Reconnaissance de mots manuscrits cursifs par modèles de Markov cachés en contexte : application au français, à l'anglais et à l'arabe

Anne-Laure Bianne Bernard

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Signal et Images »

présentée et soutenue publiquement par

Anne-Laure BIANNE BERNARD

le 21 Novembre 2011

Reconnaissance de mots manuscrits cursifs par modèles de Markov cachés en contexte

Application au français, à l'anglais et à l'arabe

Directeurs de thèse :	**Laurence LIKFORMAN-SULEM**
	Chafic MOKBEL
Co-encadrant de thèse :	**Christopher KERMORVANT**

Jury

Mme Nicole VINCENT,	Professeur, Labo. LIPADE, Université Paris Descartes	Examinateur
M. François YVON,	Professeur, groupe Trt. du Langage Parlé, Université Paris Sud	Président
M. Rolf INGOLD,	Professeur, groupe DIVA, Université de Fribourg	Rapporteur
M. Alessandro VINCIARELLI,	Lecturer, Dpt. Computing Science, Glasgow University	Rapporteur

T H È S E

Télécom ParisTech - Grande école de l'Institut Télécom - membre fondateur de ParisTech
46, rue Barrault - 75634 Paris Cedex 13 - Tél. + 33 (0)1 45 81 77 77 - www.telecom-paristech.fr

À Elizabeth, Scarlett et Gabrielle,
mes héroïnes.

À Laurent.

À Papa.

Remerciements

Je souhaiterais remercier Laurence Likforman et Christopher Kermorvant dont l'enthousiasme a motivé mon choix de suivre la voie du doctorat. Leur encadrement au fil des jours et leur foi en mon travail m'ont permis de toujours avancer et de vivre pleinement ces trois années. Chafic Mokbel a lui aussi largement participé à la réussite de cette thèse même s'il était plus loin géographiquement et je le remercie de tous les conseils qu'il a pu me livrer.

Je remercie vivement Alessandro Vinciarelli d'avoir accepté d'être rapporteur de ma thèse. Ses travaux ont toujours été une grande source d'inspiration pour moi et ses remarques sur mon manuscrit m'ont permis d'entrevoir de nouvelles perspectives de recherche exaltantes. Je remercie également Rolf Ingold d'avoir pris place aux côtés de M. Vinciarelli en tant que rapporteur. M. Ingold faisant partie de l'historique de la société A2iA, cela a été une grande fierté de pouvoir à mon tour l'associer à mes travaux de recherche.

Je remercie Nicole Vincent et François Yvon d'avoir accepté de faire partie de mon jury. Leurs questions et remarques lors de la présentation de mes travaux m'ont permis d'envisager des ouvertures enthousiasmantes pour mes travaux futurs.

Je voudrais remercier l'ensemble de mes collègues pour leur soutien ces dernières années. En particulier, Farès et Patrick qui m'ont continuellement encouragée et m'ont permis de toujours repousser mes propres limites (et d'améliorer ma précision au lancer de sucre). Merci aussi à Romain, au 1er étage et en général à tout A2iA pour l'ambiance unique de travail qu'ils créent et qui donne envie de se dépasser. Je souhaite aussi remercier tous mes collègues du laboratoire TSI de Télécom ParisTech, et en particulier Sarah et Emilie qui m'ont permis de connaître enfin les joies d'un environnement féminin au travail.

Le maître en CE2 l'avait dit, « *Cette petite, elle a du potentiel, il faut juste qu'elle l'exprime.* » Papa, Maman, Lulu, voilà qui est fait. Merci infiniment pour toute votre patience ces vingt-sept dernières années, merci pour vos encouragements et votre confiance en moi. Je n'y serais pas arrivée sans vous. Merci en particulier à Maman d'avoir tout relu et corrigé patiemment et à Lulu pour ses encouragements continuels. Et merci à toi Papa de m'avoir poussée dans cette voie. Je suis très fière de donner son premier titre de docteur à la famille, même si je pense que tu l'avais mérité avant moi. Merci aussi à Laurent (l'autre), Théo, Dominique, Jean, Pauline, Mathilde et Bonaventure de m'avoir accompagnée pendant ces trois années difficiles. Merci à tous mes amis de leur soutien indéfectible et merci à tous ceux qui, finalement, ont retenu l'intitulé de mes travaux.

Enfin, merci à toi, mon sel et ma lumière, qui me donne ce nom à rallonge et qui m'aime et me supporte chaque jour...

Anne-Laure

Résumé

La reconnaissance de l'écriture manuscrite est aujourd'hui un domaine de recherche très actif et le spectre de ses applications est très large. L'objectif de cette thèse est d'élaborer un système de reconnaissance de mots manuscrits pouvant être appris et appliqué sur différents styles d'écriture.

L'approche utilisée est une approche analytique : les mots sont découpés en sous-parties (caractères ou graphèmes) à modéliser. Le découpage est effectué de manière implicite par l'utilisation de fenêtres glissantes. Celles-ci permettent de transformer les images de mots en séquences. La méthode choisie pour apprendre les modèles de caractères utilise les modèles de Markov cachés (HMMs), qui sont à ce jour l'un des outils les plus puissants pour la modélisation de séquences. Chaque caractère est représenté par un HMM de type Bakis, ce qui permet d'absorber les variations d'écriture entre scripteurs. Les mots sont reconstruits ensuite par concaténation des modèles qui les composent.

Dans cette thèse, le choix est fait de chercher à améliorer la modélisation HMM de caractères en agissant au coeur même des modèles. A cette fin, une nouvelle approche est proposée, qui utilise l'aspect contextuel pour la modélisation : un caractère est modélisé en fonction de son contexte (le caractère précédent et le caractère suivant) et son modèle est nommé trigraphe.

La prise en compte de l'environnement d'un caractère pour sa modélisation permet de construire des modèles plus précis et plus performants. Cependant, elle implique une multiplication des paramètres HMMs à apprendre sur un nombre souvent restreint de données d'observation. Une méthode originale de regroupement de paramètres est proposée dans ces travaux : le clustering d'états par position à l'aide d'arbres binaires de décision. Ce type de clustering, inédit dans les systèmes de reconnaissance de l'écriture, a l'avantage non seulement de réduire le nombre de paramètres mais aussi de permettre au système de conserver l'un des principaux attraits des HMMs : l'utilisation d'un lexique de décodage indépendant du vocabulaire d'apprentissage.

L'amélioration apportée par la modélisation en contexte est montrée sur trois langues et deux types d'écriture différents : le français, l'anglais et l'arabe.

Abstract

Off-line handwriting recognition has become lately a very popular research area and the number of its possible applications is very large. This thesis aims at elaborating a new handwritten words recognition system that can be learned and applied on any handwriting style and any alphabet.

An analytic approach is used. Words are divided into subparts (characters or graphemes) that have to be modelled. The division is made implicitly thanks to sliding windows, which transform the word images into sequences. Hidden Markov Models, widely known as one of the most powerful tools for sequence modelling, are chosen to model the characters. A Bakis-type HMM represents each character. This enables the model to absorb variations in handwriting. A word model is built by concatenating its compound characters models.

In this thesis, the choice is made to strengthen the HMM modelling by acting directly within the models. To this end, a new approach is proposed, using context knowledge : each character model depends on its context (its preceding and following characters). This new character model is named trigraph.

Taking into account the characters environment allows more precise and more effective models to be built. However, this implies a multiplication of HMM parameters to be learned (often on a restricted number of observation data). An original method for parameter grouping is proposed in this thesis to overcome this issue : a state-based clustering, performed on each state position and based on binary decision trees. This type of clustering is new in the handwriting recognition field. It has many advantages, including parameter reduction. Moreover, the use of decision trees allows the HMMs to keep one of their most interesting attributes : independence between training and testing lexicon.

Context-dependent character modelling has shown to give an increase of performance. In this thesis, results are reported on three different databases and two different alphabets, latin and arabic.

Table des matières

Table des figures

Liste des tableaux

Introduction

La reconnaissance de l'écriture manuscrite connaît un regain d'activité depuis quelques années et est devenue un domaine de recherche très actif. Le spectre de ses applications est très large. Les logiciels de reconnaissance de montants sur les chèques ou de reconnaissance d'adresse postale en sont les deux applications historiques [60, 127, 128, 157, 30, 147, 16, 15]. Plus récemment, la reconnaissance automatique de l'écriture a permis par exemple aux grandes entreprises d'améliorer leur processus de traitement automatique de courrier entrant. Enfin, la reconnaissance d'écriture manuscrite est utilisée dans de plus en plus de projets de numérisation et d'analyse de documents historiques.

La multiplication de projets français, européens et internationaux ces dernières années pour mutualiser la recherche en traitement automatique de document est la preuve du dynamisme et de l'engouement suscité par ce domaine. Par exemple, le projet ANR Digidoc [138] a été lancé en 2011 à la suite du projet ANR Navidomass [120] (2007-2010) pour la valorisation du patrimoine écrit français. Ces deux projets sont inscrits dans la même optique que les initiatives internationales Europeana [73] et Gallica [69]. Tous ont pour but de numériser, analyser et mettre à disposision du grand public le contenu d'archives françaises et européennes. La tâche est d'autant plus ardue que le nombre de documents à traiter est colossal : par exemple dans le projet Gallica, $1,5$ millions de documents ont été numérisés, un document pouvant être entre autres un livre ou un manuscrit de plusieurs pages. La reconnaissance d'écriture sur documents anciens a d'ailleurs d'autres applications, par exemple la généalogie [72]. Mais les récents travaux dans le domaine ne se sont pas uniquement concentrés sur les documents historiques. Ainsi, des projets concernant la défense nationale ou le renseignement militaire tels le projet MADCAT (Etats-Unis) [74] pour la reconnaissance de l'écriture manuscrite arabe montrent que les applications de ce domaine de recherche sont très étendues.

L'état de l'art de la reconnaissance d'écriture manuscrite est en constante amé-

lioration depuis le milieu des années 2000. L'organisation fréquente de compétitions de systèmes de reconnaissance permet aux acteurs du domaine de se comparer régulièrement et de proposer des systèmes de plus en plus performants. Ainsi ont déjà eu lieu depuis 2005, entre autres, quatre compétitions de reconnaissance de mots isolés français (campagnes Rimes 2006 [4], 2008 [63], 2009 [64] et 2011), quatre compétitions de reconnaissance de noms de villes arabes (campagnes IFN-Enit 2005 [106], 2007 [104], 2009 [38] et 2010 [105]) et une compétition de reconnaissance de mots manuscrits arabes dans une base de très grande taille (compétition OpenHart 2010 [167], dans le cadre du projet MADCAT). Ces compétitions assurent le dynamisme de ce domaine de recherche et permettent d'élever progressivement les performances afin d'obtenir, à la fin, des logiciels fiables de reconnaissance d'écriture, adaptables à tous langages et à tous types de documents.

L'objectif de cette thèse est d'élaborer un système de reconnaissance de mots manuscrits pouvant être appris et appliqué sur différents styles d'écriture.

Le choix de reconnaître de mots isolés est motivé par le fait que la reconnaissance de lignes ou de paragraphes peut se baser sur un classifieur de mots grâce à une segmentation explicite en mots et à l'utilisation de modèles de langage. Ce point a été largement illustré dans les systèmes de reconnaissance de la parole [156, 17, 149] et les modèles de langage ont déjà prouvé qu'ils pouvaient être utilisés avec succès pour la reconnaissance de l'écrit [111, 108, 159].

Il existe deux approches pour la reconnaissance de mots manuscrits : l'approche globale et l'approche analytique. L'approche globale (ou approche holistique) consiste à apprendre et reconnaître un mot dans son ensemble. Souvent utilisée pour des tâches de reconnaissance à vocabulaire restreint, elle est peu robuste face à des tâches plus difficiles où le vocabulaire contient plus d'une centaine de mots. Nous n'avons pas envisagé l'utilisation de l'approche globale dans nos travaux car notre système de reconnaissance de mots s'inscrit dans un projet de traitement global de documents, nécessitant un vocabulaire de travail très grand. L'approche analytique quant à elle propose de segmenter le mot en sous-parties (caractères ou graphèmes) à modéliser. Les mots sont reconstruits ensuite par concaténation des modèles qui les composent : cela permet l'utilisation de lexiques libres, à condition qu'ils soient basés sur l'alphabet (ou les graphèmes) appris. Pour segmenter les images de mots, il est possible de procéder à un découpage explicite – sujet à erreurs – ou à une segmentation implicite. C'est cette dernière approche que nous avons choisie, illustrée par l'utilisation de fenêtres glissantes parcourant les images de mots dans le sens de

lecture afin de les transformer en séquences.

La méthode choisie dans notre système pour apprendre nos modèles de caractères utilise les modèles de Markov cachés [139] (en anglais *Hidden Markov Models*, HMMs). Leur utilisation se justifie par le fait que les HMMs sont à ce jour l'un des outils les plus puissants pour la modélisation de séquences. Chaque caractère est représenté par un HMM de type Bakis (modèle gauche-droit à saut d'état autorisé), ce qui permet d'absorber les variations de longueur des séquences rencontrées au fil des données : les scripteurs n'écrivent pas tous de la même manière ni avec la même amplitude, ainsi la segmentation implicite d'un mot peut associer entre 5 et 15 fenêtres glissantes à un même caractère, selon la personne qui a écrit et les conditions d'écriture. De plus, les HMMs profitent d'algorithmes d'apprentissage et de décodage très performants, largement éprouvés et optimisés au fil des années : algorithme EM (*Expectation-Maximisation*) ou Baum-Welch pour l'apprentissage [7] et, pour le décodage, algorithme de Viterbi par exemple [164]. Toutes ces raisons nous ont motivés à choisir les HMMs pour notre modélisation.

Afin d'être modélisés par le système HMM, les images de mots sont donc segmentées implicitement à l'aide de fenêtres glissantes. Dans chaque fenêtre, des valeurs numériques censées représenter mathématiquement le comportement des pixels et des formes présents dans la fenêtre sont extraites : ce sont les vecteurs de caractéristiques. L'ensemble de ces vecteurs forme les séquences d'observations modélisées par les HMMs de caractères. Les valeurs extraites peuvent représenter des caractéristiques haut niveau (présence de boucles, de jambages, de traits dans la fenêtre par exemple) ou bas niveau (statistiques des pixels : moyenne, variance, centre de gravité des pixels d'écriture de la fenêtre entre autres).

Récemment, les efforts se sont concentrés sur l'élaboration de caractéristiques plus fiables [39, 142, 19] et sur l'hybridation de HMMs avec d'autres modèles tels les des réseaux de neurones [141, 5, 44] entre autres. Peu de systèmes proposent d'agir au niveau même de la modélisation HMM, même si des améliorations ont été proposées comme l'adaptation de la topologie des HMMs à la longueur des caractères [171, 146], la modélisation de silences intra-mots (pour l'arabe [33]) ou encore l'apprentissage de modèles HMMs de bi-lettres [35].

Nous pensons que toutes les améliorations possibles au coeur de la modélisation HMM n'ont pas été explorées alors qu'il est certain qu'une modélisation plus précise et plus robuste permettrait aux systèmes HMMs de gagner en performance.

A cette fin, nous proposons dans nos travaux de recherche une nouvelle approche

de reconnaissance de mots manuscrits par HMMs avec l'utilisation de contextes pour la modélisation des caractères. Ainsi, nous faisons le choix de modéliser des caractères en fonction de leur voisinage, représenté pour chaque caractère par ses deux caractères adjacents (le caractère précédent et le caractère suivant). Nous nommons ces nouveaux modèles les trigraphes. La prise en compte de l'environnement d'un caractère pour sa modélisation nous permet de construire des modèles plus précis et plus performants. L'amélioration qu'ils apportent est montrée sur trois langues et deux types d'écriture différents : le français, l'anglais et l'arabe.

La présentation de cette thèse se déroule en cinq chapitres.

Dans le premier chapitre une introduction à la reconnaissance d'écriture manuscrite avec HMMs est proposée. Nous montrons d'abord comment envisager la reconnaissance de mot isolé dans le cadre plus large de traitement de document entier. Puis la transformation des images de mots est discutée, notamment l'extraction de caractéristiques avec une présentation de l'état de l'art des caractéristiques pouvant être extraites. Enfin, nous présentons en détail le fonctionnement des HMMs et proposons une étude comparative des systèmes de reconnaissance d'écriture manuscrite les utilisant.

Le deuxième chapitre présente notre système générique à base de HMMs (indépendants du contexte), optimisé pour la modélisation de caractères. L'extraction de caractéristiques que nous proposons mutualise les connaissances de l'état de l'art. Afin de rendre compte des liens entre les fenêtres glissantes consécutives, nous présentons l'intérêt du calcul d'une régression sur les vecteurs de caractéristiques pour la modélisation plus précise de caractères. Enfin, dans le souci de construire des modèles de caractères optimaux, nous proposons une manière originale d'adapter la topologie des HMMs à la longueur des caractères. Ce chapitre présente donc l'élaboration d'un système de reconnaissance de mots à partir de HMMs de caractères et propose des idées originales afin d'améliorer la modélisation.

Le troisième chapitre présente la principale contribution de cette thèse : la modélisation de caractères dépendants de leur contexte, que nous nommons les trigraphes. Ce chapitre part du système et des idées proposées au Chapitre 2 et poursuit les travaux de recherche vers une stratégie de modélisation de plus en plus précise. L'utilisation de modèles de trigraphes est d'abord discutée puis, constatant que cette modélisation est synonyme de multiplication de paramètres HMMs à calculer, une méthode de regroupement de paramètres est proposée : le clustering d'états par position à l'aide d'arbres binaires de décision. Ce type de clustering, inédit dans les

systèmes de reconnaissance de l'écriture, a l'avantage non seulement de réduire le nombre de paramètres (et donc de construire des modèles robustes) mais aussi de permettre au système de conserver l'un des principaux attraits des HMMs : l'utilisation d'un lexique de décodage indépendant du vocabulaire d'apprentissage.

Dans le quatrième chapitre, nous menons des expériences sur trois bases de données représentant trois langues, deux styles d'écriture et deux alphabets. A travers ces expériences, nous montrons que notre modélisation contextuelle non seulement améliore les performances d'un système générique de HMMs de caractères mais surtout qu'elle est généralisable à plusieurs langages et types de données. Ceci montre la robustesse de notre approche. Dans ce chapitre d'expériences, nous introduisons aussi deux méthodes de post-traitement pour améliorer les performances : la combinaison de systèmes de reconnaissance et l'utilisation de modèles de langage pour le décodage de lignes. Ces méthodes ont permis à notre système original à base de HMMs contextuels d'être présent en première ou en deuxième place des dernières compétitions internationales de reconnaissance d'écriture [64, 38, 74].

Dans le cinquième chapitre, nous introduisons le principe d'adaptation au scripteur et évaluons son influence sur notre système contextuel. L'adaptation au scripteur est l'une des approches les plus utilisées en reconnaissance de la parole pour améliorer les performances d'un reconnaisseur HMM. Nous verrons dans ce chapitre qu'elle a sa place en reconnaissance de l'écriture, à condition de disposer de données suffisantes en nombre pour s'adapter aux scripteurs.

Le document se termine par un chapitre de conclusion qui reporte les travaux de cette thèse et affirme l'intérêt de la modélisation contextuelle. Des perspectives sont données sur l'utilisation des HMMs pour la reconnaissance d'écriture manuscrite, notamment l'adaptation aux scripteurs. Les conclusions données dans ce chapitre nous permettent d'élaborer un plan de recherches pour nos travaux futurs.

Chapitre 1

Introduction à la reconnaissance de mots manuscrits

Introduction

Dans nos travaux, nous nous intéressons à la reconnaissance de mots manuscrits. Cette tâche s'inscrit au sein d'une plus grande problématique qui est la reconnaissance de documents numérisés en général, c'est-à-dire savoir décrire et transcrire toutes les informations contenues dans une image de document, composé d'une ou de plusieurs pages, contenant des graphiques, des paragraphes d'écriture et d'autres informations. Aujourd'hui, s'il est possible d'analyser la structure d'un document et de reconnaître l'écriture manuscrite et imprimée sur des images pré-segmentées, la construction d'un système global de traitement de document n'est pas encore un problème résolu.

Notre travail de reconnaissance de mots manuscrits s'inscrit dans le cadre de l'élaboration d'un système de traitement de documents générique et efficace. En construisant un classifieur de mots robuste, le chemin vers un reconnaisseur de lignes (et donc de paragraphes) est très proche : l'utilisation systématique de modèles de langage pour la reconnaissance de la parole par exemple montre que la reconnaissance de phrases est une application directe pour un classifieur de mots. La transcription des données contenues dans un document permet ensuite sa classification et facilite son traitement. Dans ce chapitre, nous présenterons comment passer d'un document manuscrit aux images de mots qui seront traitées par notre système (Section 1.1). Puis dans la Section 1.2 nous verrons comment transformer les images de

mots obtenues afin que le classifieur puisse les traiter.

Pour la construction d'un classifieur robuste, nous avons envisagé dans nos travaux une approche utilisant des modèles de Markov cachés (*Hidden Markov Models* en anglais, soit HMMs), où chaque caractère est modélisé par un HMM. L'avantage de cette approche est que les HMMs permettent d'absorber les disparités entre les données (un mot est écrit différemment selon le scripteur ou sa place dans la phrase). De plus, cette modélisation permet l'utilisation de lexiques libres, à partir du moment où ils sont basés sur l'alphabet appris. Nous discutons en Section 1.3 de l'état de l'art des classifieurs existant aujourd'hui basés sur les HMMs pour la reconnaissance de mots manuscrits. Ceci nous permet d'introduire notre approche, présentée dans les Chapitres 2 et 3.

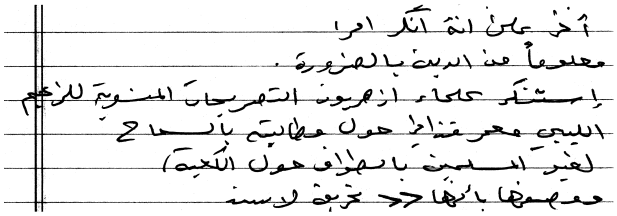
1.1 Le prétraitement des images

Les premières étapes d'un système de reconnaissance d'écriture manuscrite consistent le plus souvent à prétraiter les données. Les prétraitements effectués sur une image facilitent l'étape suivante d'extraction de caractéristiques. Ils permettent en outre d'améliorer significativement les résultats de reconnaissance. Une grande variété de prétraitements existent, par exemple :

- découpage d'une image de texte (courrier, page de livre, etc.) en lignes et/ou en mots (voir Section 1.1.2),
- binarisation (Section 1.1.3),
- extraction des lignes de base (Section 1.1.4),
- correction de la pente et de l'inclinaison des caractères (Section 1.1.5),
- normalisation de la taille des mots (Section 1.1.6).

Lorsque c'est nécessaire, ces prétraitements sont précédés ou accompagnés d'un nettoyage d'image : pour des images de documents historiques dont le fond est dégradé (taches, pages rongées ou vieillies) ou pour des images dont le fond n'est pas uniforme (page de cahier avec lignes ou quadrillage) par exemple (voir Figure 1.1). En général, le nettoyage est effectué par un filtrage de l'image : filtre passe-haut pour éliminer les taches ou les trous sur les documents (Feldbach [47]), filtre de Kalman pour éliminer des lignes (Likforman-Sulem [95]) ou encore techniques à base de filtres et d'ondelettes pour éliminer les caractères du verso visibles par transparence (Lamouche et Bellissant [88], Tann *et al.* [158]).

(a) extrait d'une page manuscrite ancienne de recensement français : le papier est rongé



(b) extrait d'une page manuscrite arabe sur un papier avec lignes et marge

FIGURE 1.1 – Exemples d'images dégradées nécessitant un nettoyage avant d'être prétraitées.

Dans le cadre de notre travail, nous utilisons des images propres (écriture sur fond blanc). Si nous n'avons pas besoin de nettoyer nos images, il nous faut cependant les transformer afin de pouvoir les traiter. Nous présentons donc dans cette Section les principaux prétraitements utilisés en reconnaissance de l'écriture. Certains d'entre eux sont utilisés pour notre tâche de reconnaissance de mots manuscrits.

1.1.1 Notations

Dans nos travaux de recherche, nous utilisons des images en niveau de gris. La valeur des pixels varie de 0 à 255. Plus la valeur est proche de zéro, plus le pixel est foncé et, inversement, plus la valeur est haute, plus le pixel est clair. Nous nommons *pixel inverse* du pixel pix le pixel de valeur $255 - pix$. Les pixels inverses permettent d'accentuer l'influence des pixels les plus foncés (pixels d'écriture, contenant l'information utile) et sont utilisés dans certains calculs des Sections suivantes.

Nous notons une image I. I est de taille n_l lignes et n_c colonnes. Un pixel sur la ligne j et la colonne i est noté $I(i, j)$. $i = 1$ représente la colonne de pixels la plus à gauche et $i = n_c$ la colonne la plus à droite. Parallèlement, $j = 1$ représente la rangée de pixels la plus haute de l'image et $j = n_l$ la rangée de pixels la plus basse.

1.1.2 De la page au mot

De manière générale, le but d'un système de reconnaissance d'écriture est, pour une image scannée de texte en entrée – que ce soit une page d'un livre ancien ou

bien un courrier envoyé à une entreprise – de retranscrire intégralement le texte écrit afin, ensuite, de pouvoir le traiter. Afin d'être traitées, les images sont analysées et découpées en blocs de texte, puis en mots ou en lignes, qui sont les entités utilisées par les systèmes de reconnaissance d'écriture.

Sur la Figure 1.2 sont représentées les différentes étapes pour passer d'une page manuscrite à des images de mots. La page est extraite de la base de données Rimes dont le scénario est celui du courrier entrant d'une entreprise. On peut constater que cette image contient plusieurs blocs de texte : l'adresse du client, la date, l'objet du courrier, la signature (encadrés en bleu) et le texte principal (encadré en rouge). Une première étape est donc de séparer ces blocs de texte afin de les traiter un par un. Suite à cette étape, on peut extraire les lignes des blocs détectés. Plusieurs algorithmes d'analyse de structure de document (*DLA : Document Layout Analysis*) existent à ce jour qui atteignent de bonnes performances pour la distinction de blocs. Un état de l'art se trouve dans Cattoni *et al.* [21] ou Mao *et al.* [103]. Certains algorithmes de DLA extraient même directement les lignes de texte de l'image sans passer par une découpe préalable de l'image en blocs (Li *et al.* [94], O'Gorman [125], Nagy *et al.* [117]). Pour les documents imprimés, l'utilisation d'histogrammes de projection est en général suffisante pour la détection de lignes (Plamondon et Srihari [132]). Pour l'écriture manuscrite, des méthodes à base de projection peuvent aussi être utilisées (Manmatha et Srimal [102], Marti et Bunke [108]) mais, lorsque l'orientation globale du texte n'est pas horizontale au départ, un prétraitement par transformation de Hough doit être appliqué avant le calcul des projections (Shapiro *et al.* [151]). La méthode de Hough [68] est d'ailleurs souvent utilisée pour la détection de lignes droites dans une image (Likforman-Sulem *et al.* [97]). D'autres méthodes peuvent encore être citées pour la détection de lignes, comme les méthodes de groupement de composants (Likforman-Sulem et Faure [96]) ou encore les méthodes à base d'étalement (en anglais *smearing methods*, voir Wong *et al.* [166], LeBourgeois *et al.* [90]). Les difficultés de la détection de lignes dans un document manuscrit sont clairement explicitées dans Likforman-Sulem *et al.* [98], où l'on voit d'ailleurs que la liberté d'écriture dans un système cursif est souvent synonyme de lignes non horizontales, voire courbes. Récemment, des compétitions de segmentation de texte en lignes ont été conduites (Gatos *et al.* [53]) ; les excellents résultats de certains systèmes (Shi *et al.* [152, 153]), y compris sur des images difficiles, montrent que les méthodes de l'état de l'art sont aujourd'hui compétentes pour régler cette tâche.

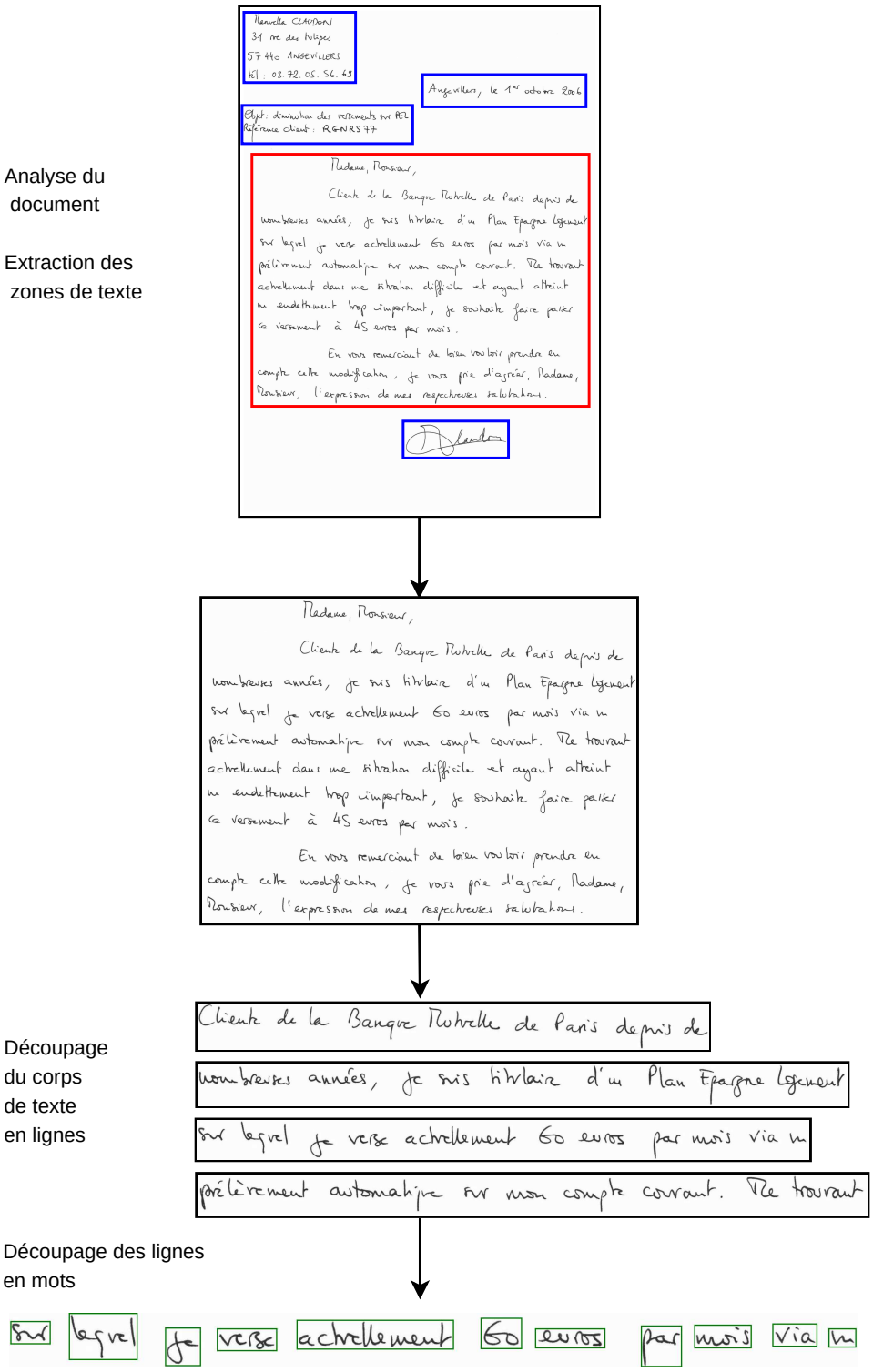


FIGURE 1.2 – Illustration des étapes pour passer d'une page manuscrite à des images de mots.

