelligence artificielle, la reconnaissance de l'écriture est considérée comme l'un des problèmes les plus difficiles. Deux types distincts de reconnaissance d'écriture existent: reconnaissance hors-ligne et reconnaissance en-ligne. Le premier, consiste à déterminer les lettres ou les mots présents dans une image numérique d'un document manuscrit. Le second est relativement plus simple car il peut utiliser des informations supplémentaires non disponibles pour les systèmes hors ligne tels que l'information temporelle comme la pression et l'ordre séquentiel de l'écriture. En outre, le cas hors ligne est celui qui correspond à la tâche de lecture classique effectuée par les humains. Plusieurs domaines démontrent l'importance des systèmes similaires, tels que l'authentification de chèques bancaires, l'échange à distance des fichiers informatiques pour les télécommunications, l'authentification et l'identification des manuscrits, la reconnaissance des cartes cartographiques et la lecture automatique de documents administratifs [85, 83, 81]. Les techniques d'OCR (Optic Character Recognition) arabe ne sont pas encore bien développées, parce que la nature cursive de l'écriture arabe induit de nombreuses difficultés techniques. Les systèmes OCR commencent par l'acquisition d'une image brute, puis procède à sa binarisation, et son codage. Ensuite, les images de l'écriture passent par une phase de segmentation afin d'extraire les segments de base. Par la suite, ses segments sont analysés pour extraire les caractéristiques essentielles. Puis, ces caractéristiques passent par une phase d'apprentissage où l'utilisateur doit introduire un ensemble de données types qui sera utilisé par un classificateur (par exemple, ANNs: Artificial Neural Networks, HMMs: Hidden Markov Models, KNNs: K-Nearest Neighbors ou SVMs: Support Vector Machine) pour créer des modèles afin de prendre des décisions durant la phase de décision. Après cela, de nombreux systèmes appliquent une étape de post-traitement pour faire deux tâches principales: la reconstruction d'unités de texte et la sélection de mot. La première, reconstruit les différentes unités du texte (caractères, mots et linges) à partir d'un ensemble de formes de base prédéfinies ou des étiquettes des segments en se basant sur les modèles du langage (règles de composition et dictionnaires). Tandis que la seconde, sélectionne les meilleurs mots cibles en utilisant plusieurs méthodes de sélection des mots telles que le

vote et l'information contextuelle[2, 6].

Problématique

Le script arabe, est par nature cursif, et écrit de droite à gauche. Souvent, les représentations des caractères ou leurs formes change avec chaque occurrence, et dépend de ses positions dans le mot (début, milieu, fin et isolée), ce qui rend l'Arabe très différentes des autres langages. Un texte arabe est constitué de plusieurs lignes, et chaque ligne contient un ensemble de mots, qui sont en outre divisés en composants connectés ou sous-mots, appelée parfois Peace of Arabic Words (PAWs). Dans un texte imprimé, l'espace entre mot est souvent plus grand que celui entre les sous-mots. Par contre, dans un texte manuscrit, il varie, ce qui conduit à une incohérence dans le processus de segmentation des mots et PAWs. Cette irrégularité génère plusieurs problèmes tels que [61, 60]:

- des chevauchements et des coupures entre et dedans les caractères.
- des chevauchements entre les mots.
- espaces entre mots irréguliers.
- des chevauchements entre les lignes.
- fluctuations des lignes (interlignes irréguliers).

Pour ces raisons, la reconnaissance du manuscrit arabe reste un défi, et nécessite des recherches futures. L'objectif de ce travail de thèse est d'étudier la possibilité d'utiliser un algorithme puzzle en post-traitement pour améliorer le taux de reconnaissance d'un texte manuscrit arabe. Ce post processeur va résoudre les problèmes des chevauchements et des coupures entre et dedans les caractères, en considérant un texte manuscrit comme un jigsaw puzzle [59, 87, 116].

Contributions

Dans ce travail, nous allons proposer deux contributions afin de remédier au problème de limites non claires entre caractères causée par les coupures et chevauchements entre caractères. La première contribution, présente une méthode structurelle basée multi-scans (N-Scans), qui a pour rôle de préparer les différents segments de base présentant par la suite l'état initial d'un puzzle. Premièrement, cet algorithme segmente le texte en un ensemble de PAWs ou composants connectés. Ensuite, ces PAWs sont divisés en graphèmes de base prédéfinies (voir le tableau 2.4). La deuxième contribution, présente un post-processeur à base de puzzle permettant de reconstruire les mots, ainsi que de sélectionner le mot adéquat à une séquence d'étiquette. Ce post-processeur est capable

de corriger certains cas d'erreurs en utilisant trois étapes de puzzling: Fusion, Division et Morphisme (voir section 4.1.6).

Organisation de la thèse

Le reste du travail est organisé comme suit:

Le premier chapitre présente un état de l'art sur les systèmes de reconnaissance de l'écriture manuscrite. D'abord, nous allons présenter la situation et les problèmes liées à ces systèmes. Ensuite, nous allons donner un aperçu sur l'architecture d'un OCR. Par la suite, nous allons décrire quelques caractéristiques de la langue Arabe. Finalement, nous donnons les problèmes ou les challenges qui confrontent la réalisation d'un système de reconnaissance de l'écriture manuscrite arabe.

Le deuxième chapitre présente un état de l'art sur la segmentation de l'écriture manuscrite. Dans ce chapitre, nous allons voir la problématique posée par l'écriture arabe cursive. Ensuite, nous présentons les différentes approches et stratégies de segmentation. Puis, nous décrivons la première contribution présentée par une méthode de segmentation structurelle multi-scans.

Le troisième chapitre va mettre l'accent sur les différentes méthodes de classification, parmi les quelles nous trouvons la technique de Machines à Vecteurs de Support (SVM) utilisée par notre système. En plus, nous allons voir les différentes méthodes de sélection de mot en post-traitement. Enfin, nous présentons la deuxième contribution présentée par l'intégration de l'algorithme du puzzle pour la construction et la sélection des mots.

Le quatrième chapitre présente la validation des résultats. Dans lequel, nous allons voir comment fonctionne globalement notre système. Parmi ces étapes nous présentons l'algorithme de segmentation d'un texte arabe manuscrit en un ensemble de segments de base. En plus, nous exposons en détails les étapes du puzzle. Enfin, nous présentons les résultats obtenus ainsi que les discussions faites.

Nous terminons le travail par une conclusion sur les résultats obtenus par les méthode utilisées, et des perspectives de ce travail.

Chapitre 1

Reconnaissance de l'écriture manuscrite

Introduction

La reconnaissance de l'écriture consiste à transformer l'image d'un texte (imprimé ou manuscrit) donnée en un texte dans un fichier sous format numérique, en essayant d'imiter la faculté de lecture de l'être humain. Des systèmes pareils sont très importants pour des différents domaines tels que la lecture automatique des chèques, des formulaires administratifs et des adresses postales. Malheureusement, jusqu'aujourd'hui la réalisation d'un système de bonne performance reste un challenge ouvert pour plusieurs langues (telles que l'Arabe). L'acquisition de l'image du texte à traiter peut être réalisée au cours de l'écriture dans le cas des systèmes en-ligne, ou depuis un document déjà écrit dans le cas des systèmes hors-ligne. Le premier cas est relativement plus simple vu la possibilité d'avoir les coordonnés et d'autres informations en temps réel. En outre, le second cas est le plus proche à la tâche de lecture des humains.

L'objectif de ce chapitre consiste d'une part à introduire et à présenter un état de l'art du domaine de la reconnaissance de document, ce qui nous permettra de situer le problème de l'OCR, et d'autre part à exposer les différentes approches, méthodes et techniques réalisées depuis plusieurs années dans le domaine de reconnaissance du manuscrit.

1.1 Situation du problème

Dans cette section, nous allons voir la différence entre la reconnaissance et la production d'un document, ainsi que la différence entre la structure physique et la logique de celui-ci. Ensuite, nous allons présenter les problèmes majeurs à surmonter pour la réalisation des systèmes de reconnaissance du manuscrit.

1.1.1 Production et reconnaissance

La reconnaissance consiste à transformer une image d'un document en une représentation sous forme d'un fichier informatique. Donc, il s'agit de faire l'inverse des étapes du processus de production (voir la figure 1.1). D'abord, elle fait l'acquisition de l'image d'un document au moyen d'un outil d'acquisition tel que: scanner, camera ou à partir d'une base de données. Souvent, cette image contient des bruits qui doivent être éliminés au maximum pour obtenir une image avec une qualité améliorée, et prête pour des traitements futurs (voir la figure 1.2).

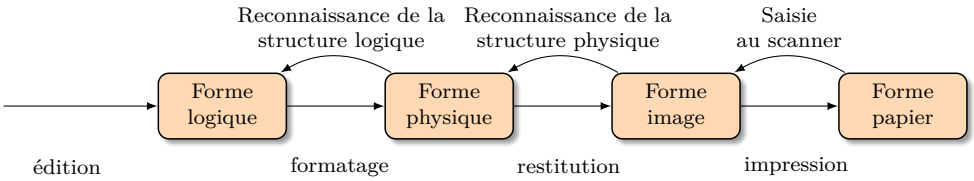


Figure 1.1: Processus de production et de reconnaissance de documents [96].

Ensuite, l'image pré-traitée doit subir à une séquence d'opérations afin d'identifier les différentes parties ou segments (zone de texte, image, graphique, tableau, ... etc.). Cette tâche est appelée la reconnaissance de la structure physique. A partir de cette dernière le système est capable de donner une étiquette aux différents objets. En plus, il réorganise ces objets selon leur flux de lecture afin de construire une structure logique du document source [96, 67].

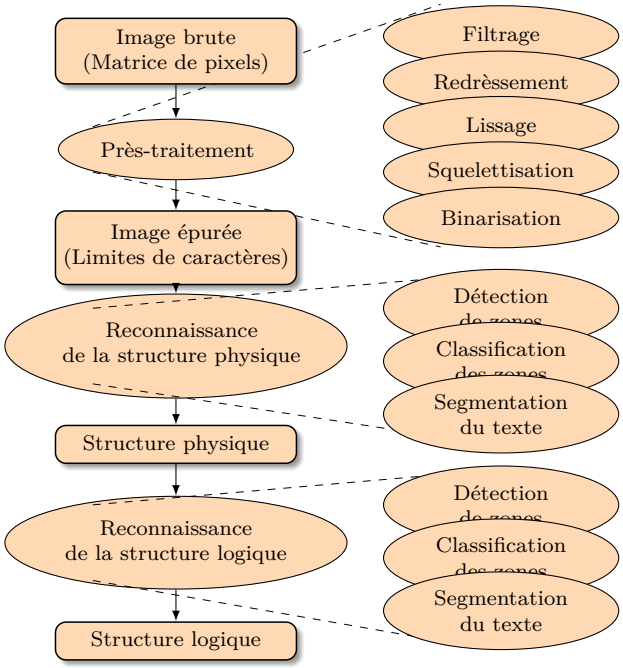


Figure 1.2: Etapes de la reconnaissance de documents [96].

La structure d'un document peut être vue selon le nombre d'objets composant le document, ce qui donne une autre classification selon le niveau de complexité [55, 96]:

- Structure simple, où le document contient uniquement des objets simples, par exemple: texte.
- Structure complexe, document contenant des objets complexes, par exemple: texte, images, schémas... etc.

1.1.2 Problèmes liés à l'OCR

Plusieurs problèmes peuvent compliquer la tâche d'un système OCR, parmi ces problèmes nous pouvons citer [22]:

- Disposition spatiale du texte,
- Nombre de scripteurs,
- Taille du vocabulaire.

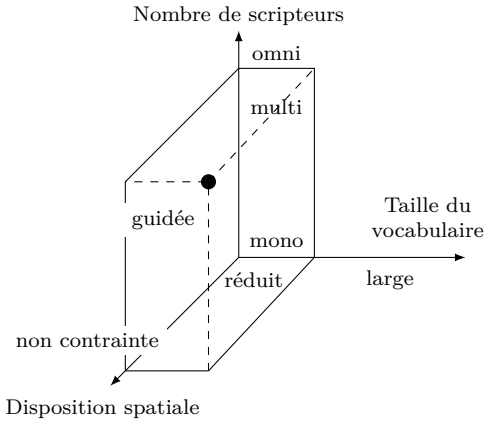


Figure 1.3: Graphe de complexité des systèmes de reconnaissance [22].

A. Disposition spatiale du texte

Principalement, deux types de variation peuvent influencer sur l'écriture manuscrite : variations propres au scripteur, et variations propres à l'écriture manuscrite. La première, se présente par le style d'écriture personnel en termes de rapidité, de continuité et de régularité. Ces facteurs influent considérablement sur les formes des caractères et les liaisons entre ces caractères [108]. La seconde, présente le changement des formes des caractères et des caractères voisins à cause de leurs positions dans le mot (début, milieu, fin). Donc, le même caractère peut avoir plusieurs représentations appelées allo-graphes (voir figure 1.4) [22].

B. Nombre de scripteurs

Le nombre de scripteurs présente un défi qui s'accroit selon ce nombre, trois classes peuvent être retirées : mono, multi et omni scripteurs. La première, est la plus simple

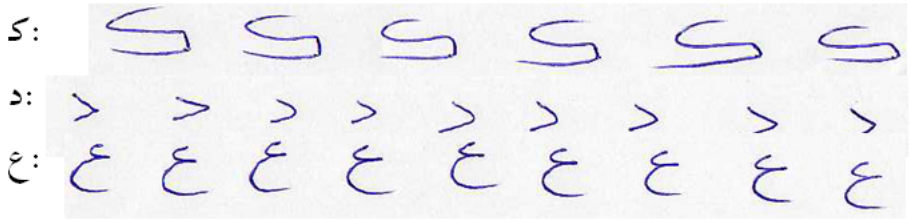


Figure 1.4: Exemples d'allographes des caractères arabes.

parce qu'un seul style d'écriture est présent dans le texte. Par contre, dans le second cas il faut que le système s'adapte avec plusieurs styles d'écriture. La troisième classe, présente le cas le plus complexe parce que le système doit être capable de reconnaître tous les types d'écriture [22].

C. Taille du vocabulaire

La taille du vocabulaire influe considérablement sur la nature du système à développer. Dans le cas d'un vocabulaire limité (inférieur de 100 mots), nous pouvons remarquer que le système est relativement simple, et entre généralement dans le cadre des systèmes globaux. Par contre, avec un vocabulaire étendu (supérieur à 10 000 mots) le système nécessite des traitements complexes, et il doit intégrer des mécanismes plus sophistiqués, il est alors classé dans la catégorie des systèmes analytiques [22].

1.2 Classification des OCRs

Généralement, la classification des OCRs est faite selon trois axes principaux : l'outil d'acquisition, les approches de reconnaissance utilisées et les techniques d'extraction des caractéristiques (voir la figure 1.5). Le premier axe, donne deux classes : reconnaissance en-ligne (en anglais on-line), et reconnaissance hors-ligne (en anglais off-line). Le second axe, les classes sont définies selon les modèles de traitement d'image. Si le traitement se fait sans segmentation la reconnaissance est dite globale. Dans le cas contraire, elle est considérée comme analytique. Le dernier axe, voit le système de reconnaissance selon les méthodes d'extraction des caractéristiques utilisées : statistiques, structurelles ou stochastiques [56, 38, 9].

1.2.1 En-ligne vs hors-ligne

La reconnaissance en-ligne consiste à réaliser parallèlement ou en temps réel les traitements avec l'écriture. Elle est réservée à l'écriture manuscrite. Dans ce cas, il est nécessaire d'utiliser un équipement tel qu'une tablette graphique ou un stylo électronique. Elle possède donc l'avantage d'avoir des informations supplémentaires telles que les mouvements et la pression [119, 48]. En plus, elle permet de corriger et modifier l'écriture de manière interactive [38]. Par contre, La reconnaissance hors-ligne (ou en différé, ou encore statique) utilise un scanner ou camera pour acquérir une image binaire

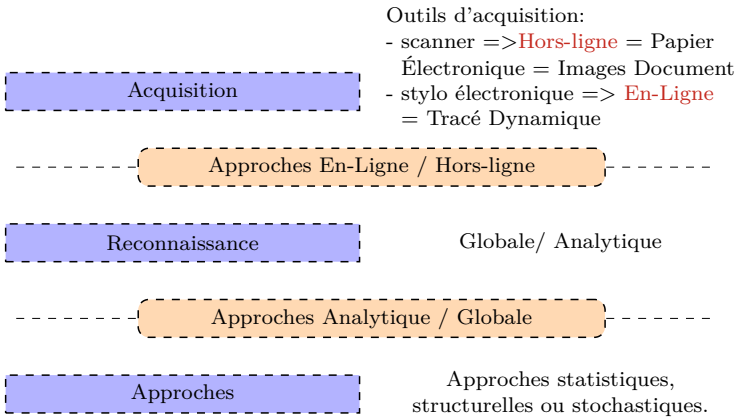


Figure 1.5: Classification des OCRs [56].

ou en niveaux de gris d'un document existant. En outre, le cas hors-ligne est celui qui correspond à la tâche de lecture classique effectuée par les humains [56]. Dans ce cas, toutes informations temporelles sur l'ordre des points du tracé sont perdues. En plus, il faut aussi considérer le problème de variabilité de l'épaisseur et la forme des lignes d'écriture surtout l'écriture cursive [22]. Une brève comparaison est illustrée par le tableau 1.1.

Tableau 1.1: Comparaison brève entre les approches en-ligne et hors-ligne [9].

Critère de comparaison	En-ligne	Hors-ligne
Outils d'acquisition	– Stylo électronique plus tablette graphique.	– Scanner ou caméra.
Bruit d'image	– Faible	– existence d'un bruit important.
Informations disponibles	– la position, – la direction du mouvement, – les points de fin, – les points de début, – ordre des traits.	– absence d'informations contextuelles.

1.2.2 Approches de reconnaissance

A ce niveau, l'approche de segmentation permet de faire la différence. Pour un système qui cible un vocabulaire limité il est possible d'avoir des taux de reconnaissance acceptables sans avoir à faire une segmentation. Ce qui donne naissance au système de reconnaissance globale où on traite l'image du texte source directement sans faire une extraction des lignes, mots ou caractères, ce qui permet une modélisation plus efficace des variations de l'écriture et des dégradations qu'elle peut subir [108]. Par contre, si le vocabulaire est grand par exemple tous les mots de la langue, il est nécessaire de faire une segmentation pour extraire les différents segments de base à reconnaître. Souvent, on utilise comme unité les caractères ou des fragments morphologiques significatifs inférieurs au caractère appelés graphèmes. Par la suite, les mots et lignes sont reconstruis progressivement durant le processus de reconnaissance. Donc, on parle d'un système de reconnaissance

analytique [4, 111].

1.2.3 Nature des traits caractéristiques

Les approches d'extraction des caractéristiques d'une image d'un texte manuscrit varies beaucoup, mais, généralement nous pouvons les voir selon quatre catégories, à savoir [108, 35, 72]:

- caractéristiques topologiques,
- caractéristiques structurelles,
- caractéristiques statistiques,
- superposition des modèles et corrélation.

1.3 Architecture d'un OCR

D'abord, un système de reconnaissance de l'écriture manuscrite doit acquérir l'image du texte en utilisant une unité d'acquisition telle qu'un scanner ou une caméra. Ensuite, cette image est pré-traitée afin d'améliorer sa qualité. Par la suite, l'image pré-traitée passe par une phase de segmentation pour extraire les segments de base du système. Puis, une phase d'extraction des caractéristiques est nécessaire afin d'avoir une description des différents segments. Ces descriptions sont passées à un classificateur qui va attribuer un label ou étiquette à chaque description d'un segment. Finalement, des opérations de post-traitement peuvent être faites pour juger la classification ou pour faire des opérations de reconstructions des différents objets du document, par exemple : mots, lignes du texte,... etc [108, 9].

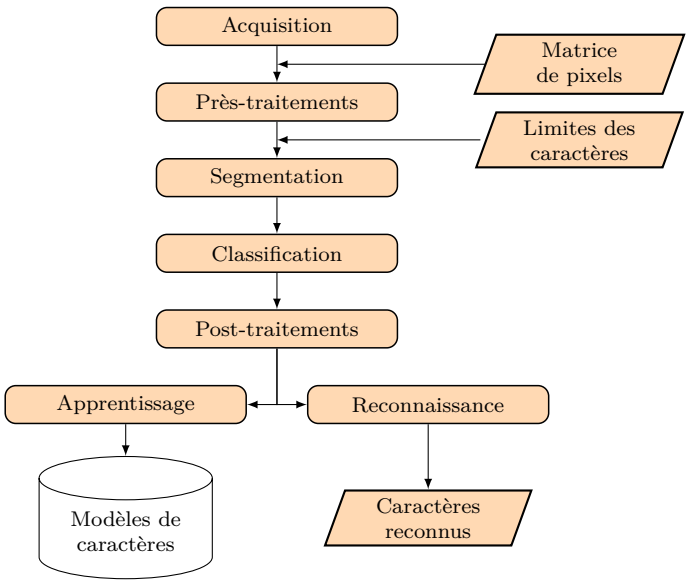


Figure 1.6: Schéma général d'un système de reconnaissance de caractères.

1.3.1 Acquisition

Cette phase utilise des unités de numérisation telle qu'un scanner ou une caméra afin de faire l'acquisition de l'image d'un texte écrit sur papier. Cette opération doit être faite avec le minimum de dégradations possible [108, 89].

1.3.2 Pré-traitements

La phase de pré-traitement consiste d'une part à améliorer la qualité de l'image captée en ne gardant que l'information significative de la forme représentée. Ce bruit peut être dû aux conditions d'acquisition (éclairage, mise incorrecte du document, ...) ou encore à la qualité du document d'origine [107, 89]. D'autre part, cette phase permet de créer d'autres représentations de l'image originale telle que la création d'une forme squelettique ou forme du contour afin de simplifier la phase d'extraction de caractéristiques [9]. Parmi les opérations de pré-traitement généralement utilisées nous pouvons citer :

- l'extraction des composantes connexes,
- le redressement de l'écriture,
- le lissage,
- la normalisation, et
- la squelettisation.

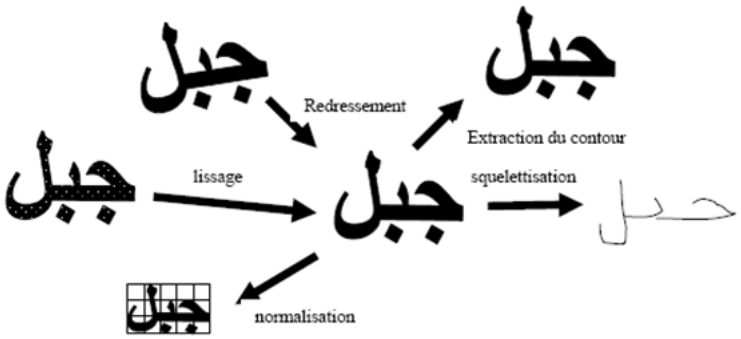


Figure 1.7: Effet de certaines opérations de prétraitement [108].

A. Extraction de composantes connexes

Dans le cas d'une écriture manuscrite cursive, les meilleures approches de segmentation sont celles basées sur l'extraction des composantes connexes. Ce dernier se représente par un ensemble de points dans le plan. Cet ensemble peut être un point diacritique, un accent, au corps d'un caractère ou d'une chaîne de caractères. Par la suite, ces CCs sont utilisés pour reconstruire le mot original [22].

B. Redressement de l'écriture

Le redressement de l'écriture consiste à faire une rotation isométrique sur l'image du texte avec une angle angle $-\alpha$ si α est l'angle d'inclinaison, dans le but de rendre les lignes du texte horizontales. Cette inclinaison peut être à cause de la saisie, si le document a été placé en biais, ou être intrinsèque au texte. [108].

$$\begin{cases} x' = x\cos\alpha + y\sin\alpha \\ y' = y\cos\alpha + x\sin\alpha \end{cases}$$

C. Lissage de l'écriture

Plusieurs facteurs tels que la qualité du document et l'outils d'acquisition peuvent causés une absence ou surcharge des points [108, 107]. Pour cela, le lissage vient pour rendre plus homogène les différentes parties de l'image sources. En plus, il élimine les fortes variations d'intensité lumineuse ponctuelles non significatives. Souvent, un filtre passe bas est appliqué, mais comme effet secondaire il rend l'image plus floue [78].

D. Normalisation

Elle consiste à rendre les différentes images des segments de mot en une taille standard. Cette normalisation est très importante pour certains systèmes, par exemple un système utilisant les réseaux de neurones. Elle permet d'augmenter la ressemblance entre les différents images traitées [107, 22].

E. Squelettisation

La squelettisation permet de créer une structure filiforme ou squelette du mot à partir de l'image du texte. Le principe est de trouver un axe médian possédant une égalité de distance par rapport aux pixels de frontières qui les entourent [4]. Elle peut causer des pertes d'information telle que les points diacritiques qui se transforment en un seul [108].

1.3.3 Segmentation

Cette phase consiste à extraire de l'image pré-traitée les différents segments essentiels aux traitements ultérieurs. Principalement, deux classes d'approches sont utilisées : approches ascendantes et approches descendantes. La première, consiste à extraire des éléments plus petits que le mot tel que graphèmes. Puis, reconnaître et construire le mot à partir de ces éléments progressivement. Contrairement, la seconde approche détecte les blocs de texte et des blocs graphiques. Ensuite, elle extrait les lignes de chaque bloc de texte. Finalement, de chaque ligne elle extrait les mots, et depuis ces derniers les caractères [78, 11].

1.3.4 Extraction des caractéristiques

Cette phase vise à donner une description aux différents segments en générant un vecteur de primitives de taille fixe. Ces primitives doivent permettre de distinguer entre les différentes classes. Devijver et Kittler dans [9], suggèrent d'extraire des primitives à base des informations significatives et appropriées afin de minimiser la variabilité entre les exemples de la même classe, et de maximiser la variabilité entre les exemples des classes [90]. Quatre groupes de caractéristiques peuvent être distingués :

- caractéristiques topologiques,
- caractéristiques structurelles,
- caractéristiques statistiques,
- superposition des modèles et corrélation.

A. Caractéristiques topologiques

La première catégorie, commence par la normalisation des matrices des images des segments du texte de tailles différentes en des matrices de tailles égales. Ensuite, chaque image est divisée en un nombre fixe de bandes horizontales et verticales pour maintenir un vecteur de taille fixe. Les caractéristiques sont les moyennes des valeurs des densités de pixels sur ces bandes [72]. Dans ce type de primitives, on compte également les profils et histogrammes. Les caractéristiques peuvent être définies par le nombre des trous, l'évaluation des concavités, mesures des pentes et autres paramètres de courbures et évaluer des orientations, mesurer la longueur et l'épaisseur des traits, détecter les croisements et les jonctions des traits, mesures les surfaces et les périmètres,... [35].

B. Caractéristiques structurelles

Ces caractéristiques sont extraites à partir d'une image contenant le squelette ou le contour et non pas de l'image brute. Dans ce cas, on s'intéresse à décrire la forme du squelette ou du contour. Cette définition intègre [35]:

- Les traits et les anses dans les différentes directions ainsi que leurs tailles.
- Les points terminaux.
- Les points d'intersections.
- Les boucles.
- Le nombre de points diacritiques et leur position par rapport à la ligne de base.
- Les symboles diacritiques et les zigzags (hamza).
- La hauteur et la largeur du caractère.

- La catégorie de la forme (partie primaire ou point diacritique, etc.).

Plusieurs autres caractéristiques peuvent être tirées, suivant qu'elles soient extraites d'une courbe, un trait ou un segment de contour.

C. Caractéristiques statistiques

Elles décrivent la forme d'un texte en utilisant des fonctions (ex : transformée de Fourier et transformée de Hough) statistiques. Parmi les caractéristiques utilisées dans la reconnaissance du manuscrit arabe on trouve [108]:

- Le zonage (zoning), consiste à superposer une grille $n \times m$ sur l'image du caractère et pour chacune des régions résultantes, calculer la moyenne ou le pourcentage de points en niveaux de gris, donnant ainsi un vecteur de taille $n \times m$ de caractéristiques.
- caractéristiques de lieu géométrique: en utilisant la méthode Loci qui est basée sur le calcul du nombre de segments blancs et de segments noirs le long d'une ligne verticale traversant la forme, ainsi que leurs longueurs.
- La méthode des moments : les moments d'une forme par rapport à son centre de gravité sont invariants par rapport à la translation et peuvent être invariants par rapport à la rotation. Ils sont aussi indépendants de l'échelle. Ces caractéristiques peuvent être facilement et rapidement extraites d'une image de texte, ils peuvent tolérer modérément les bruits et les variations.

D. Superposition des modèles et corrélation

La méthode de "template matching" appliquée à une image binaire (en niveaux de gris ou squelettes), consiste à utiliser l'image de la forme comme vecteur de caractéristiques pour être comparé à un modèle (template) pixel par pixel dans la phase de reconnaissance, et une mesure de similarité est calculée [108].

1.3.5 Classification

La phase de classification permet d'attribuer une label ou étiquette à chaque vecteur caractéristique obtenu dans la phase précédente [9]. Elle travaille selon deux principaux modes : le premier est l'apprentissage et le second est la reconnaissance et décision [94].

A. Etape d'apprentissage

Cette étape doit être réalisée afin d'initialiser la base des connaissances du système et éventuellement créer des modèles du langage ciblé. L'apprentissage est dit supervisé quand le concepteur indique l'étiquette de la forme en entrée. Par contre, lorsque le processus est automatisé, on dit que l'apprentissage est non supervisé [94].

B. Étape de reconnaissance et décision

Dans cette étape, le système attribut une label ou étiquette à des nouveaux exemples en se basant sur la base des modèles références du langage créer durant l'étape précédente. La réponse du système peut être unique dans le cas où un seul modèle répond à la description de la forme en question, comme elle peut être multiple ce qui conduit à une confusion. En plus, elle peut être un rejet s'il n'y a pas de modèle qui sont proches du modèle d'entrée.

C. Méthodes de classification

Plusieurs approches ont été développé dans les systèmes de la reconnaissance de l'écriture manuscrite cursive afin de réaliser la tâche de classification. Parmi les méthodes les plus utilisées en classification on trouve [108, 94]:

- Méthode bayésienne,
- Méthode du plus proche voisin,
- Réseaux de neurones,
- Machines à Vecteurs de Support (SVM).

D. Méthode bayésienne

Dans cette méthode, l'étiquette choisie est celle qui possède une suite de primitives avec une plus forte probabilité à posteriori par rapport aux modèles préalablement appris [66, 52].

E. Méthode du k plus proches voisins: kppv

La méthode du k plus proches voisins (En anglais KNN: K Nearest Neighbors) consiste à calculer une distance entre l'exemple en entrée et les exemples dans la base des modèles. Cette distance est obtenue en utilisant une fonction de mesure de similarité par exemple euclidien, gaussien, Manhattan, ... etc. Ensuite, elle attribue à l'exemple en question l'étiquette des k modèles de référence les plus proches. Elle permet d'avoir de bons résultats, avec une simplicité de réalisation. Mais, la vitesse de traitement s'affaiblie à des calculs de distances quand la taille de base des modèles devient importante [108, 66].

F. Réseaux de neurones

Ils se composent d'un ensemble de neurones artificiels ou formels. Ce dernier, permet de créer un graphe orienté pondéré. Chacun de ces neurones possède un état interne et une fonction d'activation lui permettant d'affecter l'états des autres neurones. Cette activité se propage dans le graphe le long d'arcs pondérés appelés liens synaptiques [103, 71]. Pour ce classifieur, les entrées sont les caractéristiques des images des segments obtenus précédemment, la sortie activée correspond à l'étiquette choisie en décision. Une bonne

architecture doit tenir en compte la complexité des calculs et le taux de reconnaissance du réseau à réaliser. Par ailleurs, le point fort des réseaux de neurones est la capacité de générer une région de décision de forme quelconque au prix de l'intégration de couches de cellules supplémentaires dans le réseau [52, 15].

G. Machines à Vecteurs de Support (SVM)

D'abord, L'algorithme a été développé par le russe Vladimir N. Vapnik et Alexey Ya. Chervonenkis en 1963. Après 29 ans, l'idée d'utiliser un noyau pour définir un hyperplan à marge maximale a été découverte par Bernhard E. Boser, Isabelle M. Guyon et Vladimir N. Vapnik. Ensuite, Corinna Cortes et Vapnik ont proposé une version à marge douce, et ont pu publier le travail en 1995. Au début, l'algorithme effectue une classification binaire supervisée. Les SVM sont capable de traiter des problèmes avec des descripteurs de grande taille, et en assurant une solution unique (pas de problèmes de minimum local comme pour les réseaux de neurones). Les bons résultats obtenus sur des problèmes réels les ont rendu célèbres [79, 21].

Ce classificateur consiste à trouver une frontière séparant deux classes de données, avec la possibilité d'enrichir le modèle par une projection du problème dans un autre espace dans lequel les données peuvent être séparer facilement [75, 53].

1.3.6 Post-traitement

Le post-traitement est effectué quand le processus de reconnaissance aboutit à la génération d'une liste de lettres ou de mots possibles, éventuellement classés par ordre décroissant de vraisemblance. Le but principal est d'améliorer le taux de reconnaissance en faisant des corrections orthographiques ou morphologiques à l'aide de dictionnaires de digrammes, tri-grammes ou n-grammes. Quand il s'agit de la reconnaissance de phrases, le processus consiste à faire une sélection de la solution en utilisant des niveaux d'information plus élevés (syntaxique, lexicale, sémantiques. . .). Le post-traitement se charge également de vérifier si la réponse est correcte (même si elle est unique) en se basant sur d'autres informations non disponibles au classifieur [22].

1.4 Caractéristiques de l'écriture arabe

L'écriture arabe a été développée à partir d'un type d'Araméen. La langue araméenne comporte moins de consonants que l'arabe, alors de nouvelles lettres ont été créée en ajoutant des points aux lettres déjà existantes. D'autres petites marques appelées diacritiques sont utilisées pour indiquer de courtes voyelles, mais elles ne sont généralement pas utilisées [108, 110].Elle possède les caractéristiques suivantes [4, 54, 1]:

- nature, le script arabe est cursif,

- le texte arabe est écrit de droite à gauche,
- avec 28 lettres de base,
- plus de la moitié des lettres arabes sont composées d'un corps principal et des composantes secondaire: par exemples la lettre Beh (ب) et possède un point au-dessus de son corps principal, Teh (ت) possède deux points et Kef (ك) possède un zigzag ci-joint avec le corps principal [1],
- le type et la position de la composante secondaire sont des caractéristiques très importantes des lettres arabes. Par exemple, la reconnaissance de deux points au-dessous du corps principal est suffisant pour reconnaître la lettre Yeh (ي) parce que c'est la seule lettre possédant deux points au-dessous de sa corps principal. En outre, certaines lettres ne peuvent être distinguées que par les composantes secondaires. Par exemple : Teh (ت) et Theh (ث) se diffèrent seulement par le nombre de points au-dessus du corps principal, et Teh médiane (تـ) Yeh médiane (يـ) se diffèrent seulement par la position des deux points [1],
- il y'a une variation importante dans le dessin des composantes secondaires (voir tableau 1.8) [1],

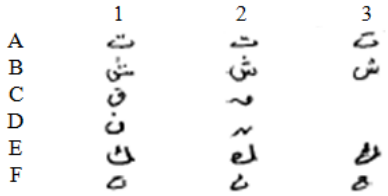


Figure 1.8: Exemples de variations en dessin des composantes secondaires [1].

- parmi les lettres arabes 16 possèdent des points,
- Les points peuvent être un, deux ou trois, comme ils peuvent être au-dessus ou au-dessous de la ligne de base,
- D'un autre côté la largeur des lettres arabes diffère d'une lettre à une autre; dans les scripts manuscrits. En plus, habituellement il y'a des différences en formes des lettres d'une personne à une autre,
- Il y'a un nombre petit de lettre qui possèdent la même forme quelque soit la position comme : " ر " et " و ",
- La plupart des lettres des mots arabes sont liés entre eux, dépendant de leur position (Début, Milieu, Fin) (voir tableau 1.2),
- Ces liaisons sont basées sur une ligne que nous l'appelons ligne de base (voir figue 1.9),

Tableau 1.2: Les différentes formes de caractères selon la position dans le mot [108].

Caractère \ Position	Début	Médiane	Finale	Isolé
Alif			ـا	ا
Beh	بـ	ـبـ	ـب	ب
Teh	تـ	ـتـ	ـت	ت
Theh	ثـ	ـثـ	ـث	ث
Jim	جـ	ـجـ	ـج	ج
Ha	حـ	ـحـ	ـح	ح
Kha	خـ	ـخـ	ـخ	خ
Del			ـد	د
Thel			ـذ	ذ
Ra			ـر	ر
Zey			ـز	ز
Sin	سـ	ـسـ	ـس	س
Chin	شـ	ـشـ	ـش	ش
Sad	صـ	ـصـ	ـص	ص
Dhad	ضـ	ـضـ	ـض	ض
Tad	طـ	ـطـ	ـط	ط
Dha	ظـ	ـظـ	ـظ	ظ
Ayn	عـ	ـعـ	ـع	ع
Ghayn	غـ	ـغـ	ـغ	غ
Fa	فـ	ـفـ	ـف	ف
Qaf	قـ	ـقـ	ـق	ق
Kaf	كـ	ـكـ	ـك	ك
Lam	لـ	ـلـ	ـل	ل
Mim	مـ	ـمـ	ـم	م
Noun	نـ	ـنـ	ـن	ن
He	هـ	ـهـ	ـه	ه
Waw	ـو	ـوـ	ـو	و
Ya	يـ	ـيـ	ـي	ي

Figure 1.9: Exemple d'écriture arabe montrant la ligne de base [4].

- les lettres arabes peuvent nécessiter 1, 2 ou 3 directions roulées par le stylo au cours de l'écriture (voir figure 1.10),

Figure 1.10: Directions roulées au cours de l'écriture [4].

- il existe trois types de mots en arabe:
  1. mots avec des caractères isolés: **ودود**
  2. mots avec des caractères liés: **هبة**
  3. mots avec des pseudo mots: **مرحبا**
- la distance entre les pseudos mots peut être plus grande que celle entre les mots,

- certains lettres possèdent un Hamza (forme zigzag) et med (voir les tableaux 1.3),

(a)

Caractère \ Position	Début	Médiane	Finale	Isolé
Alif+med			ـآ	آ
Alif+hamza			ـأ	أ
Waw+hamza			ـؤ	ؤ
Ya+hamza	ئـ	ـئـ	ـئ	ئ

(b)

Caractère \ Position	Début	Médiane	Finale	Isolé
Ta			ـة	ة
Lamalif			ـلا	لا

(c)

Caractère \ Position	Début	Médiane	Finale	Isolé
Lamalif+med			ـلآ	لآ
Lamalif+hamza			ـلأ	لأ

Tableau 1.3: (a) Les caractères additionnels, (b) et (c) Hamza et Med et les positions qu'elles occupent avec Alif, Waw et Ya.

- certains lettres sont combinées ou ligaturés [9].

Tableau 1.4: Caractères susceptibles d'être ligaturés verticalement [108].

لـ ، مـ	مـ ، جـ	فـ ، جـ	قـ ، جـ
لـ ، حـ	مـ ، حـ	فـ ، حـ	قـ ، حـ
لـ ، جـ	مـ ، خـ	فـ ، خـ	قـ ، خـ
لـ ، خـ			

1.5 Problèmes posés

Les documents arabes incluent des caractéristiques spécifiques telles que la présence de plusieurs points diacritiques et marques additionnelles. Les points sont utilisés pour distinguer entre les caractères possédant la même forme de base. D'autres caractères utilisent des marques différentes pour modifier l'accent du caractère. Les symboles diacritiques (points et marques) peuvent apparaître au-dessus ou au-dessous des caractères comme des entités isolées. Ces symboles sont écrits en gardant une distance par rapport aux caractères, ce qui génère parfois des lignes supplémentaires.

L'écriture est cursive, et produit plusieurs ascendants et descendants qui se chevauchent à travers les différentes lignes de texte. Souvent, l'interligne est rempli par des points diacritiques, marques additionnelles et composants chevauchés, ce qui rend la segmentation des documents arabe très difficile.

1.6 Bases de textes arabe

Dans la littérature, plusieurs travaux de reconnaissance hors-ligne réalisés utilisent des différentes bases de texte arabe. Le tableau 1.5 résume les bases de textes couramment utilisées dans la reconnaissance de texte hors-ligne.

Tableau 1.5: Bases de textes arabe couramment utilisées dans la littérature.