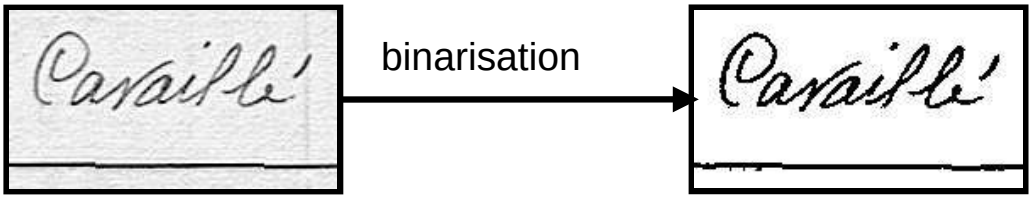


FIGURE 1.3 – Illustration de la binarisation d'une image en niveaux de gris.

Le découpage d'une ligne en mots peut ensuite se faire de plusieurs façons et, dans certains systèmes de reconnaissance d'écriture, les lignes peuvent même être directement utilisées en entrée (Natarajan *et al.* [118], Vinciarelli *et al.* [159]). Notre système est construit pour la reconnaissance de mots. Cette approche a été motivée par le fait que nous souhaitions construire d'abord un reconnaisseur robuste, et que cela commence avec la reconnaissance de mots isolés. De plus, le grand nombre de bases de données d'images de mots disponibles et référentes pour l'état de l'art des systèmes de reconnaissance d'écriture manuscrites nous ont permis de nous assurer la robustesse de notre reconnaisseur. Nous verrons cependant au Chapitre 4 que l'utilisation de modèles de langages lui permet aussi de reconnaitre des lignes.

La projection horizontale des pixels de la ligne ou bien le regroupement des composantes connexes (un regroupement = un mot) sont les deux méthodes les plus utilisées aujourd'hui pour la découpe d'une ligne en mots (Marti et Bunke [109]). D'autres méthodes existent cependant mais leur description n'est pas l'objet de ce travail. Une récente compétition de segmentation de lignes de texte en mots (Gatos *et al.* [53]) permet d'avoir un bon aperçu de l'état de l'art actuel des méthodes de segmentation de lignes.

On peut donc découper une image de texte manuscrit libre en images de mots, afin de traiter celles-ci directement avec le classifieur. Nous décrivons dans les Sections suivantes (1.1.3 à 1.1.6) quelles normalisations peuvent être appliquées lors de la découpe ou bien directement sur les images de mots.

1.1.3 Binarisation

Binariser une image en niveaux de gris consiste à la transformer en une image en noir et blanc en associant à chaque pixel un label 0 ou 1 (0 pour blanc et 1 pour noir). Ceci est illustré sur la Figure 1.3.

On peut distinguer deux types de binarisation d'images, globale et locale. La binarisation globale trouve un seuil valable pour toute l'image, tel que les pixels

dont la valeur est au-dessus du seuil sont considérés comme l'arrière-plan (blanc) et les autres comme l'information utile (appartenant aux mots écrits) (noir). La binarisation globale a l'avantage d'être rapide et, pour certains travaux comme ceux effectués sur la base Rimes, elle peut être amplement suffisante quand les images utilisées sont très propres. Un algorithme très connu et amplement utilisé est l'algorithme de séparation de classes d'Otsu présenté dans [126]. Pour appliquer cet algorithme, on calcule l'histogramme des valeurs des pixels en niveau de gris. L'algorithme Otsu sépare cet histogramme en deux classes (foncé et clair) à partir des moments des deux premiers ordres.

Il existe cependant des types d'images pour lesquels une binarisation globale n'est pas possible ou donne des résultats désastreux, comme par exemple des images de documents historiques dont le fond est taché : si le niveau de gris de la partie endommagée est faible, il peut être en dessous du seuil de binarisation ; la partie dégradée est ainsi considérée comme de l'information utile, alors qu'elle correspond au fond de l'image. Des algorithmes de binarisation utilisant des seuils locaux ont donc été proposés (Niblack [121], Sauvola et Pietikäinen [145], Kim *et al.* [84]), permettant de traiter des images extrêmement bruitées et de régler le problème de contraste de luminosité sur une même page. Des compétitions de binarisation d'images ont régulièrement lieu (Gatos *et al.* [54], Pratikakis *et al.* [137]), permettant de comparer les derniers systèmes publiés. Ainsi, en 2009 et en 2010, le même système est sorti vainqueur, basé notamment sur un calcul local de seuils et une estimation d'arrière-plan (Su *et al.* [155]).

En fonction de la base de données de travail, il est possible de choisir une méthode de binarisation particulière. Dans le cadre de cette thèse, nous utilisons des images relativement propres donc une binarisation globale nous suffit. Nous utilisons dans nos travaux la méthode dérivée de l'algorithme d'Otsu [126].

1.1.4 Calcul des lignes de base

Les lignes de base d'une image de mot sont les deux lignes qui délimitent la partie haute et la partie basse du corps central d'un mot. On peut considérer que ces lignes suivent la forme du mot ou bien les approximer par deux droites, comme illustré sur Figure 1.4.

Il est souvent utile de calculer les lignes de base d'une image de mot (ou de ligne) car elles peuvent être ensuite utilisées pour la correction de pente d'un mot,

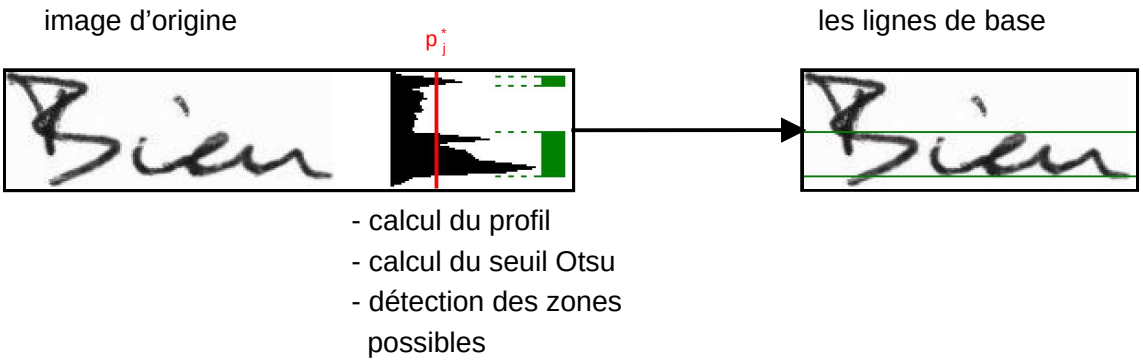


FIGURE 1.4 – Calcul des lignes (droites) de base d'un mot avec la méthode de [160].

la normalisation de l'image en taille ou bien le calcul de caractéristiques. Les lignes de bases permettent en outre de repérer les caractères ascendants et descendants : ce sont les caractères ayant des parties non situées dans le corps central des lignes de base (par exemple l est ascendant et g descendant).

Il existe plusieurs méthodes pour extraire ces lignes, souvent basées sur l'analyse de l'histogramme de projection horizontale des pixels de l'image sur un axe vertical, comme les algorithmes présentés par Blumenstein *et al.* [13] ou Vinciarelli et Luettin [160]. Dans nos travaux, nous utilisons [160] et avons adapté cet algorithme (initialement développé pour des images binaires) à des images en niveau de gris de la manière suivante : le profil horizontal de l'image en entrée est calculé. Pour la ligne numéro j de l'image, p_j est la valeur moyenne des valeurs des pixels inverses de la ligne, normalisée entre 0 et 1 :

$$p_j = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{255 - I(i,j)}{255}$$

Grâce à l'algorithme de séparation de classes d'Otsu [126] appliqué sur l'histogramme des p_j, $1 \leq j \leq n_l$, un seuil critique p_j^* est trouvé, qui divise les lignes en deux classes : les lignes potentiellement entre les deux lignes de base, et les autres. Ce seuil est illustré sur la Figure 1.4 par la ligne rouge verticale. Les lignes consécutives dont le profil p_j est supérieur au seuil p_j^* sont détectées (zones vertes sur la Figure) et la zone contenant le plus de lignes consécutives $\{p_j > p_j^*\}$ est la zone définissant les lignes de base.

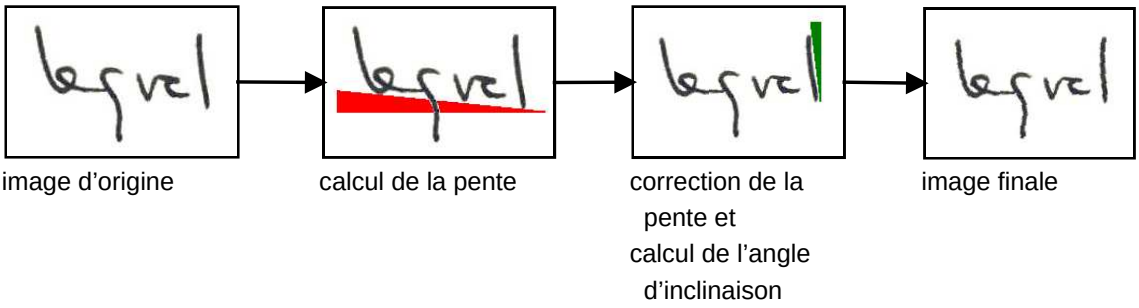


FIGURE 1.5 – Correction de la pente et de l'angle d'inclinaison d'un mot

1.1.5 Correction de la pente d'écriture et de l'angle d'inclinaison des caractères

Un exemple de correction de la pente d'un texte (en anglais *skew*) et de l'angle d'inclinaison de l'écriture (en anglais *slant*) est donné sur la Figure 1.5. Ces deux normalisations d'images sont nécessaires pour réduire la variabilité des images lors de l'apprentissage et de la reconnaissance.

En général, les méthodes de correction de pente utilisent un calcul de lignes de base non horizontales puis redressent l'image (par une rotation) jusqu'à ce que les lignes soient horizontales (Bozinovic et Srihari [14], Senior et Robinson [150]). Ces algorithmes sont cependant très dépendants du calcul de lignes de base penchées et donc potentiellement fragiles. D'autres méthodes utilisent une maximisation de l'histogramme de projection horizontale sur un axe vertical des pixels de l'image tournée selon différents angles (Côté *et al.* [26], Vinciarelli et Luettin [160]). Ces dernières, plus robustes grâce à leur indépendance vis-à-vis du calcul des lignes de base, sont celles utilisées dans nos travaux. Une fois l'image tournée pour avoir une ligne d'écriture horizontale, les lignes de bases sont recalculées (si ce n'est pas déjà fait avec l'algorithme) et on peut calculer l'angle d'inclinaison des caractères.

Les algorithmes d'estimation d'inclinaison effectuent souvent une moyenne d'angles d'inclinaisons locaux, estimés sur des parties ascendantes ou descendantes de caractères (Bozinovic et Srihari [14], Senior et Robinson [150], El-Yacoubi *et al.* [41], Marti et Bunke [111]). L'inconvénient de ces méthodes réside dans le fait qu'elles nécessitent une bonne estimation au préalable des ascendants et descendants et ne sont pas robustes aux changements de stylos ou de style d'écriture. Pour pallier à cela, Buse *et al.* [18] et Vinciarelli et Luettin [160] proposent d'utiliser le profil de la projection verticale des pixels de l'image sur l'axe horizontal. Pour notre travail,

nous avons choisi la méthode [18] qui traite directement les images en niveau de gris et ne nécessite donc pas de binarisation.

Pour un angle α donné, $-45° < \alpha < 45°$, l'image en entrée est cisaillée (en anglais *sheared*), c'est-à-dire que la nouvelle image $I_\alpha(x, y)$ est décalée de l'image d'origine $I(x, y)$ selon les équations suivantes :

$$\begin{aligned} x_\alpha &= x - y.tan(\alpha) \\ y_\alpha &= y \end{aligned}$$

Le profil horizontal de chaque image I_α est calculé : pour la colonne numéro i de l'image, p_i est la valeur moyenne normalisée des niveaux de gris de la colonne ($\sum_i p_i = 1$) :

$$p_i = \frac{1}{C}\, {p_i}^* \quad \text{avec} \quad {p_i}^* = \sum_{j=1}^{n_{l,\alpha}} \frac{255 - I_\alpha(i,j)}{255 \times n_{l,\alpha}}$$

$$\text{et} \quad C = \sum_{i=1}^{n_{c,\alpha}} {p_i}^*$$

On calcule alors pour chaque α la valeur H_α :

$$H_\alpha = -\sum_{i=1}^{n_{c,\alpha}} p_i * log(p_i)$$

où $(n_{l,\alpha}, n_{c,\alpha})$ est la taille (nombre de lignes, nombre de colonnes) de l'image I_α. L'angle α^* qui minimise H_α est l'angle d'inclinaison des caractères. Minimiser H_α revient en effet à choisir une image contenant un maximum de colonnes de pixels unies – c'est à dire des colonnes de fond ou des colonnes d'écriture (dans ce cas, p_i est très proche de 0 ou de 1, et donc $p_i log(p_i)$ est proche de zéro). Ainsi les images pour lesquelles les traits issus des ascendants et des descendants sont verticaux sont favorisées : c'est le but de l'algorithme.

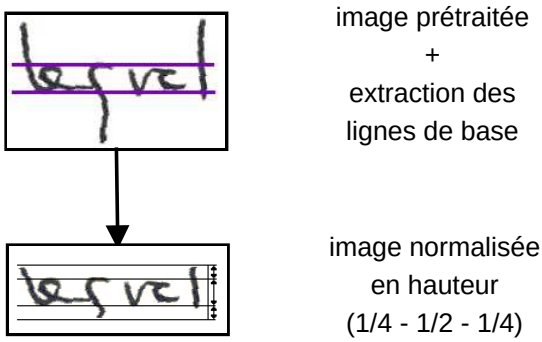


FIGURE 1.6 – Normalisation de la taille des images par reproportionnement des zones au-dessus, entre et en dessous des lignes de base.

1.1.6 Normalisation de la taille des images

La normalisation de la taille des images cherche à réduire les variations entre images dues à la taille des mots afin d'améliorer les performances du reconnaisseur. En général, la normalisation consiste à forcer les images à avoir une hauteur identique ([33, 82]). Certains systèmes ([31]) proposent en plus de reproportionner les images avec l'aide de leurs lignes de base comme illustré sur la Figure 1.6. Les proportions de la normalisation ont pour effet d'accentuer l'importance de la zone entre les lignes de base par rapport à sa taille initiale. Sur la Figure 1.6, les proportions sont 1/4 - 1/2 - 1/4.

1.1.7 Discussion

Cette Section a montré comment passer d'un problème de reconnaissance de document entier à un problème de reconnaissance de mots manuscrits isolés. Nous avons également présenté différentes formes de normalisation possibles pour une image, par exemple la binarisation, la correction de pente et d'inclination des caractères ou la modification de la taille des images. Ces normalisations sont utilisées dans la majorité des systèmes de l'état de l'art. Pourtant nous nous demandons si elles sont toutes utiles.

Par exemple, la binarisation rend parfois les contours des caractères très bruités. De plus, le classement des pixels en deux classes (noir et blanc) donne moins d'information qu'une distribution de valeurs de pixels sur 256 niveaux. De la même manière, la normalisation en hauteur des caractères est souvent utilisée mais elle dépend du calcul des lignes de base. Or des erreurs sont possibles lors de ce calcul,

ainsi la normalisation rendrait l'image en entrée « illisible » pour un oeil humain et donc a fortiori pour un système de reconnaissance. De plus, trop de normalisation peut faire perdre des informations précieuses, comme pour des tâches de reconnaissance de scripteur : si les données en entrées sont trop semblables, il devient difficile de discriminer des scripteurs.

Ainsi nous verrons dans le Chapitre 2 que nous avons choisi d'utiliser le plus possible les pixels des images non modifiées (en niveau de gris), afin de conserver le maximum d'information contenue dans l'image pour notre extraction de caractéristiques et de n'utiliser l'image binarisée que pour certains types de caractéristiques. De plus, nous ne normalisons pas la taille de nos images en entrée mais utilisons plutôt des caractéristiques dépendantes des lignes de base. Cela nous permet de conserver les proportions originales de l'image tout en disposant des informations sur les ascendants et descendants éventuellement présents dans le mot.

1.2 Extraction de caractéristiques

1.2.1 Présentation

Avant de pouvoir être interprétées par un système de reconnaissance d'écriture manuscrite, les images sont transformées. La retranscription correspond à un ensemble de *caractéristiques* extraites des images ou de parties des images.

Il existe plusieurs façons d'extraire des caractéristiques d'une image de mot, soit par fenêtre glissante, soit par segmentation explicite de l'image (en graphèmes), soit directement sur l'image complète. Quelle que soit la manière de segmenter les images, des propriétés des fragments d'image sont ensuite évaluées et représentées numériquement dans un vecteur de taille n. On dit alors que le nombre de caractéristiques extraites est n. L'ensemble des vecteurs extraits d'une image est appelé *la séquence de vecteurs de caractéristiques de l'image.*

Le principe de segmentation explicite d'une image est illustré sur la Figure 1.7. Un algorithme permet d'isoler des fragments de mots appelés graphèmes (parties de caractères ou caractères entiers) à partir de points particuliers du contour. Le principe de la fenêtre glissante est illustré sur les Figures 1.8 et 1.9. Une fenêtre glissante est une fenêtre de largeur fixe qui parcourt l'image de gauche à droite (ou de droite à gauche) telle que deux fenêtres consécutives se chevauchent. En général, la hauteur de la fenêtre est fixe et égale à la hauteur de l'image comme

FIGURE 1.7 – segmentation explicite en graphèmes (ou caractères)

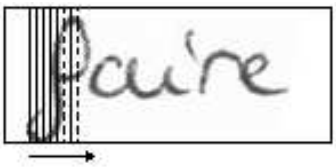


FIGURE 1.8 – segmentation implicite par fenêtres glissantes de taille fixe

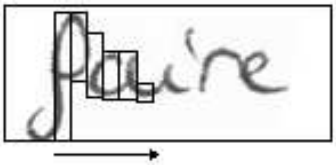


FIGURE 1.9 – segmentation implicite par fenêtres glissantes de taille variable

illustré sur la Figure 1.8 (El-Hajj *et al.* [40], Rodriguez et Perronnin [142], Dreuw *et al.* [33], Wienecke *et al.* [165]). Certains systèmes font évoluer la hauteur de leur fenêtre glissante avec la hauteur des caractères de l'image (Figure 1.9), par exemple Vinciarelli *et al.* [159].

Les caractéristiques extraites peuvent être de *bas niveau* : certains systèmes utilisent directement les pixels de la fenêtre comme caractéristiques, d'autres utilisent la distribution des pixels dans la boîte ou la fenêtre et analysent la géométrie qu'ils forment. Elles peuvent être aussi de *haut niveau*, c'est à dire qu'elles rendent compte globalement de la forme des pixels dans la boîte d'extraction, comme par exemple la présence de jambages et d'ascendants ou le nombre de caractères dans la fenêtre.

Dans cette Section, nous avons choisi de ne présenter qu'un nombre limité de caractéristiques que nous jugeons représentatives de l'ensemble des caractéristiques utilisées aujourd'hui dans l'état de l'art de la reconnaissance de mots manuscrits. Nous avons séparé les caractéristiques présentées en deux types, sachant que toutes sont de bas-niveau : les caractéristiques géométriques et statistiques basées sur l'analyse de pixels et de leur configuration et les caractéristiques directionnelles issues des descripteurs SIFT (en anglais *Scale Invariant Feature Transform*). Les SIFT sont des descripteurs d'image indépendants de variations possibles telles la translation, la rotation, l'homotétie d'une partie de l'image, etc. (Lowe [99]).

1.2.2 Caractéristiques géométriques et statistiques

Plusieurs systèmes de l'état de l'art utilisent des caractéristiques que nous qualifions de statistiques. Ces caractéristiques dépendent directement de la distribution des pixels dans la fenêtre glissante. Par exemple, Dreuw *et al.* [33], Keysers *et al.* [82] appliquent un filtre dérivatif de Sobel horizontal et vertical à une image préalablement forcée à une hauteur de 16 pixels et font glisser une fenêtre de largeur $w = 1$

pixel sur les images dérivées. Les valeurs des pixels des images dérivées sont utilisées directement en tant que caractéristiques (ou après une de réduction de dimension par analyse en composantes principales).

Dans Vinciarelli *et al.* [159], l'image est binarisée puis une fenêtre de hauteur variable (qui suit la distance entre le pixel noir le plus haut et le pixel noir le plus bas de la fenêtre, cf Figure 1.9) est divisée en 4∗4 cellules chevauchantes. Dans chaque cellule, le nombre de pixels noirs est calculé et les 16 valeurs obtenues définissent le vecteur de caractéristiques.

Les caractéristiques statistiques ont l'avantage d'être robustes face au bruit ou à la variabilité de l'écriture car elles utilisent des valeurs réelles au lieu de décrire des formes. De plus elles sont souvent normalisées par une division par la taille de l'image ou de la fenêtre d'extraction : ceci permet de se passer de la normalisation en taille des images et donc d'éviter une étape de prétraitement (sauf pour [33]). Cependant, il est intéressant d'avoir dans le vecteur de caractéristiques des informations sur la géométrie des pixels présents. Ainsi Marti et Bunke [111] proposent un ensemble de 9 caractéristiques géométriques pour images binarisées, extraites de fenêtres glissantes de largeur 1 pixel :

- le nombre de pixels noirs et les moments du premier et du second ordre de ce nombre,
- la position des contours supérieur et inférieur, ainsi que leur dérivée au premier ordre,
- le nombre de transitions noir/blanc dans la colonne de pixels,
- le nombre de pixels noirs entre les contours haut et bas.

Il a été montré que l'utilisation d'un mélange de caractéristiques statistiques et géométriques permet d'améliorer considérablement les performances du système de reconnaissance De Oliveira *et al.* [28], El-Hajj *et al.* [39]. Dans leurs travaux [39, 40], El-Hajj *et al.* utilisent des fenêtres glissantes de hauteur égale à la hauteur de l'image et de largeur w pixels, $w > 1$. Les fenêtres parcourent l'image de gauche à droite pour l'écriture latine et de droite à gauche pour l'écriture arabe. Elles sont divisées verticalement en cellules de même hauteur. Dans chaque fenêtre sont extraites $w + 20$ caractéristiques, dont certaines dépendent de la position des lignes de base. Nous notons (geo) et $(stat)$ les caractéristiques respectivement géométriques et statistiques :

- 2 comptent le nombre de transitions caractère/arrière-plan : l'une dans la fenêtre, l'autre au-dessus de la ligne de base basse(geo).

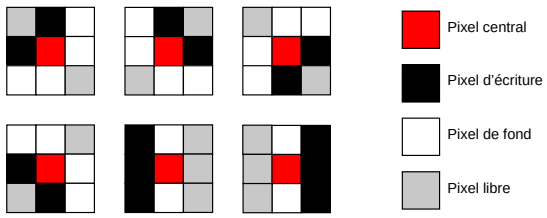


FIGURE 1.10 – Configurations de pixels comptées dans les caractéristiques géométriques de El-Hajj *et al.* [39]

- 12 sont reliées aux concavités présentes dans la fenêtre. Les 6 configurations de pixels sont illustrées sur la Figure 1.10. Pour chacune des configurations, le nombre de pixels de la fenêtre glissante lui correspondant est compté ainsi que le nombre d'occurrences entre les lignes de base haute et basse (*geo*).
- 3 sont liées à la position du centre de gravité : l'une donne la zone du centre de gravité dans la fenêtre (au-dessus de la ligne de base haute, en dessous de la ligne de base basse ou entre les deux). Une autre donne sa position par rapport à la ligne de base basse (en terme de distance de pixels). La dernière enfin est dérivative (différence des positions verticales du centre de gravité de deux fenêtres consécutives) (*geo et stat*).
- w caractéristiques correspondent aux moyennes des valeurs des pixels des w colonnes de la fenêtre glissante (*stat*).
- les 3 dernières caractéristiques sont reliées directement aux densités de pixels : la densité globale de pixels dans la fenêtre et les densités au-dessus et en dessous de la ligne de base basse (*stat*).

Grâce à l'ajout de caractéristiques dépendantes des lignes de base, des informations morphologiques sont ajoutées sur les données en entrée du système (détection de jambages et ascendants notamment). Ainsi El-Hajj *et al.* [39] proposent des caractéristiques alliant données statistiques et données géométriques et leur approche permet d'éviter des étapes de prétraitement comme la normalisation de la taille d'une image ou bien son reproportionnement. Cet ensemble de caractéristiques est l'un des plus robustes et représentatif des images initiales en ce qui concerne les caractéristiques statistiques et géométriques.

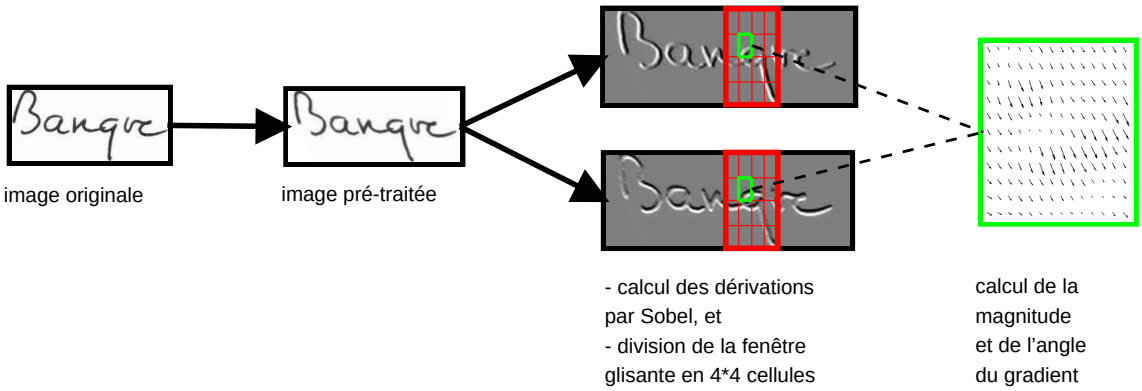


FIGURE 1.11 – Extraction des caractéristiques d'histogramme de gradient présentées dans [142] : extraction des gradients des pixels

1.2.3 Caractéristiques directionnelles

En 1999 puis en 2004, Lowe [99] présente l'algorithme SIFT pour la détection d'invariants entre deux images. Brevetée depuis, l'idée de descripteurs robustes à diverses variations subies par une image a été utilisée dans de nombreux domaines du traitement d'image, dont la détection d'objets ou encore la recherche d'image par contenu. Récemment, Rodriguez et Perronnin [142] ont proposé des caractéristiques d'histogramme de gradient poursuivant cette idée de descripteurs invariants pour la détection de mots dans une image.

L'extraction de ces caractéristiques est schématisée sur les Figures 1.11 et 1.12 et est expliquée brièvement ici. Une fenêtre glissante de largeur $w = 16$ pixels et de hauteur fixe (la hauteur de l'image) traverse l'image de gauche à droite. Cette fenêtre est divisée en $4 * 4$ cellules de taille identique et dans chacune de ces cellules sont extraites 8 valeurs décrites ci-dessous, donnant un total de $4 * 4 * 8 = 128$ caractéristiques.

Pour un pixel donné, la magnitude et l'angle du gradient sont calculés :

$$\begin{aligned} m(x,y) &= \sqrt{I_{D_h}(x,y)^2 + I_{D_v}(x,y)^2} \\ \theta(x,y) &= arctan(\frac{I_{D_h}(x,y)}{I_{D_v}(x,y)}) \end{aligned}$$

où I_{D_h} (resp. I_{D_v}) est l'image originale dérivée par un filtre Sobel horizontal ($[-1,0,1]$) (resp. vertical ($[-1,0,1]^T$)).

L'espace des angles $[-\pi,\pi[$ étant discrétisé dans 8 valeurs comme illustré sur

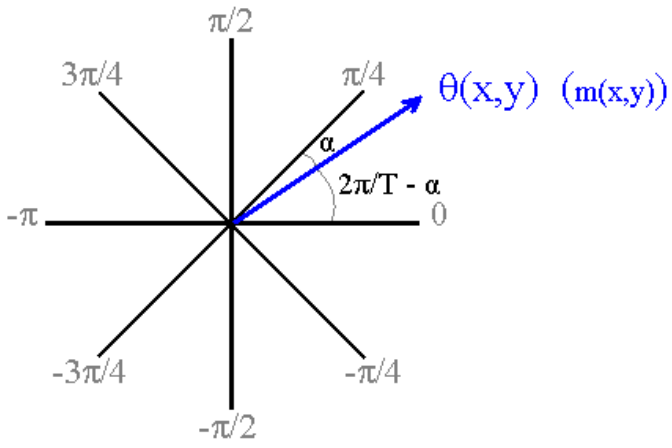


FIGURE 1.12 – Extraction des caractéristiques d'histogramme de gradient présentées dans [142] : calcul de l'histogramme

la Figure 1.12, $\theta(x, y)$ est forcément compris entre deux de ces valeurs (ou égal à une). L'apport du pixel (x, y) à l'histogramme de ces 8 directions est alors de valeur $m(x, y)$, distribué proportionnellement sur les deux valeurs entourant θ comme schématisé sur la Figure 1.12. Pour chaque cellule de la fenêtre glissante, la somme des contributions des pixels sur les 8 directions est calculée. Une fois les 128 caractéristiques calculées, elles sont normalisées pour chaque fenêtre d'extraction afin de sommer à 1.

1.2.4 Bilan

Nous avons choisi de ne pas faire une liste exhaustive des caractéristiques utilisées actuellement dans l'état de l'art car elles sont trop nombreuses pour être toutes citées. L'extraction de caractéristiques est un sujet encore brûlant aujourd'hui car aucun concensus n'a été trouvé parmi les différentes approches existantes. Une uniformisation des caractéristiques en reconnaissance de l'écriture telle qu'elle existe en parole (les MFCCs) paraît pour l'instant peu probable.

Pourtant, il est admis aujourd'hui que les caractéristiques extraites d'une image suivant certaines propriétés (comme les caractéristiques issues du mélange statistique-géométrique ou encore celles dérivées des SIFT) donnent de meilleurs résultats. C'est peut-être une première piste vers une uniformisation des méthodes d'extraction.