Base de données	Disponibilité	Taille	Utilisation
IFN/ENIT	Disponible	32,492 noms de villes et villages tunisien	manuscrit hors-ligne
KHATT	Disponible	2000 images de paragraphes de texte similaire, et 2000 images de paragraphes de texte unique	manuscrit hors-ligne
IESK-arDB	Disponible	4,000 images de mots et 6,000 images de caractères	manuscrit hors-ligne
CEDAR	Non disponible	100 pages de texte, chacune contient 150–200 mots	manuscrit hors-ligne
AHDB	Non disponible	Non exposée	manuscrit hors-ligne
AHDB/FTR	Non disponible	497images de noms de villes et villages libyen	manuscrit hors-ligne
CENPARMI	Disponible	29,498 sous-mots, 15,175 digits, et 2,499 digits	Chèques de banques et montants manuscrit hors-ligne
Arabic handwritten 1.0	Partiellement disponible	5,000 pages manuscrites	manuscrit hors-ligne
APTI	Disponisble	45,313,600 images de mots	mots arabes imprimés
ERIM	Non disponible	750 pages de texte	Texte imprimé
IBN SINA	Disponible	Plus de 1,000 ous-mots	Manuscrit arabe
IfN/Farsi	Disponible	7,271 images de mots	Manuscrit farsi
FHT	Disponible	1,000 images de formes	Manuscrit farsi

1.6.1 IFN/ENIT

Dans ce domaine, IFN/ENIT est la base la plus utilisée. Cette base est créée par l'institut de technologie de communication (IFN) à l'université technique Braunchweig en Allemagne et l'école nationale d'ingénieurs en Tunisie (ENIT). Il est signalé que 411 personnes ont participer pour générer 32,492 noms de villes et villages tunisien manuscrit. La base est disponible du lien `www.ifnenit.com`. Les images de mots viennent avec des informations sur les coordonnées de la ligne de base, le nombre de sous-mots et le nombre des caractères, mais, ne contient pas des informations concernant les limites des caractères et sous-mots.

1.6.2 KHATT

Les auteurs de [76] présentent une base de texte manuscrit arabe appelée KHATT (KFUPM Handwritten Arabic TexT). Cette base est composée de 1000 formes écrites par 1000 différentes personnes et de différents pays. Les formes ont été scannées avec des résolutions de 200, 300, et 600 dpi. Elle contient 2000 paragraphes choisis aléatoirement de 46 sources, 2000 paragraphes de texte minimal qui couvrent tous les formes des caractères arabe, et des paragraphes optionnels. Les 2000 paragraphes aléatoires contiennent 9327 lignes. Les formes de la base sont divisées aléatoirement en des ensembles de 70%, 15%, et 15% pour l'apprentissage, le test et la vérification, respectivement. La base peut être

téléchargée de http://khatt.ideas2serve.net/KHATTAgreement.php.

1.6.3 CEDAR

Dans le centre d'excellence pour l'analyse et la reconnaissance des documents (CEDAR), 10 personnes ont participées dans la création d'une base arabe en écrivant 10 pages de textes différents chacun. Chaque page contient de 150 à 200 mots. Actuellement, elle est non disponible en-ligne [44].

1.6.4 AHDB

Dans [8], la base AHDB pour le manuscrit arabe hors-ligne a été présentée. Les exemples sont rassemblés de 100 personnes. Elle contient des mots utilisés dans l'écriture des chèques de banques et montants, les mots arabes populaires et quelques pages de texte libre manuscrit. Elle est non disponible en-ligne.

1.6.5 AHDB/FTR

La création de la base est réalisée en faisant cinq élèves de différents âges et qualifications à écrire des lignes de texte, en utilisant des stylos selon leurs choix. Ensuite, ces textes sont scannés avec une résolution de 300 dpi et sauvegarder sous forme d'images BMP en niveau de gris. Elle contient 497 images de noms de villes et villages libyen. Elle est non disponible en ligne [93]

1.6.6 IESK-arDB

C'est une base de manuscrit hors-ligne. Elle contient plus de 4,000 images de mots et 6,000 images de caractères segmentés. Le vocabulaire de la base couvre les différentes parties de la parole arabe (nom, verbe, ...etc.), noms de villes, termes de sécurité et mots utilisés dans l'écriture des montants de banque. Les exemples sont collectés de 22 personnes de différents pays arabes et des pays où le script arabe est moyen [44].

1.6.7 CEMPARMI

Une autre base (CEMPARMI) de chèques manuscrits est introduit dans [101]. Principalement, elle est constituée de 3,000 images de chèques. Depuis ces images 29,498 images de sous-mots, 15,175 images de digits et 2,499 images de montants sont extraites et étiquetées. Elle est conçue pour faciliter la lecture des chèques pour les banques et les secteurs financiers. Cette base est librement disponible en ligne.

1.6.8 Arabic handwritten 1.0

L'analyse de média appliquée (AMA) a développé une base du manuscrit arabe. Elle contient 5,000 pages de texte manuscrit, écrit par 49 personnes de six pays différents. La collection composée de documents de types variés incluant des formes, des notes, des

diagrammes et des listes de nombre avec des digits arabe et indien. La base est disponible à télécharger [43].

1.6.9 IBN SINA

Les auteurs de [50] montrent le processus de création de la base IBN SINA. Elle contient 1,000 sous-mots extraits depuis 50 folios de scripts historiques manuscrits arabes. Elle est disponible pour la télécharger, avec les sous-mots et leurs informations correspondantes stockées dans des fichiers ".MATs".

1.6.10 IfN/Farsi

L'alphabet et le style d'écriture en Arabe et en Farsi sont presque les mêmes. Pour cela, les techniques OCR et les bases développées pour l'une sont pratiquement valable pour l'autre. IfN/Farsi est une base développée pour les manuscrits farsis, présentée dans [82]. Cette base est développée de façon similaire à la base IFN/ENIT. Elle contient 7,271 images binaires des exemples manuscrits de noms de villes et provinces iranienne. 600 personnes ont participé à la création de cette base. La base peut être obtenue en contactant les auteurs.

1.6.11 FHT

FHT est une autre base de manuscrits farsis, présentée par [123]. Cette base contient 1,000 formes remplis par des passages manuscrits. Ces passages sont écrits par 250 personnes des différents ages et niveaux éducatifs. Des informations supplémentaires sont offertes sous forme de texte digital. Elle peut être obtenue des auteurs.

1.7 Avancées en OCR arabe

L'Arabe, est par nature un script cursif écrit de droite à gauche. La représentation ou la forme des caractères change avec chaque occurrence et dépend de leurs positions dans le mot (début, milieu, fin ou isolé) ce qui rend l'Arabe très différente des autres langues. Un texte arabe est constitué d'un ensemble de lignes, chaque ligne contient un ensemble de mots, qui sont plus loin divisés en un ensemble de composants connexes ou sous-mots, appelés par quelques chercheurs PAWs (Pieces of Arabic Words). L'espace entre les mots est souvent plus grand que celui entre les sous-mots dans le texte imprimé, toutefois, il varie dans le texte manuscrit et résulte à une inconsistance dans le processus de segmentation des mots et PAWs (comme le montre la figure 1) ce qui rend la reconnaissance du texte manuscrit arabe un grand défit et nécessite des recherches future.

Dans ce travail, nous présentons un bilan de quelques travaux reliés à ce domaine de recherche. El Abed et al. [42] ont proposés un système de reconnaissance pour les mots manuscrits isolés arabe. Dans leur approche, ils commencent par l'utilisation d'une

méthode à base de squelette pour la détection des lignes de base et l'extraction des caractéristiques de directions. Ensuite, cinq zones horizontales d'une même hauteur ont été utilisées pour extraire la longueur du squelette dans les quatre directions. Ils sont utilisés les HMMs pour construire des modèles d'apprentissage. La base IFN/ENIT est utilisée pour l'évaluation et elle a donnée un taux de reconnaissance de 89.1% pour top1 et 96.4 pour top 10. Benourah et al. [23] ont proposés un système hors-ligne basé sur les HMMs semi-continus avec une duration d'état explicite pour la modélisation et la reconnaissance des mots arabe sans-contraintes. Un schéma combiné des caractéristiques intégrant des caractéristiques et structurelles, avec une segmentation des mots implicite sont utilisés. Les images sont divisées en un ensemble de cadres verticaux avec des longueurs constantes et variables pour l'extraction des caractéristiques. Une analyse de minima et maxima de la projection verticale de l'histogramme en considérant la complexité morphologique des caractères. En utilisant la base IFN/ENIT des taux de segmentation uniforme de 81.02% pour top1 et 91.71% pour top 10 sont achevés, et des taux de reconnaissance 83.79% pour top1 et 92.12% pour top10, sont atteints. Les auteurs de [65] ont proposés un système à base de DST (Dempster Shafer Theory) pour améliorer la précision et la fiabilité d'un système. Le système travail en deux étapes: premièrement, une méthode de combinaison est utilisée pour combiner les sorties probabilistes des classificateurs HMMs. Ensuite, un ensemble de stratégies décisions (acceptation / rejet) est utilisé comme un module de post-traitement global pour améliorer la fiabilité du système. Finalement, le taux de reconnaissance est amélioré en se basant sur les HMMs multi-stream comme une alternative de traitement pour les exemples rejetés. Le système s'était capable d'achever des de 82% pour top 1 et 86.53% pour top 2 avec la base IFN/ENIT, et des taux de 68.3% pour top 1 et pour top 2 avec la base RIMES.

Les DTW (Dynamic Time Warping) présentent une autre alternative pour d'autres développeurs. Moghaddam et al. [80] ont proposé un système de spotting des mots pour les documents historiques. Il utilise un processus indépendant du langage afin d'extraire un ensemble d'informations saillantes sans segmentation. Six caractéristiques sont utilisées pour créer une bibliothèque multi-classes des composants connexes (CCs) , utilisant une mesure de distance euclidienne améliorée par les rotations et les transformations DTW. La base du centre Juma Al Majid est utilisée pour l'évaluation, avec un taux d'auto-spotting approximé de 98.05%. Dans [99], les auteurs proposent un algorithme de recherche rapide des PAWs manuscrits dans un lexique large. Trois étapes sont nécessaires pour atteindre les résultats de la recherche. Premièrement, les différentes représentations de chaque PAW sont sauvegardées dans un lexique dans la même espace euclidienne est faite en se basant sur un processus DTW durant le calcul d'une distance de similarité. Ensuite, les PAWs sont normalisées aux mêmes tailles, et stockées en utilisant un arbre k-d (k-d tree). Une petite liste des candidats est préparée pour la

troisième étape à l'aide d'une approximation rapide des KNNs (K-Nearest Neighbors). Finalement, les candidats sont examinés par une DTW active pour définir les résultats de matching finals. La démonstration de l'algorithme est faite par la base IFN/ENIT avec une accélération d'ordre de 5 et précision avec une réduction de 3.8% de 89.4% avec une DTW traditionnelle.

Un système combiné est développé par Rodriguez-Serrano et al. [97] qui ont proposé une méthode de mesure de similarité entre les séquences de vecteurs modélisés avec les HMMs semi-continus. Dans ce type des HMMs, les probabilités émis dans chaque état sont un mélange de gaussiens partagés, ce qui offre deux profits. Premièrement, l'ensemble commun des gaussiens mène à des estimations plus précises des paramètres du HMM avec des informations à priori. Deuxièmement, le coût du calcul entre le mélange des poids de deux HMMs semi-continus peut être réduit par l'utilisation d'une DTW. Trois bases de données ont été utilisées parmi les IFN/ENIT, et les tests montrent que la similarité proposée avec le taux 91.2% dépasse la DTW traditionnel avec un taux de 87.4%. Les SVMs et RDA (Regularized Discriminant Analysis) sont utilisés par les auteurs de [69] dans un système de spotting de mots à base d'apprentissage pour les documents manuscrits arabe. Il utilise un ensemble de caractéristiques de gradient, depuis qu'ils sont indépendants du langage. Le système est basé sur un modèle de segmentation de PAWs ou les sous-mots et implémenté en utilisant un classificateur hiérarchique intégrant les SVMs et RDA. La base de document CENPARMI est utilisée pour tester l'algorithme de segmentation qui atteint une performance prometteuse de 95.4% et de 96% de précision et recall respectivement, et un taux de reconnaissance de 84.56% avec SVM et 66.4% avec RDA. Une distance d'éditions ou distance de Levenshtein ont été vu comme une bonne opportunité pour Sari et al. Dans [105] ont proposé un moteur de recherche pour les images de documents arabe manuscrits et imprimés sans la reconnaissance des patterns contextuelles. Premièrement, une approche ascendante est appliquée pour extraire les composants connexes (CCs), ensuite, chaque CC est représenté par des caractéristiques structurelles globales comme les cercles, les ascendants et les descendants, ce qui donne une représentation des WSTs (Word Shape Tokens). Les caractéristiques extraites sont sauvegardées dans un fichier ASCII. Après ça, une distance Levenshtein est calculée pour comparer les chaînes de caractères. Une précision approximative de 77.78% est achevée durant l'évaluation. Ainsi, [63] ont proposé une méthode d'extraction des caractéristiques structurelles pour la reconnaissance des noms de personnes manuscrits arabe dans le but de maximiser le taux de reconnaissance avec le minimum d'éléments. La méthode incorpore des informations structurelles comme : les cercles, stems, pieds et diacritiques. Les résultats d'extraction des caractéristiques sont évalués en calculant une distance de Levenshtein entre deux chaînes. La distance est définie par le nombre minimum de changements (permutation, suppression, ajout,...etc.) nécessaires pour transformer une séquence en une autre. Les résultats sont testés en utilisant deux bases : les noms de

personnes de l'archive national de Tunisie avec un taux de 89%, et la base IFN/ENIT avec un taux de 80%. Une technique de ranking a été utilisée par Srihari et al. [113] dans système de spotting pour des documents manuscrits arabe. L'approche fonctionne en deux étapes : premièrement, en utilisant des requêtes saisies, le système lance une sélection de prototype afin d'extraire un ensemble d'exemples de manuscrits. Ensuite, un processus de matching de mot est lancé pour trouver chaque occurrence de ces exemples dans la base. Des caractéristiques globales de forme de mot sont utilisés durant le processus de ranking entre les mots prototypes et mots candidats. Une forme de base de données pour dix écrivains est utilisée pour évaluer la performance du système ce qui a donné un taux de reconnaissance de 70%. AlKhateeb et al. [13] ont proposé un système holistique hybride pour la reconnaissance de mots manuscrits arabe. Les caractéristiques sont extraites en utilisant des fenêtres glissantes de gauche à droite appliqué à des images MWI (Mirrored Windows Image), en plus de différentes caractéristiques structurelles comme le nombre de régions aux dessus et aux dessous de la ligne de base. Un modèle de classification combiné est utilisé intégrant une technique de re-ranking et un classificateur HMM. Des caractéristiques d'intensités sont utilisées pour entrainer le classificateur HMM, et des caractéristiques topologiques pour le re-ranking afin d'améliorer la précision. Le système est testé sur la base IFN/ENIT, et il a pu d'achever un taux de reconnaissance de 89.24% pour top 1 et 95.15% pour top 10. Les auteurs de [98] ont proposé un modèle de segmentation à trois étages afin d'améliorer la précision de la segmentation du texte manuscrit arabe hors-ligne. Le premier étage est la segmentation du texte en lignes, où les lignes du texte sont extraites en utilisant la densité des lignes pour prédire l'espace entre les lignes. Le second étage, est l'extraction des mots ou sous-mots (PAWs) en utilisant les connectivités de huit voisins. Le dernier étage, est l'extraction des caractères par la recherche des points de segmentation parmi les points de branchement dans la ligne de base. Trois bases de données sont utilisées pour tester les capacités du système : une base locale, la base de Hamed et Zitar et la base IFN/ENIT avec les taux 89%, 81% et 78% respectivement.

1.8 Système OCR proposé

Dans cette section, nous allons présenter l'idée de base du système OCR proposé à l'instar de l'étude faite sur les différentes problèmes et solutions réalisées pendant ce dernier décennie dans ce domaine.

Souvent, la nature cursive de l'écriture manuscrite génère trois types de problèmes ou d'irrégularités de formes, à savoir :

1. des caractères qui se chevauchent (figure 1.11-(A));
2. des caractères coupés en parties (figure 1.11-(B));

3. des caractères mal écrits, alors mal représentés, ce qui donne l'apparence d'un autre caractère (figure 1.11-(C));

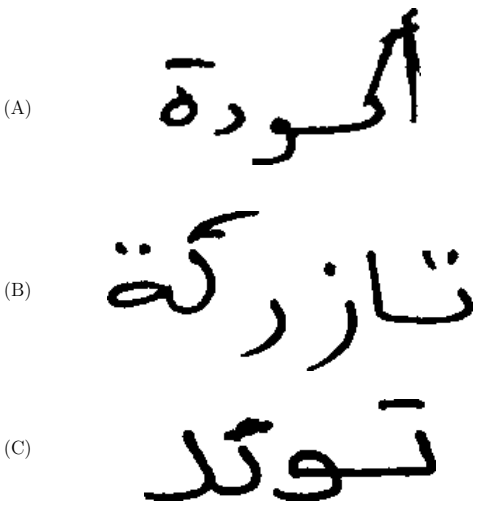


Figure 1.11: Illustration des problèmes de l'écriture manuscrite cursive:(A) Exemple de chevauchement, (B) Exemple de caractères coupés, et (C) Exemple de caractères nécessitant le changement de nature.

La résolution de ces problèmes en structure des caractères nécessite de faire le processus inverse de chaque fait (chevauchement, coupure et mal écriture) afin de restaurer la structure correcte des caractères. Le premier problème, qui concerne le chevauchement des caractères ne peut être résolut qu'en divisant (Division) les segments de caractères chevauchés. De même, le second problème de coupures dans les caractères nécessite de fusionner (Fusion) les segments des caractères sur-segmentés. En plus, le troisième problème des caractères mal représentés ne peut être résolut qu'en changeant la nature du caractère complètement (Morphisme).

La réalisation de ces opérations de correction nécessite la connaissance de la nature (étiquette ou label) des segments de caractères traités. Ces informations ne peuvent être disponible qu'après la phase de classification. Par conséquent, les opérations de corrections indiquées ci-dessus doivent être faites durant la phase de post-traitement, comme les opérations de vérification syntaxique ou sémantique (voir la figure 1.12).

Une image de texte non reconnu ressemble beaucoup à un jeu puzzle (jigsaw puzzle) non résolut. En plus, les opérations de correction présentent beaucoup de taits communs avec les étapes de résolution de ce jeu puzzle (voir la section 4.1.6).

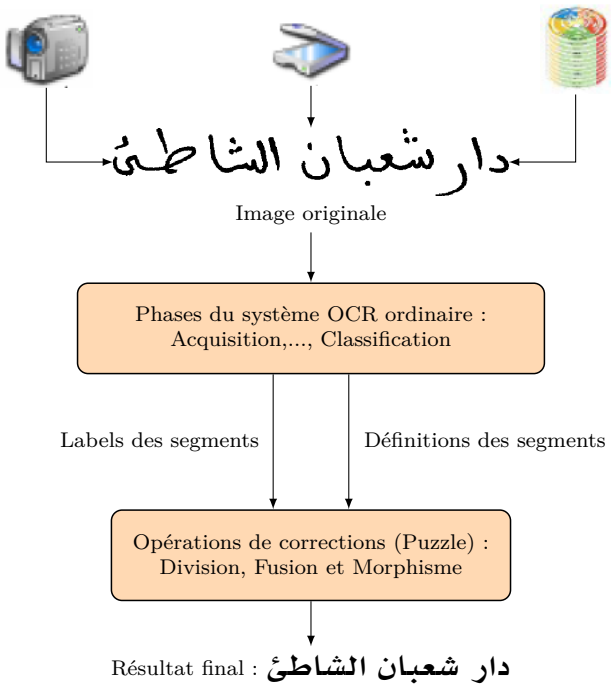


Figure 1.12: Architecture du système OCR intégrant les opérations de correction.

Conclusion

Dans ce chapitre, nous avons vu un état de l'art sur les systèmes OCR. Pour les quels, nous avons présenté les différents phases et approches qui entrent dans le processus de la reconnaissance de l'écriture manuscrite. De plus, nous avons présentés quelques caractéristiques de la langue arabe et montrer les difficultés qu'il faut surmonter pour réaliser un OCR arabe, ces difficultés sont directement liées à la nature de langue arabe et au peu de recherches effectués sur la problématique, comparativement aux autres langues. Puis, nous avons décrit quelques travaux récents dans ce domaines en effectuant une classification et en indiquant les taux de reconnaissance obtenus par rapport aux benchmarks existants. Enfin, nous avons présenté la problématique et les motivations de cette thèse.

Chapitre 2

Technique de segmentation N-Scans

Introduction

Dans le chapitre précédent, nous avons présenté, un état de l'art sur les systèmes OCR, nous avons vu l'architecture globale de ce dernier, ces différents types, les avancés dans ce domaine, et également la problématique et les motivations de cette thèse.

Le but de ce chapitre, est de présenter les techniques ou approches utilisées dans la littérature pour la segmentation du texte manuscrit cursive et en particulier la segmentation du mot en caractères, ainsi que les avantages et les inconvénients de chacune d'elles. L'idée principale est de donner un aperçu de l'état de l'art afin de justifier le choix de notre première contribution, qui consiste en la proposition d'une approche de segmentation et les hypothèses sur lesquelles nous avons construit notre idée de segmentation multi-scans ou N-Scans.

2.1 Problématique posée par l'écriture du cursif

L'image d'un mot cursif est représentée par un signal bidimensionnel dans lequel, aucune information d'ordonnancement n'est présente, bien que par convention, il existe dans ce mot une séquence de lettres dont l'ordre logique d'interprétation va de la droite vers la gauche [38].

En regardant la figure 2.1 on constate qu'il n'est pas possible de déterminer a priori si le point B se trouve ou non dans une lettre antérieure au point A.
La seule façon de restituer l'ordre cohérent du tracé est de segmenter le mot en lettres. Or dans ce type d'écriture, la localisation précise du début et de la fin d'une lettre s'avère très difficile à réaliser, voire impossible. C'est pourquoi, le problème de la segmentation se trouve au cours de la reconnaissance hors-ligne de mots cursifs [38].

Généralement, l'écriture cursive se caractérise par les problèmes suivants :

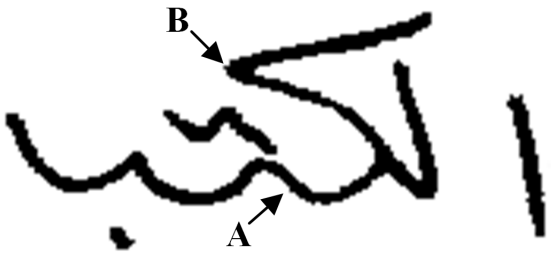


Figure 2.1: Mot cursif "الكتب"

1. Caractères avec des segments supplémentaires
2. Alignement non respecté
3. Espace entre caractères irrégulier
4. Cercles des caractères avec des segments liés
5. Caractères avec des dents invisibles
6. Caractères malformés
7. Caractères sur-segmentés
8. Caractères sous-segmentés

2.1.1 Caractères avec des segments supplémentaires

Parfois, dans les mots cursifs on trouve des caractères avec des petits segments supplémentaires, ce qui rend la reconnaissance de ces caractères plus compliquée. La figure 2.2 illustre un exemple de caractère avec un segment supplémentaire.

Figure 2.2: Caractère avec un segment supplémentaire [IFN-ENIT].

2.1.2 Alignement non respecté

A l'inverse à l'écriture imprimée ou les lignes du texte (lignes de bases) sont droites, l'écriture cursive possède des lignes de base non droites. En plus, les caractères sont pour pas mal de fois superposés (voir figure 2.3).

2.1.3 Espace entre caractères irrégulier

L'écriture manuscrite, présente souvent le problème de non régularité dans l'espace entre les différents caractères à cause de la rapidité d'écriture (voir figure 2.4).

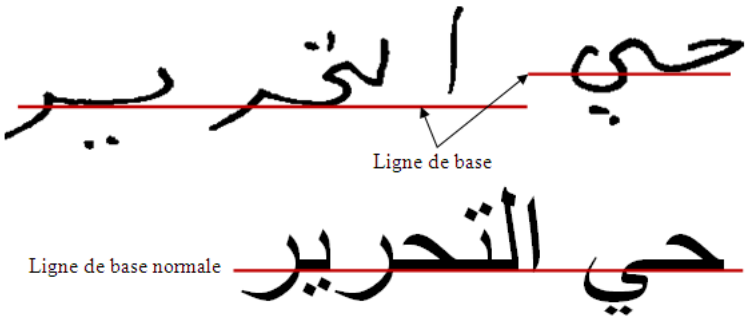


Figure 2.3: Alignement en écriture manuscrite [IFN-ENIT].

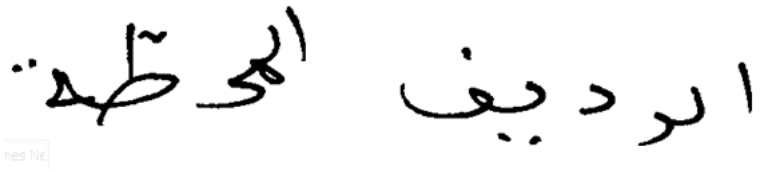


Figure 2.4: Exemple d'espace irrégulier entre caractères [IFN-ENIT].

2.1.4 Cercles des caractères avec des segments liés

Les caractères s'écrivant avec des cercles tels que : هـ, ع, ف, ق, م, peuvent avoir le problème de liaison des segments composants ces cercles tel que le présente l'exemple dans la figure 2.5.

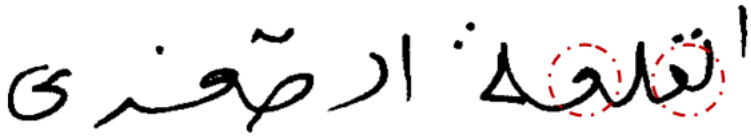


Figure 2.5: Exemple des cercles avec des segments liés (ع, ق) [IFN-ENIT].

2.1.5 Caractères avec des dents invisibles

Le problème des dents invisibles concerne les caractères avec des dents tels que : بـ, س, ش, ou parfois quelques dents n'apparaissent pas bien et par conséquent le caractère devient ininterprétable (voir la figure 2.6).

Figure 2.6: Exemple de caractères avec une dent invisible (ش, ب) [IFN-ENIT].

2.1.6 Caractères malformés

Un caractère est considéré comme étant malformé s'il perd sa représentation normale comme le montre l'exemple de la figure 2.7 le caractère "ى" est devenu "ه".

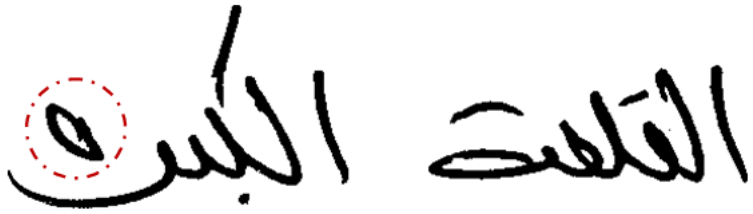


Figure 2.7: Exemple de caractères malformés (ى, ا) [IFN-ENIT].

2.1.7 Caractères sur-segmentés

Parmi les problèmes de l'écriture manuscrite, l'existence des petites coupures dans les segments (traits) représentant le caractère ce qui cause le problème de caractère sur-segmentés (voir les exemples présentés par la figure 2.8).

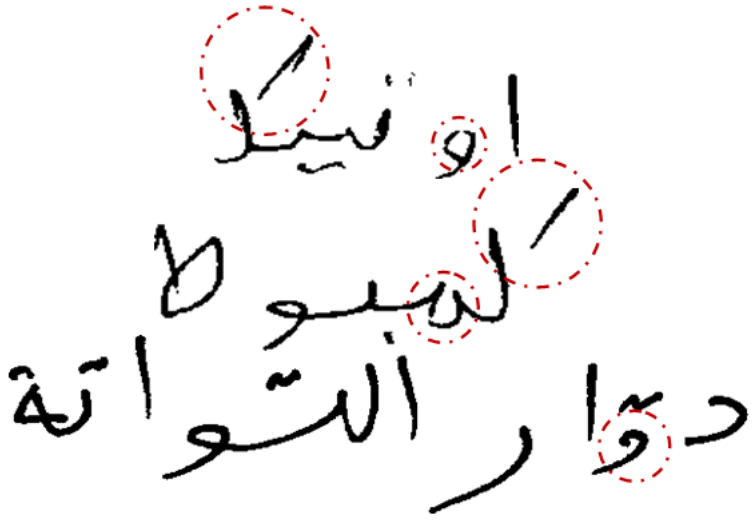


Figure 2.8: Exemple de caractères sur-segmentés[IFN-ENIT].

2.1.8 Caractères sous-segmentés

Un caractère sous-segmenté est le résultat des liaisons supplémentaires entre les segments (traits) représentant le caractère. Nous pouvons appeler aussi chevauchement de caractères (voir la figure 2.9).

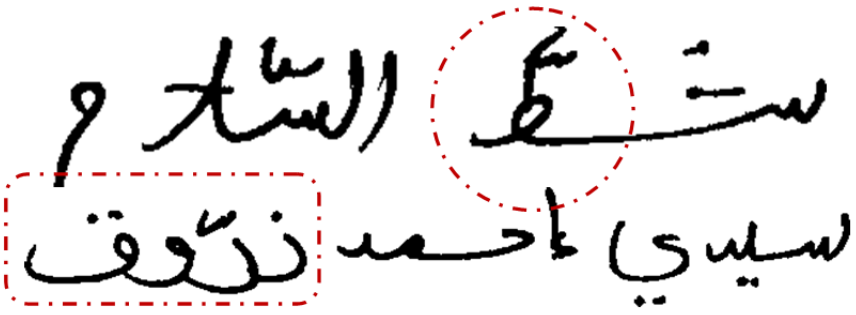


Figure 2.9: Exemple de caractères sous-segmentés [IFN-ENIT].

2.1.9 Caractères non-clair ou ambigu

L'écriture cursive, cause parfois des caractères possédant une représentation graphique non-claire et par conséquent une ambigüité dans l'interprétation du caractère (voir la figure 2.10).

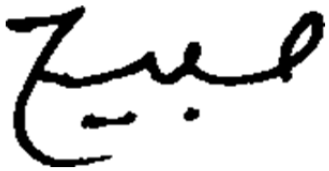


Figure 2.10: Exemple de caractères sous-segmentés [IFN-ENIT].

Généralement, l'écriture cursive se caractérise par deux problèmes majeurs, à savoir :

1. **Sur-segmentation :** elle se représente par des petites coupures dans les segments (traits) représentant le caractère.
2. **Sous-segmentation :** elle se représente par des liaisons supplémentaires entre les segments représentant le(s) caractère(s).

Le tableau ci-dessous présente les principaux facteurs causant ces deux phénomènes.

Tableau 2.1: Facteurs derrière la sur-segmentation et sous-segmentation.

	Sur-segmentation	Sous-segmentation
Rapidité d'écriture	✓	✓
Effets de prétraitement	✓	✓
Stylo de mauvaise qualité	✓	✗
Des petites parties grattées	✓	✗
Taches sur papier	✗	✓

2.2 Structure physique et structure logique

Dans le domaine de l'analyse de documents, nous pouvons identifier deux types de recherches :

1. l'analyse de composition, et
2. l'interprétation du document.

Ces deux systèmes de traitement permettent de faire la distinction entre une information physique (correspondant aux objets physiques présents dans le document) et une information logique (liée à l'interprétation de l'organisation des objets du document) [96]. Le premier niveau de données accessibles au système d'analyse est la structure physique du document. Il concerne la répartition spatiale de l'information du document. La structure logique se rapporte au sens de cette organisation. La connaissance de la structure physique permet de déduire la structure logique si les règles de présentation et de composition sont claires et connues [94].

2.3 Principe général de la segmentation

Le rôle de la segmentation est l'extraction des différents zones ou segments d'images utiles pour des traitements futurs. C'est une phase très cruciale parce qu'une bonne segmentation va déterminer les bons segments et par conséquent un bon taux de reconnaissance, à l'inverse, une mauvaise segmentation quant à elle va entraîner une chute du taux de reconnaissance [25].

Généralement, nous pouvons distingués quatre niveaux de segmentation, comme suit:

1. **Segmentation de la page:** Elle permet de localiser et de déterminer la nature du media représenté dans chaque zone (texte, graphique, photographie etc.) [96]. Ensuite, chaque classe est orientée vers un système spécialisé dans l'analyse de chaque type de media [112].

2. **Segmentation de texte en lignes:** Elle consiste à extraire les différents lignes du texte. Malgré que la plupart des études optent une décomposition de l'image en composantes connexes [89], On trouve d'autres qui utilisent des techniques basées sur les histogrammes des projections horizontales pour déterminer les lignes d'un texte donné [78].

3. **Segmentation de lignes en mots:** Elle consiste à extraire les différents mots d'une image de ligne. Dans le cas d'une approche descendante il faut déterminer l'histogramme des projections verticales des lignes pour détecter les espaces entre les mots et pouvoir les séparer [88]. Par contre, avec une approche ascendante d'autres techniques sont utilisées telles que : le suivi du contour, détermination du squelette (voir section 2.5.2) ou la détermination des composantes connexes ... [112].

4. **Segmentation de mots en caractères:**

   La segmentation des mots en caractères est une opération qui tente de décomposer une image de séquence de caractères (mot) en sous-images de symboles individuels [25].

La figure 2.11 montre le passage entre ces étapes.

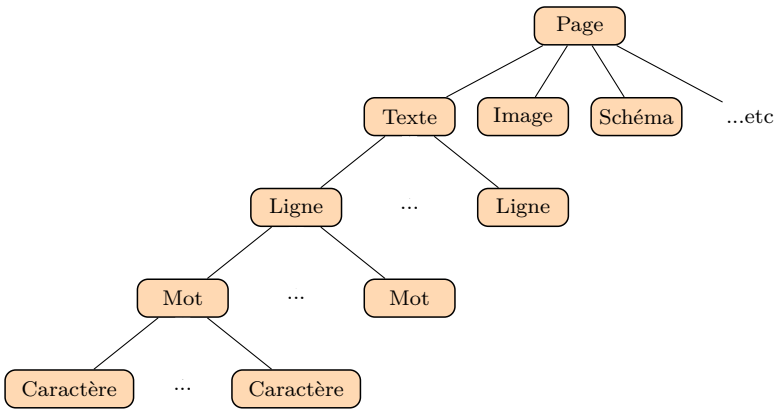


Figure 2.11: Illustration du processus de segmentation.

2.4 Approches de segmentation

La segmentation peut être réalisée séparément ou avec la reconnaissance. Pour cela, nous trouvons les termes : segmentation interne et externe ou straight segmentation et segmentation recognition pour exprimer le même sens[108]. Casey et Lecolinet [25], considèrent que cette dernière n'est pas une bonne classification, parce que le résultat de classification peut être altéré par l'ajout d'un post-processeur, par exemple il peut suggérer de substituer une lettre sortie par le classifieur par deux lettres, et cela est en fait une utilisation d'une segmentation de la sous image [22]. Ils proposent une classification basée sur la façon avec laquelle la segmentation et la classification interagissent durant tout le processus de reconnaissance [89, 25], comme suit:

- ***Approche analytique explicite***, dans laquelle les segments sont identifiés à base de propriétés de ressemblance des caractères. Elle utilise une technique de découpage de l'image en composants significatifs elle est appelée dissection.
- ***Approche analytique implicite***, dans laquelle le système cherche des composants qui correspondent à son alphabet dans l'image.
- ***Approche globale***, dans lesquelles le système essaye de reconnaître le mot comme un tout. Evitant ainsi le besoin de segmentation en caractères.
- ***Approches hybrides***, combinant des proportions différentes de ces trois approches élémentaires..

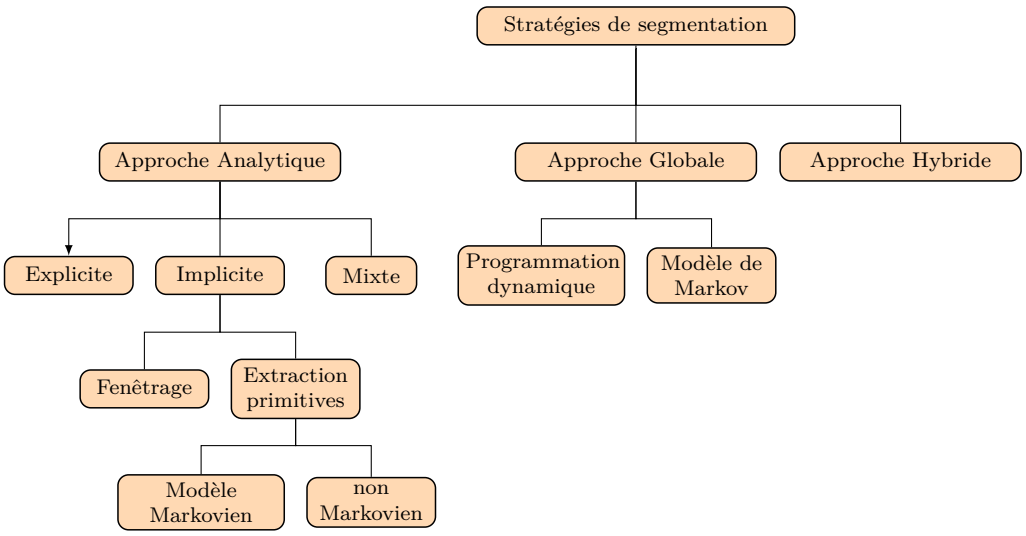


Figure 2.12: Hiérarchie des méthodes de segmentation selon R.G.Casey [38].

2.4.1 Approche analytique explicite

Casey et Lecolinet, l'appellent aussi dissection. Elle est basée sur des critères topologiques ou morphologiques. Elle repose sur l'utilisation des éléments proches des lettres appelés graphèmes [25]. Les graphèmes sont reconnus individuellement. Ensuite, le mot est reconstruit à base des informations contextuelle[38].

2.4.2 Approche analytique implicite

La segmentation s'effectue pendant la reconnaissance. Le système recherche dans l'image, des composants ou des groupements de graphèmes qui correspondent à ses classes de lettres [104]. Classiquement, il peut le faire de deux manières [38]:

1. Fenêtrage, le principe est d'utiliser une fenêtre mobile de largeur variable pour trouver des séquences de points de segmentations potentiels qui seront confirmés ou non par la reconnaissance de caractères (voir section 2.5.1).

2. Recherche de primitives, qui consiste à détecter les combinaisons de primitives qui donnent la meilleure reconnaissance possible. Pour faire la reconnaissance, différentes techniques sont employées dont les modèles de Markov cachés, réseaux de neurones. . . etc.

2.4.3 Approche analytique mixte

La plupart des méthodes actuelles pratiquent un mélange des deux stratégies précédentes [96]. D'abord, une phase de pré-segmentation est appliquée pour sur-segmenter l'image. Puis la meilleure combinaison de reconnaissance est choisie, en considérant les combinaisons possibles de 1, 2,. . . n graphèmes [38].

2.4.4 Approche globale

Dans l'approche globale, le mot est reconnu par sa forme générale. Il n'y a pas de segmentation (ni explicite, ni implicite) [89], c'est pourquoi cette approche est censée être robuste au bruit ou aux imperfections du signal. Habituellement, la reconnaissance dépend d'un lexique (une liste de modèles de mots) [92]. Puisque cette méthode nécessite un modèle pour chaque mot du lexique et que chaque modèle doit être appris, elle est plutôt utilisée pour des applications où le lexique est restreint comme c'est le cas dans le traitement automatique des chèques. Ici aussi, différentes techniques sont employées dont la programmation dynamique et les modèles de Markov cachés [38].

2.4.5 Approches hybrides

Les approches hybrides combinent plusieurs stratégies de reconnaissance afin d'exploiter les forces et les faiblesses d'approches complémentaires (stratégies ascendantes et descendantes). Il y a une pré-segmentation d'où sont tirées des hypothèses qui seront validées par la suite. L'ajout d'informations supplémentaires sur le style du scripteur tel que proposé par Crettez, peut contribuer également à améliorer les performances en choisissant un extracteur de primitives spécifique à chacun des styles d'écriture [38].