1.3 Les méthodes de reconnaissance de mots manuscrits à base de modèles de Markov cachés

Nous présentons dans cette Section quelques-unes des méthodes de l'état de l'art en reconnaissance d'écriture manuscrite, que nous jugeons appropriées pour être comparées à notre travail. Nous insistons particulièrement sur les méthodes à base de Modèles de Markov Cachés (HMMs), présentés en Section 1.3.1, parce que ce sont les méthodes les plus utilisées aujourd'hui et qu'elles atteignent les meilleures performances. Les systèmes de l'état de l'art les utilisent seuls ou bien dans des systèmes hybrides, comme nous le verrons dans la Section 1.3.2. A la fin de cette Section, nous discutons (Section 1.3.3) d'autres méthodes qui n'utilisent pas de HMMs afin de proposer un état de l'art le plus représentatif possible des méthodes actuelles.

1.3.1 Méthodes HMMs : généralités

Les modèles de Markov cachés sont une méthode utilisée depuis longtemps pour la modélisation de séquences. Leur simplicité s'applique avec succès à une grande diversité de tâches : la reconnaissance de la parole, la reconnaissance de l'écriture manuscrite ou imprimée ou encore la bio-informatique [6].

Principe. Les modèles de Markov cachés sont un processus doublement stochastique, largement utilisé pour modéliser statistiquement des séquences. D'une part, la dynamique de la séquence à modéliser est représentée par une chaîne de Markov, c'est-à-dire un ensemble d'états et de transitions entre états ; la particularité des chaînes de Markov d'ordre 1 est l'hypothèse que la probabilité de se trouver dans un état donné à un instant donné ne dépend que de l'état à l'instant précédent. D'autre part, des densités de probabilité sont associées aux états afin de modéliser les observations. La dénomination « modèles de Markov *cachés* » vient du fait que la séquence d'états correspondant aux données observées est inconnue de l'observateur, qui n'a accès qu'aux observations elles-mêmes. Ainsi, par construction, les HMMs ne peuvent pas fournir la séquence exacte d'états qui génère la séquence d'observation $\boldsymbol{O_T} = o_1 o_2 \dots o_T$ mais il est possible de calculer la séquence d'états la plus vraisemblable selon les données observées.

Un modèle HMM discret [1], que nous notons $\boldsymbol{\lambda} = \{N, M, A, B, \Pi\}$, est défini par :

- le nombre d'états N du modèle. Les états sont nommés $s_1, s_2, \ldots s_N$ et q_t est l'état au temps t. Q_T est la séquence d'états $q_1, q_2, \ldots q_T$.
- le nombre d'observations par état M, nommées $v_1, v_2, \ldots v_M$.
- la matrice des probabilités de transition d'état à état $A = \{a_{ij}\}$, où $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$ et $1 \leq i, j \leq N$. Par définition, $0 \leq a_{ij} \leq 1$ et $\sum_{j=1}^{N} a_{ij} = 1$.
- la distribution de probabilité $B = \{b_j(k)\}$ des symboles observés dans l'état j, où $b_j(k) = P(o_t = v_k | q_t = s_j)$.
- la distribution des états à l'instant initial $\Pi = \{\pi_i\}$ où $\pi_i = P(q_1 = s_i)$.

Les HMMs *continus*, très utilisés dans les systèmes de reconnaissance de l'écrit ou de la parole, diffèrent des HMMs discrets dans le sens où il est imposé aux b_j une forme continue. Ainsi on ne parle plus de la probabilité d'émission d'un vecteur d'observations mais de la vraisemblance qu'un tel vecteur soit émis alors que le système est dans un état donné. Dans le cas *classique*, une loi normale multi-gaussienne est utilisée et les b_j sont de la forme :

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(o_t, \boldsymbol{\mu}_{jm}, \boldsymbol{U}_{jm})$$

où M est le nombre de gaussiennes, c_{jm} est le coefficient de la gaussienne numéro m associée à l'état s_j et $\boldsymbol{\mu}_{jm}, \boldsymbol{U}_{jm}$ sont respectivement sa moyenne et sa matrice de covariance. D'autres systèmes cependant utilisent des lois différentes, comme par exemple la loi de Bernouilli [57, 58] :

$$b_j(o_t) = \sum_{k=1}^{K} \pi_{jk} \prod_{d=1}^{D} p_{jkd}^{o_{td}} (1 - p_{jkd})^{1 - o_{td}}$$

où π_{jk} est le coefficient du $k^{ème}$ mélange et p_{jk} son prototype de Bernouilli, K le nombre de mélanges et D la dimension des caractéristiques.

En général, les systèmes de reconnaissance d'écriture manuscrite utilisent des HMMs gauche-droit (de type Bakis), qui imposent des contraintes sur A, comme

1. les observations en sortie appartiennent à un espace discret et fini

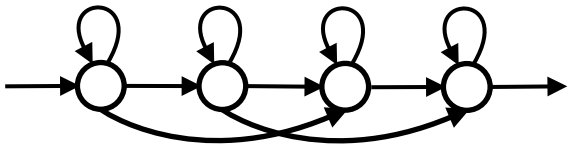


FIGURE 1.13 – HMM de type Bakis

illustré sur la Figure 1.13 :

$$a_{ij} = 0 \quad \text{si} \begin{cases} j < i & ou \\ j > i + 2 \end{cases}$$

La première condition signifie qu'il n'y a pas de transition d'un état vers un état antérieur (modèle gauche-droite) et la seconde signifie que d'un état on ne peut passer que vers lui-même, vers l'état suivant ou bien celui qui suit le suivant (modèle de Bakis). Sachant cela, on impose aussi une contrainte sur Π : la séquence d'états doit commencer par l'état 1 et terminer par l'état N, donc $\pi_1 = 1$ et $\pi_i = 0$ si $i \neq 1$.

Les HMMs *semi-continus*, quant à eux, sont des HMMs continus dont les paramètres sont partagés. Cette modélisation est avantageuse par rapport aux HMMs classiques continus car elle permet d'apprendre moins de paramètres avec autant de données et d'éviter les phénomènes de suraprentissage qui nuisent à la capacité de généralisation du modèle. En général, les HMMs semi-continus définissent un ensemble de gaussiennes et l'ensemble des états partagent ces gaussiennes. Nous verrons au Chapitre 3 que des partages de paramètres plus intéressants peuvent être effectués, qui améliorent considérablement les performances du système.

L'un des principaux avantages d'utiliser les modèles de Markov cachés est que leur apprentissage ainsi que le décodage sont basés sur des algorithmes performants et éprouvés, présentés ci-dessous.

Apprentissage. Il n'existe pas d'algorithme pour estimer les paramètres optimaux du modèle mais il est possible de trouver un maximum local, notamment à l'aide de l'algorithme EM (Espérance-Maximisation, en anglais *Expectation- Maximisation*). Les paramètres du modèle $\boldsymbol{\lambda}$ sont itérativement ajustés pour maximiser la probabilité $P(\boldsymbol{O}|\boldsymbol{\lambda})$ d'avoir généré les données observées avec le modèle. A chaque itération, on dispose du modèle précédemment calculé $\boldsymbol{\lambda}$ et on cherche $\hat{\boldsymbol{\lambda}}$ tel que

$P(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}) > P(\boldsymbol{O}|\boldsymbol{\lambda})$.

Dans le cas de l'algorithme Baum-Welch [7] (ou algorithme *Forward-Backward*), $\hat{\boldsymbol{\lambda}}$ est trouvé par maximum de vraisemblance, c'est-à-dire en maximisant la fonction auxiliaire $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}})$:

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) = \sum_{Q_T} P(q_1 q_2 \dots q_T|\boldsymbol{O}, \boldsymbol{\lambda}) log[P(\boldsymbol{O}, q_1 q_2 \dots q_T|\hat{\boldsymbol{\lambda}})]$$

Le calcul de $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}})$ est l'étape E de l'algorithme EM et la maximisation de $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}})$ est l'étape M. La procédure EM est répétée jusqu'à ce que $P(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}) - P(\boldsymbol{O}|\boldsymbol{\lambda}) < \epsilon$ pour ϵ faible ; le modèle trouvé maximise la vraisemblance.

L'algorithme Baum-Welch est à ce jour largement utilisé pour l'apprentissage des modèles HMMs. Il introduit deux probabilités : la probabilité avant $\alpha_t(i)$ (*forward*) et la probabilité arrière $\beta_t(i)$ (*backward*). $\alpha_t(i)$ est la probabilité de la séquence d'observation partielle $\boldsymbol{o_1, o_2, \dots o_t}$ d'être dans d'état s_i à l'instant t et $\beta_t(i)$ est la probabilité d'observer la séquence partielle de symboles $\boldsymbol{o_{t+1}, o_{t+2}, \dots o_T}$ en supposant être à l'état s_i à l'instant t. L'algorithme Baum-Welch se résume en trois étapes : d'abord une passe avant (*forward*) puis une passe arrière (*backward*) suivie d'une passe de lissage.

L'étape *forward* consiste à calculer par récurrence les $\alpha_t(i)$ à partir de $t = 1$. La récurrence s'initialise :

$$\alpha_1(i) = \qquad \text{pour} \quad 1 \leq i \leq N$$

Puis le passage de α_t à α_{t+1} s'effectue avec le calcul suivant :

$$\alpha_{t+1}(j) = \{\textstyle\sum_{i=1}^{N} \alpha_t(i) a_{ij}\} b_j(\boldsymbol{o_{t+1}}) \qquad \text{pour} \quad 1 \leq j \leq N, \quad t = 1 \dots T-1$$

Durant l'étape *backward*, on calcule les $\beta_t(i)$ par récurrence inverse (en commençant à $t = T$). L'initialisation est donnée en posant :

$$\beta_T(i) = 1 \qquad \text{pour} \quad 1 \leq i \leq N$$

et le passage de β_{t+1} à β_t se calcule :

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{o_{t+1}}) \beta_{t+1}(j) \qquad \text{pour} \quad 1 \leq i \leq N, \quad t = T-1 \ldots 1$$

Les probabilités avant et arrière sont ensuite combinées pour obtenir la probabilité d'être dans l'état s_i à l'instant t :

$$P(q_t = s_i | \boldsymbol{O}, \boldsymbol{\lambda}) = \frac{\alpha_t(i)\beta_t(i)}{P(\boldsymbol{O}|\boldsymbol{\lambda})}. \tag{1.1}$$

Cette probabilité est dite *lissée* car elle utilise les connaissances avant et arrière pour son calcul. Les probabilités $\alpha_t(i)$ et $\beta_t(i)$ sont enfin utilisées dans le calcul de la probabilité d'être dans l'état s_i à l'instant t **et** dans l'état s_j à $t+1$:

$$P(q_t = s_i, q_{t+1} = s_j | \boldsymbol{O}, \boldsymbol{\lambda}) = \frac{\alpha_t(i) a_{ij} b_j(\boldsymbol{o_{t+1}}) \beta_{t+1}(j)}{P(\boldsymbol{O}|\boldsymbol{\lambda})}. \tag{1.2}$$

Avec les deux valeurs définies dans les équations 1.1 et 1.2, il est ensuite possible de ré-estimer les paramètres a_{ij} et $b_j(t)$ du modèle HMM $\boldsymbol{\lambda}$.

Décodage. Le but du décodage de modèles HMMs est de choisir la séquence d'états $Q_T = q_1 q_2 \ldots q_T$ optimale correspondant aux données observées $\boldsymbol{O_T} = o_1 o_2 \ldots o_T$ sachant qu'on dispose du modèle $\boldsymbol{\lambda}$. La séquence Q_T^* est celle qui maximise $P(\boldsymbol{O_T}, q_1 q_2 \ldots q_T | \boldsymbol{\lambda})$.

Il ne serait pas correct de calculer Q_T^* en prenant, pour chaque instant t, l'état s^* qui maximise $P(q_t = s | \boldsymbol{O}, \boldsymbol{\lambda})$. En effet, les états q_t^* seraient individuellement les plus probables mais la séquence qu'ils produiraient ne serait pas viable car elle ne prendrait pas en compte, entre autres, les probabilités de transition possibles entre états.

La séquence Q_T^* se construit alors par récurrence. Supposons qu'à l'instant t on connaisse la meilleure séquence d'états Q_t^* (les états de 1 à t) et que celle-ci finisse par l'état s_i au temps t. On définit alors $\delta_t(i)$, la vraisemblance que la séquence d'observations $\boldsymbol{o_1, o_2, \ldots o_t}$ soit produite par cette séquence d'états optimale. Pour chaque étape de la récurrence, on conserve le souvenir de la séquence Q_t^* dans la

variable $\phi_t(i)$. La récurrence s'initialise alors :

$$\begin{aligned} \delta_1(1) &= 1 \\ \delta_1(i) &= 0 \qquad\qquad \text{si i} > 1 \text{ (HMMs de Bakis)} \end{aligned}$$

Le passage de l'instant t à l'instant $t+1$ se calcule ensuite :

$$\begin{aligned} \delta_{t+1}(j) &= \max_{1<i<N} |\delta_t(i)a_{ij}|b_j(o_t) \\ \phi_t(j) &= \arg\max_{1<i<N} |\delta_t(i)a_{ij}| \end{aligned}$$

La terminison s'obtient enfin :

$$q_T^* = \arg\max_{1<i<N} |\delta_T(i)|$$

L'algorithme de Viterbi se termine en effectuant un parcours en arrière sur les états optimaux. Pour chaque instant $t < T$, q_t^* s'obtient :

$$q_t^* = \phi_{t+1}(q_{t+1}^*)$$

Les modèles de Markov cachés en reconnaissance d'écriture. L'utilisation massive des HMMs dans les systèmes de reconnaissance de l'écriture se justifie par l'énonciation du problème : pour une image donnée, $\boldsymbol{O_T}$ représente l'ensemble des vecteurs de caractéristiques (en anglais, *frames*) extraits de cette image. Etant donné que l'on dispose des données observées $\boldsymbol{O_T}$, on cherche ensuite à trouver la séquence de symboles $\hat{\boldsymbol{w}}$ qui maximise la probabilité a posteriori d'avoir émis $\boldsymbol{w}$ sachant que l'on dispose des données $\boldsymbol{O_T}$, $P(\boldsymbol{w}|\boldsymbol{O_T})$. Grâce à la loi de Bayes, on peut écrire :

$$\begin{aligned} \hat{\boldsymbol{w}} &= \operatorname*{argmax}_{\boldsymbol{w}} P(\boldsymbol{w}|\boldsymbol{O_T}) \\ &= \operatorname*{argmax}_{\boldsymbol{w}} \frac{P(\boldsymbol{w})P(\boldsymbol{O_T}|\boldsymbol{w})}{P(\boldsymbol{O_T})} \\ &= \operatorname*{argmax}_{\boldsymbol{w}} P(\boldsymbol{w})P(\boldsymbol{O_T}|\boldsymbol{w}) \end{aligned}$$

$P(\boldsymbol{w})$ est la modélisation des séquences possibles de symboles $\boldsymbol{w}$. $P(\boldsymbol{O_T}|\boldsymbol{w})$ est la probabilité d'observer la séquence de vecteurs de caractéristiques $\boldsymbol{O_T}$ sachant le modèle d'écriture, que les algorithmes Baum-Welch (pour l'apprentissage) et Viterbi

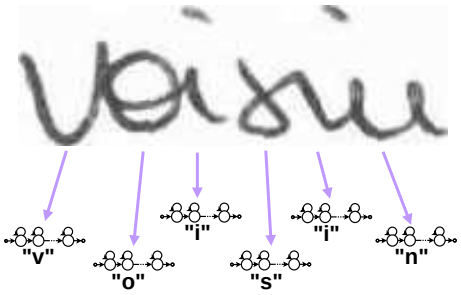


FIGURE 1.14 – Le modèle d'un mot est la concaténation des modèles HMMs des caractères le composant.

(pour le décodage) permettent de déterminer.

Dans le cas de la reconnaissance de mots, $\boldsymbol{w}$ est une séquence de caractères et $P(\boldsymbol{w})$ représente le lexique ou le modèle de langage ; un HMM modélise un caractère et le modèle d'un mot est la concaténation des modèles de caractères le composant, comme illustré sur la Figure 1.14.

1.3.2 Etat de l'art des méthodes à base de HMMs

On peut regrouper sous trois catégories les systèmes de reconnaissance de mots manuscrits : les méthodes holistiques, celles à base de segmentation explicite d'un mot en graphèmes et les méthodes à segmentation implicite.

Les méthodes holistiques considèrent un mot comme une entité. Les caractéristiques sont extraites sur le mot sans le décomposer et chaque mot correspond à un modèle unique appris. Ces méthodes ont été appliquées avec succès à des tâches très précises, comme la reconnaissance de montant sur les chèques ou bien la recherche de mot-clef dans un document mais leur utilisation reste limitée. En effet, ce type de méthode ne passe pas à l'échelle d'un plus grand lexique et exige que le nombre d'exemples d'apprentissage soit élevé pour chaque modèle, ce qui n'est souvent pas réaliste dans le cadre de la reconnaissance de mots en général. Cependant, l'utilisation d'une classifification à base de méthode holistique peut permettre pour certaines tâches de réduire le lexique de test en amont (Madhvanath et Govindaraju [101]) ou encore d'aider au nettoyage d'images dégradées (Lavrenko *et al.* [89]).

Les méthodes à base de segmentation explicite d'un mot découpent un mot en sous-parties, les graphèmes, puis des caractéristiques sont extraites sur ces

graphèmes. Les graphèmes peuvent être soit des caractères, soit des parties de caractères. Pour chaque mot, une ou plusieurs segmentations sont conduites en même temps et certains graphèmes peuvent être regroupés entre eux pour former des caractères. Les mots peuvent être modélisés de deux façons. Soit un classifieur de caractère isolé modélise (puis reconnaît) chacun des caractères proposés dans chacune des segmentations [46, 83, 41, 86]. Dans ce cas, la phase d'apprentissage et de décodage se simplifie à un classifieur de caractères isolés, pour lequel l'état de l'art atteint actuellement d'excellentes performances. Le décodage peut d'ailleurs être guidé par un lexique afin d'avoir un nombre de mots autorisés fini. De plus, il est possible d'appliquer un modèle de langage sur les n meilleures sorties du classifieur pour chaque graphème afin d'améliorer les performances. Une autre façon de modéliser les mots est d'utiliser des HMMs pour les décomposer en une séquence de graphèmes [60, 5, 113]. L'inconvénient majeur de méthodes à base de segmentation explicite est que la segmentation en elle-même est périlleuse. En effet, les méthodes de segmentation en graphèmes utilisent des heuristiques basées sur des intuitions humaines et non des règles automatiques. Ainsi il est impossible de vérifier que la segmentation en graphèmes est correcte (aucun graphème ne doit être partagé par deux caractères) ou bien que, dans toutes les segmentations et regroupements de graphèmes proposés, la segmentation en caractères est bonne [35].

Les méthodes séquentielles, à base de segmentation implicite, sont à ce jour les méthodes les plus utilisées en reconnaissance de l'écriture manuscrite et produisent la plupart des meilleurs systèmes. Elles sont utilisées avec des modèles de Markov [159, 100, 15, 40, 134, 143, 58] ou bien avec des réseaux de neurones par exemple [44]. Ces méthodes ont l'avantage de ne pas nécessiter de segmentation explicite des images en entrée, aussi l'utilisation de modèles statistiques permet d'être peu sensible aux variations de forme des caractères ou au bruit qui les entoure. De plus, ces méthodes ont une grande flexibilité vis-à-vis de la longueur des modèles, ce qui est un avantage pour la reconnaissance de l'écriture, où la taille des mots d'un document à un autre peut varier grandement. Enfin, ces méthodes considèrent les mots écrits comme des observations séquentielles, c'est-à-dire une suite d'éléments ordonnés : c'est sans doute la modélisation la plus naturelle.

Le tableau 1.1 dresse une liste non exhaustive de classifieurs aujourd'hui considérés comme à l'état de l'art et qui utilisent l'une de ces trois méthodes. Cette liste insiste plus sur les méthodes à base de modèles de Markov cachés étant donné

que c'est ce que nous utilisons dans notre système. Cette sélection est donnée par ordre de parution des systèmes – si les systèmes ont été améliorés depuis, la dernière publication le concernant (à notre connaissance) est dans le tableau. On peut voir que de nouvelles méthodes à base de segmentation explicite des mots en graphèmes n'apparaissent plus, alors que les travaux de recherche sur les méthodes à base de segmentation implicite sont toujours d'actualité.

Dans le tableau 1.1, nous utilisons certaines abbréviations explicitées ci dessous :

- HMM signifie que la méthode utilise des Modèles de Markov Cachés. Il existe plusieurs types de HMMs, comme les HMMs discrets, continus, semi-continus ou encore les HMMs à distribution de Bernouilli, comme nous l'avons décrit dans la Section 1.3.1.
- NN veut dire Réseau de Neurones (Neural Network) et les sigles RNN et SNN signifient que les méthodes utilisées dérivent respectivement des Réseaux de Neurones Récurrents et des Réseaux de Neurones Segmentaux.

1.3.3 Discussion

Nous avons présenté en Section 1.3.2 un état de l'art général de systèmes de reconnaissance de mots manuscrits à base de HMMs, dont les principes ont été développés en Section 1.3.1. Dans cette Section, nous approfondissons l'étude de la Section 1.3.2. Des méthodes citées dans le tableau 1.1, nous retenons 3 courants principaux pour les classifieurs à base de HMMs, décrits ci-dessous.

Les HMMs continus sont très répandus dans les systèmes de l'état de l'art de la reconnaissance de mots manuscrits. Ils sont le plus souvent utilisés accompagnés d'améliorations annexes. Ainsi le système de l'Université de Aachen [33, 34, 32] utilise un classifieur à base de HMMs gaussiens mais pratique un apprentissage discriminant afin d'améliorer ses performances. De plus, pour l'écriture arabe, des modèles de silence intra-mots permettent à leur système d'être à l'état de l'art lors de récentes compétitions. De même, El-Hajj *et al.* [39] utilise des HMMs classiques ; l'élaboration de caractéristiques géométriques (cf Section 1.2) ainsi qu'une combinaison des sorties de trois classifieurs HMMs à l'aide de Réseaux de Neurones donne de bons résultats. Enfin, le système utilisé par Marti et Bunke [111], qui utilise aussi des HMMs gaussiens classiques, modélise chaque caractère en fonction de sa longueur, ce qui permet d'augmenter les performances du système. Enfin, si toutes ces méthodes

Système / Exploitant	**Méthode utilisée**	**Caractéristiques, options, optimisations, etc**	**Article(s) de référence**
A2iA	segm. explicite + hybride HMM/NN	Système hybride : segmentation explicite et reconnaissance de graphèmes par NN puis passage de graphèmes à mots par HMMs	Gorski *et al.* [60], 2001 Augustin [5], 2001 Menasri [113],2008
Concordia University, Montréal	combin. HMMs et NNs segm.	Reconnaissance et segmentation de mots avec HMMs et caractéristiques haut niveau. Reco. lettres par SNN et caract. bas niveau	El-Yacoubi *et al.* [41], 1999 Koerich *et al.* [86], 2006
IAM - Bern	HMMs	La longueur des modèles dépend du caractère. Décodage avec modèle de langage	Marti *et al.* [111], 2001 Bertolami *et al.* [11], 2007 Vinciarelli *et al.* [159], 2004
BBN Technologies	HMMs	HMMs semi-continus + modèles de caractères en contexte. Décodage avec modèle de langage	Natarajan *et al.* [119], 2001 Cao *et al.* [20], 2010 MacRostie *et al.* [100], 2010
TU München	HMMs	Reconnaissance d'adresses avec n-grams de caractères et adaptation au scripteur	Brakensiek *et al.* [15], 2004
University of Massa-chusetts	HMMS	Approche holistique de reconnais--sance de documents historiques dégradés monoscripteur pour le nettoyage d'images	Lavrenko *et al.* [89], 2004
UOB et Télécom ParisTech	HMMs	Caractéristiques géométriques et combinaison de classifieurs par réseaux de neurones	El-Hajj *et al.* [39], 2005 El-Hajj *et al.* [40], 2005
TU Dortmund	HMMs	HMMs semi-continus. La longueur des modèles dépend du caractère	Wienecke *et al.* [165], 2005 Plötz *et al.* [133], 2008 Plötz *et al.* [134], 2009
University of Leeds	HMMs	Caractéristiques d'histogramme de gradient. Localisation de mots sur une page avec des HMMs semi-continus	Rodriguez *et al.* [142], 2008 Rodriguez *et al.* [143], 2009
University of Aachen (RWTH)	HMMs	Modèles pour les espaces intra-mots (arabe) + apprentissage discriminant (adaptation des modèles)	Dreuw *et al.* [33], 2008 Dreuw *et al.* [34], 2009 Dreuw *et al.* [32], 2009
University of Valencia (UPV-DSIC)	HMMs	Mélanges de distributions de Bernoulli dans les HMMs	Gìmenez *et al.* [59], 2008 Gìmenez *et al.* [57], 2009 Giménez *et al.* [58], 2010
University of Valencia (UPV-DSIC)	hybride : HMMs + NNs	Système hybride : un perceptron mutli-couches estime les proba. d'émission des états des HMMs	España-Boquera *et al.* [45], 2009, España-Boquera *et al.* [44], 2010

TABLE 1.1 – Sélection de systèmes à base de HMMs pour la reconnaissance de mots manuscrits

utilisent des mélanges multi-gaussiens pour probabilité d'émission, d'autres lois depuis ont été proposées et évaluées, comme la loi de Bernoulli, utilisée par Giménez *et al.* [58]. Le caractère même des distributions de Bernouilli permet de recevoir en entrée du système directement les pixels de l'image binarisée et donc d'éviter de passer par une extraction de caractéristiques. Comme discuté dans la Section 1.2.4, l'extraction de caractéristiques d'images pour l'apprentissage de modèles ou la classification n'est pas un problème résolu. Ainsi, des systèmes qui ne nécessitent pas une extraction explicite de caractéristiques mais qui incluent une phase d'extraction automatique et implicite, par exemple par apprentissage automatique, nous semblent prometteurs.

Les HMMs semi-continus attirent plus spécialement notre attention car ils se répandent depuis peu dans la reconnaissance d'écriture manuscrite. Le partage de paramètres permet de mieux apprendre chaque modèle, en particulier si le nombre de données d'apprentissage n'est pas élevé. Ce partage (que ce soit un partage de distributions gaussiennes entre tous les états ou bien un partage d'états entre différents HMMs de caractères) permet aussi d'élaborer des modèles plus complexes et plus précis, comme par exemple les modèles de caractères dépendants de leur contexte, qui sont au coeur de notre système de reconnaissance de mots manuscrits (voir Chapitre 3).

Les systèmes hybrides HMMs / NN sont très intéressants car la combinaison des deux méthodes permet d'éviter certains problèmes d'une méthode en utilisant l'autre. Par exemple, les réseaux de neurones peuvent être utilisés pour calculer les probabilités d'émission des états des HMMs [5, 113]. Cela permet de faire de l'apprentissage discriminant et de remplacer le calcul des mélanges gaussiens dans les HMMs classiques par un réseau de neurones, plus rapide lors du décodage. Aussi, en reconnaissance de l'écriture en ligne, les réseaux de neurones à convolution permettent de dépasser le problème de l'extraction de caractéristiques que rencontrent les systèmes HMMs en général [135]. En effet, l'extraction est effectuée par le réseau directement sur les pixels et ce type de système permet d'apprendre les caractéristiques à extraire, au lieu de forcer le système à travailler avec des caractéristiques choisies.

Nous ne pouvons clore cette Section de discussion sans évoquer une dernière méthode, qui n'utilise pas de HMMs mais obtient de très bons résultats en reconnaissance d'écriture manuscrite : *les Réseaux de Neurones Récurrents* sont en effet à ce jour le système de référence de l'état de l'art (voir Grosicki et El-Abed [64]). Ils sont présents dans le monde de l'apprentissage depuis plusieurs années (Hochreiter et Schmidhuber [67]) et le développement récent de réseaux LTSM (*long-short term memory*) a permis leur application récente à la reconnaissance de la parole (Eck *et al.* [36]) et de l'écrit (Graves et Schmidhuber [62]). Contrairement aux HMMs, les RNNs utilisent un apprentissage discriminant et les caractéristiques qui représentent les données sont adaptatives (apprises sur les données). De plus les RNNs ne font pas l'hypothèse de l'indépendence entre les observations, contrairement aux HMMS. Nous comparerons donc dans le Chapitre 4 (Expériences) notre système original à d'autres systèmes HMMs, ainsi qu'aux RNNs.

Conclusion du chapitre 1

Dans ce premier chapitre, nous avons présenté une introduction à la reconnaissance de documents manuscrits. Nous avons vu en premier lieu que le traitement d'un document manuscrit se fait en plusieurs étapes : d'abord la structure du document est extraite et ensuite les paragraphes isolés sont analysés un à un ; les lignes les composant sont détectées, afin que les mots qui les composent soient reconnus. Avant de procéder à la lecture de lignes, nous avons vu qu'il est nécessaire de prétraiter les images. Certains documents manuscrits sont en effet bruités et les systèmes de reconnaissance peuvent en être affectés.

Dans la deuxième partie de ce chapitre, nous avons décrit comment transformer une image afin qu'elle soit interprétée par un système : c'est l'extraction de caractéristiques. Plusieurs types de caractéristiques peuvent être extraites d'une image selon le système de reconnaissance choisi. Elles peuvent être haut niveau, bas niveau, liées aux statistiques des pixels ou bien à la forme des caractères présents dans l'image. Pour extraire les caractéristiques d'une image, plusieurs approches sont possibles : l'approche à segmentation explicite où l'image d'un mot est découpée en sous-parties (graphèmes), l'approche holistique et l'approche à segmentation implicite. Cette dernière est l'approche que nous choisissons. Elle a l'avantage d'éviter les difficultés de segmentation de la première et permet au classifieur d'apprendre

un plus grand nombre de modèles que la deuxième. Une segmentation implicite des images de mots est caractérisée par une extraction de caractéristiques par fenêtres glissantes. Nous avons présenté dans ce chapitre les caractéristiques rencontrées le plus fréquemment dans l'état de l'art pour les systèmes à base de fenêtres glissantes. Nous avons réutilisé certaines d'entre elles dans notre système, comme nous l'expliquerons dans le Chapitre 2.

Enfin, ce premier chapitre a permis d'introduire l'utilisation de modèles de Markov cachés pour la reconnaissance de mots manuscrits. Les HMMs représentent à ce jour la technique la plus utilisée pour cette tâche. Ceci s'explique grâce à certaines de leurs propriétés, notamment leur capacité à modéliser des séquences de longueur variable. Nous avons donné un aperçu des techniques à base de HMMs utilisées dans l'état de l'art de la reconnaissance de l'écriture manuscrite. Trois courants majeurs se dégagent : les systèmes HMMs *classiques*, où chaque caractère est représenté par un HMM dont les états sont modélisés par des distributions de probabilité indépendantes d'un état à un autre, les systèmes HMMs *hybrides*, où le calcul des probabilités d'observation se fait avec des réseaux de neurones et enfin les systèmes HMMs *semi-continus*. Ces derniers attirent plus précisément notre attention car ils permettent une modélisation plus fine des caractères tout en contrôlant la multiplication des paramètres que cette modélisation implique. Le système original que nous présenterons au Chapitre 3 se base sur cette idée tout en proposant une nouvelle façon de partager les paramètres des modèles HMMs afin d'améliorer le système.