2.5 Segmentation de l'écriture cursive

Malgré la simplicité de la tâche de lire un texte écrit pour l'être humain, savoir extraire le sens d'un texte manuscrit pour une machine reste une tâche loin d'être parfaite. Cette opération est constituée d'une séquence de tâches coopérants dans notre système cognitif, nous permettant de passer d'une forme écrite au sens en se basant sur des connaissances de nature syntaxique, grammaticale, sémantique, culturelle. Dans le but de mécaniser ces opérations, une multitude de travaux ont été consacrés à la reconnaissance des lettres successives constituant les mots. La segmentation de l'écriture cursive est une opération très délicate et coûteuse parce que les écritures sont variées, et les lettres souvent contenant des coupures ou liées les unes aux autres. Le résultat de cette phase influe considérablement sur le rendement des prochaines phases.

Principalement, trois approches de segmentation existent : les méthodes descendantes ou top-down, les méthodes ascendantes ou bottom-up, et les méthodes hybrides. La première, consiste à subdiviser itérativement l'image entière en segments plus petits pour extraire les parties de texte désirés. Alors que la seconde, utilise des éléments d'image de bas niveau tels que les composantes connexes ou PAWs. Ensuite, ces composants sont regroupés pour reconstruire des éléments plus grands, tels que des caractères, des mots, des lignes et des blocs de texte. Enfin, la troisième combine les avantages des deux méthodes pour atteindre de meilleurs résultats. Généralement, les approches ascendantes sont adoptées en écriture parce qu'ils sont capables de mieux faire face au le bruit et

la variation de l'écriture. Dans cette section, nous exposons certains travaux antérieurs concernant les méthodes de segmentation. Ces travaux sont classés selon qu'ils sont utilisés pour segmenter un texte en lignes ou pour segmenter un mot en caractères.

2.5.1 Segmentation de texte en lignes

L'une des tâches les plus importantes et les plus difficiles dans l'analyse de documents textuels est de réaliser une segmentation de ligne précise, en particulier lorsque le document est composé de texte manuscrit cursive. Plusieurs méthodes ont été développer dans ce contexte, parmi les nous trouvons :

1. Segmentation par histogrammes de projection horizontale;
2. Segmentation basée sur les fenêtres glissantes;
3. Segmentation par écoulement d'eau;
4. Segmentation basée peinture Piece-Wise.

A. Segmentation par histogrammes de projection horizontale

C'est l'approche la plus utilisée par les systèmes de reconnaissance, en raison de sa grande vitesse de récupération des résultats, mais elle est sensible aux chevauchements et fluctuations des lignes du texte. Elle consiste à faire une segmentation du texte en combinant l'utilisation d'une projection horizontale (voir l'équation 2.1), et la classification des lignes selon leurs poids. Selon cette méthode, le document acquis par scanner contient trois types de lignes [28, 120]:

1. Lignes diacritiques : elles se caractérisent par des petits poids. Ces lignes sont considérées comme bruit.
2. Lignes normales : elles possèdent des poids moyens. Cette approche calcule les poids moyens de toutes les lignes du document.
3. Lignes chevauchées : elles possèdent des grandes hauteurs, ce sont le résultat du chevauchement des caractères ascendants et descendants, mais aussi des caractères collés des lignes adjacentes (voir la figure 2.13).

$$f(y) = \sum_{x=0}^{W} I(x, y) \tag{2.1}$$

Afin de déterminer le nombre des lignes dans un bloc chevauché il faut diviser la hauteur du bloc sur la hauteur d'une ligne, et ensuite diviser le bloc sur le nombre des lignes obtenu. L'inconvénient de cette approche réside dans le cas où les lignes ne possèdent pas une hauteur fixe, par exemple dans le cas où une ou plusieurs lignes sont composées seulement des caractères avec une petite hauteur. Dans ce cas, le système considère ces lignes comme des lignes diacritiques [120].

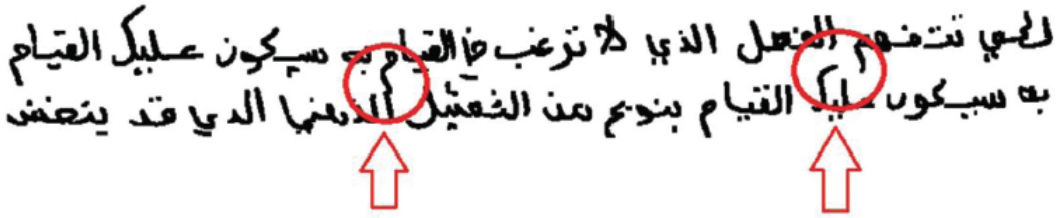


Figure 2.13: Exemple de chevauchent entre des lignes du texte [120].

Figure 2.14: Exemple d'histogramme de projection horizontale [28].

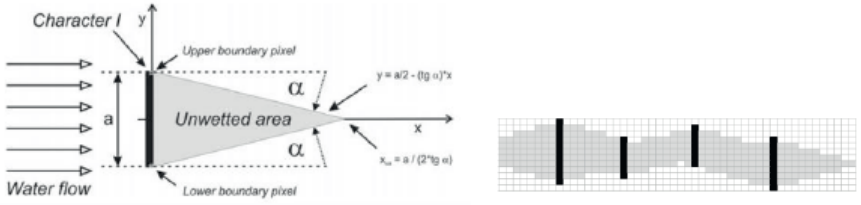
B. Segmentation basée sur les fenêtres glissantes

Elle consiste à subdiviser l'image du texte en colonnes par une fenêtre de taille fixe (comme peut être variable). Cette fenêtre est déplacée avec un pas souvent la moitié de la fenêtre. Ensuite, un diagramme de projection horizontale est calculé à chaque déplacement de la fenêtre dans le but de détecter les lignes du texte. Après, la taille moyenne pour les lignes qui possèdent une taille moins d'un seuil choisit (souvent 1/3 du taille moyenne) [89, 38]. Cette technique est simple, robuste au bruit et indépendante des connexités. Néanmoins, la détermination de la taille de la fenêtre appropriée n'est pas facile. En plus, quand il y'a beaucoup de segments liés, on risque de joindre deux entités différentes [35, 72].

Figure 2.15: Segmentation à base de fenêtre glissante : découpage du mot en bandes verticales [38].

C. Segmentation par écoulement d'eau

L'algorithme d'écoulement d'eau suppose que des débits d'eau hypothétique sous quelques angles de l'image du document à partir de gauche à droite et de haut en bas. Ce processus génère des régions mouillées et d'autres non mouillées. Dans le cas où l'eau s'écoule de gauche à droite, la situation est montrée par la figure 2.16(a). De plus, ce flux d'eau hypothétique devrait combler les écarts entre les lignes du texte consécutives. Par conséquent, les régions non mouillées restantes sur l'image indiquent les lignes du texte. Les régions non mouillées sont étiquetées afin d'extraire les lignes du texte. Une fois cet étiquetage terminé, l'image va être divisée en deux types de bandes. Le premier contient les lignes du texte. Le second présente l'espacement entre les lignes. L'angle d'écoulement de l'eau hypothétique peut être obtenu en utilisant une fonction mathématique selon l'application. L'union des régions non mouillées est présentée par la figure 2.16(b) [77].



(a) (b)

Figure 2.16: Exemples de segmentation des caractères ligaturés [77].

D. Segmentation basée peinture Piece-Wise

Cette technique est proposée par Alei et al. [10] dans le but de segmenter en lignes des textes perses manuscrits. Elle commence par la décomposition verticale du bloc de texte en bandes parallèles. Chaque ligne de chaque bande est colorée par une intensité de gris, qui est la valeur d'intensité moyenne des valeurs de gris de tous les pixels présents dans cette bande de lignes. Par conséquent, les bandes colorées sont convertis à une représentation lissées en deux couleurs (voir la figure 2.17). Les espaces blancs / noirs dans chaque bande de l'image lissée sont analysés pour obtenir une courte ligne de séparation (PPSL : Piece-wise Potential Separating Line) entre deux espaces noir consécutifs. Les PPSLs sont concaténées pour produire une segmentation de texte en lignes (voir la figure 2.18).

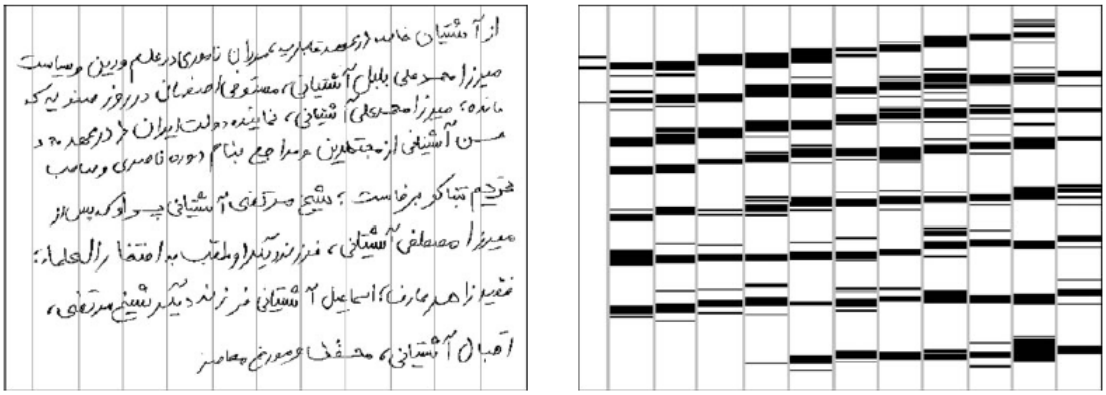


Figure 2.17: Exemple de segmentation avec une technique Piece-Wise [10].

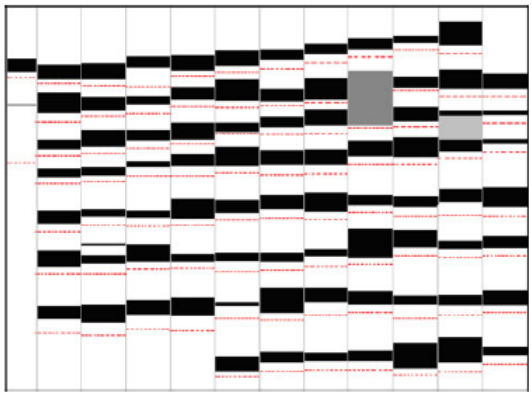


Figure 2.18: Exemple d'image de texte lissée [10].

E. Segmentation stochastique

Elle est basée sur des algorithmes stochastiques, afin de construire des chemins non linéaires entre les lignes de texte chevauchés. Ces lignes sont extraites en utilisant une modélisation de Markov cachée (HMM). De cette façon, l'image est divisée en petites cellules, chacune correspond à un état du HMM. Les meilleurs chemins de segmentation sont recherchés de gauche à droite. Dans le cas des composants chevauchés, le chemin avec la probabilité la plus élevée croise le composant chevauché avec le moins de points en pixels noirs. Cependant, la méthode peut échouer dans le cas où le point de contact contient beaucoup de pixels noirs [77, 28].

F. Segmentation par groupement

Cette méthode consiste à construire les lignes en agrégeant des unités dans une approche ascendante. Des unités telles que des pixels, des composants connectés ou des blocs sont ensuite réunies pour former des lignes. Likforman-Sulem et Faure [28] proposent une approche basée sur un groupement perceptuel des composants connectés par des pixels noirs. Les lignes de texte sont itérativement construites en groupant les composants voisins connectés en se basant sur des critères perceptuels comme la similarité, la continuité et la proximité. Par conséquent, les contraintes locales sur les composants

voisins sont combinées avec des mesures de qualité globales. Pour gérer les conflits, la technique fusionne une procédure de raffinement combinant une analyse globale et une analyse locale. Cependant, la technique proposée ne peut pas être utilisée sur des documents dégradés ou mal structurés.

G. Segmentation par une technique de smearing

Dans cette méthode, les pixels noirs consécutifs le long de la direction horizontale sont enduits consécutivement. L'espace blanc entre les pixels noirs est rempli de pixels noirs. Elle n'est valide que si leur distance est comprise dans un seuil prédéfini. De cette façon, des zones agrandies de pixels noirs autour du texte sont formées. C'est ce qu'on appelle les régions de frontières croissantes. Ces régions englobent les lignes de texte séparées [77]. Li et al. [95] proposent une technique de smearing à base de la méthode level set. D'abord, ils convertissent l'image du texte en une image en niveau de gris en utilisant une fenêtre gaussienne, ce qui améliore les structures de lignes de texte. Les lignes de texte sont extraites en faisant évoluer une estimation initiale en utilisant la méthode level set (voir la figure 2.19).

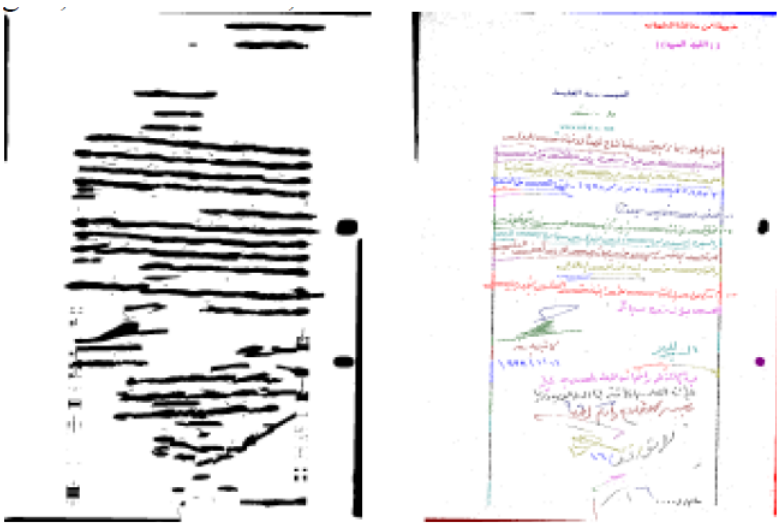


Figure 2.19: Technique de smearing avec l'utilisation de la méthode level set [95].

2.5.2 Segmentation en mots et caractères

A ce niveau le système de reconnaissance examine l'image du texte ou l'image d'un segment de ligne afin d'extraire les différents mots ou/et caractères composants le texte, selon l'approche de segmentation : descendante ou ascendante. Les méthodes de cette classes peuvent être classées en deux catégories, comme suit :

1. Segmentation sans classification : dans ce cas le système n'applique pas des méthodes de classification dans le processus de segmentation;
2. Segmentation avec classification : c'est le cas inverse, dans lequel le système utilise des bases de connaissances et des classifieurs durant la segmentation.

A. Segmentation sans classification

Les méthodes de cette catégorie évitent l'utilisation des bases de connaissances et des classificateurs afin d'alléger la segmentation. Généralement, ces méthodes sont basées sur les histogrammes de projections verticales et des balayages avec marquages colorés de l'image de texte, ou même sur des balayages d'une représentation en squelette en contour de l'image source.

A.1 Segmentation à partir du squelette

Dans ce cas, le système commence par la création d'une image contenant le squelette du texte à traiter. Ensuite, à partir de ce squelette cherche à identifier les points de liaison entre les caractères. Cette opération fait introduire des calculs de courbures et d'angles basées sur des seuils prédéfinis [9]. X.Dupré [38] indique que cette approche est erronée à 10% . Cette méthode est limitée par la qualité du squelette obtenu. Des lettres enchevêtrées (comme les "tt") ou des lettres à liaison haute (comme 'b', 'o', 'v', 'w') avec leur successeur sont très difficile à être segmentées [72].

Figure 2.20: Segmentation à base du squelette [72].

Elzobi et al. [44] proposent une approche de segmentation topologique optimisée. D'abord, le système applique une opération d'amincissement sur une image filtrée en utilisant un filtre médian pour obtenir une représentation squelette du contenu traité. Ensuite, applique une segmentation à deux étapes. Premièrement, il analyse l'axe des x des composants connectés pour extraire les sous-mots des images sans chevauchement. Puis, il extrait des segments de base en fonction des caractéristiques topologiques pour les représentants des caractères.

A.2 Segmentation à partir du contour

Dans cette approche, le système génère à partir de l'image source du texte une image représentant son contour. Par la suite, un balayage du contour est réalisé afin de détecter les points de liaison des caractères ou graphèmes en se basant sur les extrema locaux du contour, qui sont associées selon un critère de proximité (voir figure 2.21). Ces derniers sont faciles à ajuster lorsque la qualité de l'écriture est bonne, comme ils peuvent y avoir des comportements erratiques lorsque l'écriture est de mauvaise qualité [72].
Abandah et al. [2] proposent un système de reconnaissance qui utilise une segmentation basée graphèmes et réseaux de neurones récurrents. Il estime la ligne de base du mot en utilisant une méthode de projection horizontale, ce qui permet d'identifier les

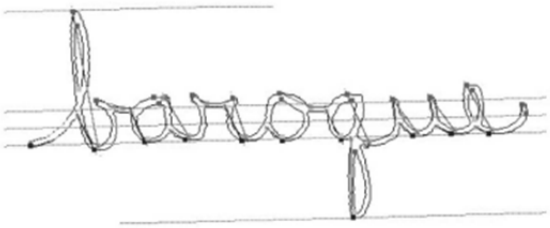


Figure 2.21: Segmentation à base du squelette [72].

caractéristiques des corps secondaires et autres configurations. Les corps principaux et secondaires sont obtenus en utilisant un algorithme à base de contour. Ensuite, une étape d'étiquetage est faite pour associer les corps secondaires à leurs corps principaux possibles.

A.3 Segmentation à partir des histogrammes de projection verticales

Cette approche est proposée par B. Yanikoglu et P. Sandon [38]. Elle consiste à calculer des histogrammes de projection horizontal pour identifier les différentes lignes du texte, et les histogrammes de projection vertical pour identifier ces différents caractères [89]. Les droites de segmentation sont celles qui contiennent le moins de pixels noirs, avec une contrainte d'espacement régulier dans l'image (voir figure 2.22). Néanmoins, Cette méthode montre des limites lorsque les lettres sont très proches ou enchevêtrées [78].

Figure 2.22: Exemple de projection verticale d'un mot [78]

A.4 Segmentation basée sur des réservoirs

Cette technique est utilisée dans la segmentation des manuscrits Gurmukhi et Oriya . Les caractères sont segmentés à base des points de la région de base du réservoir. Elle fonctionne comme suit : si l'eau coule de haut en bas du caractère, les régions de la cavité (trou, vide) des caractères où l'eau est stockée sont considérées comme des réservoirs (voir la figure 2.23). On nomme les réservoirs de haut les réservoirs obtenus quand l'eau coule du haut en bas. On n'utilise pas tous les réservoirs, seulement ceux qui sont plus grand d'un seuil vont être utilisés dans des traitements futurs. La valeur du seuil est 1/10 du caractère . Avec cette méthode nous pouvons arriver à un taux de segmentation de 93.5%, même si les caractères sont coupés ou chevauchés [30, 115].

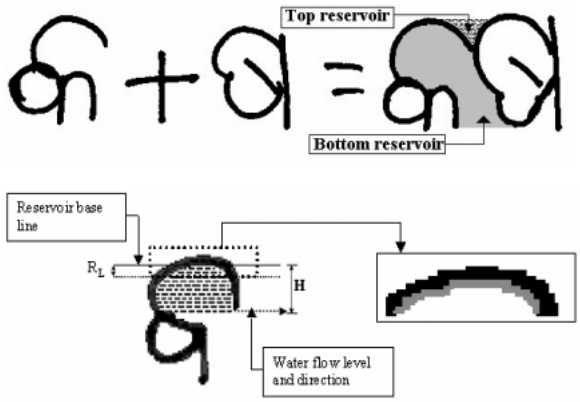


Figure 2.23: Un réservoir obtenu d'un flot d'eau du haut et marqué par des points [115].

B. Segmentation avec classification

Les techniques de cette catégorie font introduire l'utilisation d'une base de connaissance (des différents caractères ou même graphèmes) avec un classifieur (tel que ANN, HMM,...etc) afin d'obtenir un meilleur taux de segmentation.

B.1 Segmentation par propagation d'affinité (Affinity Propagation)

Segmentation par propagation d'affinité (AP : Affinity Propagation) c'est une méthode de clustering [118, 70]. Elle consiste à trouver un ensemble d'exemplaires (prototypes) dans les données et d'assigner d'autres points de données aux exemplaires les plus appropriés, puis à partitionner l'ensemble de données en clusters. L'objectif de propagation d'affinités c'est de maximiser la somme des similarités (affinités) entre les points de données et ses exemplaires [16].

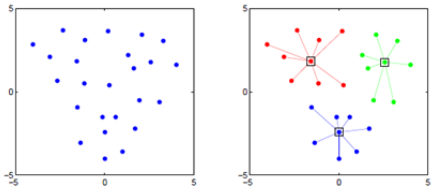


Figure 2.24: Exemple de points de données à clustrer (gauche). Une solution d'exemplaires basée sur le clustering par AP [16].

B.2 Segmentation à base de forêts aléatoires

Les auteurs dans [3] proposent une approche de segmentation des mots manuscrits arabes améliorée en utilisant les forêts aléatoires (RF: Random Forests). Elle fonctionne en deux phases : sur-segmentation de mots et validation des points de segmentation (SPs: segmentation points). La première phase dissèque les mots en différents composants sans laisser des caractères liés. Au cours de cette procédure certains caractères sont divisés en plus d'une partie, ce qui nécessite une étape de validation. Cette dernière, permet d'extraire des caractéristiques de zoning, en raison de leur force dans la capture des

caractéristiques locales et globales, pour les utiliser dans l'apprentissage des classificateurs efficaces de forêts aléatoires. Ensuite, les classificateurs RF vérifient chaque segment de caractère et valide les SPs.

B.3 Segmentation à base de Spline en Plaque Mince

Aouadi et al. [18, 17] présentent une approche de segmentation de texte en lignes. L'approche segmente les composants connexes (CCs) en deux étapes. Premièrement, elle cherche à trouver dans un dictionnaire un modèle similaire avec une segmentation correcte pour un CC. La recherche est basée sur le contexte de la forme et et la La méthode de Spline en Plaque Mince (SPM). Ensuite, elle segmente la CC en utilisant le point central des pièces de modèle similaire.

B.4 Segmentation à base de réseaux de neurones

Cheng et al.[34] proposent une approche qui améliore la segmentation à base de réseaux de neurones pour la reconnaissance de l'écriture manuscrite. La méthode utilise les deux valeurs de confiance de caractère gauche et du centre pour les fusionner avec les sorties d'une procédure de validation de segmentation. En outre, il utilise des caractéristiques de direction modifiées pour représenter les caractères et leurs points de segmentation, ce qui améliore le processus global. En plus, Cheng et Blumenstein [33] présentent une heuristique segmenteur améliorée à base de neurones pour l'amélioration de la segmentation des mots cursifs, et la validation des points de segmentation possibles. Premièrement, il utilise la détection des ligatures à l'aide de neurones pour localiser les points éventuels. Par la suite, il fusionne les valeurs de confiance de la reconnaissance des caractères de centre et gauche pour examiner et valider les points de segmentation. Khan et Mohammad [68] proposent une nouvelle méthode rapide pour segmenter les mots manuscrits cursifs en caractères. Dans le but d'accroître l'efficacité de l'approche un réseau de neurones artificiel est entraîné manuellement avec un grand nombre de points de segmentation valables à partir de mots cursifs. Ensuite, l'ANN entraîné utilise une analyse de ligature et de forme pour extraire les points de segmentation incorrects des nouveaux exemples de mots cursifs rapidement. En raison de la nature des caractères tels que m, n, u, v et w, une sur-segmentation se produit ce qui induit une inexactitude dans la localisation des limites des caractères.

2.6 Technique de segmentation proposée

La segmentation de l'écriture cursive est une phase très cruciale parmi les phases de la reconnaissance du manuscrit parce qu'elle extrait les principaux segments d'image nécessaire pour des traitements ultérieurs. Toutefois, le développement de ces techniques OCR reste insuffisant parce que les mots arabes manuscrits, les sous-mots, et les limites entre les caractères sont difficile à identifier avec l'existence de coupures et des

Tableau 2.2: Comparaison des méthodes de segmentation.

Méthode de segmentation	Critères de comparaison			
	Seuils	Balayage	Support des caractères enchevêtrés	Reconnaissance
A partir du squelette	Seuils prédéfinis	Rapide	Oui	Non
A partir du contour	Seuils dynamiques	Lourd	Oui	Non
A partir des histogrammes	Seuils prédéfinis	Très rapide	Non	Non
Basée sur les réservoirs	prédéfinis	Lourd	Oui	Non
Basée sur les fenêtres glissantes	prédéfinis	Lourd	Oui	Non
Propagation d'affinité	Seuils prédéfinis	Très lourd	Oui	Oui
basée forêts aléatoire	Seuils prédéfinis	Très lourd	Oui	Oui
basée sur les Spline en Plaque Mince	Seuils prédéfinis	Très lourd	Oui	Oui
basée sur les réseaux de neurones	Seuils dynamiques	Très lourd	Oui	Oui
Notre méthode	Seuils dynamiques	Rapide	Oui	Non

chevauchements dans les segments de lignes. De plus, les points permettent de distinguer entre les caractères de la même forme. Ces points sont présents comme des entités isolées au-dessus ou au-dessous de la forme de base, mais, peuvent être loin ce qui parfois crée une confusion entre les caractères adjacents. Souvent, l'écriture Arabe manuscrite produit plusieurs ascendants et descendants qui se touchent quand les lignes du texte sont proches, Ce qui rend la segmentation des documents Arabe très difficile [107, 60].

Dans ce travail, nous proposons une méthode de segmentation structurelle basée sur le principe d'appliquer une succession de scans afin d'identifier les différents points d'intérêts. Ces derniers sont de deux types : des points d'inclinaison permettant de détecter les liaisons ou connecteurs des caractères, et des points de structure permettant de représenter la forme d'un caractère). Le rôle principal de cette technique est de préparer un ensemble d'images représentant l'ensemble initial d'un puzzle. Ce dernier, va utiliser par la suite la définition de cet ensemble pour résoudre le problème des coupures et chevauchement dans les segments de lignes des caractères. Le choix de cette technique est fait en essayant d'avoir un compromis entre les critères suivants : utilisation des seuils, vitesse de traitement ou balayage, le support des caractères enchevêtrés et l'utilisation des classificateurs (voir le tableau 2.2).

La méthode passe par cinq étapes principales comme le montre la figure 2.25, à savoir:

1. Extraction des PAWs ;
2. Détermination des points de nature verticale (PNVs) et points de nature horizontale (PNHs);
3. Détermination des connecteurs des caractères (CCs);
4. Détermination des niveaux de complexité;
5. Scans des cercles cachées.

Le système utilise de manière récursive les huit voisins d'un pixel noir pour générer les différents composants connexe ou PAWs. Ensuite, pour chaque composant connexe,

cherche à trouver les points d'intérêt en utilisant parallèlement un scan vertical et un scan horizontal. Ces derniers utilisent des masques prédéfinies (voir les figures 2.27, 2.28 et 2.30). Les points détectés sont classés en quatre : points de fusion vertical (VFPs), points de division vertical (VDPs), points de fusion horizontal (HFPs) et points de division horizontal (HDPs) (voir la section 2.6.2). Ces différents points doivent être filtrés pour éliminer les points anormaux, par exemple deux points proches (distance<PS). Par la suite, si le nombre des VFPs et HFPs est égal à zéro le segment courant est considéré simple, et doit subir des scans pour détecter l'existence des cercles cachées. Sinon, le système cherche à trouver des connecteurs de caractères dans l'ensemble des points. La dernière opération permet d'avoir un ensemble de segments de base du segment source. Cet ensemble doit passer par un test de niveau de complexité afin de déterminer la complexité de chacun des segments. Si un segment est de structure simple ($s \in SS$), il est nécessaire d'appliquer des scans des cercles cachées pour détecter la présence des cercles cachées dans la structure du segment. Sinon, le système rajoute ce segment à l'ensemble des segments de base.
Les caractères de la langue Arabe peuvent être classés comme suit:

Tableau 2.3: Catégories des caractères arabes.

Catégorie	Caractères
Charactères Simple	ـكـ, م, ل.
Caractères Composés	ي, ن, ق, ف, غ, ظ, ض, ش, ز, ذ, ج, ث, ت, ب, أ, إ, آ.
Caractères Ambigus	ى, ا, و, ع, ط, ص, س, ر, د, ح.

En appliquant les mêmes étapes de segmentation sur ces trois catégories nous pouvons trouver les différentes formes présentées dans le tableau 2.4.

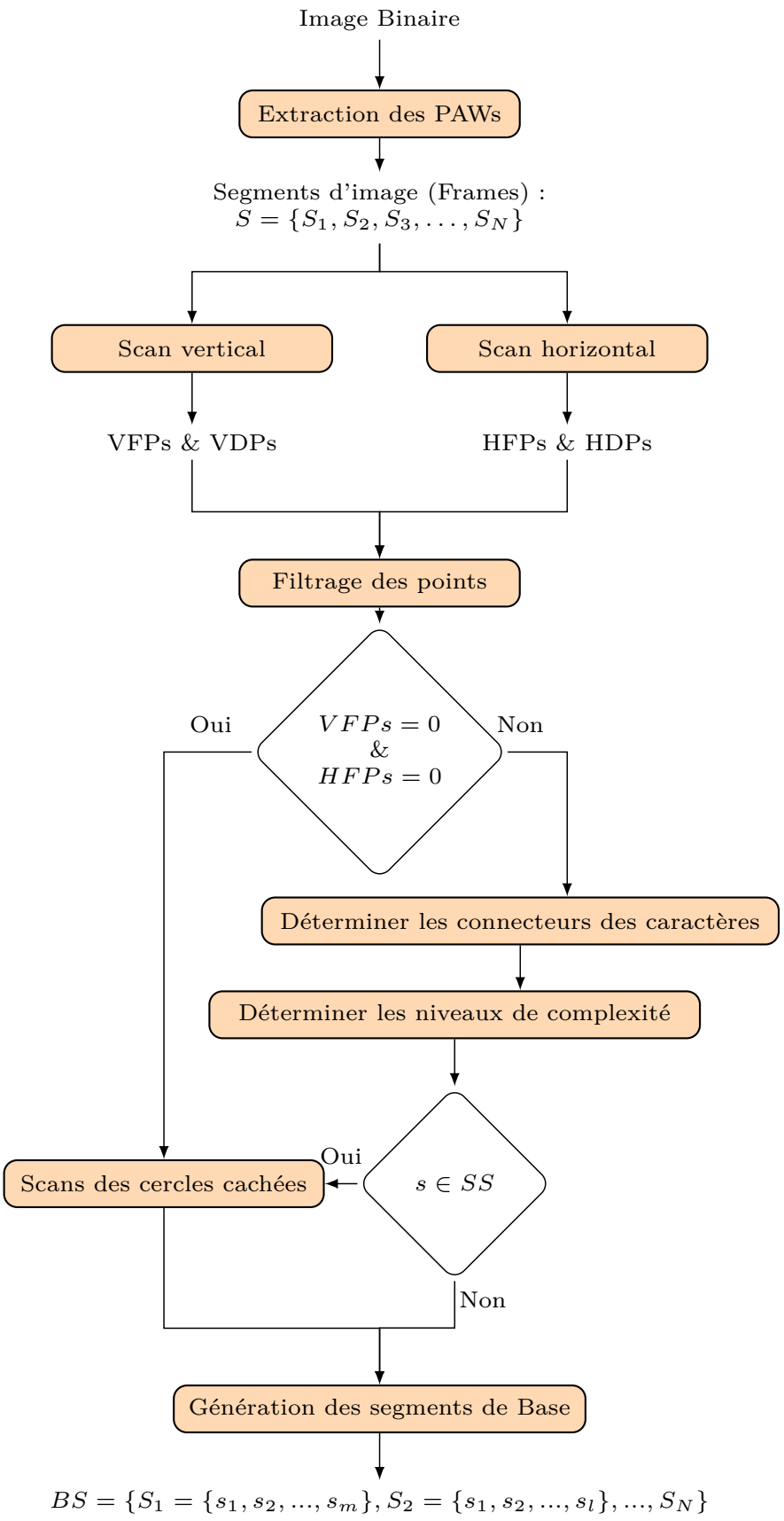


Figure 2.25: Extraction des segments de base.

Tableau 2.4: Formes principales utilisés pour la composition des mots arabes [47].

Forme \ Position	Début	Milieux	Fin	Isolée
Alif	ﺍ	ﺎ		ﺍ
Ba	ﺑ	ﺒ	ﺐ	
Hha	ﺣ	ﺤ	ﺞ	ﺢ
Del			ﺪ	ﺪ
Fa	ﻓ	ﻔ		
Ain	ﻋ	ﻌ	ﻊ	ﻉ
Ha	ﻫ	ﻬ	ﻪ	ﻩ
Kaf	ﻛ	ﻜ		ﻙ
Khaf		ﺧ		
Lam	ﻟ	ﻠ	ﻞ	
Mim	ﻣ	ﻤ	ﻢ	ﻡ
Ra			ﺮ	ﺭ
Sad	ﺻ	ﺼ		
Tta	ﻃ	ﻄ	ﻂ	ﻁ
Waw			ﻮ	ﻭ
1 Point				◆
2 Point				◆◆
Hamza				ء
Med				~

2.6.1 Extraction des PAWs

Le script arabe est constitué de deux types de PAWs: les grands PAWs qui représentent les caractères ou la forme principale d'un caractère, et les petits PAWs qui représentent les points diacritiques. Généralement, nous pouvons différencier entre ces deux types par la taille et la position. Néanmoins, dans ce cas de manuscrit il est difficile d'utiliser ces informations comme une règle.

Le système commence par un balayage de l'image binaire de droite à gauche afin d'avoir le premier point noir (pixel noir), ensuite il utilise les huit points voisins et une fonction récursive pour traquer le reste du PAW (voir figure 2.26).

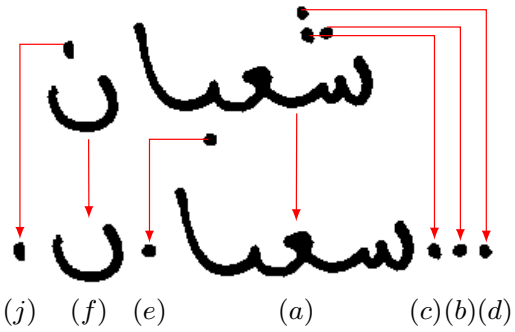


Figure 2.26: Segmentation du mot "شعبان" en PAWs [46].

Cette technique ne segmente pas tout le mot en formes de base des caratères, comme c'est le cas pour les PAWs (a) et (f), mais permet d'extraire les syllabes et les diacritiques.

Cette tâche est réalisée à l'aide de la fonction récursive 2.1:
Où:
Img: image source,
$Tmpimg$: image résultante de la segmentation,
$getWidth$: fonction permet d'obtenir la largueur d'une image (ex: img.),
$getHeight$: fonction permet d'obtenir la hauteur d'une image (ex: img.),
$getRGB(x, y)$: fonction permet d'obtenir les composantes du couleur du pixel en position (x,y),
$setRGB(x, y, v)$: fonction permet de changer les composantes du couleur du pixel en position (x,y) à la valeur v (v appartient à [0, 16777215]), IN_BF: fonction permet de connaître si la couleur d'un pixel est dans la plage du couleur noir ou non.

Jusqu'ici, nous obtenons un ensemble d'images nommée «$Image\_NumSéq.gif$» et des fichiers nommés «$Segment\_NumSeq.pos$» sauvegardant la position de chaque image produite dans l'image originale.

Algorithme 2.1: Fonction de construction des PAWs

```
Fonction ConstructSegment(p : Point): Point
    FULL_MARKER ← false : booléen
    m : matriced'entier[3, 3]
    x, y, h, k : entier
Début
                          /* Initialiser l'indicateur du marquage */
    FULL_MARKER ← faux
    Pour h ← 0 Jusqu'à 3 Faire
        Pour k ← 0 Jusqu'à 3 Faire
            x ← (p.x − 1) + h
            y ← (p.y − 1) + k
            Si ((x >= 0) et (x <= img.getWidth())) et
                ((y >= 0) et (y <= img.geHeight())) et
                (IN_BF((img.getRGB(x, y))) Alors

            Si ((x <> p.x)ou(y <> p.y)) Alors m[h, k] = 1;
            Sinon m[h, k] = 0;

            FULL_MARKER ← vrai;
            Si (x < Xmin) Alors Xmin = x;
            Si (y < Ymin) Alors Ymin = y;
            Si (x > Xmax) Alors Xmax = x;
            Si (y > Ymax) Alors Ymax = y;
              /* Changer le pixel correspondante de l'image temporaire en noir=0 */
            tmpimg.setRGB(x, y, 0);
            /* Changer le pixel correspondante de l'image source en blanc = 16777215
            */
            img.setRGB(x, y, 16777215);
            Sinon m[h, k] = 0;
        FinSi
        FinPour
    FinPour
    Si (FULL_MARKER) Alors  Pour h ← 0 Jusqu'à 3 Faire
        Pour k ← 0 Jusqu'à 3 Faire
            Si m[h, k] = 1; Alors
                x = (p.x − 1) + h;
                y = (p.y − 1) + k;
                ConstructSegment(nouveau Point(x, y));
            FinSi
        FinPour
    FinPour

    return Point(x, y);
Fin
```

2.6.2 Détermination des PNVs et PNHs

Cette étape utilise les deux masques présentés dans la figure 2.27 pour déterminer les deux types de points verticaux: Points de Fusion Verticaux (VFPs) en rouge et Points de Division Verticaux (VDPs) en vert.

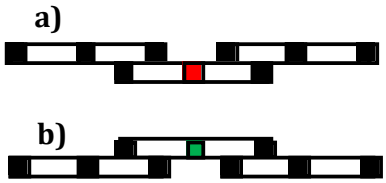


Figure 2.27: Masques de balayage vertical (a- masque de fusion b-masque de division) [47]

De la même façon, mais en utilisant des masques de balayage horizontal comme le montre la figure 2.28 il est possible de déterminer deux types de points horizontaux: points de fusion horizontaux (HFPs) en bleu et points de division horizontaux (HDPs) en jaune.

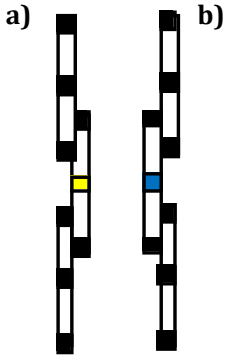


Figure 2.28: Masques de balayage horizontal (a-masque de division b-masque de fusion) [47]

Chaque masque est composé de trois vecteurs de pixels superposés. En plus, chaque vecteur possède trois pixels ou points de contrôle: point de début (SP), point de milieu (MP) et point de fin (FP). Les points générés peuvent contenir des points incorrects, ce qui nécessite une étape de filtrage. En se basant sur l'épaisseur (T: thickness) et la longueur (L: length) des lignes des segments en taille du stylo (PS: Pen Sise) il est possible d'éliminer l'un des points proche comme le montre la figure 2.29 quand $T \,\&\, L < pen\_size$. Ensuite si le nombre des points VFPs et HFPs filtrés est égale à zéro $((VFPs = 0)\, et\, (HFPs = 0))$ le PAW courant est considéré comme de structure simple (SS), et le système lance le scan des cercles cachées. Autrement, il cherche à trouver les connecteurs des caractères.

En outre, le système cherche à trouver les caractères de ligatures verticales en utilisant un masque différent. Ce masque incorpore des vecteurs verticaux avec une épaisseur égal à la taille du stylo (PS) et une longueur égale ou plus grande que la taille du stylo. Au-dessous de ces vecteurs nous trouvons un autre vecteur plus grand, qui contient la limite verticale du caractère présentée par le point rouge illustrée par la figure 2.30.

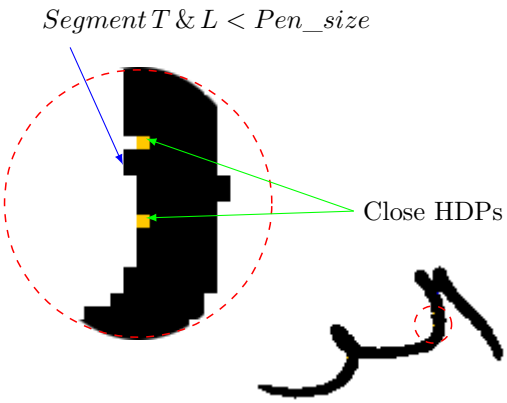


Figure 2.29: Exemple de filtrage de points .

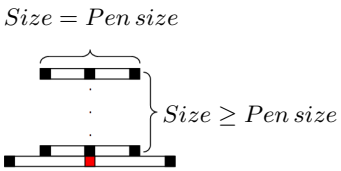


Figure 2.30: Masque des caractères de ligature verticale : ل, ب, ت,ث, ن et ي.

2.6.3 Détermination des Connecteurs de caractères

Après avoir appliqué les masques présentés ci-dessus, nous allons obtenir de l'image un ensemble de points (VFPs, VDPs, HFPs and HDPs) avec des différentes couleurs comme il est illustré dans la figure 2.31.