L'état de l'art des systèmes de reconnaissance de mots manuscrits que nous avons présenté dans ce chapitre reste dans le périmètre des systèmes à base de HMMs. Il nous faut pourtant évoquer qu'il existe d'autres types de classifieurs, appliqués avec succès à cette tâche, comme les classifieurs à base de réseaux de neurones. Lors de la présentation de nos résultats au Chapitre 4, nous reviendrons sur ces autres classifieurs, qui atteignent des performances comparables à celles des HMMs.

Chapitre 2

Système de reconnaissance de mots manuscrits à base de HMMs Gaussiens indépendants du contexte

Introduction

La plupart des méthodes de reconnaissance de mot manuscrits présentées dans le chapitre précédent partagent la même vision des images de mots en entrée du système : elles convertissent ces images en séquence d'observations. L'approche séquentielle de la reconnaissance d'écriture manuscrite est aujourd'hui la méthode qui fonctionne le mieux car elle respecte la nature même des données. Avant d'aborder les modèles en contexte au Chapitre 3, qui représentent une approche originale de l'application de HMMs à la reconnaissance de l'écriture, nous présentons dans ce chapitre un système HMM classique. L'élaboration de ce système garantit la mise en place d'un reconnaisseur HMM classique robuste, qui pose les bases du reconnaisseur avec modèles en contextes. Le système présenté dans ce chapitre profite en outre d'apports originaux, comme les caractéristiques dynamiques ou encore l'adaptation du nombre d'états à la longueur des caractères.

Le chapitre est organisé ainsi : la Section 2.1 présente les caractéristiques utilisées dans nos systèmes HMM (classique et en contexte). La Section 2.2 explicite notre façon d'utiliser les algorithmes d'apprentissage et de décodage des HMMs pour l'élaboration de nos modèles. Enfin, la Section 2.3 propose une technique originale d'amélioration de performances, optimisant le nombre d'états du HMM de chaque

caractère appris.

2.1 Extraction de caractéristiques

La première étape d'un système à base de HMMs est de prétraiter, normaliser et transformer en un signal interprétable (séquence de vecteurs de caractéristiques ou séquence de sous-parties de l'image) les images en entrée. Nous avons présenté dans le Chapitre 1 différents prétraitements applicables aux images, comme la binarisation, la correction d'angle et de pente de l'écriture ainsi que la normalisation des images en hauteur. Dans notre système, nous avons choisi de conserver les images dans leur niveau de gris initial afin de ne pas perdre d'information et de ne pas modifier la taille des images. Le seul prétraitement que nous appliquons est la détection d'angle et de pente, afin d'harmoniser le parcours des fenêtres glissantes lors de l'extraction des caractéristiques.

Nous avons aussi présenté au Chapitre 1 les caractéristiques les plus utilisées pour la reconnaissance d'écriture manuscrite. Nous présentons ici les caractéristiques que nous utilisons (voir Section 2.1.1), qui sont une combinaison des caractéristiques géométriques (cf Section 1.2.2) et des caractéristiques d'histogrammes de gradient (cf Section 1.2.3). Dans cette Section, nous présentons aussi les ajouts et changements que nous avons appliqués aux caractéristiques que nous utilisons, afin d'améliorer notre système : calcul d'une régression (voir Section 2.1.2) et analyse en composantes principales (voir Section 2.1.3).

2.1.1 Caractéristiques utilisées dans le système HMM Gaussien

Caractéristiques géométriques : L'avantage des caractéristiques géométriques présentées par El-Hajj *et al.* [39] (cf Section 1.2.2) est qu'elles dépendent des lignes de base, ainsi une normalisation de la taille des images n'est pas nécessaire. Ces caractéristiques modélisent des comportements géométriques des caractères. Nous utilisons donc cet ensemble mais nous avons ôté trois des caractéristiques proposées : les 2 correspondant au nombre de transitions arrière-plan / écriture (dans la fenêtre glissante et entre les lignes de base) et la caractéristique de position du centre de gravité de la fenêtre glissante (au-dessus des lignes de bases, en dessous ou entre les deux). Ces trois caractéristiques ont été enlevées car leurs valeurs sont discrètes et ne

pourraient être modélisées correctement par des distributions Gaussiennes. Ceci nous permet d'éliminer le paramètre n_{cells} de l'extraction de caractéristiques, dont seules 2 de ces 3 caractéristiques éliminées dépendaient. Ainsi, les caractéristiques que nous calculons ne dépendent plus que de la largeur de la fenêtre w et du chevauchement δ (et, par définition, des lignes de base de l'image).

Nous avons modifié les calculs proposés par El-Hajj pour l'extraction sur images binaires afin de travailler avec les niveaux de gris. Soit $r(i)$ la somme des pixels inverses (c'est-à-dire 255 - pixel) de la ligne i de la fenêtre courante :

$$r(i) = \sum_{j=1}^{w} 255 - I(i,j) \tag{2.1}$$

On utilise les pixels « inverses » afin de donner plus de poids aux pixels d'écriture, foncés, dont la valeur en niveau de gris est faible par rapports aux pixels de fond, clairs, dont la valeur est proche ou égale à 255.

L'ensemble des caractéristiques géométriques et statistiques calculées est défini ci-après (d'après le paragraphe 3.4.1 du manuscrit de Thèse de El-Hajj [2]) :

f_1 : densité des pixels d'écriture dans la fenêtre.

$$f_1 = \frac{1}{n_l \times w} \sum_{i=1}^{n_l} r(i) \tag{2.2}$$

f_2 : différence de position verticale entre le centre de gravité g de la fenêtre courante et celui de la fenêtre précédente. Pour la première fenêtre la différence se calcule avec la position de la ligne de base basse.

f_3 : différence de position verticale entre le centre de gravité g de la fenêtre courante et la position de la ligne de base basse.

Le calcul du centre de gravité g d'une fenêtre s'effectue ainsi :

$$g = \frac{\sum_{i=1}^{n_l} i \times r(i)}{\sum_{i=1}^{n_l} r(i)} \tag{2.3}$$

Si i_{lb} est la position verticale de la ligne de base basse, on a donc :

$$f_2 = g(t) - g(t-1) \tag{2.4}$$

$$f_3 = \frac{g - i_{lb}}{n_l} \tag{2.5}$$

f_4 : densité des pixels d'écriture au-dessus de la ligne de base basse.

f_5 : densité des pixels d'écriture en-dessous de la ligne de base basse.

$$f_4 = \frac{1}{n_l \times w} \sum_{i=1}^{i_{lb}} r(i) \tag{2.6}$$

$$f_5 = \frac{1}{n_l \times w} \sum_{i=i_{lb}+1}^{n_l} r(i) \tag{2.7}$$

f_6 à f_{11} : caractéristiques de configurations locales (concavités) relatives à l'ensemble de la fenêtre.

f_{12} à f_{17} : caractéristiques de configurations locales relatives aux pixels entre les lignes de base haute et basse.

Les caractéristiques de configurations loncales, ou concavités, sont calculées en comptant, dans la zone concernée, le nombre de pixels correspondant à certaines configurations (représentées sur la Figure 2.1) et en normalisant par la hauteur de la zone. Ainsi, si f_6 est la caractéristique représentant le nombre de configurations *haut-gauche* dans la fenêtre et f_{15} est celle pour le nombre de configurations *bas-droit* entre les lignes de base alors, après une binarisation de l'image par la méthode Otsu [126], on compte :

$$\begin{aligned} C_{hg,fen} &= \text{nombre de pixels dans la fenêtre correspondant} \\ &\quad \text{à la configuration haut-gauche} \\ C_{bg,med} &= \text{nombre de pixels dans la zone comprise entre les lignes de base} \\ &\quad \text{correspondant à la configuration bas-gauche} \end{aligned} \tag{2.8}$$

Puis :

$$f_6 = \frac{C_{hg,fen}}{n_l} \tag{2.9}$$

$$f_{15} = \frac{C_{bg,med}}{i_{ub} - i_{lb}} \tag{2.10}$$

(i_{ub} est la position verticale de la ligne de base haute)

f_{18} à f_{18+w-1} : densité des pixels d'écriture dans chaque colonne j de la fenêtre

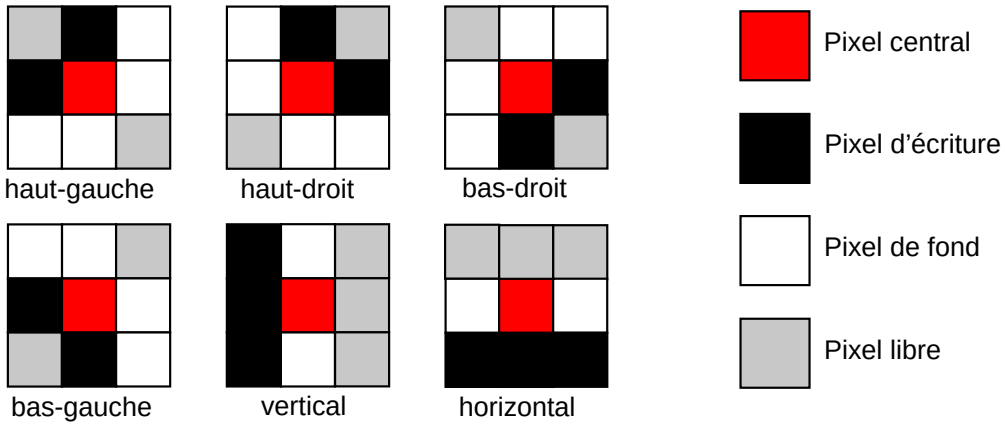


FIGURE 2.1 – Configurations locales proposées par El-Hajj [39] pour le calcul de caractéristiques géométriques.

$(1 \leq j \leq w)$.

$$f_{18+j-1} = \frac{1}{n_l} \sum_{i=1}^{n_l} 255 - I(i,j) \tag{2.11}$$

Caractéristiques directionnelles : Nous avons souhaité apporter une dimension statistique à notre extraction de caractéristiques en associant aux caractéristiques géométriques un autre type de caractéristiques : les histogrammes de gradient. Ces caractéristiques, présentées dans la Section 1.2.3, calculent l'histogramme des directions des gradients de cellules découpées dans la fenêtre glissante. Discrétisant l'espace des angles $[-\pi, \pi[$ dans 8 directions possibles, le nombre de caractéristiques extraites est donc égal à 8 fois le nombre de cellules de la fenêtre. Etant donné que les caractéristiques géométriques sont déjà nombreuses, nous avons choisi de ne pas découper la fenêtre en cellules et donc d'accumuler dans un seul histogramme l'ensemble des directions présentes dans la fenêtre. Ces deux types de caractéristiques sont extraites avec des fenêtres glissantes différentes dans les travaux qui les présentent. Nous avons donc ajusté le parcours de la fenêtre des caractéristiques d'histogramme de gradient afin qu'il suive celui des caractéristiques géométriques.

Résumé : Ainsi, pour une fenêtre de largeur w pixels, de hauteur n_{lines} pixels (la hauteur de l'image) et de décalage avec la fenêtre suivante δ pixels, les caractéristiques sont :

- $17 + w$ caractéristiques géométriques (les w caractéristiques correspondent à la valeur moyenne des pixels dans chaque colonne de la fenêtre)
- 8 caractéristiques d'histogrammes de gradient.

Nous obtenons donc un ensemble de $25 + w$ caractéristiques extraites par le système de fenêtres glissantes sur les images. Nous avons tracé sur la Figure 2.2 la distribution empirique de chacune des caractéristiques. Les distributions ont été calculées sur les caractéristiques extraites des images de mots la base Rimes [70] (environ 50000 images) avec une fenêtre glissante de largeur $w = 9$. D'après leur allure, il semble raisonnable d'approximer ces densités avec des mélanges de distributions Gaussiennes. Ainsi ces caractéristiques sont cohérentes avec notre approche utilisant des HMMs gaussiens.

2.1.2 Caractéristiques dynamiques : régression du premier et du second ordre

Le principe des fenêtres glissantes pour extraire des caractéristiques d'une image permet de capturer le côté séquentiel de l'écriture. Les fenêtres étant choisies chevauchantes en général, les liens inter et intra-caractères sont aussi pris en compte mais de manière réduite. Nous avons alors souhaité augmenter l'importance de l'information sur l'environnement d'un caractère avec une dérivation temporelle, c'est-à-dire au niveau de la succession des fenêtres lors du parcours de l'image. La dynamique des fenêtres glissantes entourant la fenêtre courante est prise en compte par le calcul de caractéristiques dynamiques.

Certains systèmes de l'état de l'art utilisent aussi le principe de la dérivation dans l'extraction de caractéristiques. Par exemple [33] calcule la différence $\boldsymbol{o_t - o_{t-1}}$ et l'ajoute à son vecteur de caractéristiques. L'approche que nous proposons utilise une régression au lieu d'une simple dérivation. La régression permet d'obtenir plus précisément la *pente* du gradient et est plus robuste [9].

Notons p l'abscisse en pixels de la fenêtre courante et K le nombre de fenêtres participant au calcul des caractéristiques dérivées. La dérivation prend alors en compte les fenêtres situées aux positions $p - \delta * i$ et $p + \delta * i$, pour $1 \leq i \leq K$, comme illustré sur la Figure 2.3. La dérivation est calculée par une régression (K représente donc la profondeur de la régression). En reconnaissance de la parole, les régressions du premier et du second ordre sont connues respectivement comme les coefficients delta et delta-delta des caractéristiques cepstrales.

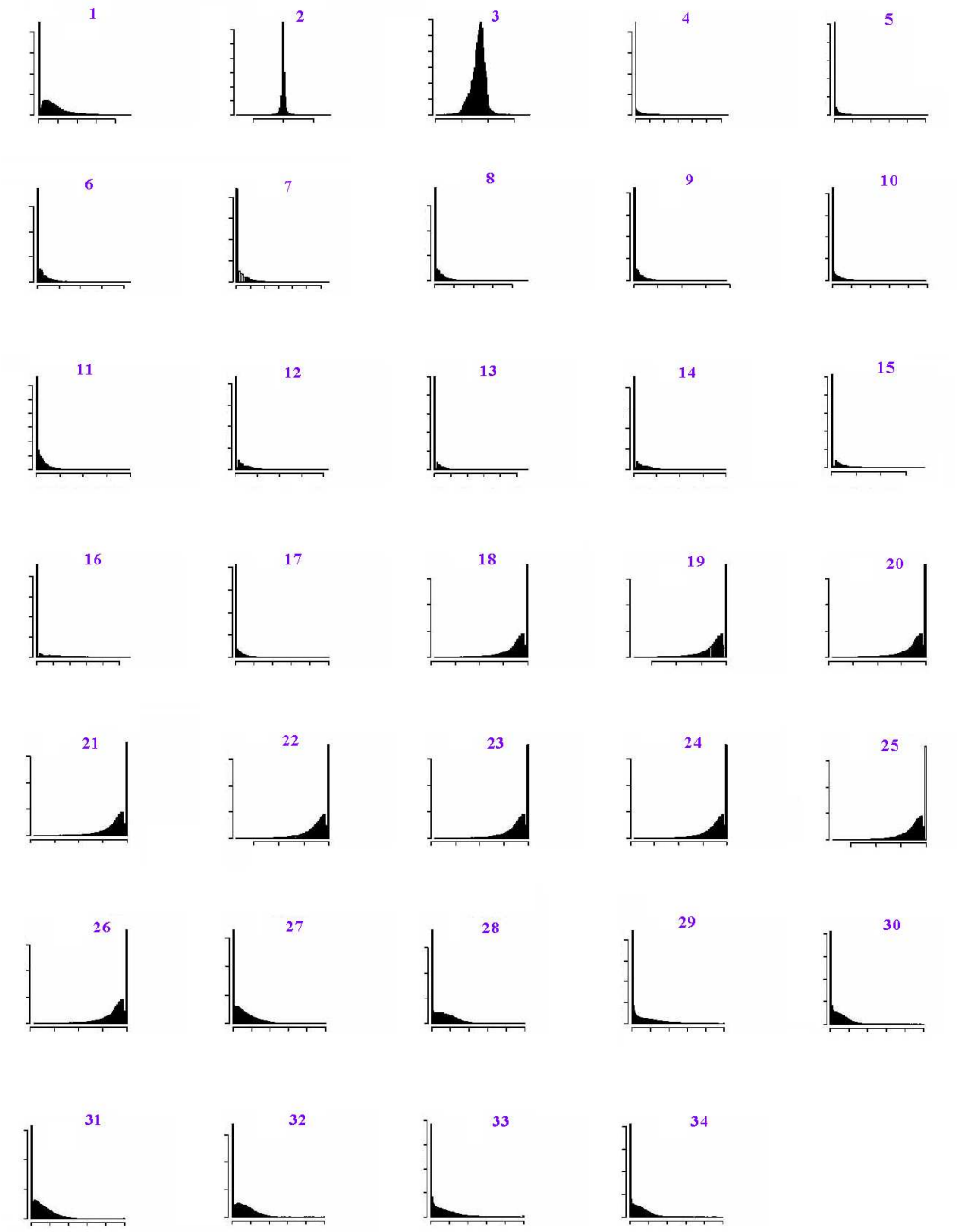


FIGURE 2.2 – Distribution des valeurs des $25+w$ caractéristiques ($w = 9$). Les caractéristiques 1 à 5 et 18 à 26 sont statistiques, les caractéristiques 6 à 17 sont géométriques (caractéristiques de concavité) et les caractéristiques directionnelles sont celles de 27 à 34.

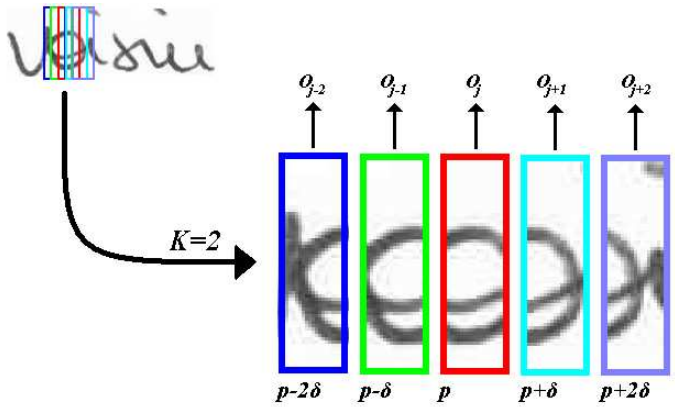


FIGURE 2.3 – Illustration du calcul de la régression sur les caractéristiques extraites par fenêtres glissantes.

Soit $\boldsymbol{o_j}$ le vecteur de caractéristiques de la fenêtre courante (à la position p) et $\boldsymbol{o_{j+i}}$ (resp. $\boldsymbol{o_{j-i}}$) celui de la fenêtre décalée de $i * \delta$ (resp. $-i * \delta$) pixels, située à l'abscisse $p + i * \delta$ (resp. $p - i * \delta$). La régression du premier ordre du vecteur de caractéristiques $\boldsymbol{o_j}$ s'écrit :

$$\Delta \boldsymbol{o_j} = \frac{\sum_{i=1}^{K} i(\boldsymbol{o_{j+i}} - \boldsymbol{o_{j-i}})}{2\sum_{i=1}^{K} i^2}. \tag{2.12}$$

$2 * K$ vecteurs o_k sont donc pris en compte pour le calcul des caractéristiques dynamiques.

La régression du second ordre $\Delta\Delta \boldsymbol{o_j}$ se calcule simplement à partir de celle du premier ordre, en remplaçant $\boldsymbol{o_j}$ par $\Delta \boldsymbol{o_j}$ dans l'équation 2.12 :

$$\Delta\Delta \boldsymbol{o_j} = \frac{\sum_{i=1}^{K} i(\Delta \boldsymbol{o_{j+i}} - \Delta \boldsymbol{o_{j-i}})}{2\sum_{i=1}^{K} i^2}. \tag{2.13}$$

Le vecteur de caractéristiques final $\tilde{\boldsymbol{o}}_{\boldsymbol{j}}$, en entrée du système, est alors la concaténation du vecteur initial $\boldsymbol{o_j}$ et de ses régressions :

$$\tilde{\boldsymbol{o}}_{\boldsymbol{j}} = \begin{bmatrix} o_1 \\ o_2 \\ \dots \\ o_{25+w} \\ \Delta o_1 \\ \Delta o_2 \\ \dots \\ \Delta o_{25+w} \end{bmatrix} \quad \text{ou} \quad \tilde{\boldsymbol{o}}_{\boldsymbol{j}} = \begin{bmatrix} o_1 \\ o_2 \\ \dots \\ o_{25+w} \\ \Delta o_1 \\ \Delta o_2 \\ \dots \\ \Delta o_{25+w} \\ \Delta\Delta o_1 \\ \Delta\Delta o_2 \\ \dots \\ \Delta\Delta o_{25+w} \end{bmatrix}$$

2.1.3 Analyse en composantes principales

Nous avons présenté dans cette Section les caractéristiques que nous utiliserons dans notre système de reconnaissance de mots manuscrits. Nous verrons, lors de la présentation de notre modélisation, que nous supposons que les dimensions des caractéristiques sont indépendantes entre elles (matrice de covariance diagonale pour les HMMs, voir Section 2.2). Cette hypothèse impose une restriction très forte sur nos caractéristiques. Un moyen répandu d'assurer l'inter-indépendance et l'orthogonalité des dimensions de notre système est d'appliquer une analyse en composantes principales (ACP ; en anglais *PCA, Pincipal Component Analysis*) sur nos données.

L'ACP a en outre l'avantage de réduire la dimension des données en perdant le moins d'information possible : les observations $\boldsymbol{o_j}$, de dimension n, sont projetées sur un sous-espace de dimension $n' < n$ dont chaque dimension correspond à une combinaison linéaire des dimensions de l'espace de départ. Chaque dimension choisie pour la projection est une *composante principale* et on peut choisir leur nombre. Les composantes sont rangées par ordre d'importance d'information qu'elles contiennent et les plus faibles sont éliminées en priorité. Nous verrons au Chapitre 4 qu'il est possible de diviser par deux le nombre de dimensions tout en conservant plus de 90% de l'information utile.

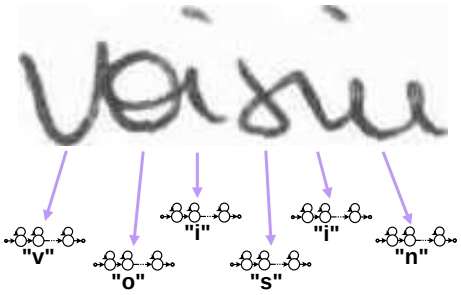


FIGURE 2.4 – Le modèle d'un mot est la concaténation des modèles HMMs des caractères le composant.

2.2 Apprentissage et décodage avec des HMMs gaussiens

Les modèles de Markov cachés sont une méthode depuis longtemps éprouvée et connue pour la modélisation de séquence, comme présenté dans la Section 1.3.1. Dans ce chapitre, nous présentons notre manière d'utiliser des HMMs continus pour la reconnaissance de mots manuscrits. Ceci nous permet de poser les bases d'un système robuste à partir desquelles nous développerons au prochain chapitre notre système original de modèles en contexte. Nous utilisons le logiciel HTK [169] pour l'apprentissage et le décodage de nos modèles.

Dans notre système, nous utilisons une approche par segmentation implicite et les HMMs modélisent des caractères. Les modèles de mots sont construits en concaténant les modèles des caractères qui les composent, comme illustré sur la Figure 2.4. Cette modélisation permet une fois un *alphabet* appris (comme le latin, l'arabe ou le cyrillique) d'utiliser n'importe quel lexique pour le décodage, pourvu qu'il soit basé sur le même alphabet. Ainsi, contrairement aux méthodes holistiques pour lesquelles le lexique de mots est prédéfini (et de taille restreinte), le lexique dans notre cas est libre. En général, on suppose que les données (de dimension n : n caractéristiques sont extraites) ont leurs dimensions indépendantes entre elles. Ainsi les matrices de covariance utilisées sont diagonales.

2.2.1 Apprentissage

Tous les modèles de caractères suivent la topologie de Bakis avec S états émetteurs (transitions gauche-droite et saut d'état autorisé). Cette topologie est illustrée

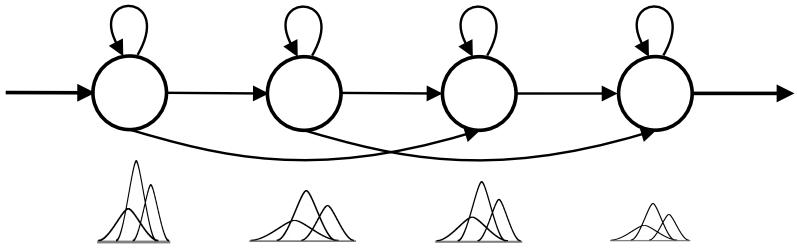


FIGURE 2.5 – Illustration de la topologie de type Bakis utilisée pour nos modèles, où chaque état est représenté par un mélange de distributions gaussiennes.

sur la Figure 2.5.

La densité de probabilité des observations est un mélange de distributions gaussiennes. Le nombre optimal N_G de gaussiennes dans les mélanges est calculé sur une base de validation. Le mélange final est obtenu par incrémentation du nombre de gaussiennes dans chaque état, depuis 1 gaussienne par état jusqu'à N_G. A chaque étape d'augmentation du nombre de gaussiennes, une fois que le nombre désiré de gaussiennes a été ajouté, les paramètres sont ré-estimés avec l'algorithme Baum-Welch.

La procédure d'ajout de gaussienne(s) à un mélange est décrite dans les Algorithmes 1 et 2. Dans l'Algorithme 1, σ_m représente la racine carrée des termes diagonaux de Σ_m : pour $i \in [1 \dots n]$, $\sigma_{m,i} = \sqrt{\Sigma_{m,i,i}}$ (n est le nombre de caractéris-

tiques extraites des images).

Algorithme 1: Procédure `Ajout1Gaussienne` pour ajouter une gaussienne à un mélange contenant n gaussiennes

1 Trouver la gaussienne $m : \mathcal{N}(\mu_m; \Sigma_m)$ au poids w_m le plus fort parmi les n gaussiennes;
$m = \arg \max_{j \in [1..n]} w_j$;

2 Scinder la gaussienne m en deux gaussiennes m_1 et m_2;
$m_1 : \mathcal{N}(\mu_m + 0.2\sigma_m; \Sigma_m)$ et $m_2 : \mathcal{N}(\mu_m - 0.2\sigma_m; \Sigma_m)$;
$w_{m_1}, w_{m_2} \leftarrow w_m/2$;

3 Supprimer m et ajouter m_1 et m_2 au mélange ;

Algorithme 2: Procédure d'apprentissage avec incrémentations successives du nombre de gaussiennes par mélange

```
Initialisation : n ← 1 ;
tant que n < N_G faire
    si n < 10 alors
        Ajout1Gaussienne ;
        n ← n + 1 ;
        Ré-estimation des paramètres avec l'algorithme Baum-Welch ;
    sinon si n < 20 alors
        répéter
            Ajout1Gaussienne ;
        jusqu'à n ← n + 2;
        Ré-estimation des paramètres avec l'algorithme Baum-Welch ;
    sinon
        répéter
            Ajout1Gaussienne ;
        jusqu'à n ← n + 4;
        Ré-estimation des paramètres avec l'algorithme Baum-Welch ;
    fin
fin
```

2.2.2 Décodage

Le décodage se fait avec l'algorithme de Viterbi, présenté en Section 1.3.1. Etant donné que nous travaillons sur une tâche de reconnaissance de mots, nous choisissons de ne donner aucune connaissance a priori sur la vraisemblance des mots du vocabulaire. Ainsi, les mots du test ont tous la même probabilité.

2.3 Adaptation du nombre d'états par caractère à la morphologie du caractère

Nous présentons dans cette Section une dernière amélioration de notre système à base de HMMs sans contextes : l'adaptation de la morphologie du modèle HMM à la longueur du caractère.

La Figure 2.6 illustre le problème discuté ici : la taille d'un caractère peut être plus ou moins grande et les majuscules ne sont pas les seuls caractères larges. Ainsi, les lettres m de « mois » et C de « Cliente » sont très longues, alors que les lettres i de « mois » et de « Cliente » et l de « Cliente » et de « la » sont au contraire très courtes. Il ne serait donc pas justifié de modéliser les caractères C et m avec le même nombre d'états que i ou l.

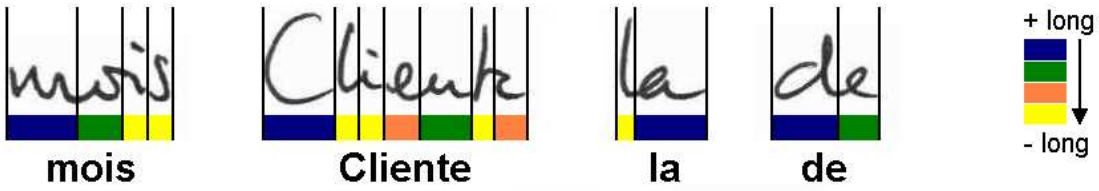


FIGURE 2.6 – Illustration des différences de longueurs des caractères latins

Les HMMs à durée variable ont été introduits par Chen *et al.* [22] et Kundu *et al.* [87] pour des systèmes de reconnaissance de mots à base de segmentation en caractères. Depuis, ils ont été introduits dans les reconnaisseurs HMMs sans segmentation [171, 146, 33]. On peut déterminer de plusieurs manières la morphologie des HMMs des caractères.

Empiriquement, il est possible de calculer la longueur moyenne $L(C)$ (en nombre de vecteurs d'observation) d'un caractère C dans la base de données et de dire que le nombre d'états de ce caractère est égal à sa longueur moyenne.

Soient $words_C$ l'ensemble des mots contenant au moins une fois le caractère C, $|w_C|$ le nombre de caractères C dans le mot w, $w \in words_C$ (il peut y avoir 0, 1 ou plusieurs caractères C dans un même mot) et $|w|$ et $L(w)$ sa longueur (respectivement en termes de nombre de caractères et de vecteurs d'observation). $|C|$ est le nombre total de caractères C dans la base de données. Alors :

$$L(C) = \frac{1}{|C|} \sum_{w \in words_C} \frac{L(w)\,|w_C|}{|w|} \tag{2.14}$$

Ce calcul donne une bonne idée des statistiques de la base de données de travail et permet de retrouver les valeurs observables graphiquement, comme par exemple de fortes valeurs pour les caractères m et C et faibles pour i et l (cf Figure 2.6). Il est souvent utilisé pour la modification de longueur de modèle. Ainsi, Zimmermann et Bunke [171] décrivent deux méthodes basées sur ce calcul empirique pour une topologie HMM sans saut d'état. Cette topologie leur permet d'établir un nombre d'états minimal pour chaque caractère. Les deux méthodes calculent $L(C)$ pour chaque caractère, puis la première choisit une fraction de ce nombre (dont la valeur est optimisée sur une base de validation) pour déterminer la topologie des modèles, alors que la deuxième utilise des quantiles. Ces deux méthodes, par leur utilisation d'une topologie sans saut d'état, choisissent cependant mal les topologies des caractères les plus courts. Dreuw *et al.* [33], quant à eux, utilisent directement les valeurs $L(C)$ obtenues pour leur topologie et permettent les sauts d'états afin de faciliter la modélisation des caractères courts.

Itérativement, il est aussi possible de déterminer les variations des longueurs des caractères. Ainsi Schambach [146] initialise les HMMs de ses caractères avec un nombre d'états identiques. Puis, à chaque itération, chaque modèle est dupliqué dans trois sous-modèles : l'un est identique, le deuxième a un état en plus et le troisième un état en moins. A la fin de l'itération, le sous-modèle ayant la probabilité d'émission la plus forte est choisi. Même si cette méthode donne de bons résultats, elle a le désavantage d'être très fastidieuse.

Statistiquement, enfin, on peut déterminer la longueur optimale d'un modèle HMM de caractère. Nous proposons ici une méthode similaire à celle de [146] mais dont la convergence est beaucoup plus rapide. Cette solution maximise directement

la morphologie des HMMs au lieu de travailler d'abord sur les caractères, comme la solution empirique.

Les HMMs des caractères sont d'abord initialisés avec un nombre d'états identiques (topologie de Bakis et 1 seule Gaussienne par état) et plusieurs itérations de l'algorithme Baum-Welch sont effectuées afin d'obtenir des modèles stables. Durant la dernière itération de l'algorithme d'apprentissage, les statistiques de passage dans chaque état sont conservées :

$$\Gamma(s) = \sum_{t=1}^{T} \gamma_s(\boldsymbol{o}_t). \tag{2.15}$$

où $\gamma_s(\boldsymbol{o_t})$ est la probabilité a posteriori que l'observation $\boldsymbol{o_t}$ soit générée par l'état s. La somme est effectuée sur l'ensembre des observations disponibles, $\boldsymbol{O_T} = \boldsymbol{o_1}, \ldots, \boldsymbol{o_T}$. Nous notons S_C l'ensemble des états du HMM du caractère C. La longueur statistique du caractère, $L_s(C)$, se calcule :

$$L_s(C) = \frac{\sum_{s \in S_C} \Gamma(s)}{|C|} \tag{2.16}$$

Le nombre d'états du caractère C est alors modifié et égal à la valeur entière de $L_s(C)$. En une itération, on obtient donc directement les nouvelles longueurs de modèles pour chaque caractère. Cette manipulation est alors répétée, jusqu'à ce que le changement de topologie ne soit plus significatif ($S - L_s(C) < \tau$, τ étant choisi sur une base de validation).

Conclusion du chapitre 2

Nous avons présenté dans ce chapitre un système de reconnaissance de mots manuscrits basé sur des HMMs gaussiens pour modéliser des caractères. Une première étape consiste à extraire les caractéristiques des images par des fenêtres glissantes. Les caractéristiques extraites sont un mélange de caractéristiques géométriques, statistiques et directionnelles pour à la fois intégrer les particularités de l'écriture mais aussi les caractéristiques des distributions de pixels dans une image. Nous avons introduit par la suite des caractéristiques dynamiques par l'ajout de régression aux

caractéristiques initiales. Ceci nous a permis d'ajouter des informations sur les fenêtres avoisinantes dans la fenêtre courante. Ensuite, nous avons proposé d'effectuer une analyse en composantes principales de l'espace des caractéristiques afin de projeter les données sur un nombre réduit de dimensions et donc de réduire la dimensionnalité du problème ainsi que d'assurer la légitimité de l'utilisation d'une matrice de covariance diagonale dans les HMMs gaussiens (inter-indépendance des dimensions des observations). L'ACP assure une réduction de complexité tout en conservant un maximum d'information.

Dans une deuxième partie, nous avons présenté notre procédure d'apprentissage incrémental des HMMs de caractères, ainsi que le décodage. Les observations en entrée du système sont alors soit les caractéristiques calculées sur les images, soit leur projection post-ACP. Enfin, nous avons proposé dans une troisième partie d'améliorer notre modélisation en adaptant de manière originale le nombre d'états pour chaque HMM de caractère.

Dans ce chapitre, nous avons donc mis en place un système à base de HMMs sans contexte et avons travaillé sur les caractéristiques et l'apprentissage afin de l'optimiser. Ce système est notre système de référence. Il sert à la fois de base pour la construction de notre système original à base de modèles de caractères en contexte présenté au chapitre suivant mais aussi de point de comparaison avec ce dernier.

Chapitre 3

Système de reconnaissance de mots manuscrits à base de HMMs Gaussiens en contexte

Introduction

Pour notre système de reconnaissance d'écriture manuscrite, nous avons choisi l'approche analytique à base de HMMs gaussiens et l'utilisation de fenêtres glissantes pour l'extraction des caractéristiques. Dans le Chapitre 2, nous avons présenté un système *générique* de reconnaissance de mots manuscrits à base de HMMs gaussiens de monographes : un modèle HMM modélise un caractère. Nous avons cherché à optimiser ce système en agissant sur les caractéristiques, la morphologie des modèles et le calcul des mélanges de distributions gaussiennes dans chaque état.

Dans ce chapitre, nous proposons une nouvelle optimisation du système, en utilisant une méthode originale pour affiner les modèles de caractères : modéliser chaque monographe en fonction de son contexte dans le mot. Le contexte d'un caractère est représenté par les deux caractères l'entourant : le caractère le précédant et le caractère le suivant. Les nouveaux modèles sont appelés trigraphes. Avec cette approche, nous cherchons à augmenter le degré de précision des modèles. Une telle modélisation implique cependant une augmentation considérable du nombre de paramètres à calculer pour le système. Celui-ci nécessite alors un très grand nombre de données d'apprentissage afin d'estimer correctement tous les contextes de tous les caractères. En pratique, cela n'est pas toujours possible. Si certains contextes sont

très fréquents (par exemple le caractère q est – presque – toujours suivi d'un u en français), la plupart d'entre eux ne possèderont pas assez d'exemples dans l'ensemble d'apprentissage pour être appris correctement.

Plusieurs solutions existent pour contourner ce problème, toutes basées sur le principe de regroupement de certains paramètres des modèles. Dans notre système, nous proposons un partage de paramètres au niveau des états. L'originalité de ce partage est qu'il est effectué non pas par calcul de distances entre les données mais avec l'aide entre autres choses de nos connaissances sur la morphologie des contextes (en anglais, *knowledge-driven clustering*). Ces connaissances nous ont permis de créer un ensemble de questions binaires pour le latin et pour l'arabe. Celles-ci servent à la construction d'arbres de décision dont les feuilles correspondent aux clusters d'états partagés.

Ce chapitre est consacré à la description des étapes de construction de notre système original. Nous présentons dans un premier temps notre principe de modélisation de caractères en fonction de leur contexte (Section 3.1), puis nous discutons en détail de la contruction et de l'utilisation des trigraphes (Section 3.2). Les arbres binaires de décision pour le clustering d'états, au coeur de notre système, sont présentés en détail dans la Section 3.2.2 et leur intérêt pour le décodage de mots avec un lexique indépendant de l'ensemble d'apprentissage est discuté en Section 3.2.3.

3.1 Les trigraphes ou modèles en contexte

3.1.1 Intérêt de la modélisation en contexte

Notre objectif étant de construire des HMMs de caractère de plus en plus précis, nous avons dans un premier temps souhaité évaluer la précision de la modélisation par monographes. Pour cela, nous avons calculé la variance de la vraisemblance des monographes. Une vraisemblance très variable est un indicateur d'un modèle peu précis de monographe et nous souhaitons éviter ce cas de figure dans notre système. Dans le paragraphe qui suit, nous avons évalué cette variance sur les caractères latins appris sur la base Rimes.

Les vraisemblances de chaque caractère sont calculées par alignement avec l'algorithme de Viterbi de chaque monographe sur les données. La variance de vraisem-

blance d'un monographe C est calculée de la manière suivante :

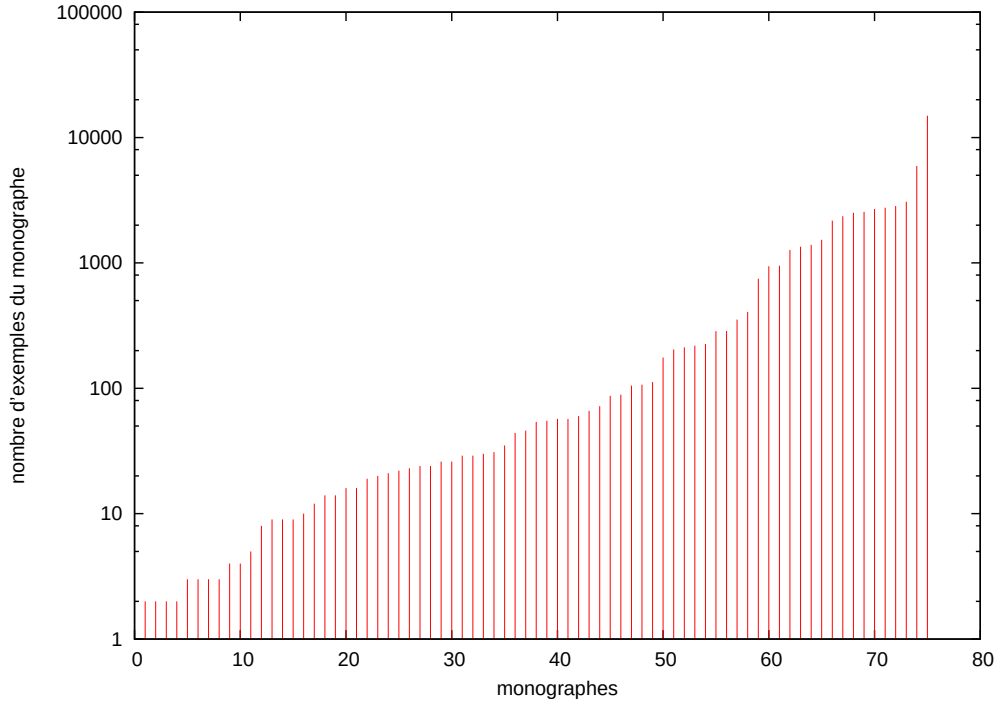
$$\sigma^2_{L_C} = \frac{1}{|C|} \sum_{j=1}^{|C|} (L_{j,C} - \hat{L}_C)^2 \tag{3.1}$$

$|C|$ est le nombre de fois où le caractère C est rencontré dans l'ensemble d'apprentissage, $L_{j,C}$ est la vraisemblance du $j^{ème}$ exemple de C ($L_{j,C}$ est normalisée par le nombre de vecteurs d'observations associés au $j^{ème}$ exemple) et $\hat{L}_C$ est la moyenne empirique des vraisemblances de C.

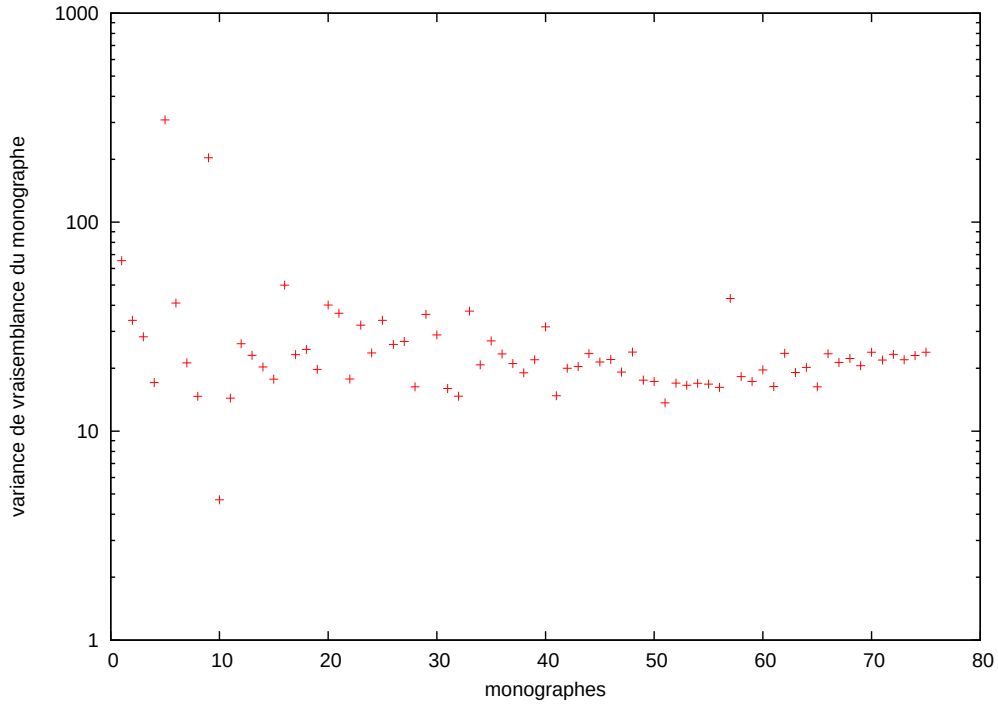
La Figure 3.1.a montre le nombre d'exemples de chaque monographe dans l'ensemble d'apprentissage. La Figure 3.1.b donne quant à elle la variance de la vraisemblance des monographes de la base d'apprentissage de Rimes (avec les monographes rangés dans le même ordre que ceux de la Figure 3.1.a). L'observation de la Figure 3.1 permet de comprendre que la variance de la vraisemblance d'un monographe n'est pas nécessairement liée au nombre d'exemples de celui-ci. Ainsi, le caractère 57 (qui correspond à g) a une forte variance (environ 45) comparativement à d'autres monographes avec un nombre d'exemples similaires (environ 300) : j, f et le symbole de l'apostrophe (sur la Figure 3.1, monographes 55, 56 et 58) ont une variance trois fois plus faible (environ 17).

Cette étude motive notre choix de modéliser plus finement les monographes. Il reste cependant à choisir quels modèles privilégier pour une nouvelle estimation. En effet, alors qu'il serait par exemple intéressant de modéliser plus finement le caractère g afin d'obtenir des « sous-modèles » plus précis (à variance de vraisemblance plus faible), on ne souhaiterait pas démutliplier les modèles des caractères avec peu d'exemples (moins de 50), même s'ils ont une très large variance (plus de 100 pour les monographes 5 et 9, correspondant à W et q).

On ne peut donc directement utiliser les données collectées par l'étude de la variance de la vraisemblance pour choisir les monographes à modéliser plus finement. Nous proposons plutôt d'agir à un niveau plus précis pour ce choix : au niveau des états des HMMs de caractères. La méthode que nous présentons dans les Sections suivantes permet d'établir quels états de quels HMMs nécessitent plus de précision.



(a) nombre d'exemples de chaque monographe



(b) variance de vraisemblance des monographes

FIGURE 3.1 – Comparaison de la variance des monographes (b) et de leur nombre d'exemples dans la base d'apprentissage (a). Les monographes sont rangés dans le même ordre dans (a) et (b).

(a) i et n dans le mot *Monsieur*

(b) i et n dans le mot *distingué*

FIGURE 3.2 – Illustration de l'influence du contexte d'un caractère en écriture. Les mots *Monsieur* et *distingué* ont été écrits par la même personne mais, dans chacun, la forme des caractères i et n change en fonction des caractères adjacents.

3.1.2 Présentation

Nous avons vu dans la Section 2.1.2 comment introduire une notion de contexte au niveau des caractéristiques avec une régression. Nous souhaitons maintenant exploiter l'observation présentée sur la Figure 3.2 : un caractère a une forme variable suivant sa position dans un mot et les lettres qui l'entourent. Le système présenté ici prend en compte cette idée en modélisant un caractère en fonction de son contexte gauche et droit : un *trigraphe*.

Une telle modélisation est utilisée en parole pour simuler l'effet de co-articulation et les modèles sont nommés triphones [91]. Mais, comme nous l'avons mentionné dans l'introduction, très peu de travaux de reconnaissance de l'écriture abordent les modèles en contexte et explorent leurs avantages. Il est vrai que l'inconvénient majeur de la modélisation des caractères en fonction de leur contexte est que le nombre de paramètres des HMMs de tous les trigraphes possibles à modéliser est très élevé et que, pour un vocabulaire de taille moyenne (4000 mots), le manque de données pour l'apprentissage devient un problème patent. Il existe pourtant des méthodes pour réduire ce nombre, basées sur le partage des paramètres entre modèles ou entre états par exemple. Nous avons choisi d'utiliser une méthode de clustering sur les positions d'états afin de pallier ce problème et d'inclure efficacement des trigraphes dans notre système.

Le prétraitement des images en entrée et l'extraction des caractéristiques (géométriques, statistiques et dynamiques) sont identiques à ceux du système générique décrit dans le Chapitre 2. L'apport majeur de notre système, outre la modélisation de caractères en fonction de leur contexte, est l'usage d'un processus de partage de

paramètres utilisant un clustering d'états basé sur des arbres de décision binaires. Ces derniers sont construits à partir de questions originales que nous avons définies pour le latin et pour l'arabe après observation de la morphologie des caractères présents dans les lexiques dont nous disposions.

3.1.3 Modélisation des caractères en contexte

Le principe de modélisation de caractères en fonction de leur contexte est assez naturel car il nous paraît logique que la forme d'un caractère varie en fonction des lettres qui l'entourent. Cette modélisation consiste à définir un mot non pas comme une succession de caractères mais comme une succession de caractères avec leur contexte. Les monographes du Chapitre 2 sont remplacés par des trigraphes, où le caractère central est influencé par le caractère le précédant (*contexte précédent* ou contexte gauche en latin) et celui le suivant (*contexte suivant* ou contexte droit en latin). D'un point de vue syntaxique, nous notons un *contexte précédent* avec un signe moins '-' et un *contexte suivant* avec un signe plus '+'. Sur la Figure 3.2.a, le modèle du n de *Monsieur* devient le modèle du trigraphe o-n+s.

Pour commencer l'apprentissage de ces nouveaux modèles, l'ensemble des trigraphes de la base d'apprentissage est listé. Typiquement, pour un lexique d'apprentissage latin de l'ordre de 4000 mots, on peut compter plus de 5000 trigraphes différents. L'apprentissage des modèles de trigraphes pourrait être effectué directement à partir des trigraphes précédemment listés, chacun étant appris sur les exemples qu'il possède dans la base d'apprentissage. Cependant, cela s'avère difficile car certains trigraphes ont trop peu d'exemples pour être appris correctement. De plus, le nombre de paramètres à calculer croît lorsque le nombre de Gaussiennes dans les mélanges des densités de probabilités augmente. Ainsi, pour une base d'apprentissage avec 5000 trigraphes différents environ, si la topologie de modèles est de S états émetteurs et N_G Gaussiennes dans chaque mélange avec S et N_G de l'ordre de 10, le nombre de paramètres à calculer monte à presque un demi-million, ce qui est tout à fait prohibitif étant donné la quantité de données disponibles dans la plupart des bases de données publiques d'écriture manuscrite. C'est pourquoi nous considérons dans notre système le partage de paramètres par un clustering d'états : celui-ci permet de pallier à la fois le problème du manque de données d'apprentissage ainsi que celui du trop grand nombre de modèles à calculer.

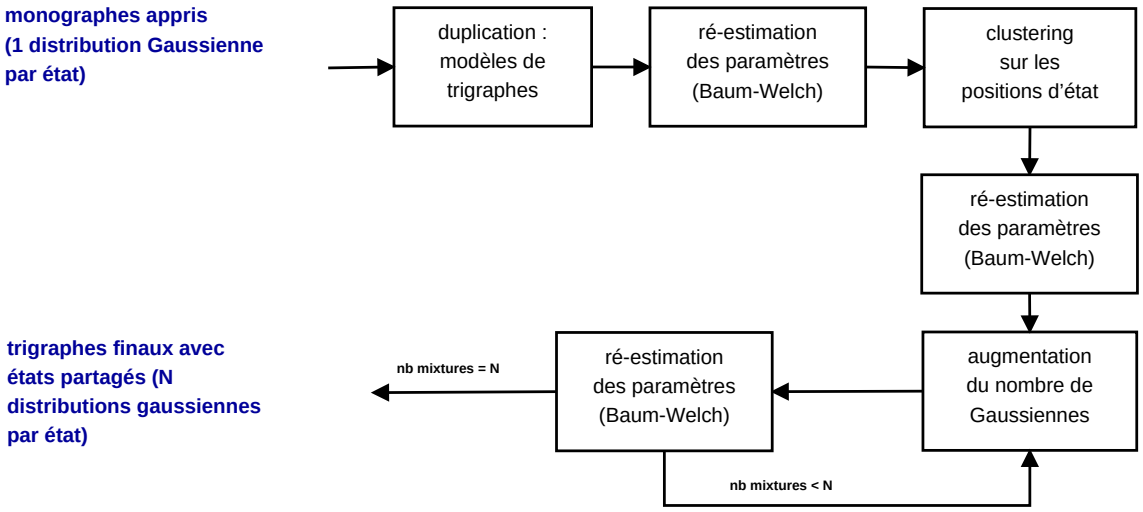


FIGURE 3.3 – Présentation générale du système à base de HMMs de caractères en contexte.

3.2 Apprentissage des trigraphes et partage des paramètres HMMs

3.2.1 Présentation générale

Dans cette Section, nous présentons en détail notre système paramétrant les HMMs de caractères en fonction de leur contexte, schématisé sur la Figure 3.3. Nous détaillons les étapes de mise en place du partage des paramètres entre trigraphes précédant le processus final d'apprentissage.

Le premier apprentissage est celui des monographes du lexique d'apprentissage. Les monographes sont initialisés avec une seule distribution Gaussienne associée à chaque état, puis plusieurs ré-estimations sont effectuées avec l'algorithme Baum-Welch. Les trigraphes sont alors eux-mêmes initialisés en copiant les modèles appris des monographes : pour un monographe donné, tous les trigraphes de la base d'apprentissage centrés sur ce caractère sont listés et le HMM du monographe est copié pour correspondre au modèle initial de chacun des trigraphes. Ensuite, une première ré-estimation (avec 2 itérations) Baum-Welch de *tous* les trigraphes est effectuée. Durant cette étape, nous avons choisi de partager les matrices de transition entre trigraphes avec même caractère central. Nous avons en effet observé qu'une fois la topologie d'un modèle choisie des modifications sur les coefficients de la matrice de transition ont une très faible influence sur la phase de reconnaissance. Considérant

cette information, nous avons alors imposé que les trigraphes ayant le même caractère central (tous les $* - C + *$ pour chaque caractère C appartenant à l'alphabet) partagent leur matrice de transition. Ainsi, le nombre de matrices à calculer se réduit au nombre initial de monographes.

La seconde étape de notre système est le clustering d'états appliqué aux trigraphes. Le principe du clustering état par état est que, pour les trigraphes associés à une lettre centrale donnée, tous les états correspondant à une même position dans le modèle HMM sont soumis à un clustering. Cette étape est illustrée plus en détails sur la Figure 3.4.

Fink et Plötz [48] et Natarajan *et al.* [118] ont montré l'efficacité du regroupement de paramètres au niveau des états pour la modélisation de caractères contextuels. Tous deux calculent un nombre prédéfini de gaussiennes (codebook) pour chaque position d'état de chaque caractère central. Les états piochent ensuite dans le codebook leur étant associé pour élaborer leur modèle. L'attribution des gaussiennes se fait par minimisation de distance euclidienne entre le modèle d'état initial du trigraphe et les gaussiennes du codebook (*data-driven* clustering). Si cette modélisation leur permet de gagner en performance, le gain obtenu n'est cependant pas rentable comparativement à l'augmentation en complexité de leur système (tous les trigraphes ont des modèles différents).

Nous différons de ce qui est proposé dans [48, 118] de deux manières. D'une part, nous calculons des clusters d'état pour chaque position d'état de chaque caractère central donné. Cela nous permet d'adapter le nombre de gaussiennes à définir au nombre réel nécessité. Soit C le caractère central considéré. N_C différents trigraphes existent dans la base d'apprentissage, centrés sur C. Donc pour chaque position d'état i des trigraphes $* - C + *$, $i \in [1 \dots S]$, N_C différents états devraient être calculés. L'utilisation de clusters d'états permet la réduction du nombre de modèles à calculer. Ainsi pour la position d'état numéro i, $n_{i,C} \leq N_c$ clusters sont sélectionnés pour représenter tous les états de la position i, définissant $n_{i,C}$ différents modèles d'états $s_{i,j}$, $j \in [1 \dots n_{i,C}]$. Un état en position i prend alors sa valeur dans l'un des $s_{i,j}$ modèles. Si $S * N_C$ est le nombre d'états à calculer sans clustering, alors le nombre final d'états à apprendre pour modéliser tous les trigraphes centrés sur le caractère C est :

$$\sum_{i=1}^{S} n_{i,C} << S * N_C \tag{3.2}$$

Considérons deux exemples. Le caractère z, qui a peu d'occurences dans l'en-

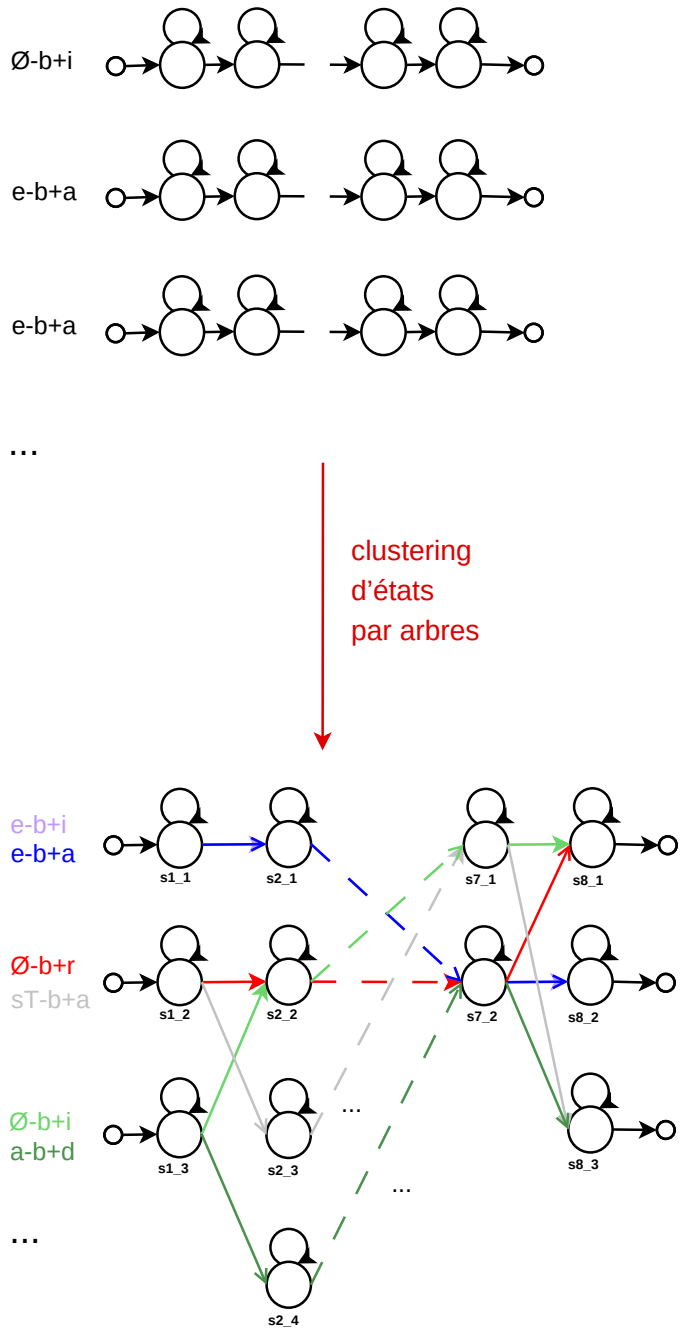


FIGURE 3.4 – Illustration du clustering d'états pour les trigraphes centrés sur la lettre *b*.

semble d'apprentissage, ne possède qu'un seul cluster pour la position d'état numéro 4 des trigraphes $* - z + *$ alors que le caractère central e en a 15 pour cette même position. Si le nombre de gaussiennes par mélange N_G est de l'ordre de 20, alors seules 20 gaussiennes seront calculées pour la position 4 des trigraphes centrés en z et 300 pour ceux centrés en e. Cette modélisation permet donc de mieux gérer le calcul de gaussiennes pour les mélanges comparativement à ce qui est proposé dans [48, 118].

D'autre part, nous différons de la modélisation de Fink et Plötz [48] et Natarajan *et al.* [118] dans l'utilisation d'arbres de décision pour calculer nos clusters au lieu de nous baser sur un clustering de type *data-driven*. Nous expliquons en détail notre méthode de calcul en Section 3.2.2. Cette étape nous a permis de réduire le nombre de paramètres drastiquement.

La dernière étape de contruction de nos modèles contextuels consiste à regrouper les trigraphes identiques. En effet, étant donné que le clustering d'états entraîne une réduction importante du nombre d'états différents, certains trigraphes se retrouvent partager exactement le même cluster pour chacune des positions d'état et donc être identiques. Cette étape permet de réduire le nombre de modèles différents définis.

Une fois que le calcul et la mise en place des clusters d'états sont terminés, le système est prêt à finir d'apprendre les modèles des caractères en contexte. (Pour des raisons de complexité, un modèle d'état est représenté par une seule distribution Gaussienne durant la phase d'initialisation des trigraphes et des clusters d'états.) En utilisant l'algorithme Baum-Welch pour les étapes de ré-estimation des paramètres des HMMs, nous augmentons de manière incrémentale le nombre de Gaussiennes dans chaque mélange, en utilisant la même procédure que celle décrite dans la Section 2.2.1 pour les monographes. Ainsi, chaque trigraphe atteint une topologie identique à celle des monographes du système générique, c'est-à-dire S états metteurs et N_G distributions Gaussiennes par état. Cela nous permettra de comparer directement les deux approches, contextuelle et non contextuelle.

3.2.2 Arbres de décision pour clustering d'états

Le clustering par arbres de décision est une alternative au clustering guidé par les données. Il se base sur une mesure de distance entre modèles utilisant la maximisation de la vraisemblance et permet une découpe de clusters contrôlée. Ce type de clustering a d'abord été proposé pour la parole par Young *et al.* [168] où il donne des

résultats au niveau de l'état de l'art avec un système moins complexe que d'autres systèmes avec modèles en contexte.

Le regroupement et la séparation de clusters d'états sont conduits par un arbre binaire dont chaque noeud correspond à une question rhétorique sur les contextes des modèles. Dans notre cas, les questions sont définies sur les caractéristiques morphologiques des contextes gauche et droit des caractères. Un arbre est construit pour chaque position d'état de tous les trigraphes correspondant à une même lettre centrale.

Pour construire un arbre, tous les états correspondant à une même position pour une lettre centrale donnée sont d'abord regroupés au noeud racine, comme illustré sur la Figure 3.5 (exemple pour le français : la Figure représente l'arbre calculé pour la position d'état numéro deux des trigraphes centrés sur la lettre *b*). Ensuite, la question binaire qui sépare le groupe en deux sous-groupes de vraisemblance maximale est choisie et la séparation est effectuée, créant deux nouveaux noeuds. Il en va ainsi de la scission de chaque noeud jusqu'à ce que l'augmentation de la vraisemblance descende sous un seuil ou bien jusqu'à ce que plus aucune question ne soit disponible qui puisse créer deux sous-groupes avec un taux d'occupation d'états suffisant.