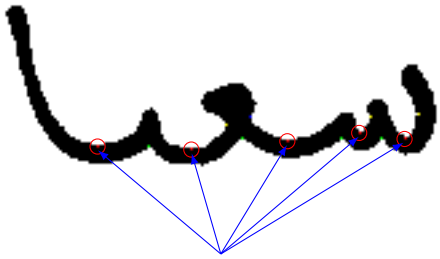


Figure 2.31: Extraction des connecteurs de caractères.

Maintenant, pour trouver les connecteurs des caractères nous appliquons les deux règles suivantes:

1. VFPs de niveau un et hors des cercles sont des connecteurs de caractères.
2. HFPs de niveau un, hors des cercles et non précédé par un connecteur VFP sont considérés des connecteurs de caractères (voir la figure 2.32).

Les VFPs sont utilisés pour regénérer le chemein de la ligne de base comme le montre la figure 2.33.
Nous pouvons remarquer que les différentes formes (voir la figure 2.31 et la figure 2.32)

(a) (b) (c) (d)

Figure 2.32: Exemples de segmentation des caractères ligaturés.

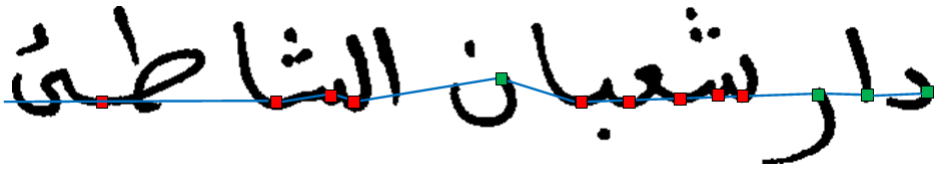


Figure 2.33: Exemple de détection de la ligne de base.

présentes entre les connecteurs des caractères sont similaires à celles dans le tableaux 2.4, et peuvent être utilisées pour reconstruire la définition des caractères, mots et texte.

2.6.4 Détermination des niveaux de complexité

Les segments des caractères obtenus possèdent deux niveaux de complexité de structure: simple ou complexe, et deux longueurs possibles: L1 pour long et L2 pour court comme le montre la figure 2.34. Ces différents niveaux sont inclus dans les caractéristiques. En plus, les segments de structure simple doivent subir un scan de cercles cachées.

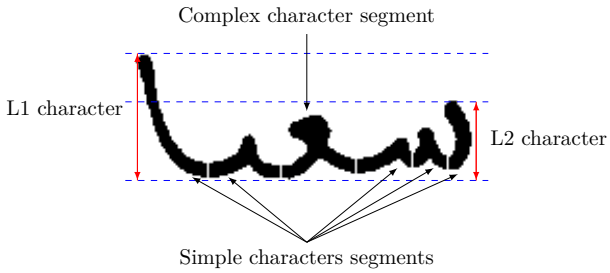


Figure 2.34: Exemple de niveaux de complexité.

2.6.5 Scan des cercles cachés

Comme il est mentionné avant, l'écriture manuscrite cursive produit des chevauchements dans les lignes des segments des caractères. Ce phénomène affecte beaucoup plus les caractères contenant des cercles comme : ـمـ, ـهـ, ـفـ, ـو,...etc. Ce qui génère un manque dans les informations des caractéristiques. Comme solution, les scans des cercles cachées sont lancés pour recréer la structure des cercles. Le processus lance un scan vertical et un autre horizontal pour trouver les lignes de segments qui garde une épaisseur égale à la taille du stylo. Si les deux scans remplissent cette condition, quatre points de contrôle sont ajoutés aux extrémités de la forme du caractère comme le présente la figure 2.35.

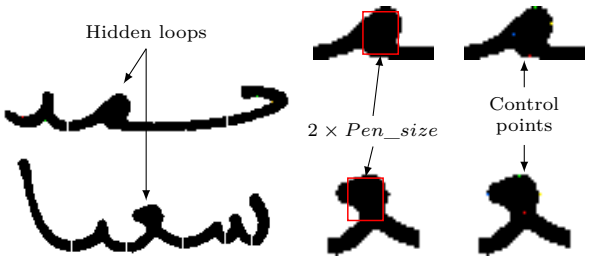


Figure 2.35: Exemple de regénération des cercles cachés.

Conclusion

Dans ce chapitre, nous avons commencé par présenter la problématique posée par la langue Arabe. Ensuite, nous avons présenté les notions de structure physique et structure logique. De plus, nous avons vu le principe général de la segmentation. Puis, nous avons présenté les différentes approches de segmentation. Nous avons exhibé quelques techniques de segmentation de l'écriture manuscrite. Enfin, dans la technique de segmentation proposée, nous avons exposé notre première contribution présentée dans une méthode structurelle basée multi-scans (N-Sans), qui a pour rôle de préparer l'ensemble initial du puzzle.

Chapitre 3

Classification, sélection des mots et post-traitement basé puzzle

Introduction

Dans le chapitre précédent, nous avons présenté premièrement les défis rencontrés lors de la segmentation de l'écriture manuscrite cursive. Ensuite, la différence entre la structure physique et celle logique d'un document, puis, nous avons vu le principe et les différentes approches et stratégies de segmentation. Finalement, nous avons exposé notre première contribution qui se résume en une technique de segmentation multi-scans.

Dans ce chapitre, nous allons voir dans un premier temps quelques méthodes de classification utilisées dans la reconnaissance du manuscrit, parmi lesquelles, on trouve la méthode des machines à vecteurs de support qui a été utilisé dans notre travail. Souvent, la phase de classification est suivie par une phase de post-traitement afin d'améliorer les résultats de classification en utilisant des informations de plus niveaux (syntaxique, sémantique,... etc.) non disponible durant la phase de classification. Pour cela, nous allons présenter quelques méthodes de sélection en post-traitement. Enfin, nous allons voir notre deuxième contribution : la technique de sélection à base du puzzle, qui permet de résoudre le problème de coupures et chevauchements dans les lignes des segments des caractères.

3.1 Méthodes de classification

Dans cette section, nous allons présenter un ensemble de classificateur couramment utilisés dans la reconnaissance de l'écriture. Nous passerons en revue les approches statistiques, les approches logiques et les approches bio-inspirées.

3.1.1 Approches statistiques

Les systèmes de Reconnaissance des formes basés sur l'approche statistique ont été utilisés avec succès dans plusieurs applications commerciales. Des concepts bien connus de la théorie des statistiques sont utilisés pour établir les frontières de décision entre les classes des formes. Dans cette catégorie on trouve:

- KNN: K-Nearest Neighbors;
- HMM: Hidden Markov Model; et
- SVM: Support Vector Machine.

A. KNN: K-Nearest Neighbors

C'est l'un des plus simples algorithmes de classification supervisé, c-à-d pas de phase d'apprentissage de paramètres. Nous pouvons l'utiliser dans les deux cas : classification et/ou régression. Il est très utilisé pour sa simplicité d'interprétation et son faible temps de calcul. Ce classificateur est très utile quand on ne dispose pas de connaissances préalables sur la distribution de probabilité des classes. Il utilise comme base l'hypothèse que les points proches dans l'espace de primitives possèdent une grande probabilité d'appartenir à la même classe. Pour cela, il calcule une distance entre un nouvel exemple x et tous les échantillons donnés pour prendre une décision. Puis, il sélectionne les K plus proches échantillons pour affecter x à la classe possédant le nombre d'occurrence maximum. L'estimation du paramètre K peut être faite en utilisant une validation croisée. Généralement k est pris de 5 à 10 pour des données de faible dimension. La distance entre deux éléments x et p est calculée en utilisant l'une des fonctions présentées ci-dessous:

1. Euclidien: $\sqrt{(x-p)^2}$;
2. Euclidien carré: $(x-p)^2$;
3. Cityblock: $Abs(x-p)$;
4. Chebyshev: $Max(|x-p|)$

A.1 Prédiction KNN

Après avoir choisi la valeur de k, une nouvelle prédiction peut être faite en utilisant les exemples de KNN. Pour la régression, les prédictions sont la moyenne des résultats des k plus proches voisins.

$$y = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{3.1}$$

Où y_i est le i ème cas de l'exemple des échantillons, et y est la prédiction. Contrairement à la régression, dans le cas des problèmes de classification, les prédictions KNN sont basées sur un schéma de vote dans lequel le gagnant est utilisé pour étiqueter la réponse.

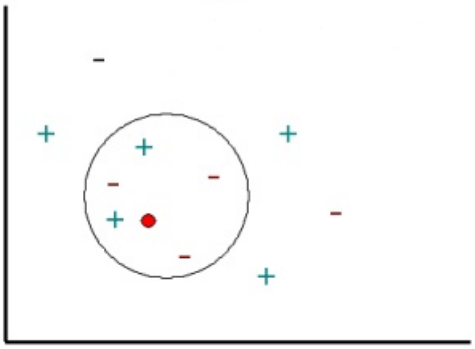


Figure 3.1: Example de classification par KNN, K=1 le résultat est le plus, K=2 va être inconnu, k=3 va être le moins.

A.2 KNN pondérée

Jusqu'ici, nous avons considérés que les K voisins possèdent la même influence sur les prédictions quelque soit leurs distances à l'exemple de test. Une autre solution (Shepard 1968) consiste à utiliser arbitrairement de grandes valeurs de K avec plus d'importance donnée aux cas plus proche de l'exemple de test. Ceci est réalisé en utilisant ce qu'on appelle pondération de la distance. Ceci peut être obtenu en introduisant un ensemble de poids W, une pour chaque voisin le plus proche, définie par la proximité relative de chacun de ses voisins par rapport au point d'interrogation. Ainsi:

$$W(x, p_i) = \frac{exp(-D(x, p_i))}{\sum_{i=1}^{k} exp(-D(x, p_i))} \tag{3.2}$$

Où $D(x, pi)$ est la distance entre l'exemple de test est le i ème cas p_i de l'exemple échantillon. Il est clair que les poids définis de cette manière ci-dessus répondront :

$$\sum_{i=1}^{k} W(x_0, x_i) = 1 \tag{3.3}$$

Pour les problèmes de classification, le maximum de l'équation ci-dessus est prise pour chacune des variables de classe. Il est clair de la discussion ci-dessus que lorsque k> 1, on peut naturellement définir l'écart-type pour les prédictions dans les tâches de régression en utilisant :

$$err\,bar = \mp\sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(y - y_i)^2}$$

B. HMM: Hidden Markov Model

Les Modèles de Markov Cachés (en anglais : hidden Markov model (HMM)) ont été inventé par Neuwirth, et développés dans les années 1960 et début 1970. Ils présentent

un modèle statistique dans lequel le système modélisé est supposé être un processus markovien de paramètres inconnus [24]. Ils sont définis par :

1. un ensemble d'états: S=$\{S_1, S_2, \ldots, S_N\}$.
2. le processus se déplace d'un état à un autre générant une séquence d'états: S_{i1}, S_{i2}, $\ldots$, S_{ik}, $\ldots$
3. probabilité de chaque état ultérieur ne dépend que de ce qui était l'état précédent:

$$P(S_{ik}|S_{i1}, S_{i2}, \ldots, S_{ik}) = P(S_{ik}|S_{ik-1})$$

4. les états ne sont pas visibles, mais chaque état génère au hasard une des M observations (ou états visibles): $\{v_1, v_2, \ldots, v_N\}$.
5. Pour définir le modèle de Markov caché, les probabilités suivantes doivent être spécifiés :
   (1) matrice des probabilités de transition A=(a_ij), a_{ij}= P($s_i \mid s_j$);
   (2) matrice des probabilités d'observation B=$(b_i\ (v_m\ ))$,$b_i(v_m\ )$ = P($v_m \mid s_i$);
   (3) un vecteur de probabilités initiales: $\pi = (\pi_i)$, π_i=P(S_i);
   (4) le modèle est représenté par M=(A, B, π).

Exemple: supposons que nous avons (voir figure 3.2):

- les deux états de pression atmosphérique: "Low" pour faible, et "High" pour haute.
- deux observations: "Rain" pour pluie, et "Dry" pour sèche.
- les probabilités de transition: P("Low"|"Low")=0.3 , P("High"|"Low")=0.7 , P("Low"|"High")=0.2, P("High"|"High")=0.8.
- probabilités d'observation: P("Rain"|"Low")=0.6, P("Dry"|"Low")=0.4 , P("Rain"|"High")=0.4 , P("Dry"|"High")=0.3.
- probabilités initiales disent:P("Low")=0.4, P("High")=0.6.

Supposons que nous voulons calculer une probabilité d'une séquence d'observations dans notre exemple, "Dry","Rain". Considérons que toutes les séquences d'états cachés possibles :

P({"Dry","Rain"})= P({"Dry","Rain"},{"Low","Low"}) +
P({"Dry","Rain"},{"Low","High"}) + P({"Dry","Rain"},{"High","Low"}) +
P({"Dry","Rain"},{"High","High"}).

Où le premier terme est:

P("Dry","Rain" , "Low","Low")= P("Dry","Rain" | "Low","Low") P("Low","Low") =
P("Dry"|"Low")P("Rain"|"Low") P("Low")P("Low"|"Low") = 0.4*0.4*0.6*0.4*0.3

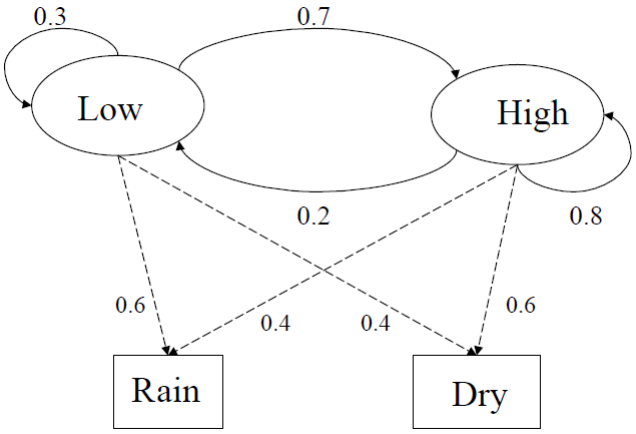


Figure 3.2: Example de modèles de markov cachés.

B.1 Problèmes de base des HMM

Les HMM souffrent de quelques problèmes de base qui peuvent être vus selon trois axes, à savoir : évaluation, décodage et entraînement. Pour chaque problème on trouve une solution proposée comme le montre le tableau 3.1.

Tableau 3.1: Problèmes de base des HMM.

Problème	Description	Solution
Évaluation	calculer la probabilité d'observation de la séquence d'observations étant donnée un HMM	Forward Algorithm
Décodage	trouver la séquence d'états qui maximise la séquence d'observations	Viterbi Algorithm
Entraînement	ajuster les paramètres du modèle HMM afin de maximiser la probabilité de générer une séquence d'observations à partir de données d'entraînement	Forward-Backward Algorithm

B.2 Reconnaissance de caractères

Dans cette section, nous allons présenter un exemple sur la reconnaissance de caractères d'un mot imprimé. On suppose que tous les caractères sont séparés tel qu'il est présenté par figure 3.3.

Figure 3.3: Mot imprimé.

Les probabilités que les sorties du système de reconnaissance des caractères peut être un caractère particulier sont P(image|caractère).

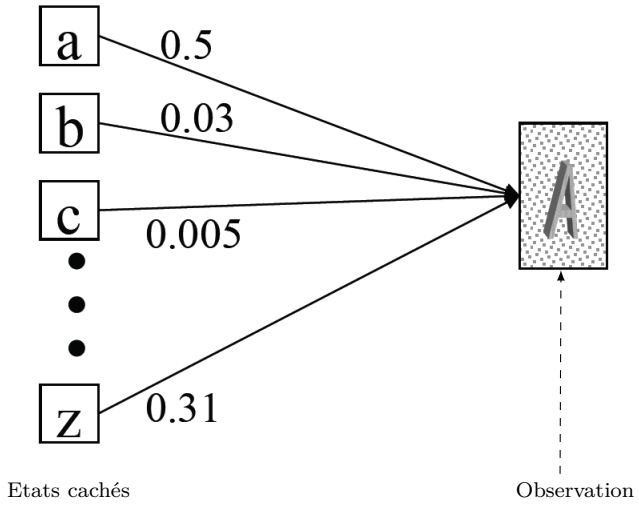


Figure 3.4: Exemple d'états cachés.

De l'exemple ci-dessus nous pouvons noter que :

1. les états cachés des HMMs sont l'ensemble des caractères.
2. les observations sont les images des caractères tapées segmentés de l'image source v_α. Il à noter qu'il y'a un nombre infini des observations
3. les probabilités d'observation sont les taux de reconnaissance des caractères.

$$\mathrm{B}{=}\mathrm{P}(b_i(v_\alpha)){=}(\mathrm{P}(v_\alpha|S_i))$$

Les probabilités de transition seront définies différemment dans les deux modèles suivants. Si lexique est donné, nous pouvons construire des modèles HMM séparés pour chaque mot de vocabulaire.

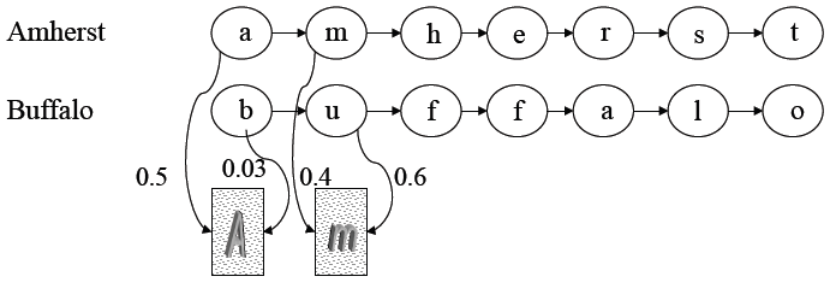


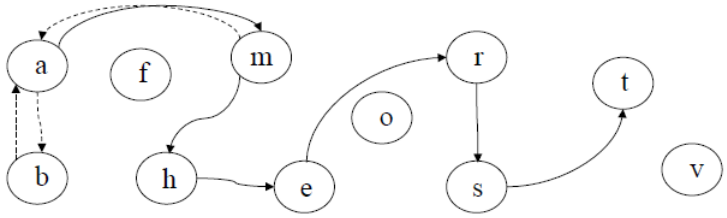
Figure 3.5: Example de comparaison en HMM.

Dans ce cas, la reconnaissance d'image de mot est équivalente au problème de l'évaluation de quelques modèles HMM. C'est une application du problème d'évaluation. Dans la réalité, nous pouvons construire un seul HMM pour tous les mots. Pour cela, nous allons considérer que :

1. Les états cachés sont tous les caractères de l'alphabet.

2. Les probabilités de transition et les probabilités initiales sont calculées à partir du modèle du langage.

3. Les observations et les probabilités d'observation sont comme avant.



Ici, nous devons déterminer la meilleure séquence d'états cachés, celui qui produit au mieux l'image du mot. C'est une application du problème de décodage. Maintenant, nous pouvons choisir une structure des états cachés. Ensuite, Les observations peuvent être des vecteurs de caractéristiques extraits à partir de tranches verticales. En plus, le mappage probabiliste des états cachés aux vecteurs caractéristiques peut être réalisé par les deux techniques suivant :

1. utiliser un mélange de modèles gaussiens.

2. Quantification de l'espace des vecteurs caractéristiques.

C. SVM: Support Vector Machine

Initialement, les SVM ont été inventé en 1963 par le russe Vladimir N. Vapnik et Alexey Ya. Chervonenkis. Ensuite, en 1992 Bernhard E. Boser, Isabelle M. Guyon et Vladimir N. Vapnik ont suggéré un moyen de créer des classificateurs non linéaires en appliquant l'astuce du noyau aux hyperplans à marge maximale. La version standard actuel à marge douce a été proposée par Corinna Cortes et Vapnik en 1993 et publié en 1995. Premièrement, elles ont été développées pour une classification binaire. Elles possèdent l'avantage de traiter des problèmes de grandes échelles, en mettant en jeu des descripteurs de grandes tailles, et en assurant une seule solution (pas de problèmes de minimum local comme pour les réseaux de neurones). Les bons résultats de leur application sur des problèmes réels ont fait des SVM un classificateur célèbre. Le principe de base est de trouver une frontière de décision linéaire entre deux classes, le modèle est enrichi par la suite en faisant une projection dans un autre espace permettant d'augmenter la séparabilité des données, et d'appliquer le même algorithme dans le nouvel espace, ce qui se traduit par une frontière de décision non linéaire dans l'espace initial [75].

C.1 Notions de base: Hyperplan, marge et support vecteur

Les SVM cherchent à trouver une frontière permettant de séparer deux classes de données, et en garantissant une distance maximale entre ces deux classes. Cette frontière au milieu est appelée hyperplan. La figure 3.6 montre un exemple d'hyperplan séparant

deux ensembles de points [57].

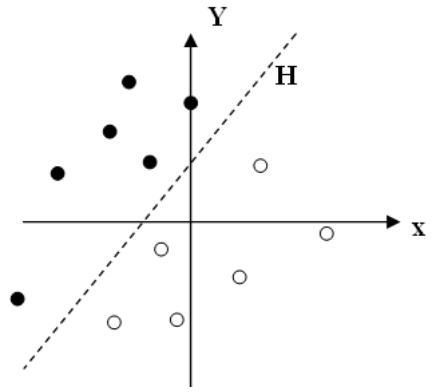


Figure 3.6: Exemple d'hyperplan [57].

Seuls les points proches sont utilisés dans la détermination de l'hyperplan. Ces ensembles de points sont appelés les vecteurs de support (voir figure 3.7).

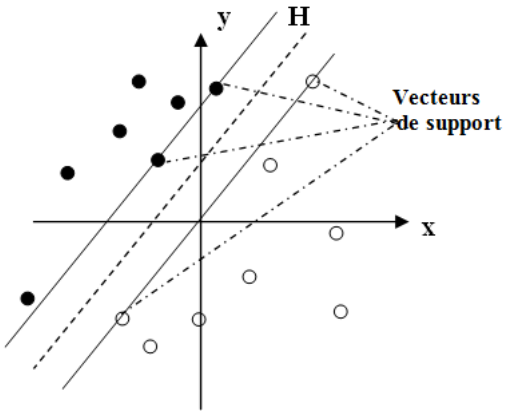


Figure 3.7: Exemple de vecteurs de support [57].

Cet hyperplan doit être optimal, c'est à dire celui qui passe au milieu des deux classes d'exemples. Cela revient à maximiser la distance ou la marge entre l'hyperplan et les exemples. C'est pour cette raison que les SVM sont appelées aussi les séparateurs à vaste marge (voir la figure 3.8) [37].
Une marge plus large permettra au mieux de classer les nouveaux exemples. La figure 3.9 montre qu'avec un hyperplan optimal, le nouveau exemple est bien classé, alors que dans la partie gauche avec une petite marge, l'exemple est mal classé [57].

Généralement, la classification d'un nouveau exemple est faite en se basant sur sa position par rapport à l'hyperplan optimal. Dans la figure 3.10, le nouveau élément sera classé dans la catégorie des « + ».

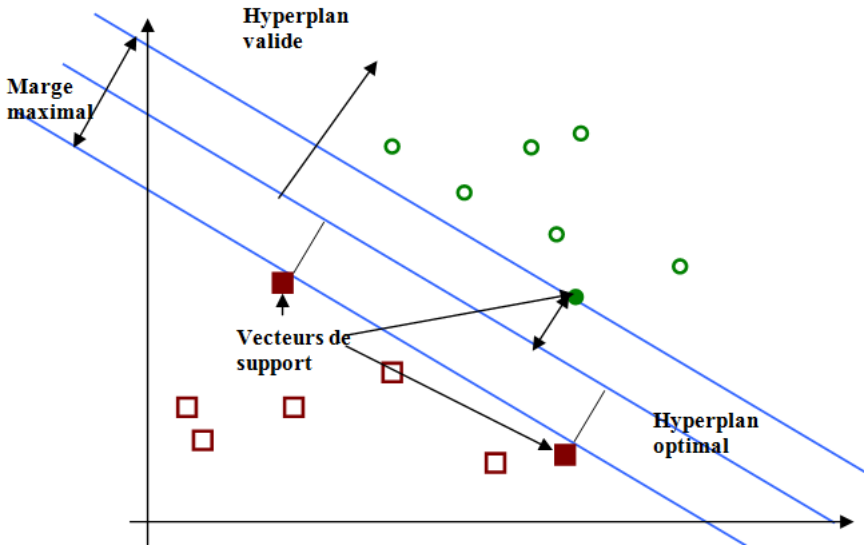


Figure 3.8: Exemple de marge maximal (hyperplan valide).

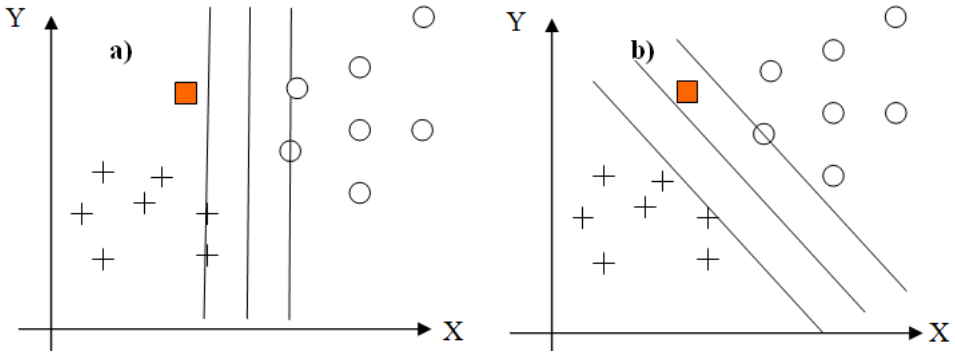


Figure 3.9: **a)** Hyperplan avec faible marge, **b)** Meilleur hyperplan séparateur [57].

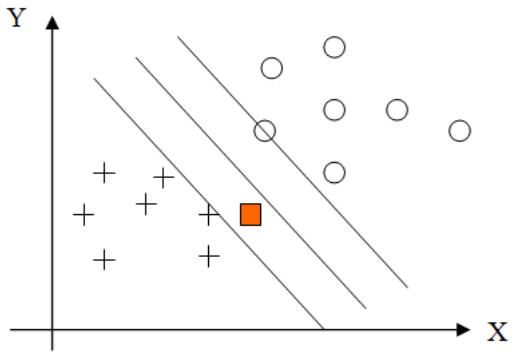


Figure 3.10: Exemple de classification d'un nouvel élément.

C.2 Linéarité et non-linéarité

Généralement, deux cas de classification sont possibles : les cas linéairement séparables et les cas non linéairement séparables. Dans le premier cas il est facile de trouver un classificateur linéaire. Autant que dans le second cas, il n'est pas possible de trouver le classificateur de marge maximale [57].

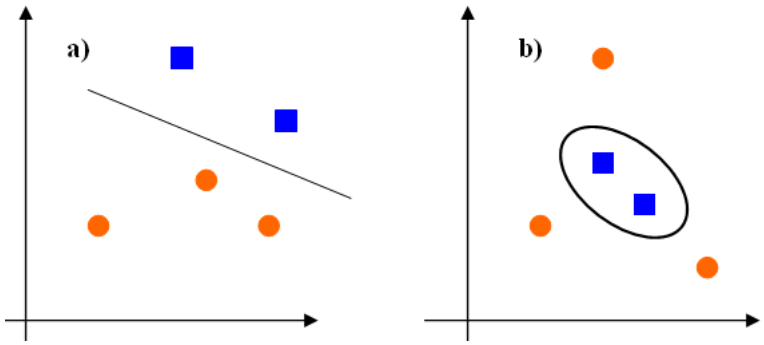


Figure 3.11: **a)** Cas linéairement séparable, **b)** Cas non linéairement séparable [57].

Afin de résoudre ce problème, un changement d'espace de données est réalisé. Cette nouvelle dimension est appelée « espace de re-description ». Il est clair que plus la dimension de l'espace de re-description est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée. Cette transformation permet de faire une séparation linéaire dans le nouveau espace (voir la figure 3.12) [57].

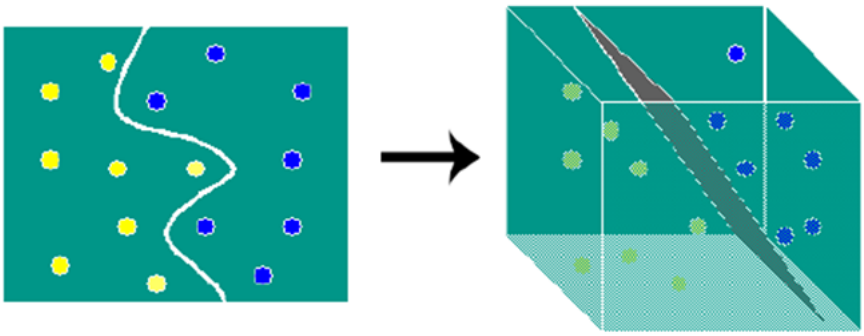


Figure 3.12: Exemple de changement de l'espace de données.

Cette transformation non linéaire est réalisée via une fonction noyau. En pratique, plusieurs fonctions existent, parmi les polynomiale, gaussien, sigmoïde et laplacien. Pour déterminer la meilleure fonction pour un système donné il faut faire des tests [57].

Tableau 3.2: Exemples de fonctions noyau [117].

Fonction linéaire	$k(x, x') = x.x'$
Fonction polynomial	$k(x, x') = (x.x')^d \quad ou \quad (c + x.x')d$
Fonction gaussien	$k(x, x') = e - \|x - x'\|^{2/\sigma}$
Fonction Laplacien	$k(x, x') = e - \|x - x'\|^{1/\sigma}$

En pratique, des noyaux simples sont combinés pour obtenir des fonctions plus complexes [26].

C.3 Algorithme générale de SVM

Avant la discussion des détails de l'algorithme SVM, nous notons que lors de l'implémentation, il faut considérer une précision pour le zéro numérique. Ceci est de

toute façon vrai pour toute implémentation. Par exemple deux données sont considérées égales si elles diffèrent de moins de 10^{-3}. Cette valeur est proposée par Platt. Cette précision est utile lors de recherche des données violant les conditions de KKT et pour contrôler l'optimisation des couples de multiplicateurs de Lagrange. Elle influe sur la vitesse de convergence de l'algorithme : plus elle est faible et plus l'algorithme est lent à converger. Cette précision est appelée zéro_ tolérance dans l'algorithme 3.1 [117].

Algorithme 3.1: Algorithme de résolution SVM.

Entrée:
Q,b,y,Cp,Cn, et an des points initial faisable alpha;
l: la taille des vecteurs et matrices;
eps: critère d'arrêt.

Sortie:
$alpha$: vecteur contenant la solution;
obj: la valeur objective;

Variables locales:
G,G_bar: vecteurs utiliser pour les calcules du gradient; $alpha\_status$: vecteur contenant l'état de alpha; $active\_size$: variable contenant la taille du vecteur $active\_set$; $active\_set$: vecteur contenant l'ensemble de travail courant;

Début

1. Initialisation des paramètres : taille de l'échantillon (l), paramètre c, largeur de bande du noyau (gaussien), précision numérique eps (zéro_tolérance).
2. Initialiser le vecteur alpha_status (vecteur contenant l'état d'alpha).
3. Initialiser la variable active_size et le vecteur active_set (utilisés durant l'opération de shrinking).
4. Initialiser le gradient.
5. Commencer l'optimisation:
   a. Faire l'opération de shrinking.
   b. Reconstruire tout le gradient.
   c. Réinitialiser active_size et choisir un ensemble de travail.
   d. Mis à jour d'α_i, α_j, G, G_bar et $alpha\_status$.
6. Calculer la valeur objective.
7. Affichage des résultats.

Fin

C.4 Versions améliorées

Les SVMs ont été amélioré pendant ces dernières décennies. Plusieurs travaux ont été faits à ce classificateur dans le but d'atteindre un temps et un taux de classification meilleur. Du et al. [40] ont combiné les SVMs avec des analyses d'ondelette afin de construire un classificateur SVM d'ondeltte (WSVM: Wavelet SVM) basé sur des fonctions noyau ondelette pour reproduire le noyau de l'espace de Hilbert (RKHS: Reproducing Kernel Hilbert Space). Ayat et al. [19] suggèrent d'utiliser un nouveau modèle de sélection de critères basé sur l'estimation des probabilités d'erreurs du classificateur SVM. Mathias Adankon et al. [7] proposent de régler les les hyper-paramètres de LS-SVM (Least Squares SVM) en utilisant les critères d'erreurs empirique dans une procédure de validation croisée qui laisse un seul paramètre(Leave-One-Out (LOO) cross-validation).

Jian et al. [13] cible le problème d'apprentissage avec noyau multiple dans le cas des LS-SVM en le formulant comme une programmation semi-définit (SDP: Semi-Definite Programming), et montre que les paramètres de régularisation peuvent être optimisés dans une framework unifiée avec le noyau , ce qui guide à un processus automatique pour la sélection du modèle. Zhou et al. [122] présentent une nouvelle approche de sélection d'exemple de noyau de coque convexe de sous-classe (Kernel Subclass Convex Hull (KSCH) sample selection approach), qui essaye de sélectionner les frontières des exemples de chaque coque convexe d'une classe. L'idée de sélection des exemples est dérivée de l'explication géométrique de SVM.

Tian et al. [114] proposent une version de noyau multiple des SVM transductive (approche basée sur l'hypothèse de cluster), qui est résolut en se basant sur la programmation de différence des fonctions convexes (DC(Difference of Convex functions) programming). He et al. [58] proposent une approche d'apprentissage multi tâches qui utilise SVM mono-classe avec l'aide d'un nouveau design du noyau. Chen et al. [32] proposent les SVM multi noyau (MK-SVM: multiple-kernel SVM) pour le datamining orientés multi tâches. Différemment, des SVM standard qui sont souvent vues comme une boite noire, les SVM multi noyau sont basées sur les systèmes de datamining utilisant les expressions génétiques qui sont applicables pour la sélection des caractéristiques, fusion des données, prédiction des classes, extraction des règles de décision, extraction des règles associées et la découverte des sous classes. Chitrakar et al. [36] proposent une approche améliorée et efficace des SVM incrémentales basée sur l'idée de conserver les exemples de données courantes et celles originales pendant tout le processus d'apprentissage.

3.1.2 Approches logiques

Dans cette catégorie nous trouvons les classificateurs utilisant un ensemble de règles logiques durant la phase de classification. Parmi ces méthodes on trouve : Classification

C-moyenne floue et Classification à base de règles.

A. Classification C-moyenne floue

La méthode C-moyenne floue utilise une partition floue tel que chaque point de l'espace de caractéristiques peut appartenir à toutes les classes (clusters) avec différents degrés d'appartenance entre 0 et 1. Cette méthode est fréquemment utilisée en reconnaissance de formes. L'objectif de C-moyenne floue est de trouver les centres des groupes (classes) qui minimisent la fonction coût (objective) suivante [86, 73, 109]:

$$J_m = \sum_{i=1}^{c} \sum_{j=1}^{n} {u_{ij}}^m \|x_i - c_j\|^2 \tag{3.4}$$

$$s.t \sum_{i=1}^{c} {u_{ij}}^m = 1, \forall j = 1, ..., n$$

Où m est un nombre réel (exposant flou) supérieur à 1, u_{ij} est le degré d'appartenance de x_i au cluster j, x_i est le $i^{ème}$ donnée, c_j est le centre du cluster j, et $\| * \|$ est la norme Euclidienne. La partition floue est exécutée à travers une optimisation itérative de la fonction coût (3.4) en ajustant les coefficients d'appartenance u_{ij} et les centres des clusters c_j par :

$$c_i = \frac{\sum_{j=1}^{n} {u_{ij}}^m x_j}{\sum_{j=1}^{n} {u_{ij}}^m} \tag{3.5}$$

et

$$u_{ij} = \frac{1}{\sum_{i=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{(m-1)}}}$$

Les itérations seront répétées tant que $(max_{ij}\left\{u_{ij}^{(k+1)} - u_{ij}^{k}\right\} > \varepsilon)$, où ε est un critère d'arrêt entre 0 et 1.

Cet algorithme a comme paramètre d'entrée le nombre de classes c. Ils divisent l'ensemble d'objets en c classes. D'abord ils proposent les centres des groupes et ensuite ils assignent chaque objet au centre le plus proche. Dans ces méthodes, le premier choix de centres est aléatoire, et les résultats sont sensibles à l'initialisation.

B. Classifiation à base de règles

Généralement, les systèmes de classification à base de règles opèrent en dérivant un ensemble de règles logiques de classification à partir d'un ensemble de données

précédemment classées. Par la suite, ces règles vont permettre de classer de nouvelles examples de données [100].

3.1.3 Approches bio-inspirées

Au contraire aux méthodes formelles qui exigent un énorme effort de calcul, et qui échouent lorsque la taille du problème est grande, les méthodes d'optimisation bio-inspirées proposent une solution informatique efficace. Elles présentent des métaheuristiques basés sur l'amélioration itérative d'une population des solutions, ou d'une solution simple et utilisant la randomisation et la recherche locale pour résoudre un problème d'optimisation donné [39, 81, 84]. La figure 3.13 illustre les différentes méthodes d'optimisation bio-inspirées.

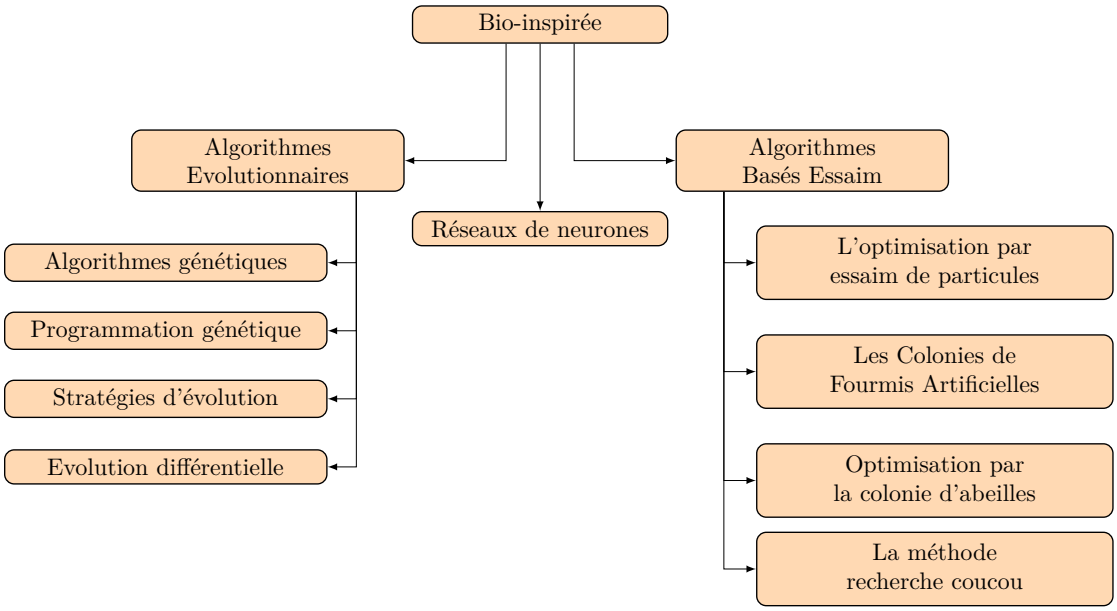


Figure 3.13: Classification de méthodes bio-inspirées [81].

A. ANN: Artificial Neural Network

Les réseaux de neurones se caractérisent par la facilité de mise en œuvre et leurs bonnes performances, ce qui les a rendus parmi les outils les plus utilisées en classification pendant les deux dernières décennies. L'idée de neurone formel vient de celle de neurone biologique (voir figure 3.14). A partir d'une description d'une observation x définit par un vecteur d'attributs numériques (x_1, x_2,..., x_d). Chaque x_i est pondérée par un poids synaptique w_i. Également un biais w_0 est ajouté. Finalement, en appliquant une fonction d'activation nous obtenons la sortie z du neurone (voir le tableau 3.3).
Souvent, un réseau de neurones possède plusieurs couches, et pour chaque couche une fonction d'activation est utilisée. Progressivement, le réseau apprend de façon itérative en se basant sur la rétro-propagation du gradient d'erreur. Les fonctions d'une couche doivent être dérivables par rapport aux entrées de la couche suivante pour assurer la

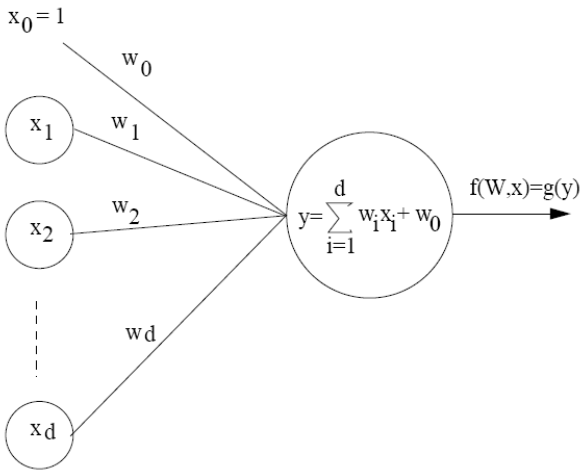


Figure 3.14: Example de neurone artificiel.