On peut formuler mathématiquement le clustering par arbres de décision de la manière suivante. Soit $\boldsymbol{P}$ l'ensemble d'états présents dans le noeud courant. $\boldsymbol{P}$ est associé au sous-ensemble $\boldsymbol{F}$ d'observations (vecteurs de caractéristiques) $\{\boldsymbol{o}_f\}_{f\in\boldsymbol{F}}$. Par définition, tous les états présents dans $\boldsymbol{P}$ sont les membres d'un même cluster, ils partagent donc les mêmes moyenne $\mu(\boldsymbol{P})$ et variance $\Sigma(\boldsymbol{P})$. Par suite, la vraisemblance de $\boldsymbol{P}$ générant l'ensemble des $\{\boldsymbol{o}_f\}_{f\in\boldsymbol{F}}$ s'écrit :

$$L(\{\boldsymbol{o}_f\}_{f\in\boldsymbol{F}};\boldsymbol{P}) = \sum_{f\in\boldsymbol{F}}\sum_{s\in\boldsymbol{P}} log(Pr(\boldsymbol{o}_f;\mu(\boldsymbol{P}),\Sigma(\boldsymbol{P})))\gamma_s(\boldsymbol{o}_f) \tag{3.3}$$

Pour plus de clarté, nous simplifierons désormais $L(\{\boldsymbol{o}_f\}_{f\in\boldsymbol{F}};\boldsymbol{P})$ en $L(\boldsymbol{P})$. Dans l'équation 3.3, $\gamma_s(\boldsymbol{o}_f)$ est la probabilité *a posteriori* que le vecteur de caractéristiques $\boldsymbol{o}_f$ soit généré par l'état $s \in \boldsymbol{P}$. En supposant que la densité de probabilité représentée par $(\mu(\boldsymbol{P}), \Sigma(\boldsymbol{P}))$ est une Gaussienne et de matrice de covariance diagonale, on peut l'estimer à partir des échantillons $\{\boldsymbol{o}_f\}_{f\in\boldsymbol{F}}$ et $L(\boldsymbol{P})$ peut être reformulé

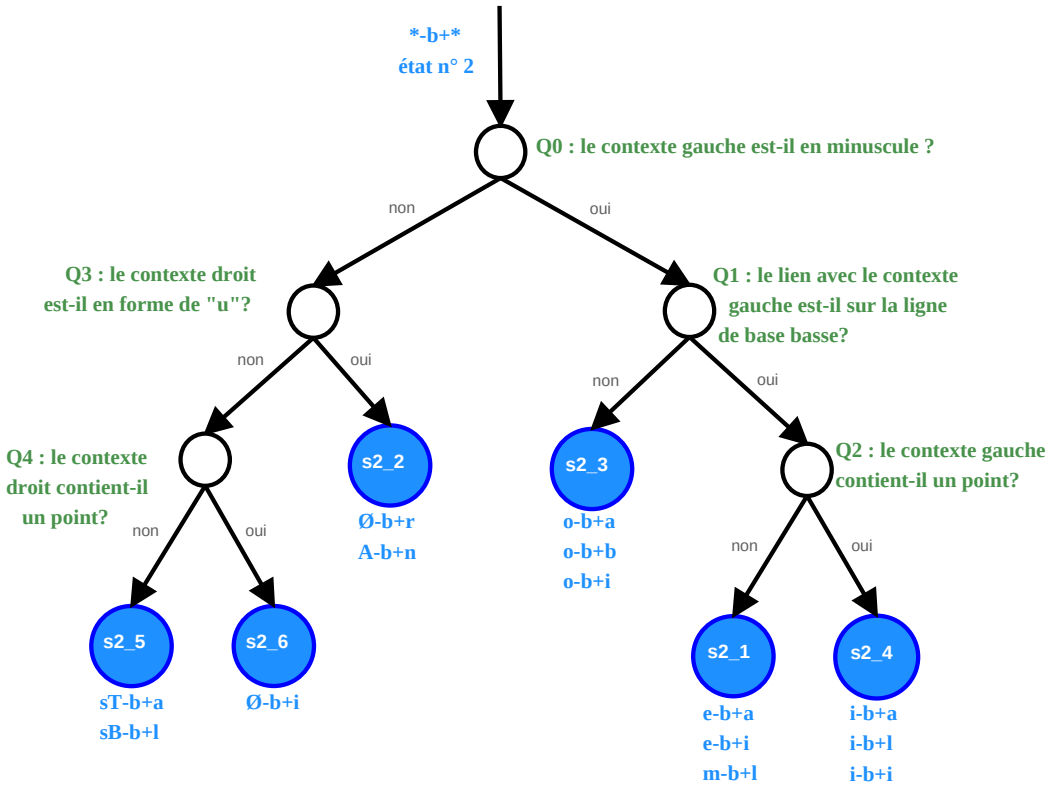


FIGURE 3.5 – Exemple d'arbre de décision pour le clustering d'états : l'ordre des questions et les clusters sont associés à un état donné (ici l'état numéro 2) de tous les trigraphes $- * b + *$.

selon Young *et al.* [168] par :

$$L(\boldsymbol{P}) = -\frac{1}{2}(log[(2\pi)^n|\Sigma(\boldsymbol{P})|] + n)\Gamma(\boldsymbol{P}) \tag{3.4}$$

n est la dimension des vecteurs $\boldsymbol{o}_f$ et $\Gamma(\boldsymbol{P})$ est le taux d'occupation d'états cumulé du noeud courant :

$$\Gamma(\boldsymbol{P}) = \sum_{f \in \boldsymbol{F}} \sum_{s \in \boldsymbol{P}} \gamma_s(\boldsymbol{o}_f). \tag{3.5}$$

La séparation de $\boldsymbol{P}$ en deux sous-groupes $\boldsymbol{P_{q+}}$ et $\boldsymbol{P_{q-}}$ est alors faite par la question q^* qui maximise ΔL_q, défini par :

$$\Delta L_q = L(\boldsymbol{P}_{q+}) + L(\boldsymbol{P}_{q-}) - L(\boldsymbol{P}) \tag{3.6}$$

La scission se fait à condition que $\Gamma(\boldsymbol{P}_{q+})$ et $\Gamma(\boldsymbol{P}_{q-})$ soient au-dessus du taux minimal d'occupation Γ_{min} et que ΔL_q soit plus grand que le seuil minimal de croissance de variance ΔL_{min}. Le choix des paramètres Γ_{min} et ΔL_{min} est effectué sur une base de validation. Pour un monographe donné, la profondeur des arbres lui étant associés (un arbre par numéro d'état) est étroitement liée à la variance de la vraisemblance et au nombre d'exemples du monographes, mentionnés dans la Section 3.1.1. En effet, on constate que lorque la variance et le nombre d'exemples d'un monographe sont élevés, alors l'arbre de chacun de ses états est profond et le nombre de trigraphes définis et modélisés différemment est grand. De même, on observe parfois des arbres réduits à leur simple noeud racine : ils correspondent aux états de monographes ayant peu d'exemples et regroupant tous leurs trigraphes correspondants dans un modèle unique.

Les questions q définissant les arbres de décision pour la reconnaissance de la *parole* ont été réalisées par des experts (voir [124]). A notre connaissance, personne n'utilise de tels arbres pour l'écrit. Nous avons donc proposé des questions en rassemblant des caractères semblables dans différents contextes. L'ensemble des questions que nous avons créées contient des questions générales, pour permettre des clusters de grande taille mais aussi des questions précises, au cas où des clusters plus petits doivent être créés et des questions intermédiaires. Les questions sont uniquement fonctions des contextes (gauche et droit). Par exemple :

– (question générale) le contexte droit est-il une lettre majuscule ? minuscule ?

Contient-il un ascendant ?

- (question intermédiaire) le contexte gauche contient-il une boucle ("o", "b", "d", etc.) ? une barre verticale ("t", "p", etc.) ?
- (question précise) est-ce que le lien avec la lettre suivante (précédente) se situe sur la ligne de base basse ("a", "c", etc.) ? sur la ligne de base haute ("v", "w", etc.) ?

L'ensemble des questions définies pour l'alphabet latin et l'alphabet arabe se trouvent en Annexe.

3.2.3 Reconnaissance de mots avec modèles de caractères en contexte

Au delà de l'apport original des arbres binaires pour le calcul des clusters, l'utilisation d'arbres de décision apporte une fonctionnalité supplémentaire très utile lors du décodage : elle permet de sélectionner les clusters qui serviront à modéliser les états des trigraphes non vus lors de l'apprentissage. En effet, nous utilisons pour nos tests des lexiques prédéterminés mais indépendants du lexique d'apprentissage (hormis l'utilisation du même alphabet). Par exemple, lors des compétitions sur la base Rimes, deux lexiques sont donnés pour le test, chacun contenant des mots non existants dans le lexique d'apprentissage.

Ainsi, plusieurs nouveaux mots et donc plusieurs nouveaux trigraphes apparaissent, qui n'ont pas été appris. Le clustering par arbres permet de les modéliser car il peut adapter les modèles appris à n'importe quel vocabulaire (basé sur les mêmes monographes de départ). Si nous avions calculé nos clusters par une distance entre les données par exemple (*data-driven clustering*), cela n'aurait pas été possible. En effet, ne disposant pas de données étiquetées pour apprendre les nouveaux trigraphes, un calcul de distance aux clusters les plus proches n'est pas envisageable.

La modélisation d'un trigraphe inconnu se fait de la manière suivante : chaque état du trigraphe est positionné à la racine de l'arbre correspondant au même numéro d'état et à la même lettre centrale. Ensuite, chaque état parcourt son arbre correspondant, répondant aux questions sur les contextes du nouveau trigraphe, jusqu'à atteindre un noeud où est positionné un cluster. Le modèle d'état représentant le cluster est alors le modèle assigné à l'état considéré. Cette procédure est illustrée sur la Figure 3.6 pour l'assignation d'un modèle à l'état numéro 2 du trigraphe inconnu $m-b+e$. L'arbre de décision pour cette position d'état a été calculé lors de

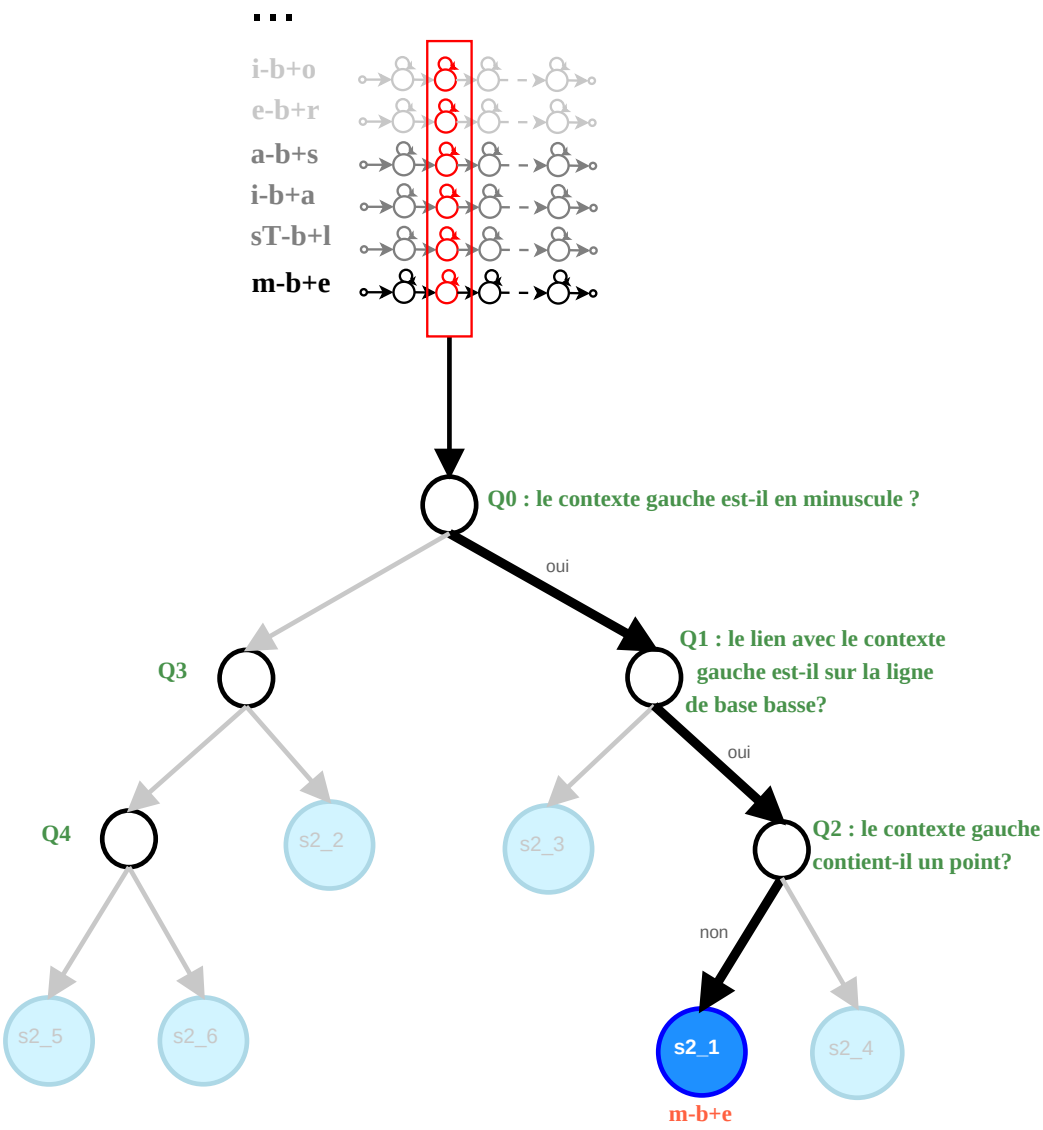


FIGURE 3.6 – Sélection de cluster pour l'état 2 du trigraphe de test $m - b + e$ non appris (absent du lexique d'apprentissage mais présent dans celui du test)

la phase d'apprentissage et représenté sur la Figure 3.5. Plaçons l'état numéro 2 de $m-b+e$, non attribué, au noeud racine de l'arbre. La première question posée est : *Le contexte gauche est-il en minuscule ?*. Etant donné que m est une minuscule, la réponse est oui, donc l'état descend vers le noeud correspondant à la réponse *oui*, où une nouvelle question est posée : *Le lien avec le contexte gauche est-il sur la ligne de base basse ?*. Après avoir répondu à cette question, le processus est recommencé jusqu'à ce que l'état atteigne un cluster. Dans notre cas, les réponses successives mènent au cluster $s2\_1$, qui est donc le modèle choisi pour l'état numéro 2 de notre trigraphe inconnu. La procédure recommence ensuite pour les autres trigraphes.

La capacité des arbres de traiter des trigraphes non vus a été très utile pour nos expériences sur le français, l'anglais et l'arabe. Les lexiques d'apprentissage et de test étant souvent différents, plusieurs centaines de nouveaux trigraphes devaient être modélisés.

Conclusion du chapitre 3

Dans ce chapitre, nous avons présenté en détail notre système original de modélisation de caractères en fonction de leur contexte. Les HMMs ne modélisent plus des caractères isolés mais des trigraphes. Les trigraphes sont la représentation d'un caractère dépendant de ses caractères voisins : dans le mot « Monsieur », le caractère n entouré à gauche d'un o et à droite d'un s est le trigraphe $o-n+s$. L'idée de la modélisation de caractères en fonction de leur contexte vient du constat simple que la forme d'un caractère varie en fonction de son voisinage. On souhaite alors pouvoir modéliser ces variations afin d'obtenir des modèles les plus précis possibles et donc d'améliorer les performances de notre reconnaisseur.

La modélisation de trigraphes est cependant synonyme d'une augmentation considérable du nombre de modèles à calculer. En effet, tous les trigraphes présents dans le lexique d'apprentissage doivent être modélisés. Expérimentalement, on observe que pour un lexique de plusieurs milliers de mots, le nombre de trigraphes à calculer se compte aussi en milliers (cf. Chapitre 4) alors que l'alphabet contient au plus une centaine de caractères différents. Cela signifie que le nombre de paramètres HMMs est multiplié par dix au moins par rapport à un système générique. De plus, étant donné que la majorité des trigraphes ont peu d'exemples dans l'ensemble d'apprentissage, une méthode doit être trouvée pour réduire le nombre de paramètres.

Dans notre système, nous proposons une approche originale pour la réduction du nombre de paramètres : le clustering d'états. Ce type de clustering consiste à calculer des clusters indépendamment pour chaque position d'état des HMMs de trigraphes centrés sur un caractère donné ; chaque état du trigraphe centré sur le caractère prend son modèle dans l'un des clusters correspondant à sa position et au caractère central. L'originalité de notre approche réside dans l'utilisation d'arbres binaires de décision pour le calcul des clusters d'états. Un arbre est défini pour chaque numéro d'état de tous les caractères centraux existants. Les noeuds des arbres sont scindés en deux par des questions binaires sur les contextes gauche et droit pouvant être présents dans un trigraphe et leurs feuilles sont les clusters d'état. Pour construire les arbres, nous avons créé un ensemble de questions binaires pour le latin et l'arabe à partir de nos connaissances et obervations morphologiques sur ces deux écritures.

L'avantage de l'utilisation d'arbres de décision pour le clustering est qu'ils permettent de gérer des lexiques de décodage contenant des mots et donc des trigraphes non vus lors de l'apprentissage : par construction, il est possible d'associer à chacun des états de ces trigraphes un cluster (il suffit de « descendre » dans l'arbre associé à chaque position et au caractère central en répondant aux questions binaires). Le trigraphe est alors associé à un modèle appris et peut être utilisé dans un mot du lexique de décodage.

Nous verrons dans le prochain chapitre que les expériences conduites sur diverses bases de données montrent l'intérêt de la modélisation de caractères en fonction de leur contexte. Pour le français, l'anglais et l'arabe, cette modélisation améliore systématiquement les performances du système.

Chapitre 4

Application du système dynamique sur le français, l'anglais et l'arabe

Introduction

Ce chapitre est consacré à l'illustration des idées développées dans les précédents chapitres par des expériences sur des bases de données publiques : la base de courriers manuscrits français Rimes [4, 63], la base en anglais IAM [110] et la base de documents manuscrits arabes de l'évaluation OpenHart [74]. Dans chacune de ces bases, nous présentons nos résultats pour la tâche de reconnaissance de mots isolés. Nous utilisons le logiciel HTK [169] pour l'apprentissage et le décodage de nos modèles.

La base Rimes nous sert de base de référence. A travers les expériences menées sur cette base, nous montrons en Section 4.1 comment nous avons établi et optimisé l'extraction de caractéristiques sur les images, choisi la topologie des modèles HMMs de caractères et surtout comment nous avons utilisé les modèles en contexte proposés au Chapitre 3. Les résultats de notre système original pour les ensembles de test des compétitions 2009 et 2011 sont ensuite présentés et comparés aux performances d'autres systèmes de l'état de l'art.

Les étapes d'élaboration du système à base de HMMs contextuels sont détaillés de manière exhaustive sur la base Rimes afin d'illustrer notre démarche. Celle-ci étant identique pour toutes les bases de données sur lesquelles nous avons testé notre système, nous ne donnons que les détails essentiels de chaque étape pour la base IAM dans la Section 4.2 et pour la base OpenHart en Section 4.3. Pour cette dernière,

nous avons dû prendre en compte les spécificités de l'écriture arabe (qui diffère sur certains points de l'écriture latine) afin de construire nos modèles. Ces spécificités sont introduites dans la Section 4.3.1 avant la présentation des résultats sur la base OpenHart (Section 4.3.3). Finalement, nous élargissons la portée de notre système de reconnaissance de mots à la reconnaissance de lignes grâce à l'indroduction de modèles de langage dans notre classifieur arabe (Section 4.3.5). Ceux-ci nous permettent d'améliorer significativement nos résultats sur la base OpenHart et ouvrent la discussion sur les nouvelles possibilités d'amélioration de notre système.

4.1 Illustration de la mise en place complète du système en contexte avec la base Rimes

Dans cette Section, nous développons en détail la mise en place d'un système à base de HMMs en contexte pour la reconnaissance de mots manuscrits. Pour illuster cela, nous utilisons la base de données Rimes, que nous présentons en Section 4.1.1. Dans un premier temps, un système robuste à base de HMMs classiques est construit : les paramètres d'extraction de caractéristiques sont optimisés (Sections 4.1.2, 4.1.3 et 4.1.4) et la topologie optimale des modèles est évaluée (Section 4.1.5). Une fois ce système mis en place, le système présenté au Chapitre 3, à base de HMMs en contextes est construit, utilisant les caractéristiques et la topologie optimisée du système HMM simple (Section 4.1.6). Enfin, une fois les paramètres finaux du système choisis sur une base de validation, nous évaluons nos deux systèmes sur une base de test (Section 4.1.8) et comparons leurs résultats ainsi que leur combinaison aux autres systèmes de l'état de l'art (Sections 4.1.7, 4.1.9 et 4.1.10).

4.1.1 La base de données Rimes

La base de données Rimes a été rendue disponible en 2006 [4]. Des exemples sont donnés sur la Figure 4.1. Elle rassemble plus de 12500 documents manuscrits écrits par environ 1300 volontaires. Elle a été créée pour répondre au besoin récurrent de bases complètes d'images propres et de taille suffisante. Aucune contrainte n'a été imposée aux scripteurs, les documents sont donc très variables, ce qui rend la base réaliste. A partir des documents collectés, il est possible d'évaluer des systèmes de reconnaissance de logo, d'analyse de structure de texte et de reconnaissance de mot,

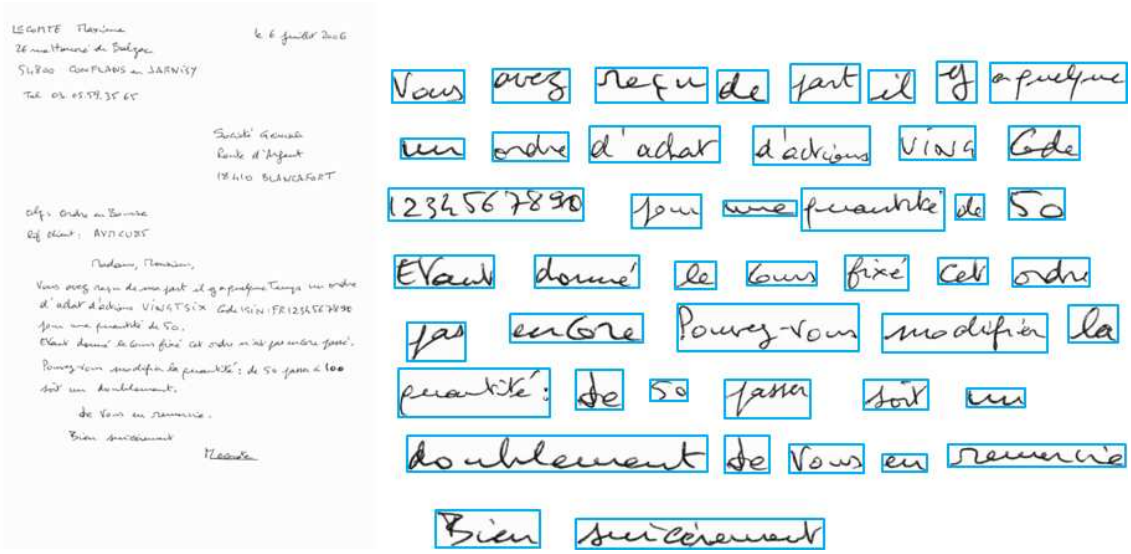


FIGURE 4.1 – Exemple de courrier de la base Rimes et d'images de mots extraites de ce courrier.

caractère, ligne ou paragraphe. Notre travail se concentre sur la reconnaissance de mots isolés. Des campagnes d'évaluation ont eu lieu depuis 2007, ce qui permet de comparer nos résultats à l'état de l'art.

Pour la mise en place de notre système (choix des caractéristiques, topologie des HMMs par exemple), nous utilisons le découpage de la base de données et la procédure d'évaluation de la campagne Rimes-ICDAR 2011, à laquelle nous avons participé. L'ensemble d'apprentissage est composé de 51738 images de mots isolés pour un lexique de 4942 mots différents. Un ensemble de validation de 7464 mots issus d'un lexique de taille 1612 est donné. L'ensemble de test est quant à lui composé de 7776 images de mots isolés. Dans les expériences présentées ci-après, sauf précision contraire, les résultats sont donnés indépendamment de la casse ($a = A$) mais une erreur sur un accent est comptée ($a \neq à$).

Pour l'apprentissage des modèles indépendants du contexte, nous avons distingué 78 caractères différents sur la base Rimes (nous n'avons créé de modèles que pour les caractères ayant au moins un exemple dans l'ensemble d'apprentissage) :

- les 26 lettres de l'alphabet latin en majuscule,
- les 26 lettres de l'alphabet latin en minuscule,
- caractères accentués : À, à, ç, É, é, è, ê, ë, î, ï, ô, ù, û,
- 8 chiffres : les chiffres de 0 à 9, excepté le 0 et le 5, qui partagent leur modèle respectivement avec O (o majuscule) et S (s majuscule)

– 4 signes de ponctuation : le trait d'union (-), l'apostrophe ('), la barre oblique (/) et le symbole degré (°).

Le $78^{ème}$ monographe est le modèle correspondant à l'espace entre deux mots, caractérisé par des colonnes successives de pixels de « fond blanc ».

4.1.2 Extraction des caractéristiques

Nous avons présenté notre extraction de caractéristiques géométriques, statistiques et directionnelles au Chapitre 2, Section 2.1. Les caractéristiques présentées dépendent de deux paramètres : la largeur de la fenêtre glissante w (en pixels) et le décalage entre deux fenêtres consécutives δ. Dans cette Section, nous évaluons les meilleurs paramètres possibles pour w et δ, pour construire un reconnaisseur robuste.

Nous avons fait varier w entre $w = 5$ pixels et $w = 14$ pixels et δ tel que $\delta \leq w/2$. Afin d'accélerer la procédure et d'éliminer les valeurs de w et de δ donnant les moins bons résultats, nous avons dans un premier temps fait des expériences sur un nombre d'images restreint : nous avons isolé une sous-base de 10000 images de la base d'apprentissage et une de 2000 images de la base de validation. Ensuite, nous avons procédé à des apprentissages rapides de modèles HMMs simples, où chaque caractère est modélisé par le même nombre d'états et où les états sont représentés par des mélanges de 5 gaussiennes. Les modèles des caractères de ponctuation ont un nombre d'états inférieur (fixé à $S = 4$) et ils ont aussi un mélange de 5 gaussiennes par état. La Table 4.1 donne les taux de reconnaissance pour chaque ensemble de caractéristiques extrait. Le lexique utilisé pour le décodage est le lexique de la base de validation de Rimes, de 1612 mots.

La Table 4.1 nous montre que les fenêtres d'extraction trop larges ($w > 10$ pixels) donnent de moins bons résultats que les fenêtres plus étroites et que des valeurs de δ trop larges détérioraient les résulats ($\delta \geq 4$). Nous concentrons donc notre recherche sur des fenêtres dont la largeur varie entre $w = 5$ et $w = 9$ pixels, pour $\delta \leq 3$. Pour chaque couple (w, δ), nous renouvelons alors l'expérience effectuée avec le meilleur δ d'après la Table 4.1 mais cette fois-ci en utilisant les bases d'apprentissage et de validation complètes de la base Rimes. La Table 4.2 donne les résultats d'un décodage pour un nombre d'états fixe par HMM de caractère et un mélange de 5 gaussiennes par état. Le lexique utilisé pour le décodage est toujours celui de la base de validation, de taille 1612.

paramètres w et δ	**nombre d'états des HMMs**	**taux de reconnaisance**
$w = 5$ $\delta = 2$	16	67.5%
$w = 6$ $\delta = 2$	16	66.15%
$w = 6$ $\delta = 3$	13	65.05%
$w = 7$ $\delta = 3$	13	64.0%
$w = 8$ $\delta = 3$	13	63.75%
$w = 8$ $\delta = 4$	10	58.7%
$w = 9$ $\delta = 3$	12	66.1%
$w = 9$ $\delta = 4$	10	58.45%
$w = 10$ $\delta = 4$	10	61.6%
$w = 10$ $\delta = 5$	8	60.2%
$w = 11$ $\delta = 4$	9	61.6%
$w = 11$ $\delta = 5$	8	59.0%
$w = 12$ $\delta = 4$	10	61.5%
$w = 12$ $\delta = 6$	6	52.1%
$w = 13$ $\delta = 5$	8	58.05%
$w = 13$ $\delta = 6$	6	53.65%
$w = 14$ $\delta = 5$	8	58.3%
$w = 14$ $\delta = 7$	6	52.85%

TABLE 4.1 – Comparaison de différentes valeurs de w et δ pour l'extraction des caractéristiques sur un **sous-ensemble** de la base Rimes. Les HMMs de caractères ont le même nombre d'états et un mélange de 5 distributions gaussiennes par état.

paramètres w et δ	**nombre d'états des HMMs**	**taux de reconnaisance**
$w = 5$ $\delta = 2$	16	68.14%
$w = 6$ $\delta = 2$	16	68.36%
$w = 7$ $\delta = 3$	13	67.02%
$w = 8$ $\delta = 3$	13	66.96%
$w = 9$ $\delta = 3$	12	**69.52**%

TABLE 4.2 – Comparaison de différentes valeurs de w et δ pour l'extraction des caractéristiques sur la base **complète** Rimes. Les HMMs de caractères ont le même nombre d'états et un mélange de 5 distributions gaussiennes par état. Ils sont appris sur la base d'apprentissage de Rimes et testés sur la base de validation.

caractéristiques	**taux de reconnaissance**
géométriques et statistiques	67.19%
directionnelles	64.11%
géométriques, statistiques et directionnelles	**69.52%**

TABLE 4.3 – Comparaison de différents ensembles de caractéristiques avec $w = 9$, $\delta = 3$ et $S = 12$. Les HMMs de caractères ont le même nombre d'états et un mélange de 5 distributions gaussiennes par état. Ils sont appris sur la base d'apprentissage de Rimes et testés sur la base de validation.

D'après les résultats obtenus, **nous choisissons les paramètres $w = 9$ pixels, $\delta = 3$ pixels, donnant 34 caractéristiques différentes. Le nombre d'états émetteurs optimal par caractère est $S = 12$ et $S = 4$ pour la ponctuation.**

Intérêt de l'ajout des caractéristiques directionnelles. Nous avons proposé au Chapitre 2, Section 2.1.1, d'ajouter aux caractéristiques géométriques et statistiques des caractéristiques directionnelles issues des SIFT. Les résultats donnés précédemment les utilisent mais nous souhaitons montrer ici que l'ajout de caractéristiques directionnelles améliore les résultats du système. La Table 4.3 donne les résultats de chaque ensemble de caractéristiques pris séparément et les compare au résultat obtenu en les assemblant, pour les mêmes conditions que celles établies ci-dessus ($w = 9$, $\delta = 3$ et $S = 12$).

On voit que les caractéristiques directionelles seules donnent des résultats inférieurs aux caractéristiques géométriques et statistiques mais leur ajout à ces dernières permet au taux de reconnaissance sur la base de validation de Rimes de passer de 67.19% à 69.52%, soit une réduction relative de mots mal reconnus de 3.5%. Un test de t-Student apparié démontre que cette différence de performances est statistiquement significative avec un taux de confiance de 99.9% [1]. Ceci légitimise l'utilisation de l'ensemble de caractéristiques proposé au Chapitre 2, Section 2.1.1.

1. La valeur critique du test de Student entre les caractéristiques complètes et les caractéristiques directionnelles seules est de 10.16 et elle est de 5.28 avec les caractéristiques géométriques et statistiques seules. Le seuil critique du test de Student pour un taux de confiance de 99.9% et pour un nombre d'échantillons supérieur à 1000 est de 3.291, soit largement inférieur aux valeurs trouvées.

type de régression	**nombre de caractéristiques**	**taux de reconnaissance**
pas de régression	34	69.52%
régression du 1^{er} ordre	68	**73.16%**
régression du 1^{er} + 2^{nd} ordre	102	72.72%

TABLE 4.4 – Utilisation de régression sur les caractéristiques. Les HMMs de caractères ont le même nombre d'états (12) et un mélange de 5 distributions gaussiennes par état. Ils sont appris sur la base d'apprentissage de Rimes et testés sur la base de validation.

4.1.3 Caractéristiques dynamiques

Nous avons proposé au Chapitre 2, Section 2.1.2, d'ajouter de l'information sur les caractéristiques en utilisant une régression. Nous avons utilisé les paramètres optimaux pour l'extraction de caractéristiques obtenus précédemment ($w = 9$ et $\delta = 3$) et un nombre d'états émetteurs fixe par caractère égal à 12 afin d'évaluer l'influence de cette régression sur les performances du reconnaisseur. Les résultats pour un mélange de 5 gaussiennes par état est donné dans la Table 4.4.

On voit que l'ajout d'une régression du 1^{er} ordre permet au taux de reconnaissance de passer de 69.52% à 73.16% sur la base de validation de Rimes, ce qui représente un gain relatif de 5.2%. L'ajout en sus d'une régression du 2^{nd} ordre apporte quant à elle une amélioration de 4.6% en relatif par rapport à un système sans régression. Si l'amélioration obtenue pour l'ajout d'une régression du 2^{nd} ordre est moindre, la différence de performances entre la régression du 1^{er} ordre et la régression du 1^{er} et du 2^{nd} ordre n'est pas statistiquement significative[2]. Cependant, l'ajout d'une régression du 2^{nd} ordre augmente le nombre de paramètres à calculer et ralentit le processus d'apprentissage et de décodage : elle ne sera donc pas utilisée par la suite.

A la suite de ces expériences, **nous choisissons donc de conserver une régression du premier ordre sur notre ensemble de caractéristiques, ce qui amène les vecteurs de caractéristiques à être de dimension $2 * 34 = 68$.**

2. La valeur critique du test de Student entre la régression du 1^{er} ordre et la régression du 1^{er} et du 2^{nd} ordre est de 0.40. Elle est largement inférieure au seuil critique pour un taux de confiance de 99.9% (3.291) ainsi qu'au au seuil pour un taux de confiance de 95% (1.96).

4.1.4 Analyse en composantes principales

Dans le Chapitre 2, Section 2.1.3, nous avons présenté le principe de réduction de dimension des caractéristiques grâce notamment à une analyse en composantes principales. Nous avons effectué cette analyse sur le même jeu de données que celui utilisé jusqu'à présent, c'est-à-dire les bases d'apprentissage et de validation de la campagne Rimes 2011.

Nous avons évalué les paramètres de projection de l'analyse en composantes principales sur une sous-partie de l'ensemble d'apprensissage (30% des données, soit plus de 15000 images de mots, correspondant à $5e^{05}$ vecteurs de caractéristiques environ). Nous avons ensuite projeté les données selon les valeurs trouvées. Nous les avons projetées sur un nombre variable de dimensions afin de étudier l'influence du pourcentage d'information utile conservée sur les performances du système.

En ce qui concerne les caractéristiques dynamiques obtenues par régression, nous avons procédé à deux expériences : dans la première, la PCA est calculée sur les n caractéristiques initiales, puis une fois les données projetées une régression est calculée. C'est le procédé *PCA-reg*. Dans la deuxième expérience, nous calculons la PCA sur les caractéristiques avec régression, donc $2n$ dimensions, et mettons leur projection en entrée du système (sans régression supplémentaire) : c'est le procédé *reg-PCA*. Pour chacun des procédés, les deux systèmes issus de différentes projections sont illustrés sur la Figure 4.2 et comparés à un système sans PCA. Pour chacune des projections, le pourcentage d'information conservé est donné en abscisse et la dimension des caractéristiques est donnée par les barres verticales qui suivent les valeurs de l'ordonnée de droite. Pour le procédé *PCA-reg*, le nombre final de dimensions est donné, égal au double de la dimension de la projection.

Les résultats obtenus permettent de constater qu'il est possible de réduire la dimension des vecteurs de caractéristiques en entrée du système sans trop perdre en performance : pour le procédé *reg-PCA*, le système avec 13 dimensions, soit plus de 5 fois moins que le système initial, fait perdre moins de 1% de mots bien reconnus au système, ce qui est très peu au vu de la réduction du nombre de dimensions. Nous constatons aussi qu'il est plus avantageux de calculer une régression sur les caractéristiques avant la PCA. Nous interprétons cela en rappelant que, si le but d'une régression est d'ajouter de l'information sur les fenêtres avoisinantes dans la fenêtre courante, elle ajoute une *corrélation* temporelle entre les dimensions : une projection après analyse en composantes principales permet de conserver l'information de la

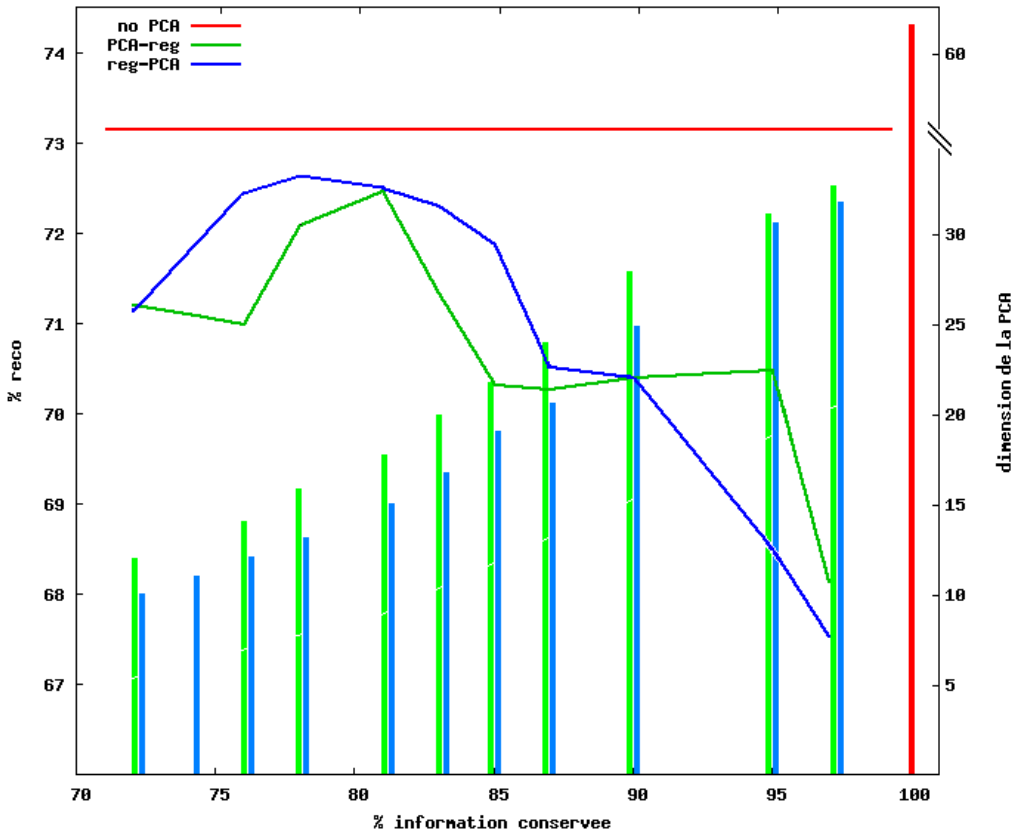


FIGURE 4.2 – Application d'une ACP sur les caractéristiques avant et après régression avec un pourcentage variable d'information conservée pour les projections. Les barres verticales représentent les dimensions de chaque système. Les HMMs de caractères ont le même nombre d'états (12) et un mélange de 5 distributions gaussiennes par état. Ils sont appris sur la base d'apprentissage de Rimes et testés sur la base de validation.

procédé	nombre de dimensions conservées	% information conservée	temps moyen de décodage
pas de PCA	68	100%	590ms
PCA-reg	18 (9x2)	81.1%	500ms
reg-PCA	13	77.9%	470ms

TABLE 4.5 – Comparaison du temps moyen de décodage d'un mot entre systèmes avec et sans PCA. Les HMMs de caractères ont le même nombre d'états (12) et un mélange de 5 distributions gaussiennes par état. Le décodage est effectué avec un lexique de 1612 mots sur la base de validation de Rimes.

régression tout en éliminant l'effet de corrélation.

On constate aussi que les systèmes réduits sont plus avantageux au niveau du temps de décodage moyen d'un mot (voir Table 4.5) : il est toujours intéressant d'accélérer le processus de décodage d'un système, surtout lorsque celui-ci doit traiter un grand nombre de données. Dans les résultats présentés par la suite, nous n'utiliserons cependant pas des caractéristiques avec PCA car nous souhaitons obtenir les meilleures performances possibles du système en vue de participer à des compétitions.

4.1.5 Choix de la topologie des HMMs de caractères sans contexte

Nous avons présenté dans le Chapitre 2, Section 2.3 un processus itératif permettant de choisir un nombre d'états optimal pour chaque HMM de caractère dans un système sans contextes. Nous présentons ici les résultats de ce processus sur le système développé dans les sections précédentes. Dans cette Section, nous établirons aussi le nombre optimal de gaussiennes dans les mélanges à chaque état. Ce nombre sera ensuite utilisé dans toutes nos expériences sur la base Rimes.

L'algorithme proposé donne la distribution d'états émetteurs pour les caractères suivants :

- les majuscules, la barre oblique ainsi que les minuscules *m, n, v* et *w* ont 14 états,
- les autres minuscules ont 10 états,
- les autres modèles de ponctuation conservent leur nombre d'états initial (4).

Cette modification de la topologie des caractères permet de gagner en relatif 0.3% de mots bien reconnus sur la base de validation de Rimes, c'est-à-dire de passer de

73.16% à 73.35% de reconnaissance sur cette base. Cette augmentation n'est pas importante pour la base Rimes mais mérite d'être notée. Comme nous le verrons plus tard (Section 4.3), l'adaptation de la topologie des modèles aux caractères apporte beaucoup à la reconnaissance de mots manuscrits arabes.

Pour finir, nous établissons le nombre optimal de gaussiennes dans les mélanges de chaque état. La Figure 4.3 illustre l'influence du nombre de gaussiennes sur le taux de reconnaissance du système sur la base de validation de la base Rimes, ainsi que sur le temps de décodage moyen d'un mot. D'après cette figure, nous choisissons de fixer le nombre de gaussiennes dans chaque mélange à 32, ce qui représente un bon équilibre entre temps de traitement et taux de reconnaissance et, surtout, un plateau en terme d'augmentation des performances. De plus, 32 gaussiennes est un nombre suffisamment peu élevé pour nous assurer d'éviter le problème du sur-apprentissage et suffisamment grand pour assurer la robustesse du système.

Le taux de reconnaissance de ce système, à base de HMMs *classiques*, est alors de 81.02% pour la base de validation de Rimes. Ce chiffre est le chiffre final du système présenté au Chapitre 2. Il nous servira de point de comparaison avec le système présenté au Chapitre 3 à base de HMMs de caractères en contexte. Tous les paramètres utilisés jusqu'ici (paramètres d'extraction de caractéristiques, topologie du modèle) sont fixés pour la suite et conservés pour être utilisés par le système en contexte.

4.1.6 Le système avec modèles en contexte

Les Sections précédentes ont permis de mettre en place un système efficace de reconnaissance de mots sur la base Rimes à base de HMMs de caractères. Les caractéristiques extraites sont un mélange de caractéristiques géométriques, statistiques et directionnelles, sur lesquelles est appliquée une régression du premier ordre. Avec ces caractéristiques, la topologie optimale des HMMs pour chaque caractère a été établie, ainsi que le nombre de gaussiennes dans les mélanges de chaque état. Maintenant que ce système stable et performant est en place, nous pouvons entamer l'élaboration du système avec modèles en contexte, basé sur celui-ci.

Comme nous l'avons présenté au Chapitre 3, la construction du système HMM en contexte commence par la duplication des modèles HMMs de chaque caractère, appris indépendamment de leur contexte. Pour la duplication, les modèles de monographes avec un mélange d'une seule gaussienne par état sont utilisés. Pour chaque

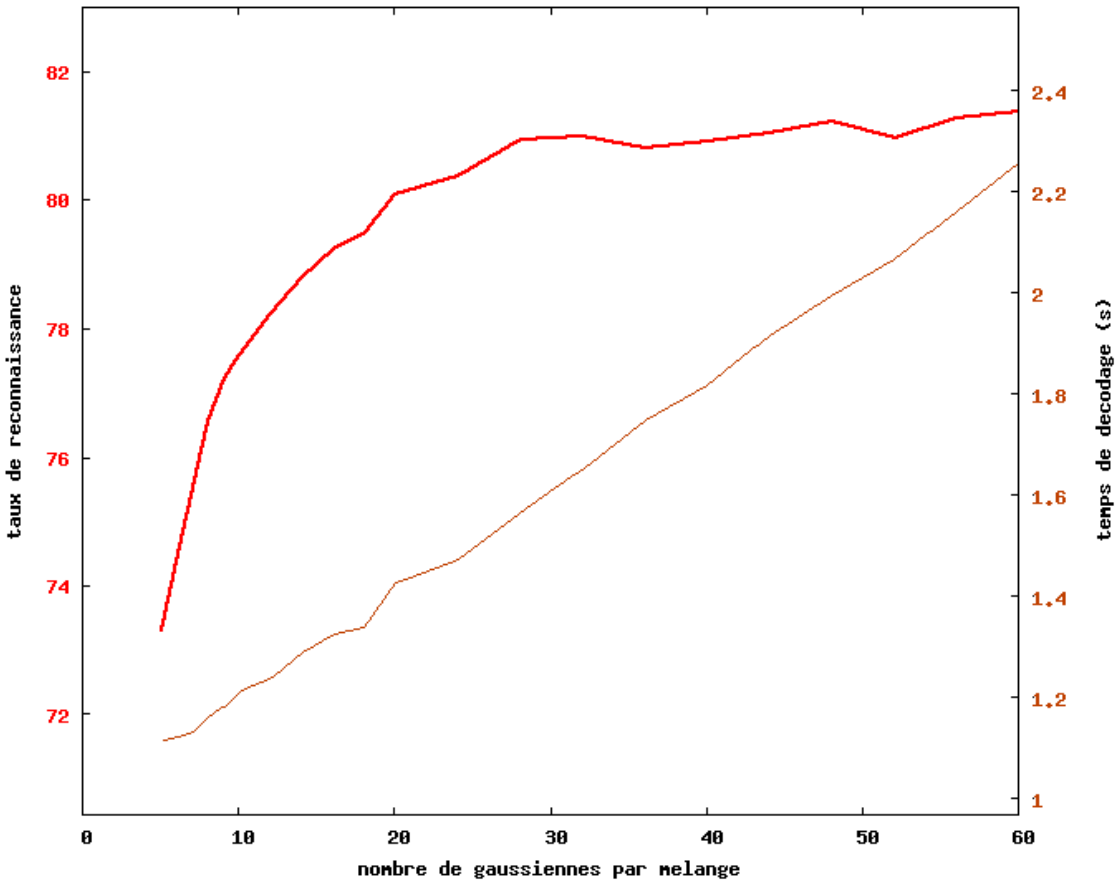


FIGURE 4.3 – Influence du nombre de gaussiennes dans les mélanges de chaque état sur la reconnaissance et le temps de décodage par image de mot pour le système à base de HMMs classiques, appris sur la base d'apprentissage de Rimes et testé sur la base de validation.

caractère, son HMM est dupliqué dans chaque trigraphe dont le caractère central lui correspond. Ensuite, les matrices de transition sont partagées de sorte qu'une seule matrice est définie pour chaque caractère (central) donné, soit 77 sur la base Rimes (le modèle du silence n'est pas contextualisé). Deux itérations de l'algorithme d'apprentissage sont alors effectuées sur cette nouvelle construction pour poser les bases du système en contexte. Au total, la base d'apprentissage de Rimes comprend 5168 trigraphes différents donc cette première étape a estimé les paramètres de ces 5168 trigraphes.

L'intérêt de l'approche avec modèles en contexte réside dans le fait qu'il est possible de faire un partage de paramètres ou plus précisément un partage d'états, décrit au Chapitre 3. Ce partage se fait à l'aide d'arbres binaires de décision qui répartissent les états dans des clusters. Un arbre est défini pour chaque position d'état de chacun des caractères centraux différents des trigraphes présents dans la base Rimes (au nombre de 77). La construction des arbres se fait en trouvant pour chaque noeud la question q^* maximisant la vraissemblance $\Delta L(q)$ de scission du noeud en deux feuilles. La scission s'effectue alors si et seulement si :

- $\Delta L(q^*)$ est supérieur à un seuil défini nommé ΔL_{min}. Si, quelle que soit la question q, $\Delta L(q) < \Delta L_{min}$, alors le noeud n'est pas scindé et devient un cluster.
- pour chaque feuille, le taux d'occupation d'états est supérieur à un seuil donné Γ_{min}. Ce seuil permet d'avoir des clusters de taille suffisante et donc des états correctement estimés.

Il nous faut alors rechercher les paramètres ΔL_{min} et Γ_{min} optimisant la répartition des clusters et donc l'apprentissage des modèles en contexte. Nous avons fait varier ces paramètres et construit autant de systèmes avec HMMs de caractères en fonction de leur contexte que de couples évalués. Pour plus de rapidité, nous avons choisi de fixer le nombre de gaussiennes par mélange à 5 pour effectuer notre comparaison. La Figure 4.4 illustre l'influence des valeurs de ΔL_{min} et Γ_{min} sur le nombre final d'états (clusters) définis par les arbres de décision et donc appris par le système. On constate sur cette Figure que plus ΔL_{min} ou Γ_{min} est élevé, moins le nombre total de clusters est grand, c'est-à-dire moins il y a d'états différents et donc de calculs à effectuer. Ceci signifie un apprentissage plus rapide certes mais une réduction trop forte du nombre d'états risquerait d'annuler l'apport donné par la modélisation en contexte des HMMs de caractère.

Pour chaque couple (ΔL_{min}, Γ_{min}), nous avons alors évalué les performances du

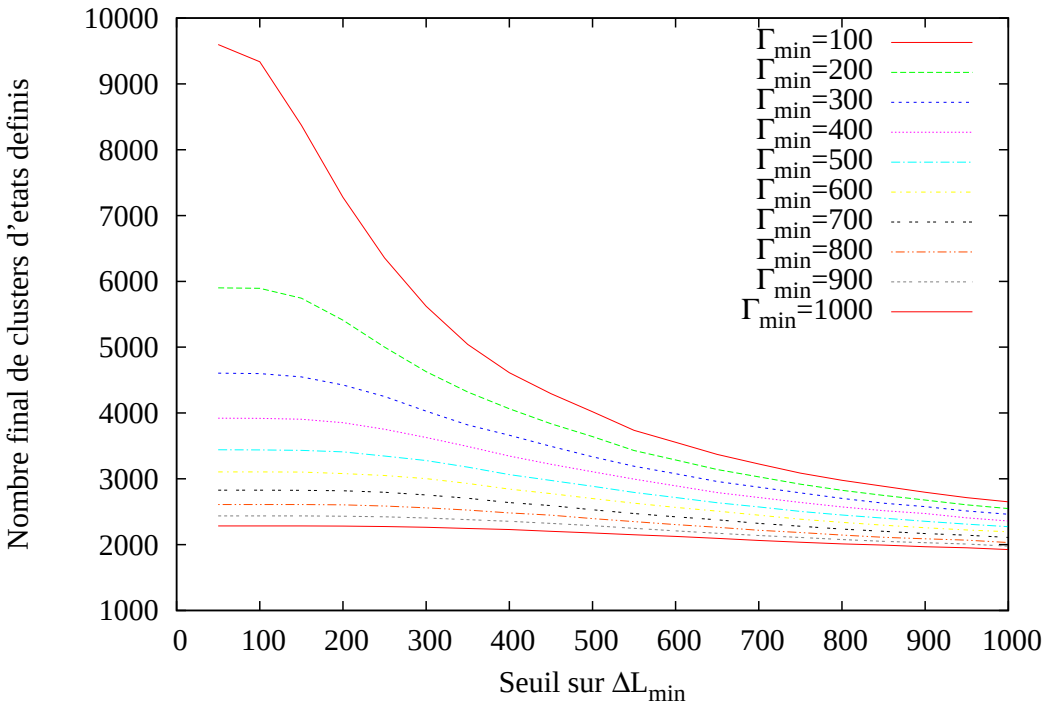


FIGURE 4.4 – Influence de ΔL_{min} et Γ_{min} sur le nombre final d'états (clusters) différents définis pour l'apprentissage des modèles sur la base Rimes.

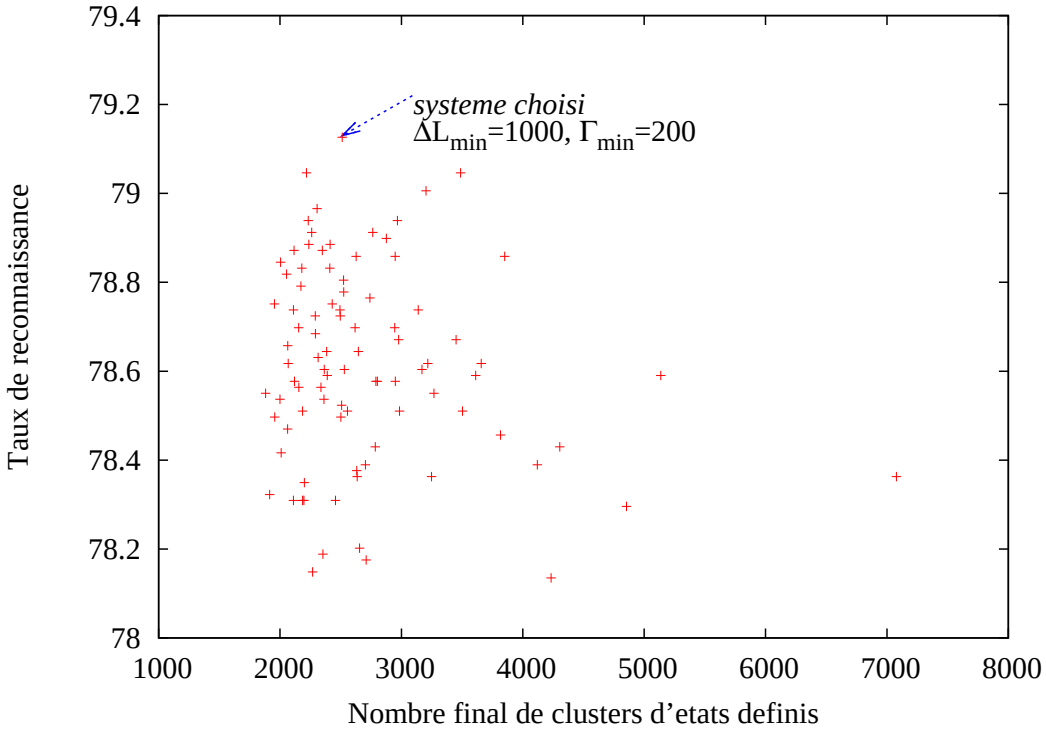


FIGURE 4.5 – Influence du nombre final de clusters sur le taux de reconnaissance du système avec HMMs en contexte sur la base de validation de Rimes. Chaque état (cluster) est un mélange de 5 distributions gaussiennes.

système appris sur la base de validation de Rimes. Or, nous avons compté plus de 350 nouveaux trigraphes présents dans le lexique de validation et non dans celui d'apprentissage. Grâce aux arbres de décision précédemment calculés (une forêt d'arbres est définie pour chaque couple (ΔL_{min}, Γ_{min})), nous avons pu allouer des modèles à ces nouveaux trigraphes, non appris (cf Section 3.2.3). Ceci constitue l'une des grandes forces du système à base de modèles en contexte : n'importe quel lexique peut être utilisé pour l'évaluation pourvu qu'il dépende du même alphabet que le lexique d'apprentissage. La Figure 4.5 montre donc l'influence du nombre d'états fixé par le couple (ΔL_{min}, Γ_{min}) sur le taux de reconnaissance sur la base de validation. Chaque point représente un couple (ΔL_{min}, Γ_{min}) donné.

Comme indiqué sur la Figure 4.5, nous avons choisi la paire ($\boldsymbol{\Delta L_{min} = 1000}$, $\boldsymbol{\Gamma_{min} = 200}$) pour notre système final. Ces paramètres représentent le meilleur équilibre entre le taux de reconnaissance (79.1% pour 5 gaussiennes sur la base de validation), le temps de calcul des clusters et celui pour le décodage. Le système choisi définit un total de 2513 états différents. Au vu du faible nombre d'états différents

par rapport aux trigraphes présents dans la base d'apprentissage, certains trigraphes se trouvent partager exactement les mêmes états. Le nombre de trigraphes différents diminue donc aussi. Pour notre système, ce nombre passe de 5168 à 1684 sur la base d'apprentissage.

Pour finir, nous avons augmenté le nombre de gaussiennes dans les mélanges de chaque état à $N_G = 32$ (nombre optimal évalué sur les monographes). **Le taux de reconnaissance final de notre système en contexte sur la base de validation de Rimes 2011 est donc de 83.32%.**

4.1.7 Comparaison avec l'état de l'art - 2009

Le résultat précédemment obtenu sur la base de validation de Rimes nous permet de nous comparer aux autres systèmes de l'état de l'art. En effet, une compétition internationale a eu lieu en 2009 sur la base Rimes lors de la conférence ICDAR [76] et la base de test de cette compétition est la base de validation que nous utilisons cette année, en 2011 (voir Grosicki et El-Abed [64]). La Table 4.6 montre la performance de notre système par rapport aux systèmes présentés lors de cette compétition, y compris les systèmes que nous avions présentés à l'époque : le *système CD 2009*, à base de modèles en contexte (les caractéristiques et l'optimisation étaient différents de ceux présentés dans ce manuscrit) et le *système combiné 2009*, produit d'une combinaison de systèmes HMMs et d'un système hybride HMM/NN élaboré par A2iA. Les taux donnés dans la Table 4.6 pour ces deux systèmes diffèrent des taux présentés dans [64] car à l'époque nos systèmes HMMs (CI et CD) ne modélisaient pas les accents. Nous avons donc réentraîné ces systèmes *dans les mêmes conditions* que celles des systèmes HMMs utilisés pour la compétition mais en ajoutant cette fois les accents dans notre lexique. Les taux de la Table 4.6 réfèrent donc à cette correction (voir Kermorvant *et al.* [80]).

Deux lexiques différents sont proposés pour le décodage : le lexique de la base de validation, contenant 1612 mots et un lexique plus large de 5334 mots. Le lexique de 1612 mots correspond exactement à tous les mots présents dans l'ensemble de validation tandis que le lexique de 5334 mots correspond à l'ensemble du vocabulaire d'apprentissage et de validation. Dans les deux cas, la couverture des mots présents dans l'ensemble des données testées est complète (pas de OOV – mots hors vocabulaire).

Le système présenté par l'UPV (*Universidad Politecnica de Valencia*) (Zamora-

système	taux de reconnaissance	
	Lexique 1612	**Lexique 5334**
TUM	93.2%	91.0%
système combiné 2009	89.1%	85.3%
UPV	86.1%	83.2%
système CD 2011	83.3%	79.1%
SIEMENS	81.3%	73.2%
système CD 2009	80.4%	75.5%
IRISA	79.6%	74.7%
LITIS	74.1%	66.7%
ITESOFT	59.4%	50.4%

TABLE 4.6 – Comparaison des performances du système proposé à base de HMMs en contexte avec les systèmes présentés à la compétition Rimes 2009 (base de validation Rimes 2011)

Martínez *et al.* [170], España-Boquera *et al.* [43]) est une combinaison basée sur des votes de trois classifieurs de type hybride HMM/NN. Les classifieurs diffèrent par le nombre d'états dans les HMMs et certains paramètres des réseaux de neurones. En amont, une diminution du lexique de test est effectuée grâce à un classifieur NN holistique qui trie les images de mots en fonction de critères haut niveau (taille de l'image, ratio, etc).

Le système présenté par Siemens est issu des travaux de Caesar *et al.* [19], Schambach [146], Kaltenmeier *et al.* [79]. Il utilise le *Hidden Markov Recognizer for Latin* (HMR), qui est la base du système de Siemens AG pour la lecture d'adresse. Les prétraitements effectués sont comparables aux nôtres et les caractéristiques extraites par fenêtres glissantes sont structurelles. Le système présenté par Siemens est aussi une combinaison de sorties de plusieurs reconnaisseurs dont les différences résident dans la topologie des modèles HMM [146].

Le système proposé par l'équipe IMADOC de l'IRISA est basé sur des HMMs à densités continues. Le prétraitement des images est standard et les caractéristiques, décrites dans [8], sont extraites par des fenêtres glissantes et dépendent des lignes de base. Le nombre d'états par caractère varie en fonction de leur longueur.

Le Litis quant à lui propose un système à base de HMM multi-flux [81]. Deux ensembles de caractéristiques (l'un basé sur les contours, l'autre sur les densités de pixels) sont extraits de deux fenêtres glissantes disctinctes. Chaque ensemble permet la modélisation d'un système HMM correspondant à un flux.

Le système présenté par Itesoft utilise les *Non-Symmetric Half-Plane Hidden Markov Model* (NSHP-HMM) [25]. Ce modèle conjugue des champs de Markov aléatoires (*Markov Random Field* (MRF)) et un HMM, et travaille directement sur des formes binaires. Par construction, leur système est insensible à la casse, ce qui détériore leurs résultats pour cette tâche.

Il faut noter que le résultat obtenu par notre système dépasse les performances de la plupart des autres systèmes de la compétition basés sur des HMMs. Le système de l'UPV a de meilleures performances (de l'ordre de 3%) mais il résulte de la combinaison de plusieurs systèmes HMMs. Cependant nous dépassons en performance certains systèmes combinés, comme le système de Siemens. Ce résultat est très positif et nous permet de démontrer la capacité de la modélisation de caractères en contextes à être utilisée pour la reconnaissante d'écriture manuscrite hors-ligne.