Tableau 3.3: Exemples de fonctions d'activation.

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

rétro-propagation du gradient d'erreur d'une couche à une autre. En plus, il est possible d'optimiser ces poids par descente de gradient par l'équation 3.6.

$$w_i(t+1) = w_i(t) - \lambda \frac{\partial Err}{\partial w_i(W)} \tag{3.6}$$

Tel que λ est le coefficient d'apprentissage. Si cette valeur trop grande, l'apprentissage risque de diverger. Par contre, si elle est trop petite il risque de converger vers un minima local.

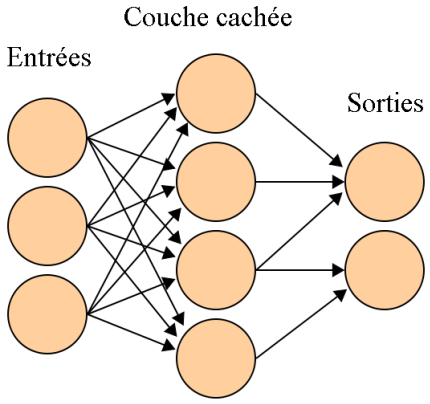


Figure 3.15: Example de neurone artificiel.

B. Algorithmes évolutionnaires

Ils s'inspirent de la théorie de l'évolution pour résoudre des problèmes connus en informatique. Dans le but de trouver le meilleur résultat ils évoluent un ensemble de solutions donné. Ils utilisent itérativement des processus aléatoires, par conséquent ils sont considérés comme des algorithmes stochastiques. Souvent, ces algorithmes sont utilisés pour résoudre des problèmes d'optimisation [39, 84]. Dans cette catégorie on trouve :

- Algorithmes génétiques;
- Programmation génétique;
- Stratégies d'évolution;
- Evolutions différentielles.

B.1 Algorithmes génétiques

Les algorithmes génétiques s'inspirent des mécanismes de l'évolution biologique pour les transposer à la recherche de solutions adaptées au problème qu'on cherche à résoudre. Premièrement, des génotypes sont choisis (intégrés aux chromosomes) sur la base de l'adaptation relative à leur environnement des phénotypes qu'ils génèrent (la qualité de cette adaptation est mesurée par la performance - fitness - relative de chaque génotype). Ensuite, les génotypes mieux adaptés à leur environnement ont une plus grande facilité à se reproduire et la reproduction (croisement) assure le croisement des gènes les plus performants dans la population [39, 84]. un algorithme génétique est défini par quatre éléments de base suivants:

1. **Individu/chromosome/séquence :** une solution potentielle du problème qui correspond à une valeur codée de la variable (ou des variables) en considération;
2. **Population :** un ensemble de chromosomes ou de points de l'espace de recherche (donc des valeurs codées des variables);
3. **Environnement :** l'espace de recherche (caractérisé en termes de performance correspondant à chaque individu possible);
4. **Fonction de performance(fitness) :** la fonction - positive - que nous cherchons à maximiser, car elle représente l'adaptation de l'individu à son environnement.

B.2 Programmation génétique

Elle consiste à faire évoluer une population composée d'un grand nombre de programmes. Premièrement, ces programmes sont créés aléatoirement. Par la suite, chaque programme est évalué selon une méthode propre au problème posé. Avec chaque itération (génération) les programmes sont classés selon les notes obtenues, ce qui génère une nouvelle

population, où les meilleurs programmes auront une plus grande chance de survivre ou d'avoir des enfants que les autres [39, 84].

B.3 Stratégie d'évolution

Cette méthode a été développée par trois étudiants (Bienert, Rechenberg et Schwefel) de l'université technique de Berlin en 1964. C'est la première véritable métaheuristique évolutionnaire (avant les algorithmes génétique). Par la suite, durant les années 1960, des développements ont été apporter sur l'algorithme par IngoRechenberg, P. Bienert et Hans-Paul Schwefel sur la conception de profils aérodynamiques [81, 85].Appliqué à l'optimisation numérique, l'algorithme manipule itérativement un ensemble de vecteurs de variables réelles, à l'aide d'opérateurs de mutation et de sélection [81]:

- L'étape de mutation est classiquement effectuée par l'ajout d'une valeur aléatoire, tirée au sein d'une distribution normale.
- La sélection s'effectue par un choix déterministe des meilleurs individus, selon l'échelle de valeur de la fonction objectif.

B.4 Évolution différentielle

C'est une méthode proposée par Storn et Price en 1995, basé sur le principe de l'évolution naturelle. Elle est utilisée pour l'optimisation mathématique des fonctions multidimensionnelles lorsque les paramètres ne peuvent pas être codés comme des vecteurs réels. Elle se base sur l'utilisation d'un opérateur de recombinaison ternaire pour la création de nouvelles générations. La création d'un nouvel individu consiste à ajouter la différence entre deux individus à un troisième. Les principales différences entre l'algorithme génétique et l'évolution différentielle sont :

- L'opérateur de Mutation est le résultat d'une petite perturbation dans les gènes du chromosome (individus), par contre dans la méthode Evolution Différentielle la Mutation est le résultat d'une combinaison arithmétique des individus.
- Au démarrage du processus d'évolution, l'opérateur de Mutation favorise l'exploration, et dans la progression d'évolution l'opérateur de Mutation favorise l'exploitation.

C. Intelligence en essaim

Elle a été introduite par Gerardo Beni et Jing Wang en 1989, dans le cadre de systèmes robotiques cellulaires, appelée Swarm Intelligence en Anglais, présente le comportement collectif décentralisé, auto-organisé des systèmes naturels ou artificiels. La source d'inspiration de cette approche vient de l'observation du comportement des insectes sociaux: une population d'agents extrêmement simples, interagissant et communiquant indirectement à travers leur environnement, constitue un algorithme

massivement parallèle pour résoudre une tâche donnée (par exemple le fourragement l'attroupement ou flocking, la division de tâches, la capture de proie ...) [39, 85].

On trouve dans cette catégorie:

- L'optimisation par essaim de particules;
- Les Colonies de Fourmis Artificielles;
- Optimisation par la colonie d'abeilles;
- La méthode Recherche coucou.

C.1 Optimisation par essaim de particules

Elle repose sur un ensemble d'individus originalement disposé de façon aléatoire et homogène, que nous appellerons dés lors des particules, qui se déplacent dans l'hyper-espace de recherche et constituent, chacune, une solution potentielle. Chaque particule dispose d'une mémoire concernant sa meilleure solution visitée ainsi que la capacité de communiquer avec les particules constituant son entourage. A partir de ces informations, la particule va suivre une tendance faite, d'une part, de sa volonté à retourner vers sa solution optimale, et d'autre part, de son mimétisme par rapport aux solutions trouvées dans son voisinage. A partir d'optimums locaux et empiriques, l'ensemble des particules va, normalement, converger vers la solution optimale globale du problème traité.

C.2 Colonies de Fourmis Artificielles

Le premier algorithme conçu selon ce modèle était destiné à résoudre le problème du voyageur de commerce. Le principe consiste à « lancer » des fourmis, et à les laisser élaborer pas à pas la solution, en allant d'une ville à l'autre. Au début, plusieurs chemins de longueurs différentes sont possibles, et les fourmis les empruntent tous en s'orientant au hasard. Ce faisant, elles laissent derrière elles des traces de phéromones. Sachant que ces traces s'évaporent petit à petit, et que le chemin le plus court permet par définition un plus grand nombre d'aller-retour. Les traces de phéromone se concentrent assez rapidement sur l'itinéraire optimal. C'est donc un algorithme qui repose sur la construction progressive de solutions [83, 81].

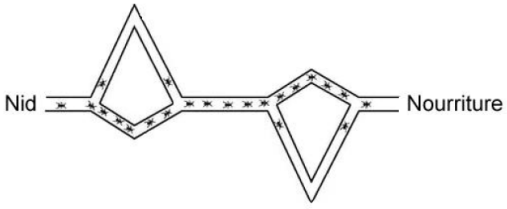


Figure 3.16: Recueil de ressources par des fourmis [81].

C.3 Colonies d'abeilles Artificielles

L'algorithme de colonie d'abeille artificielle (Artificial Bee Colony ABC en Anglais) est l'un des plus récemment introduit dans la base des Algorithmes basés essaim. Chaque solution représente une position de nourriture potentielle dans l'espace de recherche et la qualité de la solution correspond à la qualité de la position alimentaire. Agents (abeilles artificielles) recherche à exploiter les sources de nourriture dans l'espace de recherche. L'ABC utilise trois types d'agents [39, 81]:

1. les abeilles employés (Employed Bee en Anglais):sont associés à des solutions actuelles de l'algorithme,
2. les abeilles spectateur (Onlookers Bee en Anglais):choisissent parmi les postes promus en fonction de leur qualité, et
3. les scouts (Scouts Bee en Anglais):.

C.4 Recherche Coucou

L'algorithme est créé par Yang et Deb autant qu'un algorithme d'optimisation inspiré du comportement des oiseaux coucous en pondant leurs œufs dans les nids des autres oiseaux. La méthode de recherche coucou (Cuckoo Search CS en Anglais) s'appuie sur la stratégie agressive de reproduction Coucou complétée par un comportement Lévy flights dans les habitudes alimentaires des autres espèces. Ce faisant, il vise à « l'élevage » des solutions de haute qualité pour le problème d'optimisation proche. Par conséquent, il utilise deux mécanismes fondamentaux: l'intensification, en se référant à l'exploitation dans le voisinage de la meilleure solution trouvée à ce jour, et de la diversification, en se référant à une exploration efficace de l'ensemble du domaine de recherche. Dans la recherche de coucou, un " oeuf " se réfère à une solution du problème d'optimisation à portée de main. Un « oeuf coucou » se réfère à une solution vient d'être généré, et un «nid» signifie un ensemble de solutions possibles. Le nombre de nids d'oiseaux hôtes est fixé à N. En général, un nid peut contenir plusieurs ovules. Cependant, la mise en oeuvre des auteurs utilise un oeuf par nid seulement. L'idée derrière CS est de générer de nouvelles (peut-être) trouvé de meilleures solutions pour remplacer des solutions moins bonnes dans le nid. CS suit trois règles de base:

1. Chaque coucou pond un oeuf à la fois et le place dans un nid choisi au hasard. Cette intensification se fait en avançant une solution au moyen d'une marche aléatoire locale dans laquelle la longueur de pas est tirée d'une distribution (queue lourde) Lévy. Cela s'avère être plus efficace que les longueurs des pas ordinaires gaussiennes.
2. Les oeufs de haute qualité sont reportés à la prochaine itération (la survie du plus fort).

3. Un oiseau hôte découvre un oeuf de coucou avec une probabilité p. Il jette l'oeuf soit loin du coucou ou abandonne ses nids d'en construire une nouvelle ailleurs. Cette diversification se fait à faire en sorte que l'algorithme n'est pas piégé dans un minimum local.

3.1.4 Combinaison des classificateurs

L'idée consiste à améliorer la qualité globale de la classification en combinant les résultats des sorties des différents classificateurs. Dans cette approche, un grand nombre de classificateurs générés par des méthodes spécifiques tels que : Le Bootstrapping, le boosting, le bagging et le stacking. D'autres méthodes appliquent différentes conditions d'apprentissage telles que le choix aléatoire des poids initiaux pour l'entraînement d'un réseau de neurones ou le choix des dimensions des arbres de décision d'autres méthodes génèrent des classificateurs basées sur la séparation de l'espace des attributs caractéristiques en différentes classes. Les méthodes les plus simples de combinaison consiste à utiliser des règles fixes tel que le vote majoritaire (voir la section 3.3.1). Cette combinaison est faite pour plusieurs raisons [39, 83] :

- Le concepteur du système peut avoir accès à plusieurs classificateurs, développés dans des contextes différents pour des représentations/descriptions totalement différentes d'un même problème.
- On peut avoir plusieurs ensembles d'entraînement collectés/extraits à différents moments dans différents environnements, et peuvent être représentés par des paramètres différents.
- Des classificateurs différents, entraînés sur les mêmes données, peuvent présenter de grandes différences dans la qualité de la classification.
- Certains classificateurs comme les réseaux de neurones montrent des comportements différents selon la phase d'initialisation, ceci est dû à la part aléatoire du processus d'initialisation. Au lieu de ne garder qu'un seul classificateur, on peut combiner les réseaux de neurones obtenus pour bénéficier des résultats de tous les entraînements.

Principalement, nous pouvons distingué trois architectures [39]:

1. **L'architecture parallèle :** Les classificateurs sont invoqués de façon indépendante, et leurs résultats sont combinés par une fonction de combinaison. Avant cette combinaison, les sorties des classificateurs peuvent être pondérées.
2. **L'architecture en cascade :** dans ce cas, les classificateurs sont invoqués de manière linéaire afin de réduire graduellement le nombre de classes possibles pour une forme donnée. Pour des raisons d'efficacité, les classificateurs imprécis mais peu gourmands en calcul sont invoqués en premier, suivis par des classificateurs plus précis mais aussi plus lents.

3. **L'architecture hiérarchique :** Les classificateurs individuels sont combinés en formant une structure arborescente similaire aux arbres de décision. L'avantage de cette architecture est la grande efficacité et la flexibilité dans l'exploitation du pouvoir discriminant des attributs.

Il est possible de concevoir des systèmes de classification plus complexes en combinant les trois architectures basiques indiquées ci-dessus.

3.2 Bilan et technique de classification adoptée

Dans cette section nous allons voir une comparaison des différents classificateurs. Cette comparaison est réalisée à l'aide de scikit-learn version 0.18.1 sur un ensemble de données synthétiques afin de montrer la nature des frontières de décision des différents classificateurs. La figure 3.17 présente les points d'apprentissage en couleurs intenses, et de test avec des couleurs semi-transparents. Au-dessous droite se trouve le taux de classification sur l'ensemble de test [51, 91].

Une autre comparaison est présentée par [45]. Cette comparaison utilise les classificateurs : arbre de décision (DT : Décision Tree), k-NN, régression logistique (LogR : Logistic Regression), Naïve Bayes (NB), C4.5, SVM et classificateur linière (LC : Linear Classifier). Ces classificateurs sont appliqués sur des ensemble de données (DS : Data Set) DS_1 jusqu'à DS_29, et des différents nombre d'exemples 200, 500, 1000, 3000 et 5000. Le tableau 3.4 illustre les résultats obtenus par chaque classificateur.

Selon les auteurs de [51, 45] les deux classificateurs SVM et KNN montrent une bonne performance par rapport aux autres classificateurs. Dans ce travail nous avons choisit le classificateur SVM parce qu'il nécessite moins d'exemples (support vecteur) durant la phase de classification que le classificateur KNN.

3.3 Sélection de mots

Dans cette section, nous allons voir premièrement les différentes méthodes utiliser dans la littérature pour la sélection du mot dans la phase de post-traitement. Par la suite, nous allons présenter le principe de la méthode de sélection à base du puzzle proposée.

3.3.1 Méthodes de sélection de mots

Le processus de sélection consiste à appliquer des algorithmes ou des méthodes de recherche afin de choisir le meilleur candidat parmi un ensemble ou une liste de choix donnée. Plusieurs méthodes ont été développées et utilisées pour réaliser la tâche de sélection, parmi lesquelles on trouve :

A. Sélection à base de fonctions de masse.

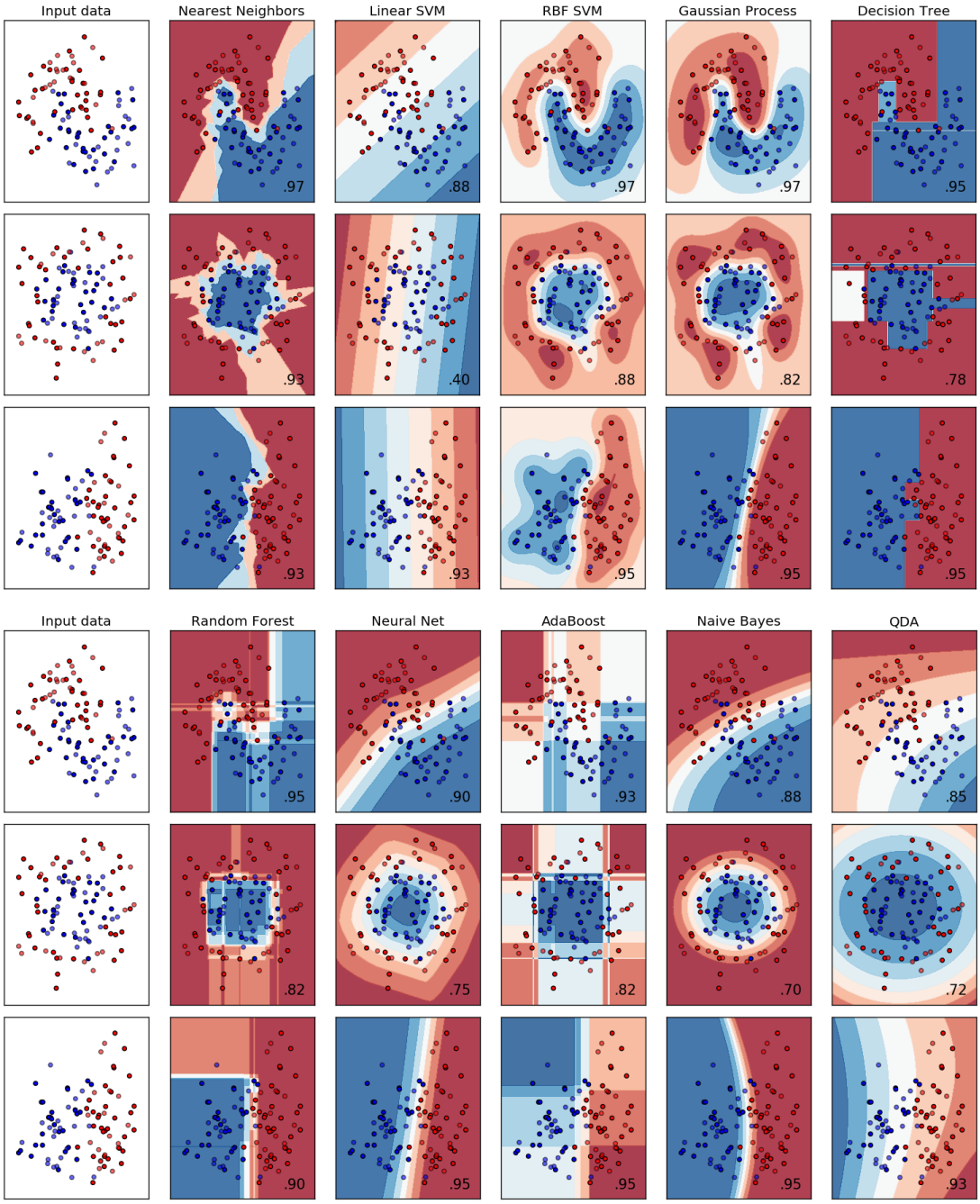


Figure 3.17: Comparaison entre différents classificateurs [51].

B. Sélection à base de DTW.

C. Sélection à base de distance de Levenshtein.

D. Sélection à base de ranking.

E. Sélection à base de RMP-PSO.

F. Sélection à base de vote.

G. Sélection à base de règles logiques.

Tableau 3.4: Résultats de classification des différents classificateurs pour les ensemble de données de DS_{19} jusqu'à DS_{29} [45].

ID	Sample size	DT	k-NN	LogR	NB	C4.5	SVM	LC
D_{19}	200	0.7941	0.7683	0.6328	0.7126	0.7452	0.7448	0.6408
	500	0.8914	0.8347	0.6117	0.6650	0.8743	0.8589	0.6075
	1000	0.9141	0.8451	0.5702	0.6607	0.9176	0.9078	0.5690
	3000	0.9865	0.9220	0.5800	0.6520	0.9917	0.9841	0.5780
	5000	0.9942	0.9453	0.5844	0.6499	0.9956	0.9927	0.5830
D_{20}	200	0.6343	0.5848	0.5828	0.6264	0.5872	0.5418	0.5727
	500	0.7984	0.7565	0.6318	0.6359	0.8034	0.7442	0.6236
	1000	0.9139	0.8208	0.5989	0.6371	0.9099	0.8609	0.5958
	3000	0.9892	0.9297	0.6241	0.6574	0.9917	0.9720	0.6211
	5000	0.9972	0.9579	0.6268	0.6517	0.9984	0.9860	0.6214
D_{21}	200	0.7206	0.7074	0.4667	0.5737	0.6547	0.7728	0.4743
	500	0.8836	0.8273	0.5468	0.5947	0.9114	0.9569	0.5095
	1000	0.9684	0.8859	0.5646	0.6074	0.9731	0.9810	0.5496
	3000	0.9978	0.9668	0.5745	0.5955	0.9984	0.9973	0.5492
	5000	0.9965	0.9924	0.5642	0.5987	0.9963	0.9951	0.5467
D_{22}	200	0.6307	0.6473	0.4789	0.4473	0.6334	0.6997	0.4731
	500	0.8869	0.8286	0.5477	0.5600	0.9079	0.9109	0.5478
	1000	0.9545	0.9221	0.6012	0.6366	0.9574	0.9577	0.6016
	3000	0.9929	0.9785	0.5559	0.5906	0.9906	0.9897	0.5495
	5000	0.9823	0.9722	0.5393	0.5743	0.9802	0.9666	0.5176
D_{23}	200	0.8670	0.8829	0.6771	0.7060	0.8698	0.8044	0.6577
	500	0.9004	0.8938	0.6249	0.6747	0.8812	0.8143	0.5798
	1000	0.9389	0.9060	0.5612	0.6347	0.9337	0.8465	0.5272
	3000	0.9907	0.9709	0.6082	0.6353	0.9913	0.9386	0.5916
	5000	0.9977	0.9840	0.5931	0.6325	0.9981	0.9766	0.5598
D_{24}	200	0.6700	0.7458	0.4791	0.5507	0.7521	0.6616	0.4500
	500	0.8269	0.8759	0.6047	0.6170	0.8745	0.7433	0.5935
	1000	0.9366	0.9294	0.6207	0.6629	0.9393	0.8467	0.6151
	3000	0.9950	0.9729	0.6117	0.6518	0.9964	0.9020	0.5993
	5000	0.9975	0.9859	0.6038	0.6539	0.9980	0.9634	0.5942
D_{25}	200	0.6422	0.8172	0.5387	0.5318	0.7597	0.4948	0.5561
	500	0.8419	0.8468	0.4988	0.5636	0.8996	0.8714	0.5154
	1000	0.9840	0.9132	0.5546	0.6172	0.9826	0.9898	0.5438
	3000	0.9947	0.9838	0.5689	0.6008	0.9941	0.9940	0.5468
	5000	0.9944	0.9978	0.5627	0.6086	0.9952	0.9937	0.5482
D_{26}	200	0.8069	0.9063	0.6679	0.6127	0.8760	0.7238	0.6564
	500	0.8901	0.9387	0.6326	0.6434	0.9312	0.7754	0.6080
	1000	0.9599	0.9584	0.6082	0.6083	0.9755	0.8081	0.5935
	3000	0.9946	0.9864	0.5608	0.6038	0.9956	0.9018	0.5321
	5000	0.9977	0.9945	0.5684	0.6187	0.9979	0.9675	0.5512
D_{27}	200	0.6746	0.8919	0.6081	0.5949	0.8567	0.5961	0.5945
	500	0.8766	0.9436	0.4900	0.5789	0.9511	0.6865	0.4917
	1000	0.9587	0.9726	0.5094	0.5822	0.9811	0.7169	0.5142
	3000	0.9981	0.9960	0.5722	0.6094	0.9981	0.9120	0.5733
	5000	0.9992	0.9986	0.5303	0.6017	0.9992	0.9634	0.5331
D_{28}	200	0.8622	0.9147	0.7635	0.6148	0.9190	0.6382	0.7384
	500	0.9185	0.9651	0.4849	0.5163	0.9639	0.6309	0.4896
	1000	0.9712	0.9782	0.5642	0.5773	0.9788	0.7655	0.5701
	3000	0.9982	0.9969	0.5339	0.5551	0.9990	0.8861	0.5260
	5000	0.9986	0.9983	0.5430	0.5620	0.9984	0.9347	0.5381
D_{29}	200	0.9669	0.9508	0.6425	0.5886	0.9661	0.6524	0.6278
	500	0.9949	0.9828	0.5558	0.5492	0.9949	0.7645	0.5570
	1000	0.9975	0.9945	0.5278	0.5141	0.9975	0.7676	0.4999
	3000	0.9985	0.9987	0.5640	0.5364	0.9983	0.8647	0.5646
	5000	0.9983	0.9993	0.5417	0.5253	0.9983	0.8796	0.5314

A. Sélection à base de fonctions de masse

Les auteurs de [65] propose un système à base de la théorie de Dempster-Shafer (DST: Dempster-Shafer Theory) pour améliorer la précision et la fiabilité d'un système de

reconnaissance du manuscrit. Le système travaille en deux étapes: premièrement, il utilise une méthode de combinaison évidentielle pour combiner les sorties probabilistes des classificateurs HMM. Ensuite, il utilise deux stratégies de décisions acceptation / rejet multiple, comme un post-traitement global pour améliorer la fiabilité du système. Finalement, le taux de reconnaissance est amélioré à base des HMM multi-stream comme un alternatif du traitement des exemples rejetés. Thomas et al. [29] proposent une nouvelle approche de rejet qui optimise la fiabilité d'un système de reconnaissance de mot manuscrit. Le système utilise la théorie de Dempster-Shafer pour combiner les sorties des différents classificateurs HMM. Par la suite, caractérise la qualité/fiabilité de la classification en se basant sur l'expressivité des fonctions de masse. Finalement, à base de cette caractérisation le système décide d'accepte de rejeter un mot de test.

B. Sélection à base de DTW

La déformation temporelle dynamique (DTW: Dynamic Time Warping) a présenté une autre alternative pour quelques développeurs. Dans [99], les auteurs proposent un algorithme pour une recherche rapide des PAWs au sein de grands lexiques. Trois étapes sont nécessaires pour achever les résultats de la recherche. D'abord, il réalise un plongement des apparences de chaque PAW dans le lexique au même espace euclidien en se basant sur un processus DTW durant le calcul de la distance de similarité. Ensuite, les PAWs sont normalisés à la même taille, et stockés en utilisant un arbre k-dimensionnel (k-d tree). Une liste courte des candidats est préparée pour la troisième étape en utilisant les approximations rapides de KPPv. Finalement, les candidats sont examinés par une DTW active pour déterminer les résultats finaux. Un système combiné est développé par Rodriguez-Serrano et al. [97] qui proposent une méthode de mesure de similarité entre les séquences de vecteurs modélisées en utilisant les HMMs semi-continu. Dans ce type des HMMs, les probabilités dans chaque état sont des mélanges de gaussiennes partagés, qui offrent deux avantages. Le premier, l'ensemble commun de gaussiennes conduit à des estimations plus précises des paramètres HMM avec des informations à priori. Deuxièmement, le coût de calcul entre les poids des mélanges de deux HMMs semi-contenus peut être réduit en utilisant une DTW.

C. Sélection à base de distance de Levenshtein

La distance d'édition ou distance de Levenshtein a été considérée comme un bon choix pour Sari et al. Dans [105] qui proposent un moteur de recherche pour les images des documents dégradés Arabe manuscrit et imprimés sans la reconnaissance des patterns textuels. Premièrement, une approche ascendante est appliquée pour extraire les composants connexes (CCs). Ensuite, chaque CC est représenté par des caractéristiques structurelles globales comme cercles, ascendants et descendants qui donnent une

représentation de la forme du mot (WSTs: Word Shape Tokens). Les caractéristiques extraites sont sauvegardées dans un fichier ASCII. Par la suite, ils calculent une distance de Lenvenshtein pour comparer entre les chaines de caractères. De plus, [63] proposent une méthode d'extraction des caractéristiques structurelles pour la reconnaissance des noms personnels Arabe manuscrit dans le but de maximiser le taux de reconnaissance avec le moins d'éléments. La méthode incorpore des informations structurelles telles que: cercles, stems, jambe et diacritiques. L'extraction des caractéristiques est évaluée en calculant une distance d'édition pour mesurer la distance entre deux séquences.

D. Sélection à base de ranking

Le ranking a été utilisé par Srihari et al. [113] dans un système de spotting de mot pour les documents Arabe manuscrit. L'approche travail en deux étapes : premièrement, basé sur une requête tapée, le système lance une sélection de prototype pour extraire un ensemble des exemples manuscrits. Ensuite, commence un processus de comparaison de mots pour trouver chaque occurrence de ces mots dans la base de données. Durant le processus de ranking, un ensemble de caractéristiques de forme global sont utilisées entre les prototypes de mots et les mots candidats. De plus, AlKhateeb et al. [12] proposent un système global hybride pour la reconnaissance de mot Arabe manuscrit. Les caractéristiques sont extraites en utilisant des fenêtres glissantes de gauche à droite appliquée sur images miroirs (WMI: Mirrored Windows Image). En plus, des différentes caractéristiques de structure sont extraites comme le nombre de régions au-dessus et au-dessous la ligne de base. Un modèle de classification combiné utilisé intégrant les HMMs et le re-ranking. Des caractéristiques d'intensité sont utilisées durant l'apprentissage des HMMs, et des caractéristiques topologiques sont utilisées pour améliorer la fiabilité.

E. Sélection à base de RMP-PSO

Salami et al. [102] proposent un système de reconnaissance des digits manuscrit Arabe/Farsi. il utilise un ensemble des classificateurs basés sur un algorithme de décomposition en valeurs singulières (SVD: singular value decomposition) parce qu'il est convenable pour la résolution des matrices creuses comme les digits manuscrit. Les décisions de SVD sont combinées ensuite en utilisant des règles de composition nommées optimisation par essaim de particules multiphase fiable (RMP-PSO: reliable multi-phase particle swarm optimization). De plus, des paramètres fiables ont été introduit pour résoudre le problème de l'optimisation par essaim de particules d'être piégée dans un minima local.

F. Sélection à base de vote

Azizi et al. [20] proposent une étude de la reconnaissance du manuscrit Arabe en utilisant une optimisation de système multi-classificateurs (MCS: multi-classifier system) basée sur des mesures de diversité. Sur la base des erreurs de corrélation, trois mesures de diversité ont été utilisées pour calculer la performance du système. Ce dernier est testé avec plusieurs critère d'optimisation et méthodes de fusion (vote, vote pondéré,...etc.).

G. Sélection à base de règles logiques

El Abed et al. [41] présentent une comparaison entre deux schémas de combinaison pour l'amélioration de la performance des systèmes de reconnaissance du manuscrit Arabe. Les sorties des différents systèmes de ICDAR 2007 sont considérés comme entrées pour leurs schémas de combinaison. Le premier schéma de combinaison est basé sur un ensemble de fusions fixes utilisant des règles logiques. Le second, utilise un ensemble de règles par apprentissage. L'amélioration du système multi-classificateurs est évaluée en termes de taux de reconnaissance avec ou sans rejet. Farah et al. [49] proposent un système multi-classificateurs qui intègre des informations de niveau syntaxique (SLI: syntactic level information) pour la reconnaissance d'un ensemble des littérales manuscrites arabes. La classification produit un ensemble des mots candidats possibles au lieu d'un seul. L'analyseur syntaxique effectue la décision finale par la vérification et la correction de la solution et la sélection du bon mot en stilisant des informations de haut niveau non disponible durant la phase de classification. Sari et al. [106] proposent une méthode contextuelle pour la correction des mots arabes générés par un système OCR. Le post-processeur correcte les erreurs de substitution et rejet en se basant sur des propriétés de langue Arabe et les erreurs plus fréquentes du système OCR. Le système utilise un ensemble de règles de production pour corriger les mots classés.

3.3.2 Technique proposée

Cette section, présente premièrement une vue générale sur l'algorithme du puzzle. Ensuite, expose le principe de la deuxième contribution qui se résume à l'intégration de l'algorithme puzzle dans le processus de reconnaissance. Cette intégration, consiste à utiliser l'algorithme comme un post-processeur pour remédier aux problèmes de coupures et chevauchement des caractères dans un texte manuscrit, et éventuellement, choisir le bon mot.

A. Puzzle

Les puzzles ont été toujours un sujet important et un problème très connu notamment dans le domaine de l'intelligence artificielle. Parmi eux, les puzzles de (N^2-1) présentent une collection de N^2-1 carreaux mobiles et numérotés de 1 jusqu'à N^2-1 avec un carreau blanc dans une grille de $N \times N$ (où $N > 2$). Le problème de 8-Puzzle est un

exemple très basique joué en une grille de 3×3. Le but est de réarranger les carreaux pour qu'ils soient en ordre en utilisant le minimum des mouvements possibles. Il est permis de glisser les carreaux horizontalement et verticalement vers le carreau blanc. Alors, on commence d'un état initial (un arrangement aléatoire), pour ensuite atteindre un état final (voir la figure 3.18) [59].

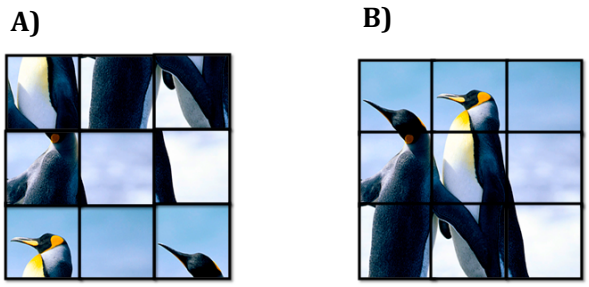


Figure 3.18: Example de 8-Puzzle, a) état initial, b) état final.

Naturellement, on commence à résoudre un puzzle comme suit :

1. Obtenir l'état initial, qui est l'état de début ou source.
2. Générer les états possibles.
3. Ajouter ces nouveaux états à la liste ouverte.
4. Ajouter l'état traité à la liste fermée.
5. Choisir l'état suivant qui a le plus petit coût dans la liste ouverte.
6. Répéter les étapes de 1 à 6 sauf si on atteint l'état final ou il n'y'a pas d'états à traiter (dans ce cas, pas de solutions)
7. Quand on obtient l'état final, on suit le chemin inverse (backtrack) au nœud de début et ça sera le chemin optimal à la solution.

Le succès de l'algorithme de recherche dépend du choix de la fonction de priorité pour un nœud de recherche. Les fonctions de priorité sont utilisées pour mesurer la distance entre l'état du puzzle et la solution:

1. Distance discrète (0 Si égale et 1 Sinon);
2. Distance de Hamming (le nombre de carreaux hors de place);
3. Distance de Manhattan (la somme des nombres d'étapes minimales pour déplacer chaque carreau à ça position correcte);
4. Distance de Manhattan + Conflit Linéaire;

5. Heuristique de base de données des Patterns Additives.

Dans ce type de puzzles la reconstruction de l'état final est basée sur le glissement des carreaux dans les quatre directions droite, gauche, haut et bas. Les Jigsaw puzzles présentent un autre type basé sur la recherche de pièce adéquate. Ils sont considérés comme une image divisée en pièces. La forme des pièces et la couleur du fond ne sont pas prédéfinies (voir la figure 3.19).

Figure 3.19: Example de Jigsaw puzzle.

Donc, deux types de puzzles existent : les puzzles picturaux où la forme et la couleur de chaque pièce sont différentes des autres, et les puzzles extra-picturaux avec des pièces de la même forme [87, 27]. L'assemblage automatique du problème en général peut être divisé en quatre sous-problèmes [87, 116]:

1. Pré-traitement: pour obtenir l'alignement approprié après avoir séparé et pivoter les pièces individuelles en scannant le puzzle original
2. Matching des bordures: réalisé par un matching des bordures de deux pièces.
3. Résolution de l'itérieur: heuristique utilisée pour réduire la complexité du problème globale.
4. Présentation de la solution: présente les pièces assemblées. Elle inclut la minimisation des chevauchements et trous par la correction de l'orientation des pièces.

La performance d'un système de résolution d'un puzzle n'est pas définit seulement par une mesure de similarité, mais aussi par la capacité de recherche globale de l'algorithme utilisé [62, 64]. Des algorithmes divers tels que : A*, algorithmes génétique, optimisation par colonie de fourmis et l'optimisation par essaim de particules ont été utilisés pour implémenter les puzzles [59, 116, 64], dans le but de réduire le coût de calcul.

En fait, un texte non reconnu peut être vu comme un puzzle, où les PAWs ou même les segments de PAWs sont considérés comme des pièces du puzzle. Au-delà, le système

de résolution du puzzle est considéré comme la partie principale (de la segmentation au post-traitement) du processus de reconnaissance. Ci-dessus, nous avons vu des types différent de puzzles et avec des règles de reconstruction différentes, mais en réalité la reconstruction est basée sur deux opérations principales : la fusion pour fusionner deux pièces, et la division pour diviser deux pièces liées.

Dans les 8-Puzzles le faite de glisser un carreau à droite, gauche, haut et bas permet de fusionner le contenu de ce carreau avec les autres, et au même temps le diviser du contenu d'autres carreau. De la même façon, dans les jigsaw puzzles le faite de mettre une pièce dans une position choisit veut dire que son contenu va être fusionné avec le contenu des pièces adjacentes et implicitement le diviser du contenu d'autres.

B. Sélection de mot basée puzzle

Dans ce travail, nous allons utiliser l'idée du puzzle en implémentant un post-processeur puzzle dans le système de reconnaissance du manuscrit arabe. Ce contrôleur de puzzle va augmenter les résultats de reconnaissance. Essentiellement, il permet de franchir le problème de chevauchements et des coupures dans les segments des caractères, ce qui cause le problème des frontières non claires dedans et entre les caractères (connu par sur-segmentation et sous-segmentation). Ce post-processeur possède une possibilité faire des feedbacks et d'intégrer des techniques de correction dans les étapes de puzzling. Ces mécanismes sont pratiquement absents des méthodes de sélection présentées ci-dessus.

En plus, il contrôle les entrées et sorties des phases de classification (classifieur SVM) et post-classification et offre un moyen de feedback pour corriger quelques erreurs de classification et segmentation. Le manuscrit arabe est en réalité un puzzle partiellement résolut parce que la plupart des PAWs et des caractères sont dans leurs positions correctes. Le contrôleur du puzzle utilise deux phases comme suit:

1. Phase de segments : dans cette phase le système créé la liste des possibilités pour fusionner ou diviser les segments pour générer des caractères. Les caractères se développent dans les directions de haut et bas.
2. Phase de mots : cette phase construit les mots des caractères générés dans la phase précédente. Le mot est développé de droite à gauche

Le système considère les labels des formes et les vecteurs de définitions des segments comme l'ensemble qui représente l'état initial du puzzle. Ces données sont passées à un module de construction des mots pour tester la séquence des caractères. Si la séquence contient des labels caractères inconnus il doit la corriger en utilisant des règles de morphisme (voir la figure 3.20). Autrement, la séquence des caractères est considérée comme un mot reconnu (MR). Ensuite, le système cherche à trouver dans la base de

modèles le mot le plus proche (MP) en calculant une distance de Levenshtein distance (LD). Dans le cas où LD=0 le résultat va être MR (MR=MP). Sinon, il considère MP comme un état final et MR comme un état initial d'un puzzle, et essaye d'appliquer autant que nécessaire des étapes de puzzle (Fusion, Division et/ou Morphism) pour changer l'ensemble du puzzle (les labels des formes) dans le but d'atteindre l'état cible MP. Si le processus réussit, le résultat va être MR' (MP), contrairement, MR va être le résultat et le mot est considéré non trouver dans la base (voir la figure 4.21).

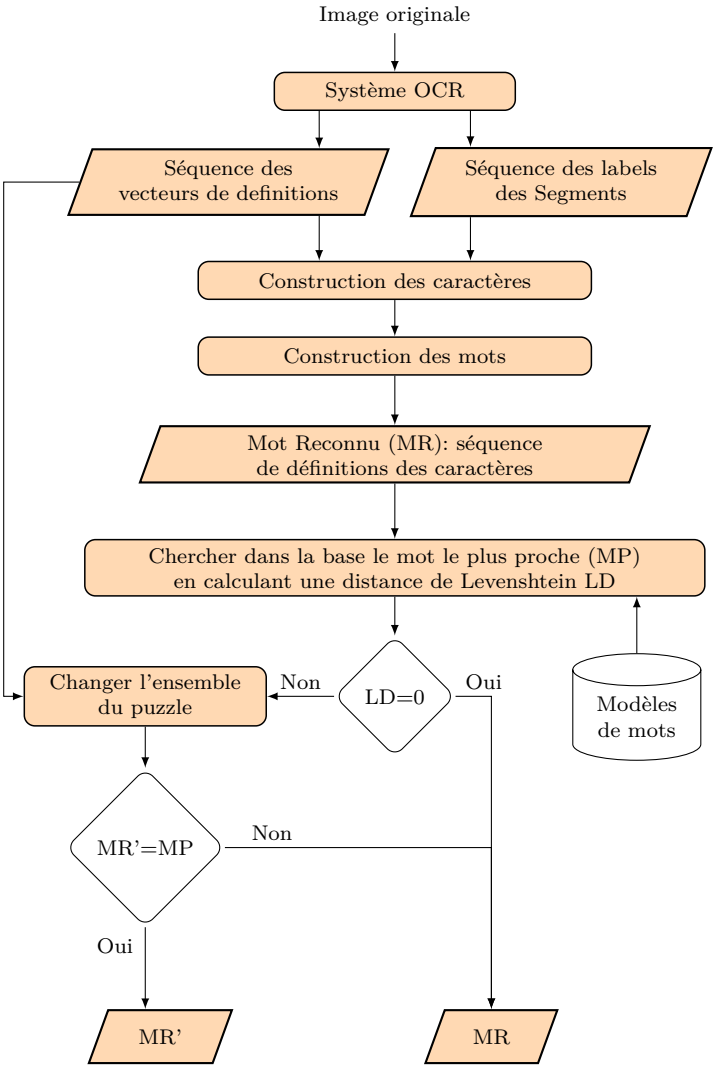


Figure 3.20: Système basé puzzle.

B.1 Morphisme

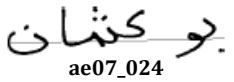
Le système utilise un ensemble de règles de morphisme qui permet dans certains cas de changer la nature d'une forme (caractère) si le dernier mot ne peut être trouvé dans la base des mots (voir la figure 4.24).

Par exemple, en examinant le premier mot dans la figure 4.25 (référencé par ae07-024) nous allons l'interpréter comme كثمان pour la première fois (le premier caractère à

R_1 : ـي <——> ـى
R_2 : ظ <——> ضـ
R_3 : ئـ <——> يـ
R_4 : ـي <——> ـى
R_5 : عـ <——> كـ
R_6 : ـد <——> ـر
R_7 : . <——> ..
R_8 : ◌ٔ <——> ر

Figure 3.21: Examples de règles de morphisme.

droite est ـكـ, mais, après la connaissance qu'il n'y'a pas de mot comme ce mot dans la base des mots (dictionnaire), avec la connaissance qu'un mot similaire عثمان avec le premier caractère à droite est ـعـ existe. Intuitivement, nous allons conclure que c'est le mot cherché. Similairement, quand le système détecte l'absence du premier mot dans la base des mots, il utilise la quatrième règle de morphisme pour ajuster la définition du mot à عثمان avec un premier caractère à droite ـعـ au lieu de ـكـ. De la même façon, il utilise la deuxième règle pour ajuster le second mot المضيلة (référencé par ae97-019 avec un quatrième caractère à droite ـضـ) pour qu'il soit le mot المظيلة avec un quatrième caractère à droite ظ au lieu de ـضـ.


ae07_024

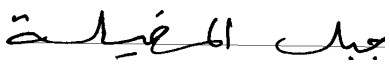

ae97_019

Figure 3.22: Exemples de mots nécessitant le morphisme.

B.2 Fusion

Cette opération permet de fusionner deux ou plus de segments autant qu'il est nécessaire pour arriver à une définition ou représentation plus correcte. Dans l'exemple présenté par la figure 3.23, le système peut considérer le second segment comme un symbole diacritique, alors le mot va être composé des segments 1 et 3. Le mot résultant va être w_1 = عسالر.

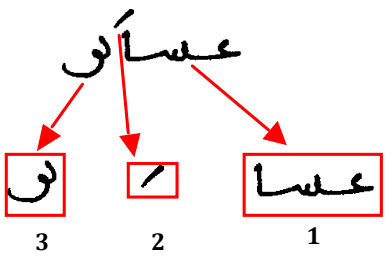


Figure 3.23: Exemple de mot segmenté en PAWs.

En examinant la liste des mots de la base IFN/ENIT, le mot w_1 = n'existe plus. Mais, un

autre mot très proche au dernier $w_2 =$عساكر existe avec une différence dans le quatrième caractère $c_4 =$ك. Dans ce cas, le système cherche à changer $c_4 =$ل de w_1 par $c_4 =$ك de w_2 en utilisant une opération de fusion, par conséquent, $w_1 = w_2$.

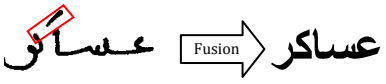


Figure 3.24: Exemple de fusion.

B.3 Division

Elle présente l'opération inverse de la fusion, parce que nous sommes intéressés à diviser un segment pour obtenir deux caractères. Deux cas sont possibles :

1. Division directe;
2. Division par Morphisme;

Dans le premier cas, le système détecte la nécessité de diviser deux segments et applique l'opération de division (voir la figure 3.25).

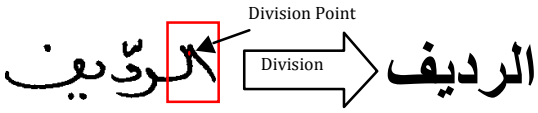


Figure 3.25: Exemple division directe.

Dans le second cas, le système détecte la nécessité de morphisme ensuite trouve qu'il est nécessaire de diviser d'autres segments pour atteindre un résultat correct (voir la figure 3.26).

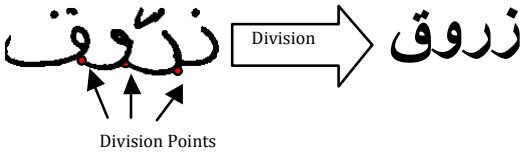


Figure 3.26: Exemple de division par morphisme.

En appliquant l'algorithme de segmentation nous allons trouver que le premier caractère $c_1 =$ن, le second $c_2 \approx$ـتـ, le quatrième $c_4 =$ق le troisième semble à une forme qui peut être ـقـ ou ـفـ ou dans le cas de morphisme و. Il est clair que la forme ne peut être que و parce qu'il n'y'a pas de symboles diacritiques au-dessus de cette forme. Maintenant, tant que $c_3 =$و, alors $c_4 =$ق et par morphisme $c_2 =$ر et $c_1 =$ز.

Conclusion

Dans ce chapitre, nous avons commencé par donner le principe de base de quelques méthodes de classification les plus utilisées dans les systèmes de reconnaissance de l'écriture manuscrite. Parmi ces techniques nous avons vu la technique des Machines à Vecteurs de Support utilisée par notre système dans la phase de classification. Ensuite, nous avons vu quelques approches de sélection de mot dans la phase de post traitements. Finalement, nous avons présenté la deuxième contribution qui vise à améliorer le taux de reconnaissance d'un système OCR hors-ligne. En effet, malgré les travaux intensifs dans ce domaine, les systèmes OCR souffrent encore du taux de reconnaissance limité à cause des problèmes de coupures, chevauchements et mal représentations des lignes de caractères. Notre contribution [121] vise à surmonter ces problèmes par l'utilisation d'un algorithme puzzle en post-traitement. Le but de notre approche est de reconstruire la structure des caractères en se basant sur le mécanisme de feedback offerte par le puzzle, et les étapes du puzzling : Fusion, Division et Morphisme. Chacune de ces étapes permet de résoudre l'un des problèmes posés par l'écriture cursive.

Chapitre 4

Mise en œuvre des contributions, résultats et bilan

Introduction

Dans le chapitre précédent, nous avons présenté les différentes méthodes de post-traitement, utilisées durant la dernière décennie ainsi que notre contribution basée sur l'algorithme du puzzle. Dans ce chapitre nous allons présenter les détails de notre système, ainsi que la validation des résultats. Nous allons commencer par montrer l'architecture globale du système proposé, où nous allons voir des exemples sur l'application des différentes étapes de la méthode de segmentation structurelle N-Scans, qui a pour but de préparer l'ensemble des segments initiale du puzzle. Ensuite, nous allons voir des exemples sur l'application des différentes étapes du post-processeur puzzle, qui permet de surmonter les problèmes de coupures, des chevauchements et des mal représentations dans les segments de lignes des formes des caractères manuscrits. Enfin, nous allons présenter un bilan des résultats de segmentation et de la classification du système dans les deux cas en activant et en désactivant le puzzle, où nous allons l'impact de ce dernier sur le processus de reconnaissance. Par la suite, nous allons faire une comparaison avec d'autres méthodes, ainsi, qu'une estimation de la complexité d'exécution du système.

4.1 Système proposé

Le système commence par l'acquisition de l'image contenant le texte source par l'intermédiaire d'une unité d'acquisition telle que : scanner, camera ou à partir d'une base de données. Ensuite, il passe par une phase de pré-traitement pour améliorer la qualité de l'image acquise, et extrait une image binaire pour la préparée à la phase de segmentation. La segmentation consiste à extraire les principaux segments, des quels, il va extraire un ensemble de caractéristiques structurelles pour obtenir une définition logique des différentes parties du texte. Ces caractéristiques sont utilisées pour créer les

modèles du langage dans une phase d'apprentissage, et de classer de nouveaux exemples en utilisant un classificateur SVM durant la phase de classification.

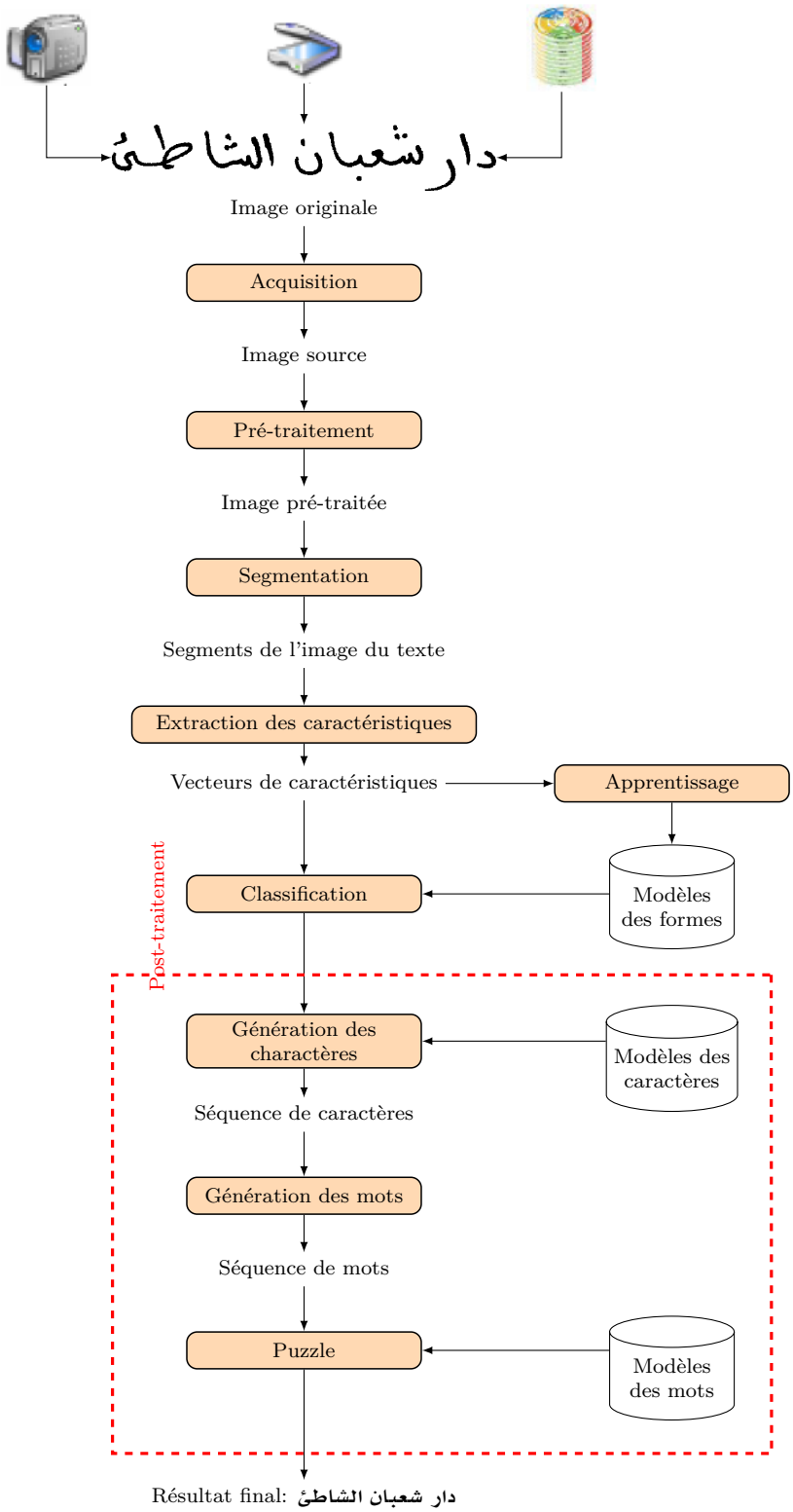


Figure 4.1: Schéma du système de reconnaissance.

Le puzzle considère les phases décrites ci-dessus comme une boite noire, qui prend en entrée une image source et donne en sortie un ensemble d'étiquettes de segments et leurs caractéristiques sous la forme de vecteur de définition pour chaque segment extrait durant la segmentation. Ces sorties sont utilisées par le puzzle pour trouver dans la base de données le mot le plus proche au mot reconnu en calculant une distance d'édition (distance de Levenstein). Quand cette distance est nulle, le système conclu qu'un mot similaire au mot reconnu est trouvé. Autrement, il essaye d'appliquer des étapes de puzzle (Fusion, Division et Morphisme) sur les vecteurs de définition pour ajuster (changer l'ensemble du puzzle) le mot reconnu pour qu'il soit comme le plus proche mot de la base. Si le processus échoue le mot reconnu est considéré comme non trouver dans la base et le résultat va être le mot reconnu.

4.1.1 Acquisition

Cette phase, consiste à charger l'image du papier et la sauvegarder en un format d'image connu (tel que : jpeg, gif, bmp...etc.) et /ou la transformer en un matrice de pixels dans le but de l'utiliser directement. Cette opération est réalisée à l'aide d'une unité d'acquisition telle que : scanner, camera,...etc. Généralement, nous utilisons des interfaces d'acquisition permettant la communication avec ces outils telle que :Twain (JTWain pour java), Morena Framework,...etc.

4.1.2 Pré-traitement

Dans cette phase le système applique des filtres à base des masques pour améliorer la qualité de l'image acquise en éliminant au maximum le bruit (voir la figure 4.2).

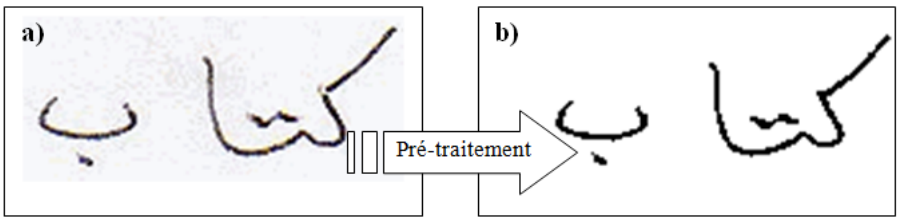


Figure 4.2: Exemple d'image prétraitée: **a)** image brute,**b)** image épurée.

Pratiquement, on passe par deux étapes principales, à savoir:

- Binairisation,
- Filtrage,

A. Binairisation

La binairisation consiste à transformer l'image en couleurs (souvent RGB) en une image binaire à deux niveaux (noir et blanc) en utilisant un seuil S pour décider qu'un pixel est noir ou blanc. Cette transformation utilise les formules de conversion RBG/YUV, comme suit :

$$Y = 299 \times R + 0.587 \times G + 0.114 \times B$$
$$U = 0.492 \times (B - Y)$$
$$V = 0.877 \times (R - Y)$$

Le passage inverse YUV/RGB est réalisé à l'aide des formules suivantes :
$$R = Y + 1.140 \times V$$
$$G = Y - 0.395 \times U - 0.581 \times V$$
$$B = Y + 2.032 \times U$$

La figure 4.3 illustre un exemple de binairisation.

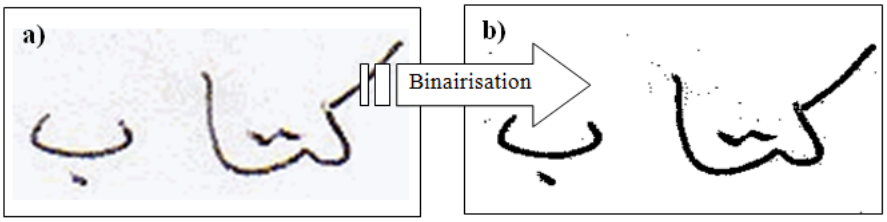


Figure 4.3: Illustration de la binairisation: **a)** image brute, **b)** image après binairisation.

*

B. Filtrage
Il consiste à utiliser des masques de pixels afin d'éliminer les taches de l'image binaire pour représenter en mieux les points d'intérêt. Le système élimine :

1. Chaque pixel noir entouré par des pixels blancs,
2. Chaque pixel blanc entre deux pixels noirs que ce soit horizontalement ou verticalement.

L'étape de filtrage est illustrée par la figure suivante :

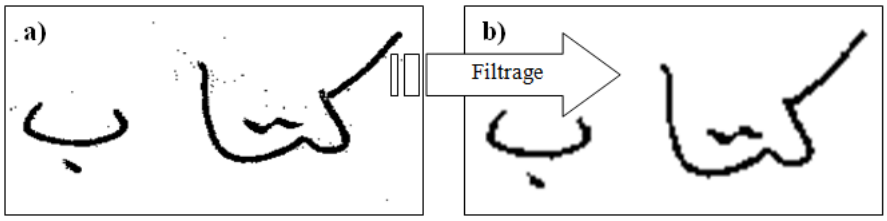


Figure 4.4: Illustration d'un exemple de filtrage: **a)** image après binairisation, **b)** image filtrée.

4.1.3 Segmentation N-Scans

Le rôle de cette phase est la préparation de l'ensemble des segments de base du puzzle (voir les formes du tableau 2.4). Cette préparation passe par cinq étapes :

1. Extraction des PAWs ;
2. Détermination des points de nature verticale (PNVs) et points de nature horizontale (PNHs) .
3. Détermination des connecteurs des caractères (CCs) .
4. Détermination des niveaux de complexité;
5. Scans des cercles cachés;

A. Extraction des PAWs

Cette phase consiste à lancer un balayage de droite à gauche sur l'image binaire afin de trouver le premier pixel noir. Ensuite, en utilisant la fonction 2.1 et les huit voisins du premier pixel, il est possible de reconstruire le reste du PAW. En appliquant cette opération sur les l'image d'entrée donnée au système dans la figure 4.1 nous allons voir comme résultat les différents segments ((a),(b),...,(q)) présentés par la figure 4.5.

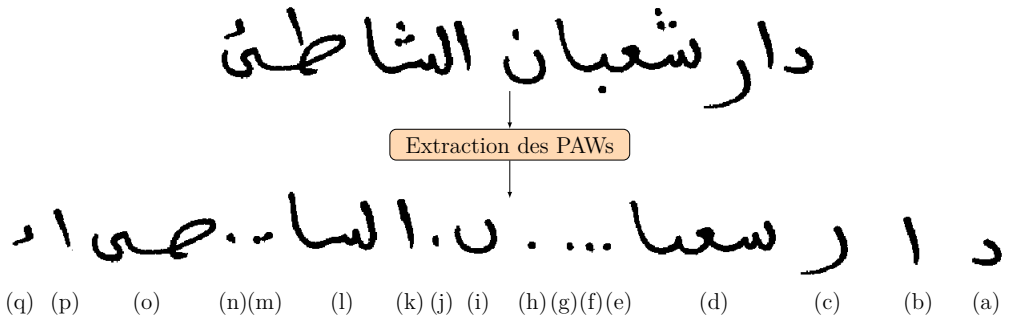


Figure 4.5: Segmentation de la phrase "دار شعبان الشاطئ" en PAWs.

A ce niveau nous remarquons qu'il y'a des segments qui ne sont pas complètement segmentés tels que les segments: (d), (i), (l) et (o). Pour cela, il est nécessaire de déterminer d'autres points de segmentation, et c'est le rôle de la phase suivante.

B. Détermination des PNVs et PNHs

Afin de segmenter le reste des segmentes le système lance en parallèle deux balayages : vertical et horizontal. Le premier utilise les masques (voir la figure 2.27) de balayage vertical afin de déterminer les points de fusion et de division verticaux. Le second utilise les masques (voir la figure 2.28) de balayage horizontal afin de déterminer les points de fusion et de division horizontaux. La figure 4.6 illustre des exemples d'application de cette opération sur les segments:(d), (i), (l) et (o).

Les deux scans vertical et horizontal génèrent une forme semi-squelettique similaire à celui qui est généré par des techniques de squelettisation. Cette structure squelettique est dessinée par les points de milieux de chaque masque. La figure 4.7 illustre un exemple. En plus, le système filtre ces points afin d'obtenir une structure plus lisse.

(d)

(i)

(l)

(o)

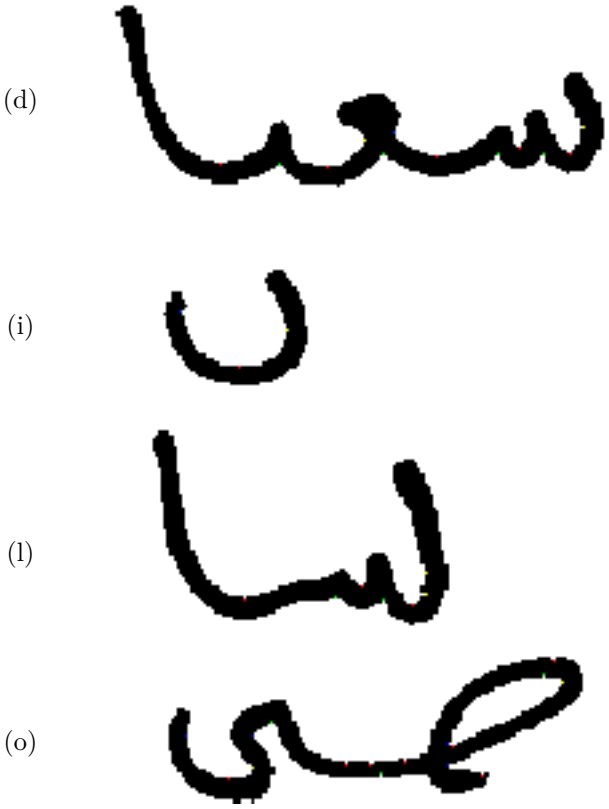
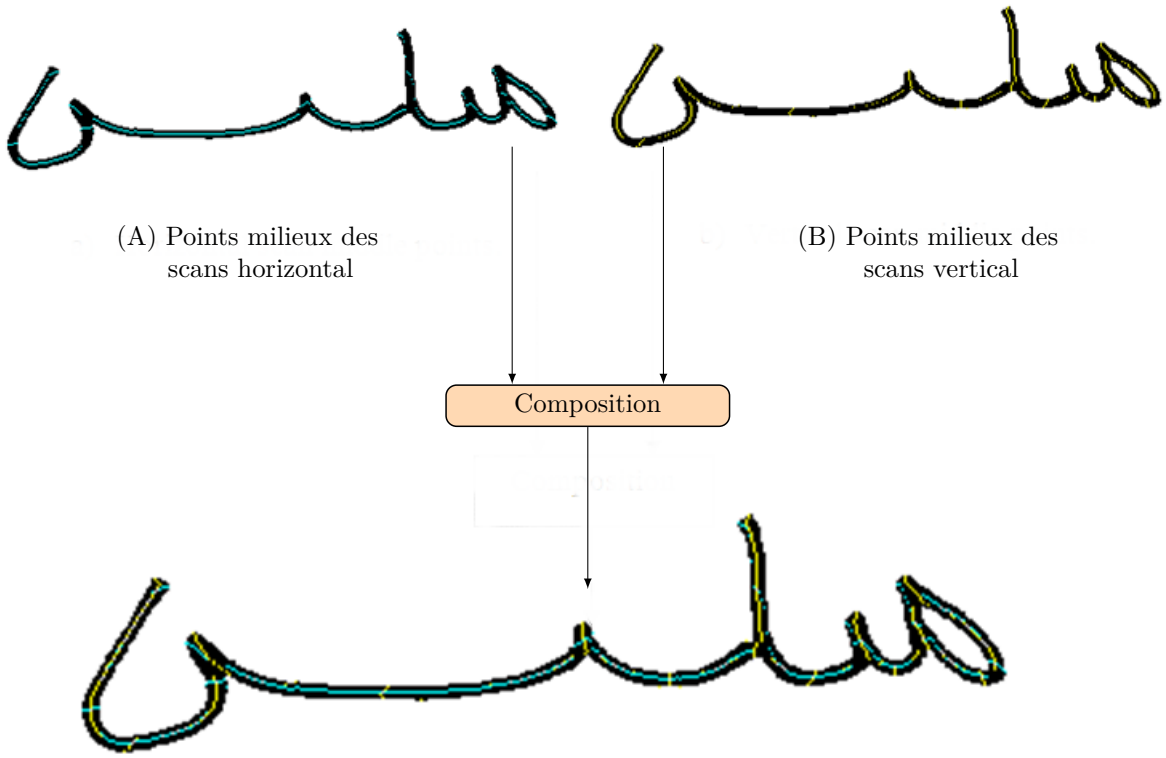


Figure 4.6: Points PNVs et PNHs des segments:(d), (i), (l) et (o).



(C) Résultat de composition

Figure 4.7: Exemple de forme semi-squelettique du mot " مثلين".

C. Détermination des Connecteurs de caractères

Après avoir effectué l'extraction des différents points de fusion et de division horizontale et verticale (VFPs, VDPs, HFPs and HDPs) avec des différents couleurs. Nous pouvons appliquer les règles décrites dans la section 2.6 pour obtenir les différents connecteurs des caractères des segments : (d), (i), (l) et (o) (voir la figure 4.8). Les différents connecteurs se situent dans des cercles rouge pointillées.

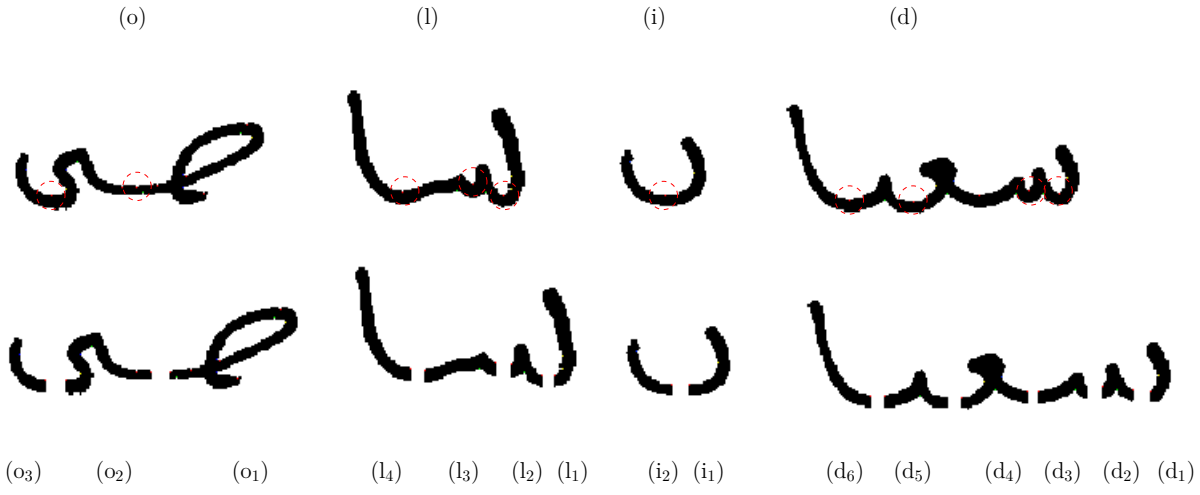


Figure 4.8: Extraction des connecteurs de caractères des segments:(d), (i), (l) et (o).

En connaissant ces points il est possible de générer les différents segments de base tel qu'il est présenté par la figure 4.8 au-dessous. Ces segments sont similaires aux segments présentés dans le tableau 2.4.

D. Détermination des niveaux de complexité

Comme nous avons décrit dans la section 2.6.4, à ce niveau le système associe un niveau de complexité et une étiquette de longueur à chaque segment générée pendant les phases précédente. Deux niveaux de complexité existent : segments simples (SS) et segments complexes (SC) qui contient souvent des cercles ou plusieurs sous-segments. En plus, deux longueurs existent : segment long L1 et segment court L2. Le tableau 4.1 illustre les différentes de complexité et de longueur pour les segments de la phrase **دار شعبان الشاطئ**.

Tableau 4.1: Etiquette de complexité des différents segments de la phrase دار شعبان الشاطئ.

Segment	Label de complexité	Label de longueur
(a)	SC	L2
(b)	SS	L1
(c)	SS	L1
(d_1)	SS	L2
(d_2)	SS	L2
(d_3)	SS	L2
(d_4)	SS	L2
(d_5)	SS	L2
(d_6)	SS	L1
(e)	SS	L2
(f)	SS	L2
(g)	SS	L2
(h)	SS	L2
(i_1)	SS	L1
(i_2)	SS	L1
(j)	SS	L2
(k)	SS	L1
(l_1)	SS	L1
(l_2)	SS	L2
(l_3)	SS	L2
(l_4)	SS	L1
(m)	SS	L2
(n)	SS	L2
(o_1)	SC	L2
(o_2)	SS	L1
(o_3)	SS	L1
(p)	SS	L1
(q)	SS	L1

E. Scan des cercles cachés

Nous avons vu dans la section 2.6.5 qu'il est nécessaire de scanner les segments de nature simple afin de déterminer et régénérer les cercles cachés. La figure 4.9 illustre comment vont être les cercles de régénérée des segments (d_4) et (q).

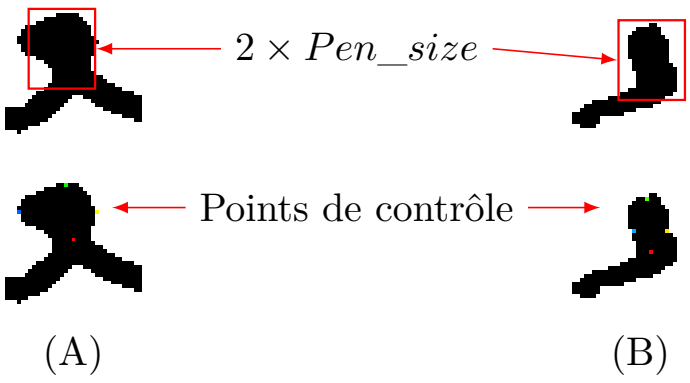


Figure 4.9: Régénération des cercles cachés des segments: (d_4) et (q).

Après la régénération, le segment (d_4) va contenir un cercle, ce qui simplifie par la suite de le classer comme Ain_M. Autant que pour le segment (q) nous voyons qu'il se transforme au symbole diacritique Dhamma comme le montre la figure 4.10, mais, tant que les segments (o_2 et o_3) au-dessous de (q) vont former la lettre ـــى avec le label Alif_Maqsoura_F, et tant que cette lettre ne nécessite pas des symboles diacritiques le

système transforme ce segment en Hamza au lieu de Dhamma. Il est à noter que la différence entre le symbole Dhamma et le symbole Hamza, est que le premier possède un cercle (voir la figure 4.10-(A)) alors que le second possède un demi-cercle (voir la figure 4.10-(B)).

(A) (B)

Figure 4.10: Régénération des cercles cachés des segments: (d_4) et (q).

4.1.4 Extraction des caractéristiques

Les caractéristiques des formes sont extraites en calculant un ensemble de valeurs qui donnent une description structurelle selon trois axes : taille, angle et forme. Durant cette phase, le système utilise les fichiers «$Image\_NumSéq.gif$» et «$Segment\_NumSeq.pos$» avec une structure nommée marqueur. Chaque marqueur contient huit éléments descriptifs, à savoir : la direction, la taille relative des segments entre eux, le nombre des boucles, la taille du segment d'image, la forme des boucles, les angles des segments, les secteurs des angles et le type des différents points constituent la boucle.

A. Direction

La direction est définit à l'aide de 03 descripteurs :

1. direction sur l'axe des X,
2. direction sur l'axe des Y,
3. direction générale, résulte de la composition des deux précédentes.

Les figures suivantes 4.11 et 4.12 montre bien cette idée :

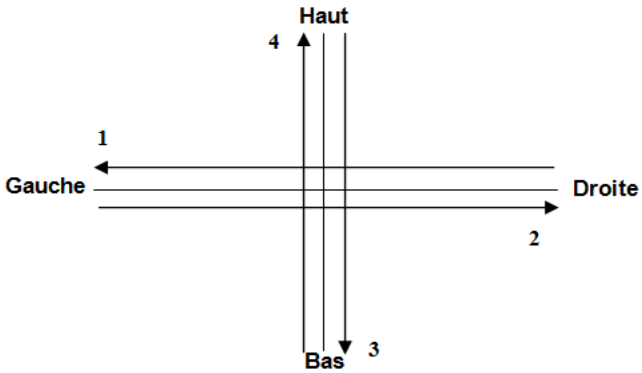


Figure 4.11: Illustration des 04 directions utilisée.

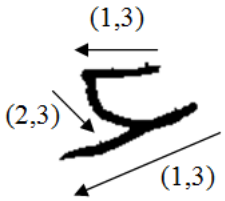


Figure 4.12: Illustration d'un exemple d'extraction de direction.

Exemple:

Direction en $X = 121$.
Direction en $Y = 333$.
Direction générale = 131.

B. Taille relatif des segments

Avant de discuter comment calculer la taille relative il faut calculer premièrement la taille de chaque segment séparément, et cela est réalisée à l'aide de la fonction suivante :

Algorithme 4.1: Fonction de calcul de longueur de segment

```
Fonction getDistance (p1, p2 : Point): Entier
    Distance, dx, dy : Entier;
Début
    dx ← p1.x − p2.x;
    dy ← p1.y − p2.y;
    Distance ← round(sqrt(dx ∗ dx) + (dy ∗ dy)));
    retourner Distance;
Fin
```

Maintenant après avoir trouvé la taille de chaque segment il est facile de calculer la taille relative entre segment par le calcul du rapport des tailles des segments.

C. Nombre de boucles

Le nombre de boucles est calculé en incrémentant un compteur de boucles à chaque fois qu'on fait une opération de fusion avec une égalité de point de départ et de fin entre deux segments à fusionner.

D. Forme de la boucle

La forme de la boucle est définie en utilisant 04 points de contrôle, le calcul de distance et angle entre ces points permet d'obtenir une description sur la forme de la boucle.
La figure suivante illustre ce principe.

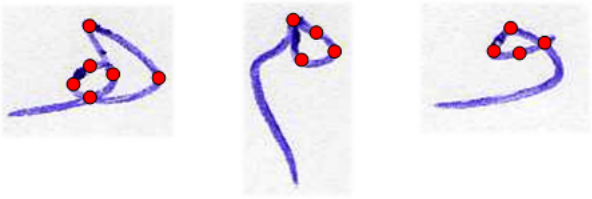


Figure 4.13: Illustration d'un exemple de points de contrôle.

E. Angle de déviation

L'angle de déviation de chaque segment est calculé à chaque fois qu'on fait une opération de fusion ou que nous voulons extraire les caractéristiques de la forme d'une boucle. Pour la calculer on fait appel à la fonction suivante :

Algorithme 4.2: Fonction de calcul d'angle de segment

```
Fonction getAngle (a, b : Point): double
    angle, dx, dy : double;
Début
    dx ← b.x − a.x;
    dy ← a.y − b.y;
    angle ← atan2(dy, dx);
    Si (dy < 0) Alors angle ← angle + (2 × PI);
    retourner angle;
Fin
```

F. Secteurs des angles

Les secteurs sont utilisés pour avoir une description présentant la relation entre les angles de déviation des segments, et au même temps on garde un certain degré de liberté.

La figure suivante illustre ce principe.

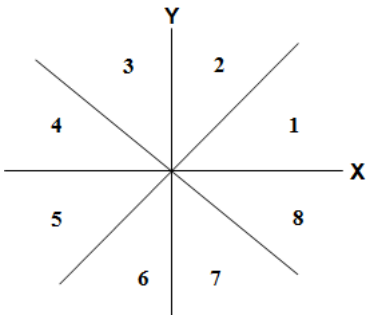


Figure 4.14: Secteurs d'angle.

Après le calcul de l'angle avec l'algorithme précédent, nous pouvons obtenir le secteur de cette angle en appelant la fonction 4.3:

Exemple: En supposant qu'un secteur est noté comme suit : s(num_secteur) nous obtenons la figure suivante:

Algorithme 4.3: Fonction de calcul du secteur d'angle.

Fonction *getAngleSector (angle : double)*: *Entier*
  rem : *double*;
  sector : *entier*;
Début
  $rem = anglemod(2 \times PI)$;
  $sector = 1 + floor(rem/(PI/4)))$;
  retourner *sector*;
Fin

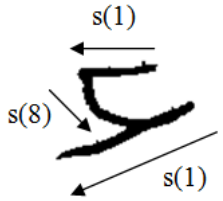


Figure 4.15: Illustration d'un exemple de secteurs d'angles.

Et par la suite, nous pouvons avoir la description suivante:

$SADescriptor = 181$, (Sector Angle Descriptor).

G. Taille du segment d'image

La taille du segment d'image est définit par la dimension de l'image (largeur, hauteur) et elle est facilement calculable par les deux fonctions (fournit par le langage de programmation – JAVA-) suivantes :

- getWidth:fonction permet d'obtenir la largeur d'une image,
- getHeight: fonction permet d'obtenir la hauteur d'une image.

En voyant bien, la figure 2.26 nous pouvons déduire que la taille du segment d'image peut être un élément clé pour différencier entre les différents formes (par exemple entre les cadres : 1, 2 et 3).

H. Type des points constituants la boucle

Le type des points constituant la boucle est obtenu au fur et à mesure qu'on fait des opérations de division ou fusion. Cette étape est réalisée à l'aide d'une classe Java appelée Tracker (nom_Image), qui va prendre chaque image et faire un suivi afin d'obtenir les caractéristiques ou vecteur caractéristique (taille, nb_Cycles, Direction...etc) de chaque caractère.

A la fin de la phase d'extraction de caractéristiques, nous disposons d'un vecteur de coefficients réels qui caractérisent un/des caractère(s). Dans la phase suivante, la méthode SVM est utilisée pour se servir de ces coefficients et reconnaître le(s) caractère(s).

4.1.5 Classification

La bibliothèque SVM dans [69] est utilisée pour la classification avec un noyau gaussien. Elle contient deux modules principaux : le premier pour l'apprentissage et l'autre pour la classification (voir la figure 4.16). Le premier, utilise les vecteurs caractéristiques (taille, nb_Cycles, Direction ...etc) obtenus dans la phase précédente et leurs labels ou étiquettes pour trouver un hyperplan séparateur durant la génération des modèles d'apprentissage pour un ensemble d'exemples donnés de la base IFN/ENIT pendant la phase d'apprentissage. Le second, permet de prendre des décisions (obtenir des labels de classes) pour identifier de nouveaux exemples de la même base pendant la phase de classification.

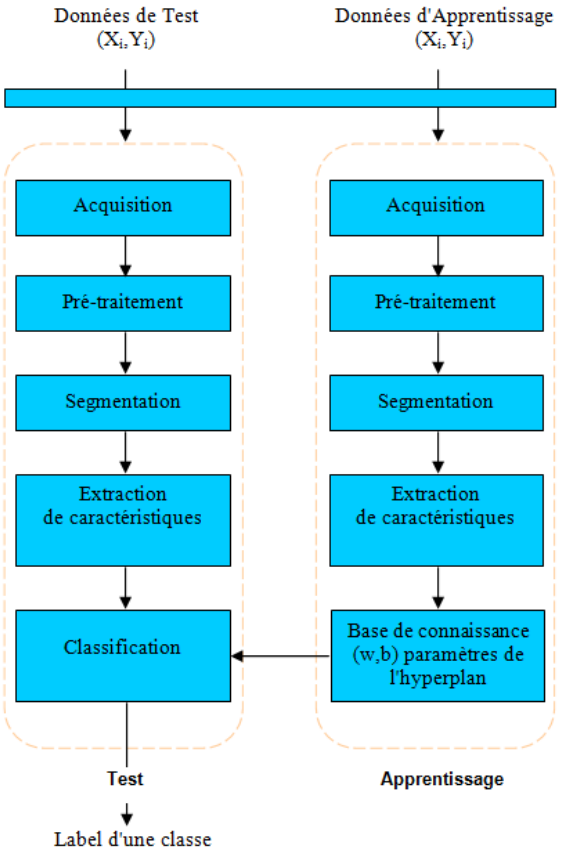


Figure 4.16: Phases utilisées par un classifieur SVM.

Où :
X_i : le vecteur caractéristique d'un caractère ;
Y_i : l'étiquette d'une classe.

Le package libsvm-2.83 de LIBSVM est utilisé comme outil de classification.Il se compose de trois couches principales (voir figure 4.17):

- **Application:** représente l'application utilisateur,
- **Interface:** permet la communication entre l'application et les modules de calcule, et
- **Calcul:** permet de réaliser les calculs nécessaires.

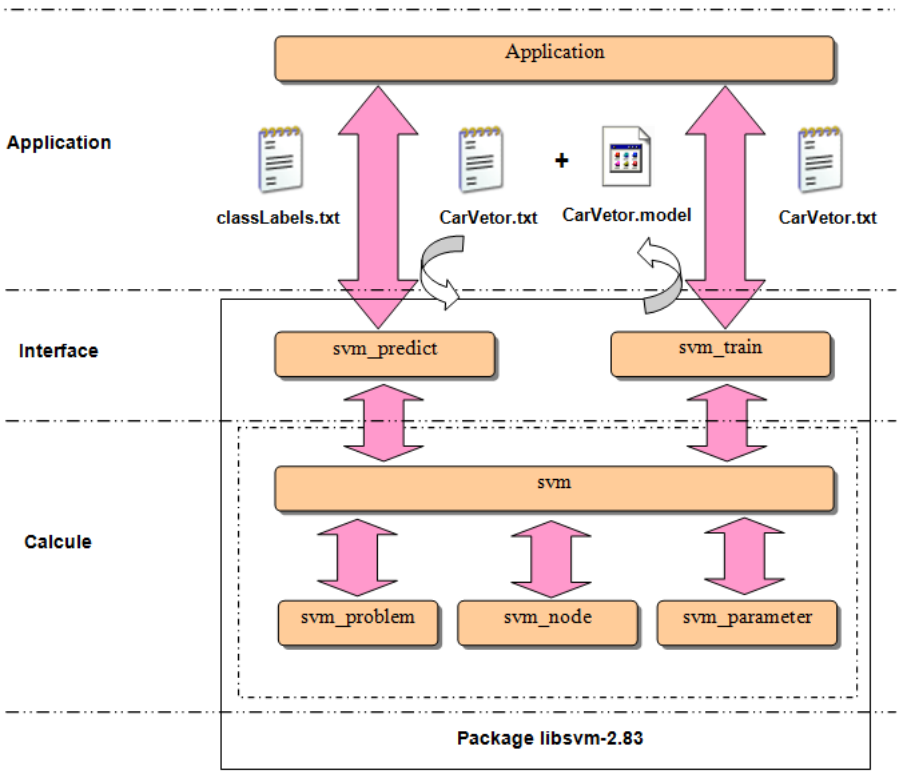


Figure 4.17: Relation entre l'application utilisateur et le package libsvm-2.83.

La couche interface est présentée par les deux modules:

- svm_train(CarVec.txt) : pour la phase d'apprentissage, et
- svm_predict (CarVec.txt, CarVector.model, classLabels.txt) : pour la phase de test (prédiction).

Où:
CarVec.txt : fichier contenant les caractéristiques extraites.
CarVector.model : fichier contenant les paramètres (w,b) après la phase d'apprentissage.
classLabels.txt : fichier contenant les étiquettes des classes après la phase de test ou décision.

On note que la couche calcul est présentée par les modules :

- svm: comme module principale,

- svm_node, svm_parameter, et svm_problem: comme des modules complémentaire.

L'utilisateur ne peut communiquer avec le package qu'à travers les deux modules de la couche interface présentée ci-dessus. Premièrement, dans la phase d'apprentissage la couche application génère un fichier texte (CarVector.txt) contenant les vecteurs caractéristiques (X_i) et les étiquettes des classes (Y_i), en utilisant ce fichier comme entrée au module svm_train nous obtenant un fichier de modèle (CarVector.model) qui contient les paramètres (w,b) nécessaire à la phase de test ou prédiction.

En revanche, dans la phase de test, on utilise le fichier de modèle généré précédemment plus un fichier texte (CarVector.txt) généré par l'application nous obtenant comme résultat un autre fichier texte (classLabels.txt) qui contient les étiquettes des classes.

Pour une classification multi-classes, nous utilisons une stratégie de vote : chaque classification binaire est considérée comme un vote, où les votes sont considérés comme une caste pour les points des données X. la classe choisi est celle qui a le nombre max de votes [31].

4.1.6 Post-traitement

Cette phase consiste à faire le processus inverse de la segmentation, en se basant sur un ensemble de règles et des modèles de la langue cible (forme, caractères et mots). Le processus de reconstruction passe par les trois étapes suivantes :

1. reconstruction des caractères,
2. reconstruction des mots,
3. post-processeur puzzle.

A. Reconstruction des caractères

Après avoir obtenu les labels des classes (le tableau 4.2 présente les labels des segments (a), (b),...,(q)) de l'opération de classification, il faut avoir maintenant les labels des caractères adéquats. Le système utilise une base des règles de composition (voir le tableau 4.3) avec les segments graphiques de base pour générer une liste des caractères possibles (en considérant le contenu du tableau 2.4 comme une matrice).

Tableau 4.2: Exemples de labels des segments.

Segment	Label
(a)	Del_I
(b)	Alif_I
(c)	Ra_I
(d_1)	Ba_D
(d_2)	Ba_M
(d_3)	Ba_M
(d_4)	Ain_M
(d_5)	Ba_M
(d_6)	Alif_F
(e)	1Point
(f)	1Point
(g)	1Point
(h)	1Point
(i_1)	Ra_I
(i_2)	Lam_F
(j)	1Point
(k)	Alif_I
(l_1)	Lam_D
(l_2)	Ba_M
(l_3)	Ba_M
(l_4)	Alif_F
(m)	2Point
(n)	1Point
(o_1)	Sad_D
(o_2)	Alif_D
(o_3)	Lam_F
(p)	Alif_I
(q)	Hamza

Tableau 4.3: Exemple de règles composition formes du tableau 2.4

Position / Caracters	start	middle	end	isolated
Alif			(1,3)	(1,4)
Ba	(2,1)	(2,2)	(2,2) + (2,3)	(2,1) + (2,3)
Hha	(3,1)	(3,2)	(3,3)	(3,4)
Sin	(2,1) + (2,2) + (2,2)	(2,2) + (2,2) + (2,2)	(2,2) + (2,2) + (13,2_2)	(2,1) + (2,2) + (13,2_2)
Fa			(5,2) + (2,3)	(5,1) + (2,3)
Khaf	(9,2)	(5,1)	(9,2) + (2,3)	(5,1) + (10,3)
Lam			(10,2) + (10,3)	(10,1) + (10,3)
Sad	(13,1) + (2,2)	(13,2) + (2,2)	(13,2) + (10,3)	(13,1) + (10,3)

Dans la figure 2.31, nous avons vu un exemple de génération de formes primitives, la fonction $reassembleCharacter(CL : entier)$ permet de faire le processus inverse (voir la figure 4.19 et la fonction 4.4).

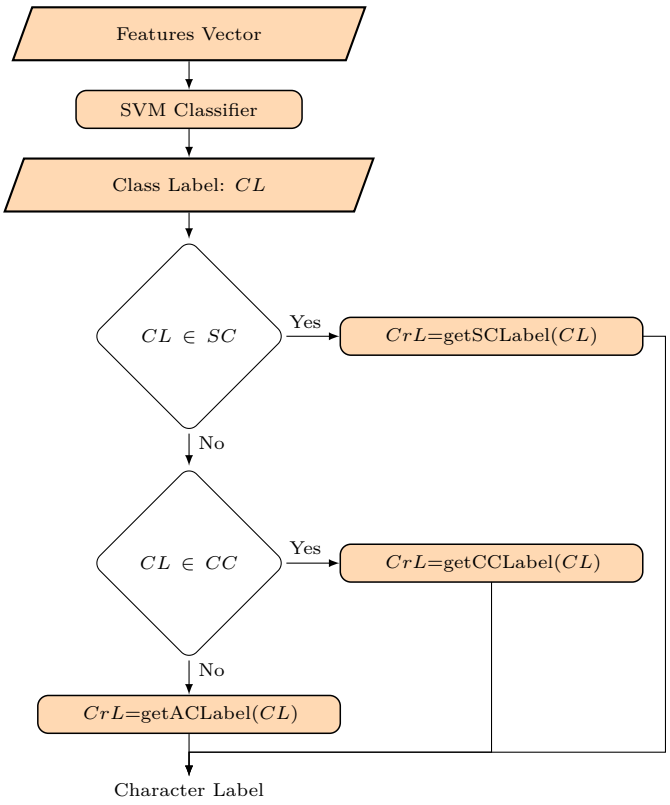


Figure 4.18: Sélection du label d'un caractère.

Figure 4.19: Exemple de composition de formes primitives.

Où:

getUpperSegmentClassLabel():permet de retourner la classe du segment d'image en haut du segment traité.
getLowerSegmentClassLabel():permet de retourner la classe du segment d'image en bas du segment traité.

Le reste des caractères est reconstruit de la même façon en donnant un code d'entrée et en obtenant un code de sortie pour le caractère adéquat.
En appliquant ce processus sur les segments du tableau 4.2 nous allons obtenir les labels de caractères indiqués par le tableau 4.4. Les segments (a), (b), (c), (d_4), (d_6), (k) et (p) vont garder les mêmes labels parce que ce sont des segments de caractères de structure simple à un segment. Par contre, les segments d=(d_1), (d_2), (d_3), (d_5), (e), (f), (g), (h), (i), (j), (l), (m), (n), (o) et (q) doivent traiter pour reconstruire les caractères qui possèdent plusieurs segments. La composition des segments (d_1), (d_2), (d_3), (e), (f) et (g)

Algorithme 4.4: Fonction de reconstruction des formes des caractères.

```
Fonction reassembleCharacter (CL : Entier): Entier
    // Character Label: label d'un caractère
    CrL : entier;
Début
    switch CL do
        case 3
            Si (getUpperSegmentClassLabel() == 19) Alors CrL = 34; //ﺃ
            Sinon
                Si (getLowerSegmentClassLabel() == 19) Alors CrL = 35; //ﺇ
                Sinon
                    Si (getUpperSegmentClassLabel() == 21) Alors CrL = 36; //ﺁ
                    Sinon
                        Ecrire("Error code ?...");
                    FinSi
                FinSi
            FinSi
        end
        // les autres cas de même façon
    endsw
    retourner CrL;
Fin
```

va générer le caractère ﺸـ avec le label Chin_D. De même, la composition des segments (d_5) et (h) va générer le caractère ﺒـ avec le label Ba_M, En plus, les segments (i_1), (i_2) et (j) vont générer le caractère ﻥ avec le label Noun_I. Les segments (l_2), (l_3), (m) et (n) vont générer le caractère ﺸـ avec le label Chin_M. La composition va donner aussi le caractère ﻁ avec le label Tad_D des segments (o_1 et (p). Les derniers segments (o_2), (o_3) et (q) vont former le caractère ﺌ avec le label Alif_Maqsoura_Hamza.

Tableau 4.4: Exemples de labels des caractères.

Groupe de segments/Caractère	Label
(a)	Del_I
(b)	Alif_I
(c)	Ra_I
(d_1, d_2, d_3, (e), (f) et (g))	Chin_D
(d_4)	Ain_M
(d_5 et (h))	Ba_M
(d_6)	Alif_F
(i_1, i_2 et (j))	Noun_I
(k)	Alif_I
(l_1)	Lam_D
(l_2,l_3, (m) et (n))	Chin_M
(l_4)	Alif_F
(o_1 et (p))	Tad_D
(o_2, o_3 et (q))	Alif_Maqsoura_Hamza

B. Reconstruction des mots et lignes

Dans cette phase, le système possède une séquence de labels qui peut être dans l'un des cas suivants:

1. la séquence contient des labels inconnus;

2. la séquence est correcte et égale à un mot de la base;

3. la séquence est correcte pas de mot semblable dans la base.

Le premier cas, implique la correction de la séquence en utilisant quelques règles de morphisme. Le second, retourne le mot sans faire des traitements supplémentaires. Tandis que le troisième, calcule une distance de Levenshtein (LD) entre le mot reconnu (MR) et le mot le plus proche (MP). Quand $LD > 0$, MP est vu comme l'état final et MR comme l'état initial d'un jigsaw puzzle. Après avoir appliquer les étapes du puzzle: Fusion, Division et/ou Morphisme pour modifier l'ensemble du puzzle (labels des segments), deux cas possibles peuvent apparaître:

1. il peut atteindre MP du MR, alors MP est retourné (MR'=MP), autrement;

2. il ne peut pas atteindre MP du MR, alors MR est retourné (MR n'est pas parmi les modèles du langages (voir la figure 4.20))

B.1 Séquence avec des labels inconnus

Dans certain cas, il n'est pas possible de passer des labels des segments aux labels des caractères parce que il y'a des segments manquant, comme c'est le cas pour le mot **زروق** dans l'image référencée par aq34_059, et présenté par la figure 4.23.

Le tableau 4.5 gauche présente les labels des segments obtenus. L'ensemble des labels de caractères correspondant se trouve dans le tableau 4.6. translation de la séquence des labels de segments en un une séquence de labels n'est pas complet parce que le label Fa_M du segment a_3 n'a pas de segment complémentaire (1Point ou 2Point) afin de le transformer au caractère Fa_M ou au caractère Khaf_M. Afin de corriger cette situation le système cherche de la base l mot le plus proche de la séquence de caractères générés, ce qui génère comme résultat le mot **زروق**. En plus, il possède dans base des règles de morphisme une règle qui permet de transformer Fa_M à Waw_I. Ces deux informations indiquent que le mot cible est bien **زروق**. Afin de confirmer cette information le système cherche s'il y'a des règles de morphisme pour le reste des caractères qui permettes cette transformation. Par conséquent, il applique les règles de morphisme suivantes:

1. Transformer Fa_M à Waw_I.

2. Transformer Noun_D à Zai_I.

3. Transformer Ta_M à Ra_I.

4. Transformer Khaf_F à Khaf_I.

Le tableau 4.7 montre la séquence de lettre finale.

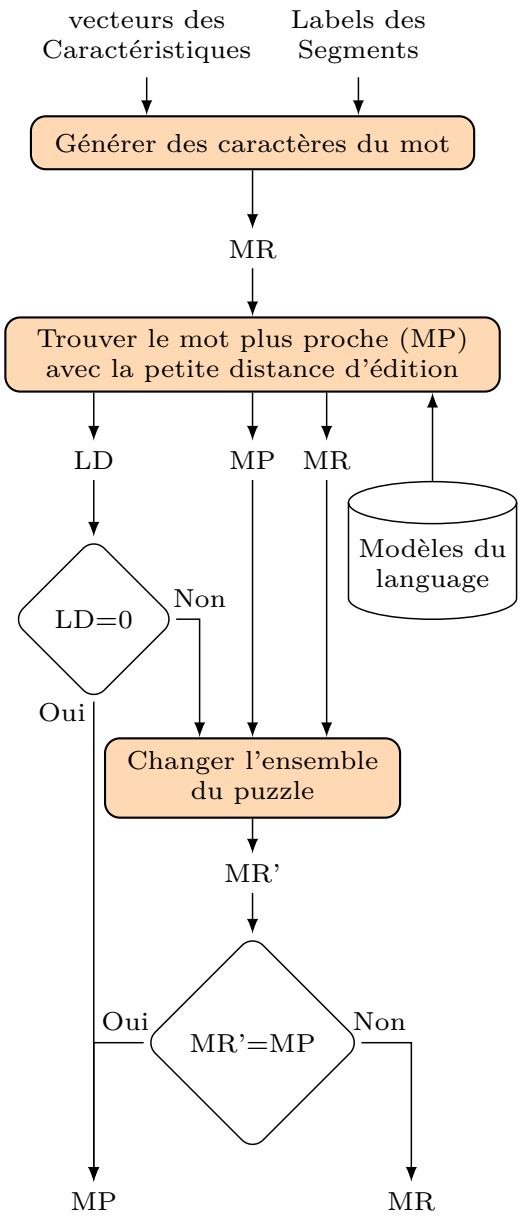


Figure 4.20: Système à base du puzzle.

Tableau 4.5: Labels des segments du mot **زروق** dans l'image référencée par aq34_059.

Segment	Label
(a_1)	Ba_D
(a_2)	Ba_M
(a_3)	Fa_M
(a_4)	Fa_M
(a_5)	Ba_F
(b)	1Point
(c)	2Point
(d)	1Point
(e)	1Point

Tableau 4.6: Labels des caractères/segments du mot **زروق** dans l'image référencée par aq34_059 avant le morphisme.

Groupe de segments/Caractère	Label	Type
(a_1 et b)	Noun_D	Caratère
(a_2 et c)	Ta_M	Caratère
(a_3)	Fa_M	Segment
(a_4, d, e et a_5)	Khaf_M	Caratère

B.2 Séquence correcte

Dans ce cas, la séquence est seulement composée de labels des caractères (pas de label de segment). A ce niveau, le système cherche à trouver une correspondance du mot construit

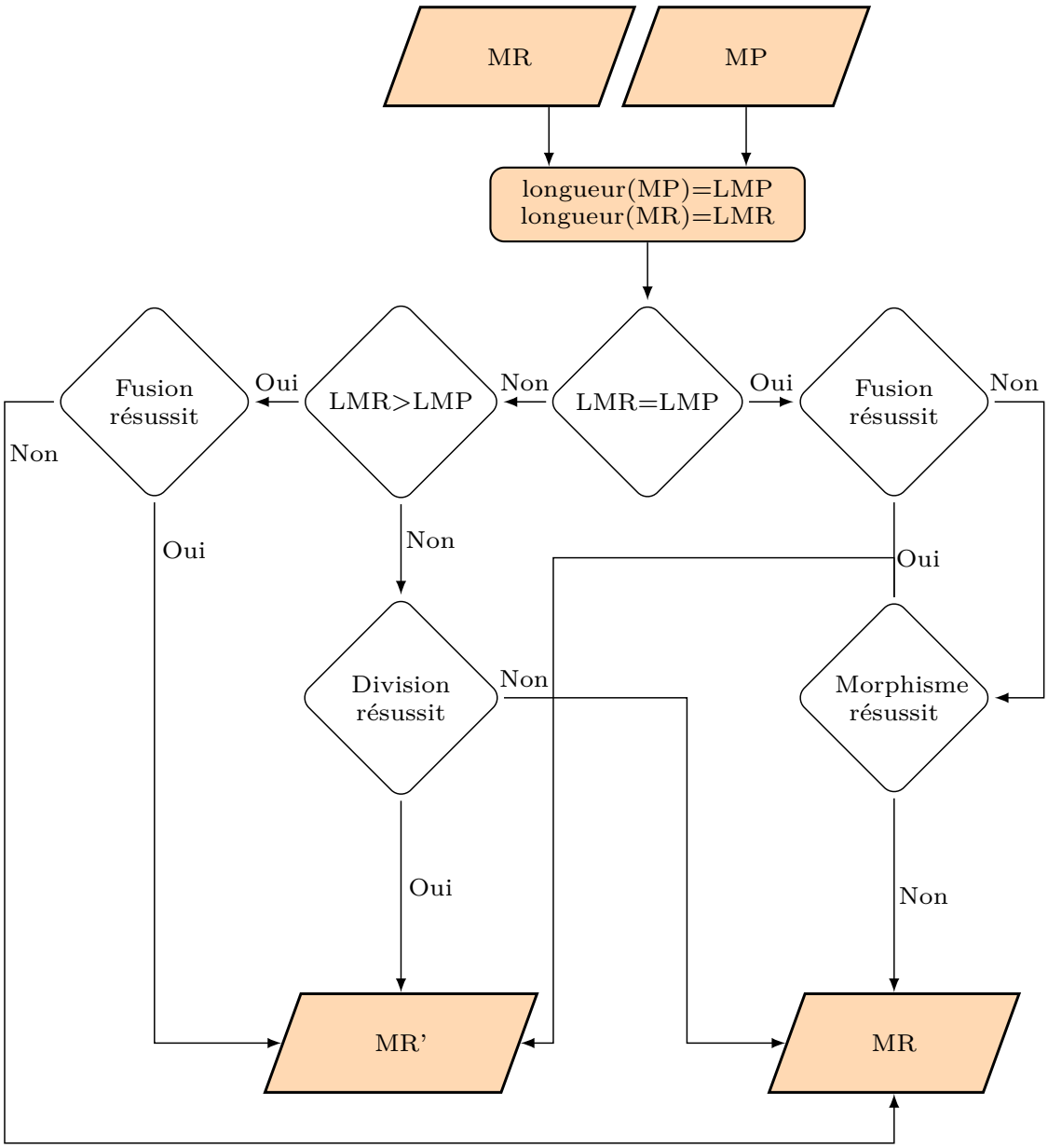


Figure 4.21: Processus du changement de l'ensemble du puzzle.

Tableau 4.7: Labels des caractères/segments du mot **زروق** dans l'image référencée par aq34_059 après le morphisme.

Groupe de segments/Caractère	Label	Type
(a_1 et b)	Zai_I	Caratère
(a_2 et c)	Ra_I	Caratère
(a_3)	Waw_I	Caratère
(a_4, d, e et a_5)	Khaf_I	Caratère

de ces caractères dans la base des mot en calculant une distance de Levenshtein (LD). Dans le cas où une correspondance est trouvé (LD=0), il considère que cette recherche confirme que le mot est bien reconnu. Par contre, si LD>0 alors il cherche à appliquer des étapes de puzzling: fusion, division et morphisme autant que possible afin de transformer le mot reconnu (RW) au mot trouvé dans la base (CW). Si cette transformation réussit ,

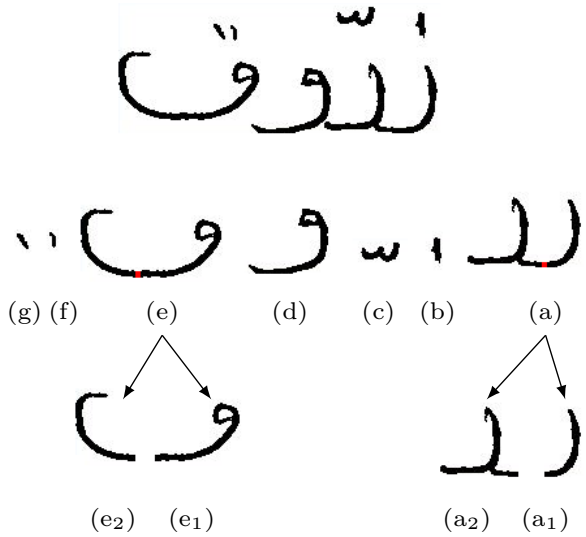


Figure 4.22: Exemple de mot avec des segments nécessitant une correction avec le morphisme.

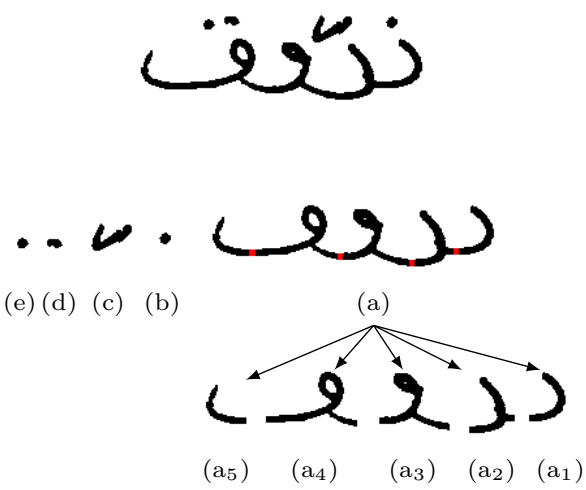


Figure 4.23: Exemple de mot avec des segments nécessitant une correction avec le morphisme.

le système retourne CW sinon il retourne RW.

B.2.1 Morphisme

L'application des phases de reconnaissance de l'acquisition jusqu'à la classification sur l'image du mot **توزر** référencée par ae97_006 génère la séquence des segments présentés par la figure 4.25 avec les labels indiqués par le tableau 4.8. La translation de ces labels en une séquence de labels des caractères est présentée par le tableau 4.9, ce qui génère comme résultat le mot **توتد**. Dans la réalité, durant la phase de recherche le système ne trouve pas un correspondant à ce mot, mais, il trouve un mot très proche qui est **توزر**. Par conséquent, il essaye d'appliquer des étapes de puzzling afin transformer **توتد** à **توزر**. A ce niveau il trouve qu'il est possible d'appliquer les règles de morphisme R_7 et R_8 pour transformer Ta_D à Zai_I, et R_6 pour transformer Del_F à Ra_I (voir la figure 4.24).

R_1 : ـي <——> ـى
R_2 : ظ <——> ضـ
R_3 : ئـ <——> يـ
R_4 : ـي <——> ـى
R_5 : ـع <——> ـگ
R_6 : ـد <——> ـر
R_7 : ◌ٔ <——> ◌ٕ
R_8 : ◌ٕ <——> ر

Figure 4.24: Exemples de règles de morphisme.

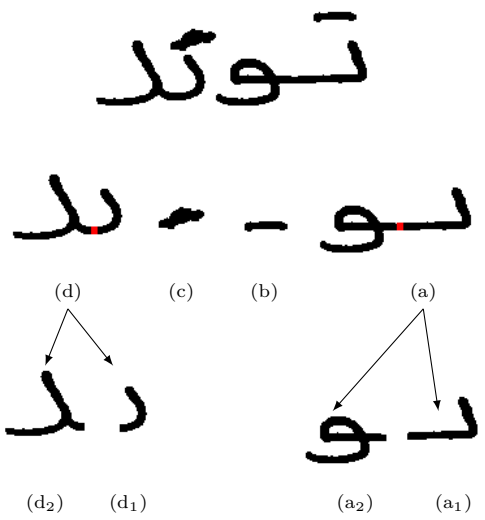


Figure 4.25: Mot avec des PAWs nécessitant un morphisme.

Tableau 4.8: Labels des segments du mot **توزر** dans l'image référencée par ae97_006.

Segment	Label
(a_1)	Ba_D
(a_2)	Waw_F
(b)	2Point
(c)	2Point
(d_1)	Ba_D
(d_2)	Del_F

Tableau 4.9: Labels des caractères/segments du mot **توزر** dans l'image référencée par ae97_006 avant le morphisme.

Groupe de segments/Caractère	Label	Type
(a_1 et b)	Ta_D	Caratère
(a_2)	Waw_F	Caratère
(d_1 et c)	Ta_D	Caractère
(a_2)	Del_F	Caratère

Tableau 4.10: Labels des caractères/segments du mot **توزر** dans l'image référencée par ae97_006 après le morphisme.

Groupe de segments/Caractère	Label	Type	Règle de morphisme
(a_1 et b)	Ta_D	Caratère	
(a_2)	Waw_F	Caratère	
(d_1 et c)	Zai_I	Caractère	R_7 et R_8
(a_2)	Ra_I	Caratère	R_6

B.2.2 Fusion

Pareillement, l'application des phases de reconnaissance de l'acquisition jusqu'à la classification sur l'image du mot **الشاطئ** référencée par aq34_047 génère la séquence des segments présentée par la figure 4.26 avec les labels indiqués par le tableau 4.11. La translation de ces labels en une séquence de labels des caractère est présentée par le

tableau 4.12, ce qui génère comme résultat le mot الشاصئ et le label Alif_I. Lorsque le système cherche une correspondance, il ne va pas le trouvé, mais, il trouve un mot très proche qui est الشاطئ. Par conséquent, quand il applique les étapes de puzzling pour transformer الشاصئ à الشاطئ, il trouve qu'il est possible de faire une fusion entre les deux segments (e_1) et (f). Le tableau 4.13 présente les labels de caractères correctes.

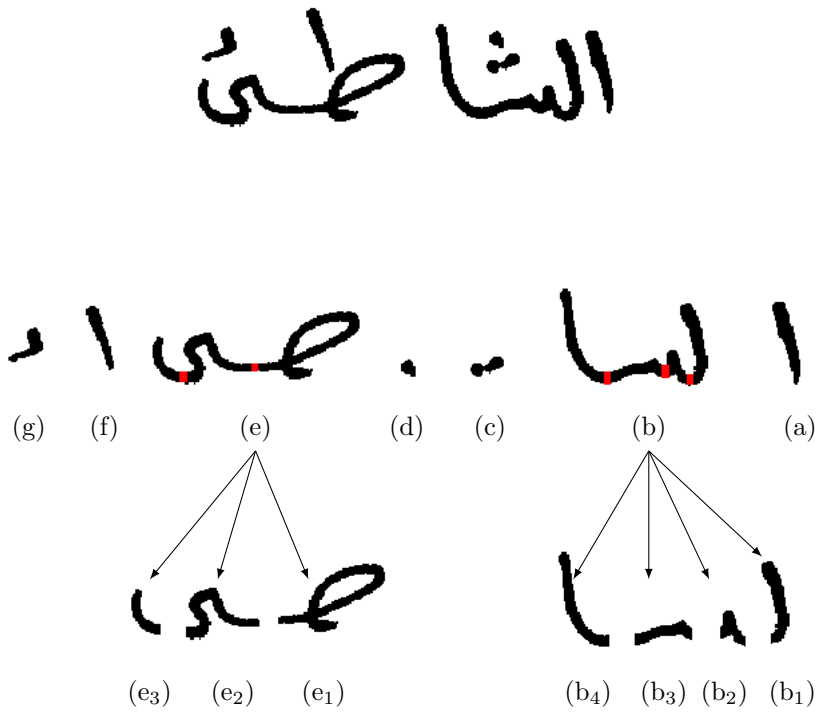


Figure 4.26: Mot avec des PAWs nécessitant une fusion.

Tableau 4.11: Labels des segments du mot الشاطئ dans l'image référencée par ae97_006.

Segment	Label
(a)	Alif_I
(b_1)	Lam_D
(b_2)	Ba_M
(b_3)	Ba_M
(b_4)	Alif_F
(c)	2Point
(d)	1Point
(e_1)	Sad_D
(e_2)	Alif_D
(e_3)	Lam_F
(f)	Alif_I
(g)	Hamza_I

Tableau 4.12: Labels des caractères/segments du mot الشاطئ dans l'image référencée par ae97_006 avant la fusion.

Groupe de segments/Caractère	Label	Type
(a)	Alif_I	Caratère
(b_1)	Lam_D	Caratère
(b_2, b_2, c et d)	Chin_M	Caratère
(b_4)	Alif_F	Caratère
(e_1)	Sad_D	Caratère
(e_2, e_3 et (g))	Alif_Maqsoura_Hamza_F	Caratère
(f)	Alif_I	Caratère

Tableau 4.13: Labels des caractères/segments du mot الشاطئ dans l'image référencée par ae97_006 après la fusion.

Groupe de segments/Caractère	Label	Type
(a)	Alif_I	Caratère
(b_1)	Lam_D	Caratère
(b_2, b_2, c et d)	Chin_M	Caratère
(b_4)	Alif_F	Caratère
(e_1 et f)	Tad_D	Caratère
(e_2, e_3 et (g))	Alif_Maqsoura_Hamza_F	Caratère

B.2.3 Division

De façon analogue, le système génère la séquence de segments présentée par la figure 4.27 pour l'image du mot الرديف référencée par aq34_042.Le tableau 4.14 présente les labels obtenus pour cette séquence. Le résultat de translation en labels de caractères est présenté dans le tableau 4.15. Le mot résultant va être عرديف. Ce ne peut pas être trouvé dans la base des mots, mais, un mot proche existe الرديف. Après ça, le système lance le processeur puzzle afin de transformer le mot عرديف à الرديف. Tant que la longueur du mot عرديف est inférieur de la longueur du mot الرديف la seule opération qui peut corriger cette situation est la division du segment (a_1) en deux. Le système est capable de trouver dans les points constituant ce segment un point de fusion et de division au même temps (le point en vert dans la figure 4.27) ce type de points est généralement observé dans le cas de chevauchement de deux segments de deux caractères. En appliquant l'opération de division, le système génère les deux segments ($a_1$1) et ($a_1$2) avec les labels Alif_I et Lam_D respectivement. Le tableau 4.16 contient la séquence de caractères du mot correcte.

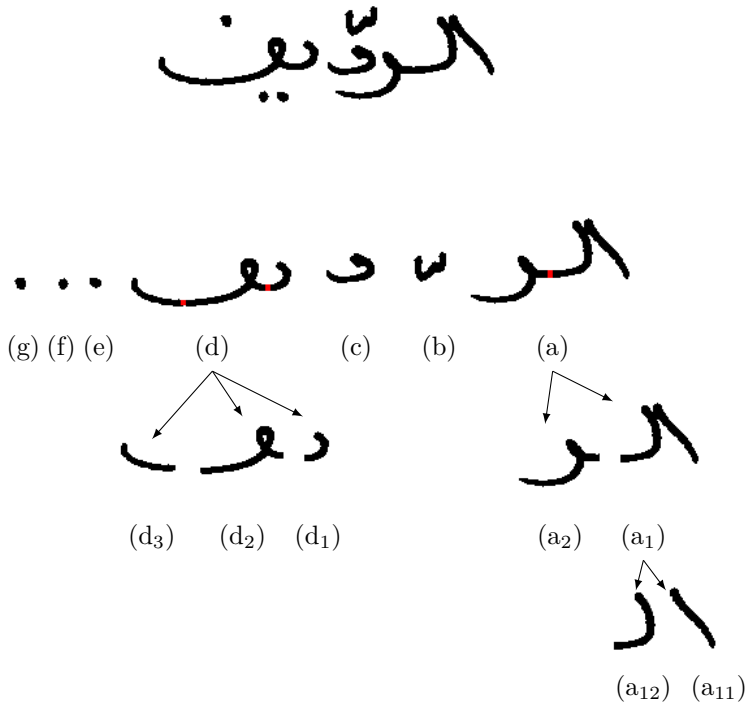


Figure 4.27: Mot avec des PAWs nécessitant une Division.

Tableau 4.14: Labels des segments du mot الرديف dans l'image référencée par aq34_042.

Segment	Label
(a_1)	Ain_D
(a_2)	Ra_F
(b)	2Point
(c)	Del_I
(d_1)	Ba_D
(d_2)	Fa_M
(d_3)	Ba_F
(e)	2Point
(f)	2Point
(g)	2Point

Tableau 4.15: Labels des caractères/segments du mot الرديف dans l'image référencée par aq34_042 avant la division.

Groupe de segments/Caractère	Label	Type
(a_1)	Ain_D	Caratère
(a_2)	Ra_F	Caratère
(b)	2Point	Segment
(c)	Del_I	Caratère
(d_1, e et f)	Ya_D	Caratère
(d_2, d_3, et g)	Fa_M	Caratère

Tableau 4.16: Labels des caractères/segments du mot الرديف dans l'image référencée par aq34_042 après la division.

Groupe de segments/Caractère	Label	Type
($a_1$1)	Alif_I	Caratère
($a_1$2)	Lam_D	Caratère
(a_2)	Ra_F	Caratère
(b)	2Point	Segment
(c)	Del_I	Caratère
(d_1, e et f)	Ya_D	Caratère
(d_2, d_3, et g)	Fa_M	Caratère

C. Reconstruction de page

Vu qu'on effectue la reconnaissance de document simple qui contiennent uniquement du texte cette étape est pratiquement réalisée automatiquement dans l'étape précédente. Les lignes obtenues sont écrites dans un fichier (docx par défaut). La conversion est réalisée en utilisant les codes de l'unicode standard version 6.3 présentés dans le tableau 4.17.

Tableau 4.17: Codes des caractères utilisés par le système.

Num	Code d'entrée	Caractère	Code de sortie
001	Kef	كـ	1603
002	Mim	م	1605
003	Lam	ل	1604
Caractères Simples			
004	Alif	ا	1575
005	Hha	ح	1581
006	Del	د	1583
007	Ra	ر	1585
008	Sin	س	1587
009	Sad	ص	1589
010	Tta	ط	1591
011	Ain	ع	1593
012	Alif_Maksoura	ى	1609
013	Waw	و	1608
014	Ha	ه	1607
015	Hamza	ء	1569
Caractères Ambigus			
016	Fa	ف	1601
017	Khef	ق	1602
018	Noun	ن	1606
019	Ya	ي	1610
020	Ba	ب	1576
021	Ta_Maftouha	ت	1578
022	Ta_Marbouta	ة	1577
023	Tha	ث	1579
024	Djim	ج	1580
025	Kha	خ	1582
026	Dhel	ذ	1584
027	Zai	ز	1586
028	Chin	ش	1588
029	Gain	غ	1594
030	Dha	ظ	1592
031	Dhad	ض	1590
032	Alif_Hamza_Haut	أ	1571
033	Alif_Hamza_Bas	إ	1573
034	Alif_Med	آ	1570
035	Waw_Hamza_Haut	ؤ	1572
035	Alif_Maksoura_Hamza_Haut	ئ	1574
Caractères Composés			

4.2 Résultats et Validation

Cette section, présentera le benchmark IFN/ENIT utilisé pendant l'évaluation du système, les résultats de segmentation, les résultats de reconnaissance et la complexité d'exécution du système.

4.2.1 Benchmark IFN/ENIT

Le système proposé est évalué en utilisant la version version v2.0p1e de la base Arabe IFN-ENIT pour les deux phases apprentissage et test, parceque c'est la base le plus utilisée dans le test des systèmes hors-ligne de reconnaissance du manuscrit arabe (voir [44], le tableau 4.18 et le tableau 4.22). Elle est créée par l'institut de communication et de technologie (IFN : Institute of Communication Technology) à l'université technique de Braunschweig en Allemagne, et l'Ecole Nationale d'Ingénieurs de Tunis (ENIT) en Tunisie. La base contient un total de 32,492 mots manuscrits et symboles, de 946 noms de villes et villages écrit par 411 personnes différentes. Ces documents sont divisés en quatre ensembles a, b, c et d et ils sont utilisés pour l'apprentissage et le test. Les exemples sont scannés avec une résolution de 300 dpi.

4.2.2 Résultats de segmentation

L'évaluation de la performance de la segmentation a été basée sur le calcul des taux de segmentation des caractères bien-segmentés (right segmented (RS)), sous-segmentés (sub-segmented (SS)) et sur-segmentés (over segmented (OS)) dans les deux cas en utilisant (APC: Activating Puzzle Controller) et sans utiliser (DPC: Deactivating Puzzle Controller) le contrôleur du puzzle.

Le taux de segmentation moyen obtenu, avec 90.5% de caractères bien-segmenté, 4.53% de caractères sous-segmentés et 5.42% de caractères sur-segmentés est expliqué par les coupures et chevauchements, qui produit des caractères partiellement combinés dans le premier cas, et deux formes de caractères fusionnés dans le second cas.

Tableau 4.18: Taux de segmentation obtenus par cross validation avec la base IFN/ENIT.

Apprentissage	Test	RS	OS	SS
bcde	a	89.93	04.03	06.04
adce	b	89.87	04.63	05.50
abde	c	90.03	05.00	04.97
abce	d	89.90	03.77	06.37
abcd	e	90.55	05.21	04.24
Taux moyen		90.05	04.53	05.42

Ces différences en taux de segmentation (voir le tableau 4.18) sont le résultat:

1. des différences en nombre d'images utilisées pour l'apprentissage et pour le test, et
2. le nombre des caractères et mots ambigus.

De plus, les résultats de segmentation de notre système ont été comparés avec les résultats de certains d'autres systèmes (voir le tableau 4.19). Le système de Cheng et al.[47] basé sur les caractéristiques et les réseaux de neurones artificiels (ANNs) donne des résultats meilleurs par rapport aux notre, mais utilise un ensemble d'évaluation de CEDAR composé de 317 mots seulement. La même remarque pour le système à base d'analyse de ligatures et formes développé par Khan et Mohammad [68]. Le système de Abandah et al.[2] basé sur une segmentation de graphèmes surpasse notre système avec son taux de segmentation de 96% mais utilise uniquement 107 exemples sélectionnés de la base IFN/ENIT. Aussi, le système dans [74] basé sur une approche de segmentation structurelle possède des résultats plus grand que le notre mais avec ensemble de test de 200 images de la base IFN/ENIT. Un autre système à base de ligature et ANNs est développé par Cheng et Blumenstein [19], qui atteint un taux de segmentation de 84.19% basé sur un ensemble d'évaluation de 317 mots de la base de CEDAR. Aouadi et al.[5] utilisent une approche de segmentation basée sur la reconnaissance et des transformations, qui réalise un résultat moyen pondéré de segmentation de 86.73% avec un ensemble de test de documents Arabe de l'archive national tunisien.Le problème majeur d'obtenir un taux d'erreur de 13.27% est la nature ambigus des mots. Une approche de segmentation dual-phase a été adoptée par Elzobi et al. [6] pour segmenter les images de 600 mots de la base IESK-ArDB. Le système a pu obtenir un taux de segmentation de 67%. La segmentation à base des Random Forests qui était choix par Abdeen et al.[97] obtient un taux de 88.71% avec 500 images de test de la base IFN/ENIT. Clairement, le taux de segmentation moyen de 90.05% obtenu par le processus de cross-validation et la méthode de segmentation proposée N-Scans est beaucoup plus robuste, et prouve que la méthode structurelle proposée peut être une source robuste pour les traitements des phases ultérieurs.

Tableau 4.19: Comparaison avec d'autres méthodes de segmentation.

Système	Approche de segmentation	BDD	Exemples	TS
Cheng et al. (2004) [47]	Basée caractéristiques+ANN	CEDAR	317 mots	95.27
Lorigo and Govindaraju (2005) [74]	Structurelle	IFN/ENIT	200 images	92.30
Cheng and Blumenstein (2005) [19]	Détection de ligature+ANN	CEDAR	317 mots	84.19
Khan and Mohammad (2008) [68]	Analyse de ligatures et formes	CEDAR	2,936 mots	91.21
Aouadi et al.(2013) [5]	Reconnaissance et tranformation	TNAs	Non donné	86.73
Elzobi et al.(2013) [6]	Dual-Phase	IESK-ArDB	600 mots	67.00
Abandah et al.(2014) [2]	Graphèmes	IFN/ENIT	107 exemples	96.00
Abdeen et al.(2015) [97]	Random Forests	IFN/ENIT	500 mots	88.71
Notre méthode	N-Scans	IFN/ENIT	a,b,c,d,e [*]	90.53

[*]: a=6,537 images, b=6,710 images, c=6,477 images, d=6735 images, e=6,033 images.

4.2.3 Résultats de Classification

Egalement, les résultats de classification (classifieur SVM et de post-classification) sont évalués avec et sans utiliser le contrôleur du puzzle. Selon les opérations appliquées, les caractères peuvent être catégorisés en trois classes: premièrement, les caractères classés de manière normale (pas de morphisme ou d'opération de Fusion/Division), deuxièmement, les caractères classés par morphisme (comme ك, ع, بـ, يـ, ظ, ض, ى, لا... etc.), et finalement, les caractères classés après des opérations de Fusion/Division, généralement, ce sont les caractères construits des segments directs (comme ط, كـ,لا... etc.).

Tableau 4.20: Résultats de reconnaissance des caractères simples et ambigus (a)

	Caractère	NE	Taux de reconnaissance	
			DPC	APC
Cars Simple	كـ	61	92.54	98.84
	ل	480	93.66	98.81
	م	224	89.34	98.76
Caractères Ambigus	ا	765	85.41	96.64
	ح	117	90.86	98.44
	د	155	85.4	96.17
	ر	312	83.74	95.75
	س	135	80.34	95.25
	ص	52	80.24	98.67
	ط	70	78.33	94.64
	ع	98	86.77	98.76
	ى	14	93.55	97.49
	و	211	84.43	97.46
	ه	58	90.00	98.13
	ء	2	91.17	98.15

Tableau 4.21: Résultats de reconnaissance pour les caractères composés (b)

	Caractère	NE	Taux de reconnaissance	
			DPC	APC
Caractères Composés	ف	79	78.56	94.38
	ق	98	77.32	94.28
	ن	236	93.00	94.57
	ي	352	93.55	96.65
	ب	176	94.15	97.21
	ت	104	94.09	97.2
	ة	197	89.7	97.29
	ث	27	93.94	97.18
	ج	68	90.58	97.72
	خ	51	90.23	97.8
	ذ	21	85.13	96.01
	ز	91	83.56	95.73
	ش	74	80.23	95.06
	غ	16	86.47	98.04
	ظ	26	78.17	94.39
	ض	36	80.08	97.61
	آ	45	86.81	96.61
	إ	12	87.53	96.27
	أ	3	85.88	96.62
	ئ	12	93.34	97.4
Taux Total			87.08	96.85

Les résultats dans les tableaux 4.20 et 4.21 montrent que l'utilisation des mécanismes de morphisme et de Fusion/Division accroisse significativement le taux de reconnaissance de 87.08% à 96.85%. Mais, une erreur de 3.15% qui veut dire que les règles utilisés par le contrôleur du puzzle n'est pas capable de traiter les caractères avec un grand chevauchement (par exemple le mot الحجام référencé par af03_046).

Tableau 4.22: Comparaison avec d'autres systèmes de reconnaissance de mots.

Système	Méthode	Database	Taux
Srihari et al.(2005) [113]	Ranking	10 writers	70.00
Farah et al.(2005) [49]	SLI	Local base	96.00
Sari et al.(2008) [105]	Levenshtein Distance	IFN/ENIT	80.00
El Abed et al.(2009) [41]	multiple accep/reject strategies	IFN/ENIT	94.71
Saabni et al.(2011) [99]	DTW	IFN/ENIT	85.60
AlKhateeb et al.(2011) [13]	Re-ranking	IFN/ENIT	95.15
Rodriguez-Serrano et al.(2012) [97]	Active DTW	IFN/ENIT	91.20
Salami et al.(2013) [102]	RMP-PSO	HODA	96.42
Kessentini et al.(2015) [65]	multiple accep/reject strategies	IFN/ENIT	87.55
Notre système sans Puzzle	Simple matching	IFN/ENIT	87.33
Notre système avec Puzzle	Puzzle	IFN/ENIT	96.93

En plus, nous avons comparé nos résultats avec les résultats d'autres systèmes utilisant des méthodes de post-traitement différentes. Nous pouvons dire que les résultats de notre système sont meilleurs par rapport aux différents ensembles de test utilisés dans la comparaison avec les travaux présentés dans le tableau 4.22.

4.2.4 Complexité d'exécution

Les tests ont été réalisé en utilisant un PC avec 4.0 GB de RAM, un processeur de 2.53 GHz, système d'exploitation Windows et un programme en Java. La complexité en temps d'exécution est calculée pour les deux phases principales influencées par le contrôleur du puzzle: segmentation et classification (classificateur SVM et post-classification). Le processus est réalisé dans les deux cas en activant (APC) et en désactivant (DPC) le contrôleur du puzzle. L'ensemble de test est composé de 100 exemples choisis, et divisés en trois catégories selon leur complexité d'écriture, comme suit: 33 exemples de mots clair (CW: Clear Words), 34 exemples de mots moyennement clair (MCW: Medium Clear Words) et 34 exemples de mots non clair (UCW: Unclear Words).

Principalement, le contrôleur du puzzle utilise deux types ou niveaux de feedback: retours de niveau un (LOS: Level One Shifts) qui représentent les retours à la liste des caractères possibles créée par les règles de morphisme, et les retours de niveau deux (LTS: Level Two Shifts), présentent les feedbacks à l'ensemble de segments de base pour appliquer des opérations de Fusion/Division. Figure 4.28 ci-dessous montre la différence en temps d'exécution durant le processus de classification. La région entre les deux courbes présente la différence en temps entre l'activation et désactivation du contrôleur du puzzle. La complexité est calculée par mot alors elle dépend du nombre de caractères

composant le mot et le nombre de feedbacks ou étapes du puzzling (qui dépend du nombre d'opérations de Fusion/Division nécessaire pour changer l'ensemble du puzzle). Le nombre d'opérations de Fusion/Division peut être vu aussi comme une métrique de la complexité d'écriture d'un mot. La complexité en temps d'exécution et celle d'écriture possèdent une relation symétrique entre eux.

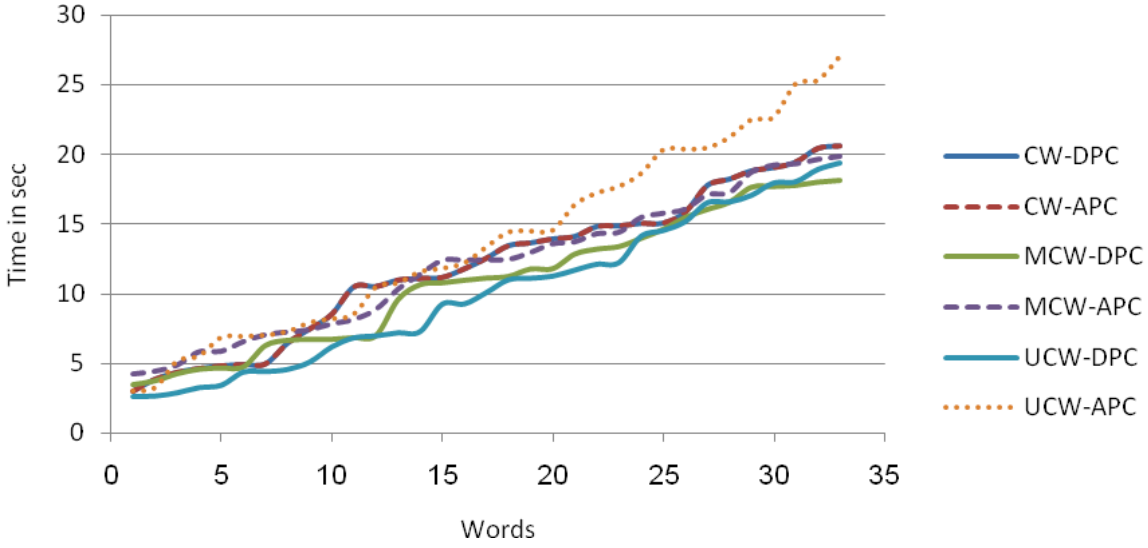


Figure 4.28: Courbes de classification en DPC et APC.

Par conséquent, si le mot est bien écrit le nombre d'opérations de Fusion/Division va être nulle ou petit parce que il y'a un nombre nulle ou petit de chevauchements ou coupures entre les segments des caractères, ce qui réduit la complexité en temps. Mais, quand le mot est mal écrit, les chevauchements ou coupures vont être nombreuse, de même le nombre d'opérations de Fusion/Division et la complexité de temps va être considérable.

Conclusion

Dans ce chapitre, nous avons commencé par illustrer l'architecture globale du système proposé. Dans laquelle nous avons présenté en détail la première contribution introduite dans chapitre 2, qui se présente en une méthode structurelle de segmentation multi-Scans permettant de préparer les segments de l'ensemble initiale du puzzle. En plus, nous avons vus les différents éléments de la deuxième contribution introduit dans chapitre 3, présentée par l'algorithme de post-traitement puzzle qui applique un ensemble d'opérations (Fusion, Division et/ou Morphisme) pour résoudre le problème de coupures et chevauchements dans un texte manuscrit cursive Arabe. Finalement, nous avons donné les résultats obtenus par les deux méthodes, ainsi qu'une validation en comparant nos résultats avec ceux d'autres travaux.

Conclusion et perspectives

L'objectif de nos travaux de thèse est l'amélioration du taux de reconnaissance d'un système hors-ligne de reconnaissance de l'écriture manuscrite en intégrant l'algorithme de puzzle en post-traitement.

Un texte arabe manuscrit est composé d'un ensemble de caractères qui possèdent une représentation irrégulière, ce qui génère des fluctuations dans les lignes, des espaces irréguliers intra et inter-mots et des chevauchements, caractères mal représentés et des coupures entre les mots et même entre les caractères d'un mot. L'image d'un texte peut être vue comme un jigsaw puzzle à résoudre, où nous devons ajuster les segments afin de restaurer le texte original, et par conséquent, résoudre le problème des coupures, chevauchements et, caractères mal représentés.

Nous avons dégagé de cette étude deux contributions qui peuvent profiter des capacités des algorithmes puzzles en feedback pour surmonter le problème d'irrégularité dans l'écriture cursive manuscrite.

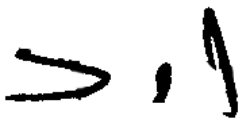
Le problème des coupures ne peut être résolut qu'avec l'utilisation d'une opération de fusion afin de fusionner les différents segments. Par contre, le problème de chevauchements nécessite une opération de division pour qu'il soit résolu. Autant que le problème de caractères mal représentés implique l'utilisation d'un ensemble de règles de morphisme permettant de changer complètement la nature du caractère dans le but de restaurer le caractère original.

La première contribution, se présente par une méthode de segmentation structurelle multi-scans (N-Scans) qui a pour rôle de créer le premier ensemble des segments représentant par la suite l'état initial du jigsaw puzzle. Autant que la deuxième est présentée par un post-processeur puzzle qui possède trois étapes principales : Fusion, Division et Morphisme. En appliquant ces étapes sur l'ensemble initial du puzzle le post-processeur est capable de régénérer des autres sous-ensembles. Ce processus permet

de surmonter de manière remarquable les problèmes posés par l'écriture manuscrite en achevant une performance importante de 90.03% de segmentation et un taux de reconnaissance de 96.93%. Les résultats présentés dans la section 4.2 prouve que la performance du système de reconnaissance peut être améliorée par l'utilisation d'un contrôleur puzzle parce que les résultats de segmentation et les décisions de classificateur SVM sont guidés par l'état objectif du puzzle (les mots modèles de la base) et jugés par le contrôleur du puzzle pour atteindre le meilleur état objectif.

Les perspectives présentées dans cette thèse sont nombreuses. L'une des perspectives les plus prometteuses est l'utilisation des étapes du puzzling (Fusion, Division et Morphisme) ou les chevauchements, coupures des segments des caractères et les caractères mal représentés comme une métrique d'évaluation de la qualité d'une écriture, ou dans l'identification des personnes ou de ses caractères à partir d'écriture (graphologie). Une autre perspective très intéressante consiste à ajouter la détection des points de croisement en utilisant un masque similaire au masque de détection des caractères de nature verticale (voir la section 2.6.2). Ces points vont permettre de surmonter le problème de division de mots avec des caractères complètement enchevêtrés. Par exemple, dans le cas du mot **الحجام** dans l'image  référencée par af03_046, il est impossible de segmenter les caractères ا, ل, ح par l'application d'une opération de division en se basant sur les points de fusion et de division précédemment indiqués. L'ensemble des points de croisements permet de générer d'autres sous-segments (la figure ci-dessous (A)) qui contiennent les caractères corrects.

(A) Segments de caractères corrects



(B) Segments non corrects

Une autre piste d'amélioration intéressante consiste à utiliser une heuristique adaptative tel que PSO afin de rendre le système capable de générer de nouvelles règles permettant d'améliorer l'ensemble du puzzle. Une autre piste non moins intéressante consisterait à étendre le travail de cette thèse à la reconnaissance des écritures manuscrites artistiques telles que la montre la figure ci-dessous.

Bibliographie

[1] G. Abandah et N. Anssari – "Novel moment features extraction for recognizing handwritten arabic letters", *Journal of Computer Science* **5** (2009), no. 3, p. 226.

[2] G. A. Abandah, F. T. Jamour et E. A. Qaralleh – "Recognizing handwritten arabic words using grapheme segmentation and recurrent neural networks", *International Journal on Document Analysis and Recognition (IJDAR)* **17** (2014), no. 3, p. 275–291.

[3] R. M. Abdeen, A. Afifi et A. B. El-Sisi – "Improved arabic handwriting word segmentation approach using random forests", *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, Nov 2015, p. 1–8.

[4] S. Abdulla, A. Al-Nassiri et R. A. Salam – "Off-line arabic handwritten word segmentation using rotational invariant segments features.", *Int. Arab J. Inf. Technol.* **5** (2008), no. 2, p. 200–208.

[5] S. Abe – *Support vector machines for pattern classification*, 2nd éd., Springer Verlag, 2010.

[6] A. A. Aburas et S. M. Rehiel – "Off-line omni-style handwriting arabic character recognition system based on wavelet compression", *Arab Research Institute in Sciences & Engineering* **3** (2007), p. 123–135.

[7] M. M. Adankon et M. Cheriet – "Model selection for the ls-svm. application to handwriting recognition", *Pattern Recognition* **42** (2009), no. 12, p. 3264–3270.

[8] S. Al-Ma'adeed, D. Elliman et C. A. Higgins – "A data base for arabic handwritten text recognition research", *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, IEEE, 2002, p. 485–489.

[9] H. Al-Rashaideh – "Preprocessing phase for Arabic Word Handwritten Recognition", *INFORMATION TRANSMISSIONS IN COMPUTER NETWORKS* **6** (2006), p. 11–19.

[10] A. Alaei, P. Nagabhushan et U. Pal – "Piece-wise painting technique for line segmentation of unconstrained handwritten text: a specific study with persian text documents", *Pattern Analysis and Applications* **14** (2011), no. 4, p. 381–394.

[11] J. H. AlKhateeb, J. Jiang, J. Ren et S. Ipson – "Component-based segmentation of words from handwritten arabic text", *International Journal of Computer Systems Science and Engineering* **5** (2009), no. 1, p. 54–58.

[12] J. H. AlKhateeb, J. Ren, J. Jiang et H. Al-Muhtaseb – "Offline handwritten arabic cursive text recognition using hidden markov models and re-ranking", *Pattern Recognition Letters* **32** (2011), no. 8, p. 1081–1088.

[13] J. AlKhateeb H, J. Ren, J. Jiang et H. Al-Muhtaseb – "Offline handwritten arabic cursive text recognition using hidden markov models and re-ranking", *Pattern Recognition Letters* **32** (2011), no. 8, p. 1081–1088.

[14] S. Alma'adeed, C. Higgens et D. Elliman – "Recognition of off-line handwritten arabic words using hidden markov model approach", *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3, IEEE, 2002, p. 481–484.

[15] A. J. Alnsour et L. M. Alzoubady – "Arabic handwritten characters recognized by neocognitron artificial neural network", *University of Sharjah Journal of Pure & Applied Sciences* **3** (2006), no. 2, p. 54–58.

[16] A. M. Andrew – "An introduction to support vector machines and other kernel-based learning methods", *Kybernetes* **30** (2001), no. 1, p. 103–115.

[17] N. Aouadi, S. Amiri et A. K. Echi – "Segmentation of connected components in arabic handwritten documents", *Procedia Technology* **10** (2013), p. 738–746.

[18] N. Aouadi et A. Kacem – "A proposal for touching component segmentation in arabic manuscripts", *Pattern Analysis and Applications* (2016), p. 1–23.

[19] N. Ayat, M. Cheriet et C. Suen – "Automatic model selection for the optimization of SVM kernels", *Pattern Recognition* **38** (2005), no. 10, p. 1733 – 1745.

[20] N. Azizi, N. Farah, M. Sellami et A. Ennaji – "Using diversity in classifier set selection for arabic handwritten recognition", Multiple Classifier Systems, Springer, 2010, p. 235–244.

[21] C. Bahlmann, B. Haasdonk et H. Burkhardt – "Online handwriting recognition with support vector machines-a kernel approach", *Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on*, IEEE, 2002, p. 49–54.

[22] A. Belaïd – "Reconnaissance automatique de l'écriture et du document", *Pour la science* (2001), p. 22 p, Article dans une revue de vulgarisation.

[23] A. Benouareth, A. Ennaji et M. Sellami – "Semi-continuous hmms with explicit state duration for unconstrained arabic word modeling and recognition", *Pattern Recognition Letters* **29** (2008), no. 12, p. 1742–1752.

[24] A. Benzenache, H. Seridi et H. Akdag – "Features modelling in discrete and continuous hidden markov models for handwritten arabic words recognition", *The International Arab Journal of Information Tdchnology* (2015).

[25] A. Boukharouba et A. Bennia – "Reconnaissance de caractères imprimés omni-fonte", *3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, Tunisia*, 2005.

[26] O. Bousquet – "Introduction au support vector machines (svm)", *Center mathematics applied, polytechnique school of Palaiseau. http://www. math. u-psud. fr/˜ blanchard/gtsvm/index. html* (2001).

[27] M. Brand – "No easy puzzles: Hardness results for jigsaw puzzles", *Theoretical Computer Science* **586** (2015), p. 2–11.

[28] D. Brodić et Z. Milivojević – "A new approach to water flow algorithm for text line segmentation", *Journal of Universal Computer Science* **17** (2011), no. 1, p. 30–47.

[29] T. Burger, Y. Kessentini et T. Paquet – "Dempster-shafer based rejection strategy for handwritten word recognition", *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, Sept 2011, p. 528–532.

[30] J. CALLUT – "Implémentation efficace des support vector machines pour la classification", Thèse, Université libre de Bruxelles, 2003.

[31] C.-C. Chang et C.-J. Lin – "LIBSVM: a Library for Support Vector Machines", Tech. report, Department of Computer Science, National Taiwan University, September 2006.

[32] Z. Chen, J. Li, L. Wei, W. Xu et Y. Shi – "Multiple-kernel SVM based multiple-task oriented data mining system for gene expression data analysis", *Expert Systems with Applications* **38** (2011), no. 10, p. 12151 – 12159.

[33] C. K. Cheng et M. Blumenstein – "The neural-based segmentation of cursive words using enhanced heuristics", *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, IEEE, 2005, p. 650–654.

[34] C. K. Cheng, X. Y. Liu, M. Blumenstein et V. Muthukkumarasamy – "Enhancing neural confidence-based segmentation for cursive handwriting recognition", *5th International Conference on Simulated Evolution and Learning, Busan, Korea, SWA-8*, 2004.

[35] S. Chevalier, M. Lemaitre, E. Geoffrois, E. Grosicki et F. Preteux – "Etude de primitives spectrales pour la reconnaissance de caractères manuscrits dans le cadre d'une approche markovienne 2d", *Actes 15ème Congrès Francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle (RFIA'2006)*, 2006.

[36] R. Chitrakar et C. Huang – "Selection of candidate support vectors in incremental SVM for network intrusion detection", *Computers & Security* **45** (2014), p. 231 – 241.

[37] A. Cornuéjols – "Une nouvelle méthode d'apprentissage: Les svm. séparateurs à vaste marge", *Bulletin de l'AFIA* **51** (2002), p. 14–23.

[38] M. Coté – "Utilisation d'un modèle d'accès lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs", Thèse, Ecole Nationale Supérieur de Télecommunications, France, Juin 1997.

[39] A. Deneche – *Approches bio-inspirées pour la reconnaissance de formes*, Mémoire, Université Mentouri de Constantine, 2006.

[40] P. Du, K. Tan et X. Xing – "Wavelet SVM in reproducing kernel hilbert space for hyperspectral remote sensing image classification", *Optics Communications* **283** (2010), no. 24, p. 4978–4984.

[41] H. El Abed et V. Märgner – "Improvement of arabic handwriting recognition systems; combination and/or reject?", *IS&T/SPIE Electronic Imaging*, International Society for Optics and Photonics, 2009, p. 72470A–72470A.

[42] H. E. El Abed et V. Märgner – "Comparison of different preprocessing and feature extraction methods for offline recognition of handwritten arabicwords", *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, IEEE, 2007, p. 974–978.

[43] M. Elleuch, R. Mokni et M. Kherallah – "Offline arabic handwritten recognition system with dropout applied in deep networks based-svms", *Neural Networks (IJCNN), 2016 International Joint Conference on*, IEEE, 2016, p. 3241–3248.

[44] M. Elzobi, A. Al-Hamadi, Z. Al Aghbari et L. Dings – "Iesk-ardb: a database for handwritten arabic and an optimized topological segmentation approach", *International Journal on Document Analysis and Recognition (IJDAR)* **16** (2013), no. 3, p. 295–308.

[45] R. Entezari-Maleki, A. Rezaei et B. Minaei-Bidgoli – "Comparison of classification methods based on the type of attributes and sample size", *Journal of Convergence Information Technology* **4** (2009), no. 3, p. 94–102.

[46] Z. Faouzi, D. Abdelhamid et B. M. Chaouki – "An approach based on structural segmentation for the recognition of arabic handwriting", *Advances in Information Sciences and Service Sciences* **2** (2010), no. 4, p. 314–325.

[47] Z. Faouzi, D. Abdelhamid et B. Mohamed Chaouki – "Methode structurelle a deux passes: verticale et horizontale pour la segmentation du texte arabe manuscrit", *1st International Symposium of Informatics and its Applications ISIA*, Msila University, Algeria, 2014.

[48] M. Farag – "Handwrittentext recognition system for automatic reading of historical arabic manuscripts", *International Journal of Computer Applications* **60** (2012), no. 13.

[49] N. Farah, L. Souici et M. Sellami – "Arabic word recognition by classifiers and context", *Journal of Computer Science and Technology* **20** (2005), no. 3, p. 402–410.

[50] R. Farrahi Moghaddam, M. Cheriet, M. M. Adankon, K. Filonenko et R. Wisnovsky – "Ibn sina: a database for research on processing and understanding of arabic manuscripts images", *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, ACM, 2010, p. 11–18.

[51] V. Gaël, M. Andreas et J. Grobler – "Classifier comparison", http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html, 2010 (visité 25 Février, 2017).

[52] P. Gallinari, H. Zaragoza et M. Amini – "Apprentissage et données textuelles", *LIP6, Université Paris* **6** (2000), no. 4.

[53] Y. Gherghout et L. Souici-Meslati – "Reconnaissance de caractères arabes manuscrits par les séparateurs à vastes marges (svm)", *Proceedings of the International Conference on Applied Informatics (ICAI), Bordj Bou Arreridg-Algeria*, vol. 3, 2009, p. 220–223.

[54] A. J. H, J. Jianmin, R. Jinchang et I. S – "Component-based segmentation of words from handwritten arabic text", *International Journal of Computer Systems Science and Engineering* **5** (2009), no. 1.

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

[56] S. HAITAAMAR – *segmentation de texte en caractère pour le reconnaissance optique de l'écriture arabe*, Mémoire, Université EL-HADJ LAKHDHAR, Batna, Juillet 2007.

[57] M. Hasan et F. Boris – "Svm: Machines à vecteurs de support ou séparateurs à vastes marges", *Rapport technique, Versailles St Quentin, France. Cité* (2006), p. 64.

[58] X. He, G. Mourot, D. Maquin, J. Ragot, P. Beauseroy, A. Smolarz et E. Grall-Maës – "Multi-task learning with one-class SVM", *Neurocomputing* **133** (2014), p. 416 – 426.

[59] K. Igwe, N. Pillay et C. Rae – "Solving the 8-puzzle problem using genetic programming", *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, ACM, 2013, p. 64–67.

[60] S. M. Ismail[1] et S. Abdullah – "Geometrical-matrix feature extraction for on-line handwritten characters recognition", *Journal of Theoretical and Applied Information Technology* **l4** (2013), no. 1.

[61] I. A. Jannoud – "Automatic arabic hand written text recognition system", *American Journal of Applied Sciences* **4** (2007), no. 11, p. 857–864.

[62] S.-Y. Jin, S. Lee, N. A. Azis et H.-J. Choi – "Jigsaw puzzle image retrieval via pairwise compatibility measurement", *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*, IEEE, 2014, p. 123–127.

[63] A. Kacem, N. Aouïti et A. Belaïd – "Structural features extraction for handwritten arabic personal names recognition", *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, IEEE, 2012, p. 268–273.

[64] K. Kamei et M. Nakano – "An approach to search the solution of a puzzle game by particle swarm optimization", *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on*, IEEE, 2014, p. 75–80.

[65] Y. Kessentini, T. Burger et T. Paquet – "A dempster–shafer theory based combination of handwriting recognition systems with multiple rejection strategies", *Pattern Recognition* **48** (2015), no. 2, p. 534–544.

[66] Y. Kessentini, T. Paquet et T. Burger – "Comparaison des méthodes probabilistes et évidentielles de fusion de classifieurs pour la reconnaissance de mots manuscrits", *CIFED (to appear, 2010)* (2010).

[67] D. Ketata et M. Khemakhem – "Un survol sur l'analyse et la reconnaissance de documents : imprimé, ancien et manuscrit", *Colloque International Francophone sur l'Ecrit et le Document (CIFED2010)* (sousse, Tunisia), Mars 2010, p. 12pages.

[68] A. R. Khan et Z. Mohammad – "A simple segmentation approach for unconstrained cursive handwritten words in conjunction with the neural network", *International Journal of Image Processing* **2** (2008), no. 3, p. 29–35.

[69] M. Khayyat, L. Lam et C. Y. Suen – "Learning-based word spotting system for arabic handwritten documents", *Pattern Recognition* **47** (2014), no. 3, p. 1021–1030.

[70] J. Kumar, L. Kang, D. Doermann et W. Abd-Almageed – "Segmentation of handwritten textlines in presence of touching components", *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, IEEE, 2011, p. 109–113.

[71] M. Kumar, M. Jindal et R. Sharma – "k-nearest neighbor based offline handwritten gurmukhi character recognition", *Image Information Processing (ICIIP), 2011 International Conference on*, IEEE, 2011, p. 1–4.

[72] T. Lebore – "Segmentation d'image application aux documents anciens", *Mémoire de Master de recherche, Université de Nante, France* (2007).

[73] C. Leila, K. Maâmar et C. Salim – "Combining neural networks for arabic handwriting recognition", *2011 10th International Symposium on Programming and Systems*, April 2011, p. 74–79.

[74] Y. Leydier, F. Lebourgeois et H. Emptoz – "Text search for medieval manuscript images", *Pattern Recognition* **40** (2007), no. 12, p. 3552–3567.

[75] P. Mahé – "Noyaux pour graphes et support vector machines pour le criblage virtuel de molécules", *Rapport de stage, DEA MVA* **2003** (2002).

[76] S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, M. Alshayeb, M. T. Parvez, V. Märgner et G. A. Fink – "Khatt: An open arabic offline handwritten text database", *Pattern Recognition* **47** (2014), no. 3, p. 1096 – 1112, Handwriting Recognition and other {PR} Applications.

[77] G. Mehul, P. Ankita, D. Namrata, G. Rahul et S. Sheth – "Text-based image segmentation methodology", *Procedia Technology* **14** (2014), p. 465–472.

[78] F. Menasri – "Contributions à la reconnaissance de l'ecriture arabe manuscrite", *UNIVERSITE PARIS DESCARTES, Thèse de doctorat* (2008).

[79] D. Meyer et F. T. Wien – "Support vector machines", *The Interface to libsvm in package e1071* (2014).

[80] R. F. Moghaddam et M. Cheriet – "Application of multi-level classifiers and clustering for automatic word spotting in historical document images", *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, IEEE, 2009, p. 511–515.

[81] S. Mohamed – *Traitement d'images par des approches bio-inspirées application à la segmentation d'images*, Mémoire, Université Constantine 2, 2014.

[82] S. Mozaffari, H. El Abed, V. Märgner, K. Faez et A. Amirshahi – "Ifn/farsi-database: a database of farsi handwritten city names", *International Conference on Frontiers in Handwriting Recognition*, 2008.

[83] S. Nebti – "Reconnaissance de caractères manuscrits par intelligence collective", Thèse, Université Ferhat Abbas de Sétif 1, 2013.

[84] S. Nebti et A. Boukerram – "Use of nature-inspired meta-heuristics for handwritten digits recognition", *Int J Comput Linguist Res* **1** (2010), no. 1.

[85] — , "Handwritten characters recognition based on nature-inspired computing and neuro-evolution", *Applied intelligence* (2013), p. 1–14.

[86] M. Nemissi, H. Seridi et H. Akdag – "One-against-all and one-against-one based neuro-fuzzy classifiers", *Journal of Intelligent & Fuzzy Systems* **26** (2014), no. 6, p. 2661–2670.

[87] T. R. Nielsen, P. Drewsen et K. Hansen – "Solving jigsaw puzzles using image features", *Pattern Recognition Letters* **29** (2008), no. 14, p. 1924–1933.

[88] H. Oulhadj, J. Lemoine, E. Petit et H. Wehbi – "Combinaison d'algorithmes pour la reconnaissance des chiffres et des lettres bâtons dans un environnement multiscripteur d'écriture mixte.", *Vision Interface'99 VI'99*, 1999.

[89] J. Park – "Hierarchical character recognition and its use in handwritten word/phrase recognition", Thèse, State University of New York at Buffalo, 1999.

[90] M. T. Parvez et S. A. Mahmoud – "Offline arabic handwritten text recognition: A survey", *ACM Comput. Surv.* **45** (2013), no. 2, p. 23:1–23:35.

[91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay – "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research* **12** (2011), p. 2825–2830.

[92] S. Quiniou – "Intégration de connaissances linguistiques pour la reconnaissance de textes manuscrits en-ligne", Thèse, INSA de Rennes, 2007.

[93] J. Ramdan, K. Omar, M. Faidzul et A. Mady – "Arabic handwriting data base for text recognition", *Procedia Technology* **11** (2013), p. 580 – 584.

[94] J. Ramel – "Lecture automatique de partitions musicales", *Mémoire de DEA ingénierie informatique, LISPI-Equipe de Reconnaissance des Formes et Diagnostics, Université Lyon* **1** (1993).

[95] Z. Razak, K. Zulkiflee, M. Y. I. Idris, E. M. Tamil, M. N. M. Noor, R. Salleh, M. Yaakob, Z. M. Yusof et M. Yaacob – "Off-line handwriting text line segmentation: A review", *International journal of computer science and network security* **8** (2008), no. 7, p. 12–20.

[96] L. ROBADEY – "2(crem): Une méthode de reconnaissance structurelle de documents complexes basée sur des patterns bidimensionnels", Thèse, Institut d'informatique de l'Université de Fribourg, Suisse, 2001.

[97] J. Rodriguez-Serrano, F. Perronnin et al. – "A model-based sequence similarity with application to handwritten word spotting", *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **34** (2012), no. 11, p. 2108–2120.

[98] E. S et A.-Z. M.A – "A three stages segmentation model for higher accurate off-line arabic handwritting recognition", *WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION* **2** (2012), p. 98–104.

[99] R. Saabni et A. Bronstein – "Fast key-word searching via embedding and active-dtw", *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, IEEE, 2011, p. 68–72.

[100] T. Saba, A. S. Almazyad et A. Rehman – "Language independent rule based classification of printed & handwritten text", *Evolving and Adaptive Intelligent Systems (EAIS), 2015 IEEE International Conference on*, IEEE, 2015, p. 1–4.

[101] H. Sajedi – "Handwriting recognition of digits, signs, and numerical strings in persian", *Computers & Electrical Engineering* **49** (2016), p. 52–65.

[102] H. Salami et D. Giveki – "Farsi/arabic handwritten digit recognition based on ensemble of svd classifiers and reliable multi-phase pso combination rule", *International Journal on Document Analysis and Recognition (IJDAR)* **16** (2013), no. 4, p. 371–386.

[103] A. Salvail-Bérard – "Réseaux de neurones", (2012).

[104] F. B. SAMOUD, S. S. MADDOURI et K. HAMROUNI – "Segmentation de chèques bancaires arabes", *3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, Tunisia*, 2005.

[105] T. Sari et A. Kefali – "A search engine for arabic documents", *Colloque International Francophone sur l'Ecrit et le Document*, Groupe de Recherche en Communication Ecrite, 2008, p. 97–102.

[106] T. Sari et M. Sellami – "Morpho-lexical analysis for correcting ocr-generated arabic words (molex)", *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, IEEE, 2002, p. 461–466.

[107] T. Sari, L. Souici et M. Sellami – "Off-line handwritten arabic character segmentation algorithm: Acsa", *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, IEEE, 2002, p. 452–457.

[108] H. Schahrazed – *Segmentation de textes en caracteres pour la reconnaissance optique de l'ecriture arabe*, Mémoire, Université El-Hadj Lakhdar Batna, 2007.

[109] S. Shelke et S. Apte – "A fuzzy based classification scheme for unconstrained handwritten devanagari character recognition", *Communication, Information & Computing Technology (ICCICT), 2015 International Conference on*, IEEE, 2015, p. 1–6.

[110] F. Slimane, S. Kanoun, J. Hennebert, A. M. Alimi et R. Ingold – "Modèles de markov cachés et modèle de longueur pour la reconnaissance de l'ecriture arabe à basse résolution", *Proceedings de Manifestation des Jeunes Chercheurs en Sciences et Technologies de l'Information et de la Communication (MajecSTIC), Avignon-France* (2009).

[111] P. Smrž, M. Martinásek et al. – "Offine recognition of cursive handwritten czech text", *SOFSEM'98: Theory and Practice of Informatics*, Springer, 1998, p. 437–442.

[112] A. Soudi, G. Neumann et A. Van den Bosch – *Arabic computational morphology: knowledge-based and empirical methods*, Springer, 2007.

[113] S. Srihari, H. Srinivasan, P. Babu et C. Bhole – "Handwritten arabic word spotting using the cedarabic document analysis system", *Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, 2005, p. 123–132.

[114] X. Tian, G. Gasso et S. Canu – "A multiple kernel framework for inductive semi-supervised SVM learning", *Neurocomputing* **90** (2012), p. 46 – 58, Advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011).

[115] N. Tripathy et U. Pal – "Handwriting segmentation of unconstrained oriya text", *Sadhana* **31** (2006), no. 6, p. 755–769.

[116] C.-W. Tsai, S.-P. Tseng, M.-C. Chiang et C.-S. Yang – "A high performance algorithm for puzzle reconstruction problem", *Machine Learning and Cybernetics (ICMLC), 2012 International Conference on*, vol. 5, IEEE, 2012, p. 1698–1703.

[117] K. Vervier – "Doctorat européen paristech these", Thèse, MINES ParisTech, 2015.

[118] C.-D. Wang, J.-H. Lai, C. Suen et J.-Y. Zhu – "Multi-exemplar affinity propagation", *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35** (2013), no. 9, p. 2223–2237.

[119] L. Wang – *Support vector machines: theory and applications*, vol. 177, Springer Science & Business Media, 2005.

[120] M. Younes et Y. Abdellah – "Segmentation of arabic handwritten text to lines", *Procedia Computer Science* **73** (2015), p. 115–121.

[121] F. Zaiz, M. C. Babahenini et A. Djeffal – "Puzzle based system for improving arabic handwriting recognition", *Engineering Applications of Artificial Intelligence* **56** (2016), p. 222 – 229.

[122] X. Zhou, W. Jiang, Y. Tian et Y. Shi – "Kernel subclass convex hull sample selection method for SVM on face recognition", *Neurocomputing* **73** (2010), no. 10–12, p. 2234 – 2246, Subspace Learning / Selected papers from the European Symposium on Time Series Prediction.

[123] M. Ziaratban, K. Faez et F. Bagheri – "Fht: An unconstraint farsi handwritten text database", *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, IEEE, 2009, p. 281–285.

Résumé

Jusqu'aujourd'hui, la reconnaissance de caractères joue un rôle important dans plusieurs domaines tels que l'authentification de chèques bancaires, l'échange à distance des fichiers informatiques pour les télécommunications et l'authentification et l'identification des manuscrits. Elle permet de convertir une image de texte imprimée ou manuscrit en un texte codé par machine.

Nombreuses recherches ont été faites durant les dernières années afin d'améliorer le taux de reconnaissance des systèmes de reconnaissance du texte manuscrit Arabe. Plusieurs systèmes ont essayé d'utiliser divers techniques de post-traitement pour la sélection de mot telles que des techniques de votes et d'information contextuelles, ... etc.

Dans nos travaux précédents, nous avons proposé une technique basée sur le classificateur SVM pour la reconnaissance du manuscrit Arabe en se basant sur une méthode de segmentation à deux passes horizontale et verticale.

Dans ce travail, nous améliorons la technique de segmentation en intégrant une technique multi-scans (N-Scans) qui consiste à remplacer la segmentation en deux passes. En plus, nous ajoutons un algorithme du puzzle qui améliore le taux de reconnaissance surtout pour les caractères ambigus.

L'approche est testée sur la base du manuscrit Arabe IFN-ENIT. Elle donne des résultats encourageants et ouvre des perspectives dans le domaine de la reconnaissance du manuscrit Arabe.

Mots Clés: OCR, Manuscrit Arabe, PAW, SVMs, Puzzle.