Il nous faut cependant compléter cette comparaison avec un dernier système présent à la compétition ; le système gagnant, à base de Réseaux de Neurones Récurrents (RNN) et présenté par TUM (Technische Universität München), donne un taux de reconnaissance remarquable, de 93.2%. Les deux principales différences entre les RNNs et les HMMs sont d'une part que l'un utilise un apprentissage discriminatif et l'autre génératif et, d'autre part, que les RNNs ne font pas d'hypothèse sur l'indépendance des observations (contrairement aux HMMs) en faisant intervenir un contexte élargi.

On voit dans la Table 4.6 que nous avions aussi proposé une combinaison de systèmes pour la compétition de reconnaissance de mots. La combinaison de systèmes est aujourd'hui l'une des méthodes les plus populaires pour obtenir les meilleurs résultats, à condition de se baser sur des classifieurs robustes [80, 37]. C'est pourquoi nous proposerons dans la Section 4.1.9 de combiner les sorties de nos systèmes HMMs non seulement avec le système hybride proposé par A2iA, mais aussi avec les RNNs. Ceci nous permet d'obtenir d'excellents résultats sur la base de test Rimes 2011.

4.1.8 Evaluation sur la base de test

Les Sections précédentes nous ont permis de montrer de manière détaillée notre protocole expérimental à l'aide de la base de validation de Rimes et d'établir deux systèmes robustes de reconnaissance de mots manuscrits, l'un à base de HMMs classiques, l'autre à base de HMMs de caractères dépendants de leur contexte. Nous

Système	**Ensemble d'apprentissage**	**Taux de reconnaissance (lexique 5744 mots)**
CI	train (51739 images)	75.2%
CI	train+valid (59203 images)	75.4%
CD	train (51739 images)	78.6%
CD	train+valid (59203 images)	**79.2%**

TABLE 4.7 – Comparaison des différents systèmes présentés (CI et CD), modélisant les caractères en fonction de leur contexte ou non. Résultats sur la base de test de Rimes 2011 contenant 7776 images.

pouvons maintenant évaluer ces deux systèmes sur la base de test de Rimes-ICDAR 2011, contenant 7776 images de mots.

La compétition 2011 ne propose qu'une expérience avec un grand dictionnaire contenant 5744 mots. Les modèles sont appris en distinguant la casse et les accents. Le taux de reconnaissance est évalué sans prendre en compte la casse mais en respectant les accents, conformément au protocole de la compétition Rimes-ICDAR 2011.

Pour rappel, le premier système (CI) modélise les caractères indépendamment de leur contexte. Décrit dans le Chapitre 2, il présente un nombre variable d'états par caractère et 32 distributions Gaussiennes par état. Le second système (CD) est celui décrit dans le Chapitre 3, modélisant les caractères en fonction de leur contexte : les trigraphes. Nous avons compté 201 nouveaux trigraphes présents dans la base de test et non dans les bases d'apprentissage ou de validation. Les arbres de décision nous ont permis de leur allouer des modèles d'états. Les paramètres du système CD ont été fixés sur la base d'apprentissage (calcul et répartition des clusters, voir Section 4.1.6) mais pour l'apprentissage final des Gaussiennes de chaque état (après l'étape de partage des paramètres), nous avons choisi d'ajouter les données de la base de validation afin d'améliorer nos performances.

Les résultats sont donnés sur la Table 4.7. Pour chaque système (CI et CD), nous montrons le taux de reconnaissance en ayant appris sur la base d'apprentissage uniquement ou bien sur l'apprentissage et la validation.

Deux observations peuvent être tirées de la Table 4.7 :

– D'une part, l'augmentation du nombre de données pour l'apprentissage permet une amélioration des performances : 0.2% pour le système CI et 0.6% pour le système CD. L'amélioration est statistiquement significative pour le système

CD avec un taux de confiance de 95%[3]. Cela nous motive à poursuivre l'ajout de données pour obtenir un meilleur apprentissage.

– D'autre part, on remarque que la modélisation des caractères en fonction de leur contexte améliore considérablement les performances : le nombre d'erreurs est diminué relativement de plus de 15% pour les systèmes appris sur le plus de données. Ainsi, même si les systèmes dits 'CD' ont plus de paramètres à calculer (2513 états, au lieu de 958 pour les monographes), ils obtiennent de meilleurs résultats. Ceci est dû à la plus grande précision dans la modélisation des caractères mais aussi à la gestion de la taille des clusters d'états, correctement paramétrée grâce à une étape de validation.

4.1.9 Combinaison de reconnaisseurs

La combinaison de classifieurs est un moyen rapide et efficace d'améliorer les performances d'un système. Elle est utilisée avec succès dans un grand nombre de domaines, tels la reconnaissance d'écriture bien sûr, mais aussi l'analyse de document, l'imagerie médicale, la vérification biométrique (visage, empreintes digitales) et la reconnaissance de la parole (voir Rahman et Fairhurst [140]).

Ayant à disposition deux systèmes distincts à base de HMMs pour la reconnaissance de mots isolés (l'un classique, l'autre à base de HMMs en contexte), nous avons donc cherché à les combiner. Nous pouvons justifier cette approche en observant les statistiques données dans la Table 4.8. Dans la première partie du tableau, le nombre de mots de l'ensemble de test correctement ou incorrectement reconnus par les deux classifieurs est donné de manière conjointe (les mots mal reconnus par les deux systèmes, ceux bien reconnus par les deux systèmes ainsi que les mots bien reconnus par l'un et non par l'autre). On lit par exemple que 478 mots ont été bien reconnus par le système CI mais ont engendré des erreurs pour le système CD. La deuxième partie de la Table 4.8 donne les taux de reconnaissance de ces deux systèmes en 1-, 10- et 50-best.

On voit sur la Table 4.8 que même si les deux systèmes ont souvent tort ou raison ensemble, ils ne sont pas d'accord sur $478 + 777 = 1255$ mots, soit plus de 16% de la base de test. Or, chacun obtient un taux de mots bien reconnus très élevé en 10-best et en 50-best. Ainsi, une combinaison de leurs sorties avec une réorganisation des

3. Le seuil critique du test de t-Student entre les troisième et quatrième lignes de la Table 4.7 est de 1.95.

classification des mots	CI incorrect	CI correct
CD incorrect	1137	478
CD correct	777	5384

taux de reco. n-best	CI	CD
1-best	75.4%	79.2%
10-best	94.2%	95.4%
50-best	97.5%	97.7%

TABLE 4.8 – Comparaison des performances des systèmes CI et CD sur la base de test de Rimes 2011 en terme de nombre de mots bien ou mal classifiés par les deux reconnaisseurs et en terme de taux de reconnaissance en n-best de chacun des reconnaisseurs.

listes $n-$best pourrait permettre de récupérer la bonne réponse pour la plupart de ces 1255 mots et donc d'améliorer les performances du reconnaisseur.

Pour cela, nous avons effectué une combinaison à base des scores des reconnaisseurs : pour chacun d'eux, la liste des n sorties les plus probables est donnée, chaque sortie étant associée à un score, normalisé entre 0 et 1 (la somme des scores des n sorties fait 1). Disposant de ces deux listes de taille n, pour chaque mot présent dans au moins l'une des deux listes, la somme des scores obtenus par chacun des systèmes pour ce mot est calculée (le score vaut 0 pour un reconnaisseur si le mot considéré n'est pas contenu dans sa n-best liste). On obtient finalement une nouvelle liste de sorties, de taille comprise entre n et $2n$. Le mot ayant le nouveau score le plus élevé est le mot final reconnu.

Ce procédé nous a permis d'améliorer notre performance en passant de 79.2% avec le système en contexte seul à 81.1% avec les deux systèmes combinés. L'amélioration que nous obtenons est encourageante pour la combinaison de systèmes mais nous espérons pouvoir encore gagner en performance en combinant nos sorties avec celles d'autres reconnaisseurs. En effet, d'après la Table 4.8, même si la combinaison des deux systèmes permettrait hypothétiquement de gagner jusqu'à 1255 mots supplémentaires bien reconnus, 1137 mots restent mal décodés par les deux systèmes quoiqu'il arrive. Ainsi, encore 14.6% de mots restent à reconnaître.

Dans [37], El-Abed *et al.* ont combiné les sorties des onze systèmes présentés lors de la compétition ICDAR 2007 de reconnaissance d'écriture manuscrite arabe. Leurs résultats montrent que la combinaison de systèmes est certes le meilleur moyen d'obtenir *in fine* les meilleurs résultats mais surtout que celle-ci est d'autant plus efficace que les classifieurs sont différents. Nous proposons donc de combiner nos

système	taux de reconnaissance
système CI	75.4%
système CD	79.2%
combinaison CI + CD	81.1%
combinaison CI + CD + graphm-A2iA	86.2%
combinaison CI + CD + graphm-A2iA + RNN (système officiel A2iA-Télécom ParisTech pour la compétition Rimes 2011)	94.9%

TABLE 4.9 – Résumé de nos résultats sur la base de test Rimes 2011.

deux systèmes à base de HMMs à d'autres systèmes, très différents.

Le premier système considéré pour être ajouté à la combinaison est un système hybride HMM/NN développé par A2iA, basé sur une segmentation explicite de mots en graphèmes [5, 80, 12]. Les probabilités d'observation sont calculées par un réseau de neurones et la décomposition d'un caractère en graphèmes est modélisée par un HMM de type Bakis. Le second système considéré est le système à base de LSTM-MDRNN (RNN multidimensionnels) proposé dans [62], élaboré avec la librairie publique donnée dans [71].

Les résultats obtenus suite à ces combinaisons sont donnés sur la Table 4.9. **Cette Table résume l'ensemble de nos résultats sur la base Rimes - ICDAR 2011.** Elle permet de constater l'évolution des performances entre un système HMM générique, un système HMM plus performant à base de modèles contextuels et la combinaison de classifieurs robustes et différents.

4.1.10 Comparaison avec l'état de l'art - 2011

Depuis la rédaction de ce manuscrit, les résultats officiels de la compétition de reconnaissance de mots manuscrits français Rimes - ICDAR 2011 sont apparus (Grosicki et El-Abed [65]). La Table 4.10 montre que le système combiné proposé dans la Section précédente est arrivé en première place. Le système original à base de HMMs contextuels est arrivé quant à lui premier parmi les systèmes non issus de combinaisons, et troisième d'un point de vue général.

Le système proposé par Jouve est une combinaison d'un classifieur à base de HMMs gaussiens et d'un classifieur de type Réseaux de Neurones Récurrents, appris sur la RNNLib. Le deux systèmes proposés par l'IRISA sont basés sur des HMMs à densités continues (Guichard *et al.* [66]). Le prétraitement des images est standard

système	**taux de reconnaissance**
A2iA	94.87%
Jouve	87.47%
système CD	79.2%
IRISA(1)	78.59%
ParisTech	75.12%
IRISA(2)	74.54%

TABLE 4.10 – Comparaison des performances du système proposé à base de HMMs en contexte et de sa combinaison avec d'autres reconnaisseurs avec les systèmes présentés à la compétition Rimes 2011.

et les 60 caractéristiques, décrites dans [8], sont extraites par des fenêtres glissantes et dépendent des lignes de base. Le nombre d'états par caractère varie en fonction de leur longueur. Lors du décodage, les scores des mots sont réévalués au niveau caractère par des SVMs. Les deux systèmes diffèrent par leur utilisation du lexique lors du rescoring. Enfin, le système de ParisTech est un système à base de HMMs contextuels.

4.1.11 Bilan

Dans cette Section, nous avons développé en détail la mise en place de notre système à base de modèles HMMs en contexte et présenté nos résultats sur la base de données Rimes. Les bases du système sont élaborées avec des HMMs de caractères qui ne dépendent pas du contexte (topologie des modèles, choix des caractéristiques) et une base de validation sert à optimiser la construction des arbres de décision pour le système en contexte. Nous avons vu que le système original que nous proposons modélisant les HMMs de caractères en fonction de leur contexte permet de diminuer relativement de plus de 15% le nombre de mots mal reconnus sur la base de test 2011.

Lorsque nous comparons nos résultats à la compétition Rimes 2009, nous constatons que notre système HMM est extrêmement compétitif par rapport aux autres systèmes utilisant des HMMs - il arrive en troisième place, les première, deuxième et troisième places étant occupées par des systèmes utilisant une combinaison de classifieurs. Ainsi, en système seul, notre système à base de modèles en contextes dépasse les autres systèmes HMM. Cependant, la comparaison à un autre classifieur (basé sur les LSTM-RNN) montre qu'il faut encore chercher à améliorer les systèmes HMMs.

Dans le chapitre suivant, nous montrerons d'ailleurs l'une des méthodes possibles pour améliorer notre système. Enfin, nous avons vu à la fin de cette Section que la combinaison de classifieurs permet d'améliorer considérablement les performances d'un système. S'il est indispensable de construire les classifieurs les plus robustes possible, il est aujourd'hui incontournable de considérer la combinaison de classifieurs pour construire le meilleur système possible. Grâce à ce procédé, le système proposé à base de HMMs contextuels, combiné à des reconnaisseurs différents (à base de réseaux de neurones entre autres) est arrivé premier de la compétition ICDAR 2011 de reconnaissance de mots sur la base Rimes. (Il est aussi arrivé premier pour la reconnaissance de lignes).

4.2 Résultats sur une autre base latine : la base IAM

La base IAM (Marti et Bunke [110], Marti et Bunke. [107]) est aujourd'hui souvent utilisée comme base de données pour l'évaluation de systèmes à l'état de l'art. Aucune compétition officielle n'a eu lieu à ce jour mais, le nombre de laboratoires utilisant cette base de données étant important, nous avons jugé utile d'évaluer notre système sur cette base afin de prouver qu'il n'est pas spécifique au français.

Nous avons travaillé sur la tâche « reconnaissance de mots isolés ». Pour l'élaboration et l'évaluation de notre système, nous avons utilisé uniquement les images de mots dont la segmentation était propre, soit 74154 images différentes, correspondant à un vocabulaire de 10212 mots. La base est divisée en une base d'apprentissage de 46901 images, deux bases de validation de 6442 et 7061 images respectivement et une base de test de taille 13750 images. La base IAM étant une base de mots anglais, nous n'avons pas modélisé de caractères accentués. Cependant, de nouveaux signes de ponctuation ont du être modélisés : . , - " ' ( ) :;!? / +. Nous avons appris un total de 75 modèles de caractères.

Comme pour la base Rimes, nous avons dans un premier temps évalué le meilleur ensemble de caractéristiques pour le système à l'aide d'un système appris sur la base d'apprentissage et testé sur les bases de validation. Les valeurs optimales trouvées sont $w = 8$ pixels et $\delta = 4$ pixels. Nous avons appliqué une régression sur les caractéristiques puis évalué la meilleure topologie possible pour chaque caractère, illustrée sur la Figure 4.6.

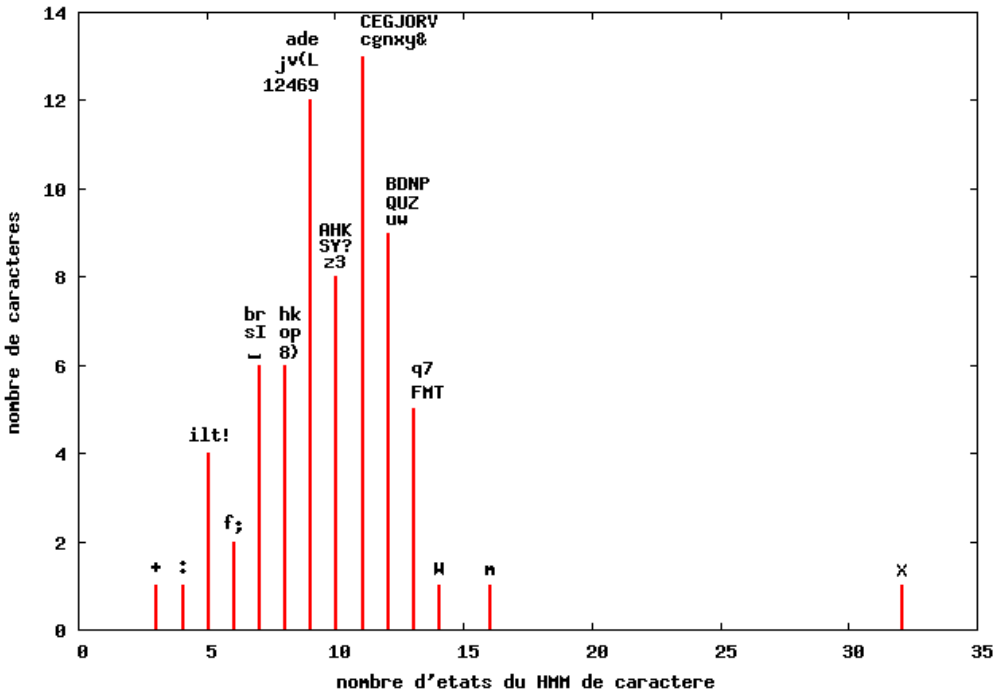


FIGURE 4.6 – Mise en place du nombre d'états optimal pour chaque HMM de caractère sur la base d'apprentissage de IAM.

Système	Ensemble d'apprentissage	Taux de reconnaissance
CI	train	66.82%
CI	train+valid1+valid2	67.86%
CD	train	67.91%
CD	train+valid1+valid2	**69.53%**

TABLE 4.11 – Comparaison des systèmes présentés, modélisant les caractères en fonction de leur contexte ou non. Résultats sur la base de test de IAM contenant 13750 images.

Le taux de reconnaissance obtenu par le système sans contexte sur les bases de validation est de 68.8%, avec un mélange de 20 gaussiennes par état.

En nous basant sur ce premier système, nous avons construit ensuite le système dynamique à base de HMMs en contexte, de la même manière qu'il a été présenté pour la base Rimes. Nous avons déterminé les seuils optimaux ΔL_{min} et Γ_{min} pour apprendre les modèles en contexte à l'aide des bases de validation. Les valeurs trouvées sont $(\Delta L_{min}, \Gamma_{min}) = (450, 50)$, définissant un total de 3053 clusters (états) et 2590 modèles de trigraphes différents.

Une fois les seuils choisis, nous avons augmenté incrémentalement le nombre de gaussiennes dans les mélanges définissant chaque état à $N_G = 20$ et avons évalué le système sur les bases de validation. Avec ces modèles en contexte, nous obtenons un taux de 70.2% de mots bien reconnus sur les bases de validation de IAM. Ceci représente une augmentation de 2% en relatif par rapport au résultat d'un système sans contexte appris sur les mêmes données.

Nous avons ensuite évalué nos modèles sur les 13750 images de la base de test d'IAM. Pour cela, nous avons appris deux systèmes différents : le premier est le système précédemment obtenu avec les seuils $(\Delta L_{min}, \Gamma_{min}) = (450, 50)$ donnant un taux de 70.2% sur les bases de validation. Le second a été appris en ajoutant les données des bases de validation pour l'apprentissage des clusters définis par les mêmes $(\Delta L_{min}, \Gamma_{min})$. Les deux systèmes ont donc le même nombre de clusters définis et le même nombre de gaussiennes dans chaque mélange. Pour compléter notre étude, nous avons aussi appris des modèles HMMs de caractère sans contexte avec les mêmes paramètres que définis plus haut mais en ajoutant les bases de validation aux données d'apprentissage. Les résultats sont donnés sur la Table 4.11.

Les résultats de la Table 4.11 nous permettent d'affirmer que l'utilisation de modèles en contexte pour les systèmes de reconnaissance de mots à base de HMMs

de caractères est très efficace, quel que soit le langage. La réduction relative d'erreurs entre les deux meilleurs systèmes - avec et sans contexte - est de 5.2%, ce qui représente un gain non négligeable pour le système présenté. Un test de t-Student apparié démontre que la différence de performances entre le système CI et le système CD (tous deux appris sur train+valid1+valid2) est statistiquement significative avec un taux de confiance de 99.9% [4].

Le système que nous avons contruit pour la reconnaissance de mots isolés sur la base IAM est difficilement comparable aux autres systèmes de l'état de l'art sur cette base car notre protocole de décodage ne correspond pas au protocole habituellement utilisé : le plus souvent, les résultats sont donnés sur les lignes de la base de test et un modèle de langage calculé sur le LOB-corpus est utilisé. Un système cependant donne des résultats sur une découpe de la base IAM proche la nôtre et un dictionnaire similaire : le sytème de l'UPV [170, 43], qui a aussi participé à la compétition Rimes 2009 [64]. Il est basé sur une combinaison de systèmes faite à partir des votes de trois classifieurs de type hybride HMM/NN. La combinaison est accompagnée d'une diminution du lexique de test avec un classifieur NN holistique en amont du décodage. Pour leurs expériences, Zamora-Martínez *et al.* [170] ont rassemblé la base de test et une base de validation pour en faire leur ensemble de test. Ils ont éliminé les mots mal segmentés ainsi que les mots contenant de la ponctuation et ont obtenu le découpage suivant : 41743 images d'apprentissage, 6313 images pour la validation, 17477 images de test et un lexique contenant 10199 mots. Leur système combiné obtient un taux de reconnaissance de 77.9% sur leur ensemble de test. Si ce résultat est difficilement comparable à ce que nous obtenons avec le système proposé à base de HMMs en contexte (système combiné, absence de ponctuation, ensemble de tests et dictionnaire différents), nous avons cependant souhaité effectuer notre propre combinaison de systèmes pour pouvoir comparer nos résultats à un certain point. Nous avons combiné les sorties de trois reconnaisseurs : nos deux systèmes HMMs (l'un indépendant du contexte et l'autre, contextuel) et un système hybride HMM/NN à base de segmentation explicite d'un mot en graphèmes, élaboré par A2iA [5]. Nous obtenons un taux de 78.1% de mots bien reconnus [12] sur les 13750 images de mots de la base de test d'IAM avec un lexique de taille 10212. Ce dernier taux montre que nos résultats sont comparables à ceux de l'état de l'art.

4. La valeur critique du test de Student entre les deux systèmes est de 4.80, soit largement supérieure au seuil critique de 3.291 pour un taux de confiance de 99.9% et un nombre d'échantillons supérieur à 1000.

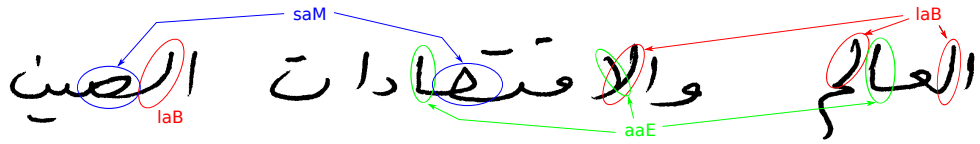


FIGURE 4.7 – Illustration de l'influence du contexte des caractères pour l'écriture arabe. Les trois mots العالم , والاقتصادات et الصين ont été écrits par le même scripteur, cependant les caractères laB , aaE et saM ont des formes différentes selon leur contexte.

Nous avons donc montré dans cette Section que la modélisation de caractères en fonction de leur contexte améliore les performances par rapport à un système HMM générique et ce, quel que soit le langage utilisé (français/anglais).

4.3 Application du système sur l'écriture Arabe

Nous avons montré dans les sections précédentes que le système que nous proposons à base de HMMs de caractères dépendant de leur contexte est performant sur des bases d'écriture latine, que le langage soit le français ou l'anglais. Dans cette section, nous généralisons ce constat à l'écriture manuscrite arabe.

Les trois mots de la Figure 4.7 montrent qu'il est pertinent d'utiliser des modèles contextuels pour l'arabe. Ecrits par un même scripteur, ils présentent des formes différentes pour les caractères ص , ل et ا selon le mot dans lequel ils sont écrits et les caractères qui les entourent.

Nous verrons dans un premier temps les spécificités de l'écriture arabe par rapport à l'écriture latine que nous avons dû prendre en compte pour notre modélisation (Section 4.3.1). Ensuite nous présenterons la base OpenHart, proposée par le NIST en 2010 dans le cadre du projet MadCat (Section 4.3.3). Cette base est notre base d'étude pour nos expériences sur l'écriture manuscrite arabe. Après avoir introduit le système élaboré en 2010 et envoyé à la compétition OpenHart de reconnaissance de mots arabes dans un document (Section 4.3.4), nous discuterons de l'influence des modèles de langage sur nos performances (Section 4.3.5). Enfin, nous étudierons dans la Section 4.3.6 l'influence de certaines des améliorations proposées dans ce manuscrit sur un sous-ensemble de la base OpenHart. Ces dernières expériences nous permettront de planifier les recherches futures à effectuer sur la base Openhart complète.

4.3.1 L'écriture arabe

Tout comme le latin, l'arabe est une écriture cursive. Cependant, l'écriture arabe est consonantique, ce qui signifie que l'alphabet ne contient que très peu de voyelles (trois) et que l'écriture utilise des signes diacritiques (traits ou points) représentant des semi-voyelles ou voyelles courtes en remplacement. Selon leur position, les diacritiques peuvent radicalement changer le sens d'un caractère et donc le sens d'un mot, ce qui rend le traitement de l'écriture manuscrite arabe difficile. De plus, si 29 caractères différents sont définis en arabe, la forme de certains caractères peut changer selon leur position dans un mot : au début, au milieu ou à la fin d'un mot, ou bien encore si ce sont des caractères isolés. En tout, 120 *monographes* différents sont définis pour l'arabe. Pour plus de détails, un bon récapitulatif des spécificités de l'écriture arabe se trouve dans Amara et Bouslama [3].

L'utilisation d'un système HMM sans segmentation pour la reconnaissance de l'arabe se justifie par la forme que peuvent prendre certains caractères. On observe d'une part un fort chevauchement entre certains caractères, qui rend une découpe presque impossible et, d'autre part, les ascendants (ou descendants) d'un caractère qui peuvent s'étendre sur (ou sous) les autres caractères du mot auquel il appartient : une segmentation aurait de fortes chances de couper le caractère de son sens et ajouterait du bruit aux autres caractères. De plus, des coupures sont souvent observées dans un mot en arabe, découpant le mot en pseudo-mots. Une approche à base de fenêtres glissantes et surtout l'utilisation de modèles HMMs qui ont le pouvoir d'absorber les variations de longueur d'un caractère sont d'autant plus justifiées pour cette écriture.

4.3.2 Les modèles en contexte pour l'arabe

Nous avons présenté au Chapitre 3 un système à base de modèles de caractères dépendants de leur contexte gauche et droit. En arabe, le sens de l'écriture étant de droite à gauche, nous noterons par la suite, pour plus de simplicité, les caractères encadrant une lettre comme « contexte précédent » (signe '-' dans notre notation) et « contexte suivant » (signe '+' dans notre notation) au lieu de contextes gauche

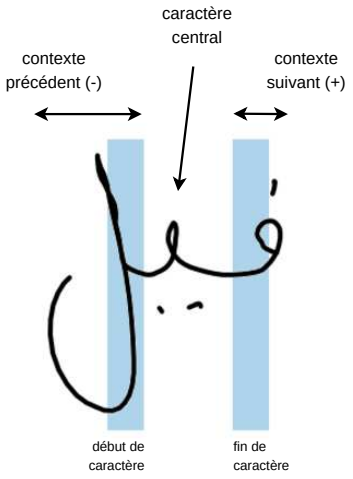


FIGURE 4.8 – Exemple de contexte précédant et suivant pour le caractère ـيـ dans le mot فيل

et droit. Par exemple sur la Figure 4.8 le caractère ـيـ dans le mot فيل est entouré par فـ (contexte précédent) et ـل (contexte suivant). Pour écrire les caractères arabes aisément en ASCII, nous utilisons la translitération proposée pour la base de données IFN-Enit [131]. Ainsi, sur l'exemple précédent, le trigraphe défini se note : faB-yaM+laE.

Afin de construire nos modèles HMMs contextuels, nous avons utilisé nos connaissances sur la forme des lettres et de leurs extrémités gauche et droite. Nous avons élaboré l'ensemble des questions binaires utilisées dans la construction des arbres de clustering en suivant deux hypothèses :

- les caractères dont la partie finale ont une forme de trait similaire auront tendance à influer de la même manière sur le caractère central suivant
- les caractères dont la partie initiale ont une forme de trait similaire auront tendance à influer de la même manière sur le caractère central précédent

Ainsi, les caractères arabes ayant une forme similaire [112] ont plus de chance d'être rassemblés dans des questions. Des fins de caractère similaires conduisent à des regroupements dans des questions se rapportant au « contexte précédent » (P_QS) et les débuts de caractères similaires à des questions sur le « contexte suivant » (S_QS).

Par exemple, les caractères ڧ , ڤ , ف , ق ont leur début (partie droite) qui se ressemblent. Ils sont regroupés dans une question de type S_QS : { *+faM , *+kaM , *+faE , *+kaE }. L'ensemble des questions créées pour l'écriture arabe peut être trouvée en Annexe A.2.

4.3.3 La base OpenHart

La base OpenHart a été proposée en 2010 par NIST dans le but d'effectuer une évaluation des systèmes de l'état de l'art sur une base manuscrite arabe de très grande taille [74, 167]. Les données sont séparées en deux phases (Phase 1 et Phase 2), chacune possédant un ensemble d'apprentissage (Train_PhaseX), de validation (DevTest_PhaseX) et de test (Eval_PhaseX). L'ensemble de test utilisé pour la compétition est Eval_Phase2, dont les données n'ont été distribuées qu'au moment de la compétition. Le nombre total d'images de mots contenues dans la base OpenHart peut se compter en millions et correspond à un lexique d'une centaine de milliers de mots :

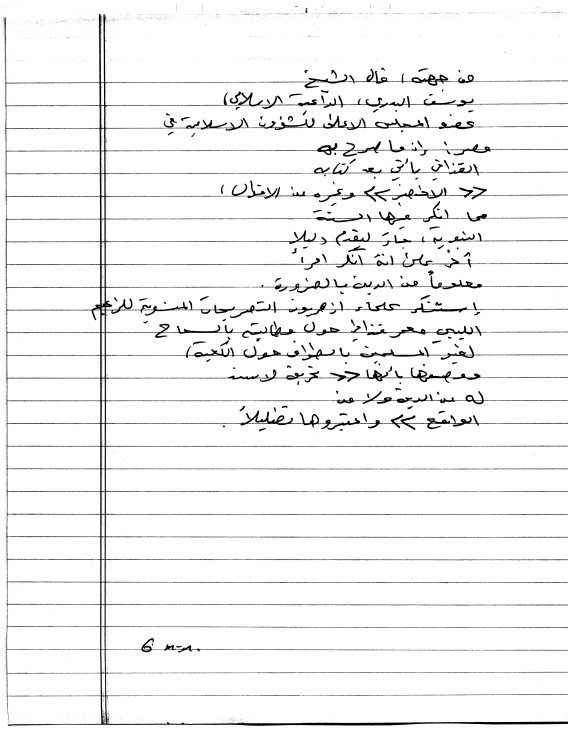
- 758 936 images pour Train_Phase1,
- 84 405 images pour DevTest_Phase1,
- 48 342 images pour Eval_Phase1,
- 2 975 417 images pour Train_Phase2,
- 57 462 images pour DevTest_Phase2,
- 64 359 images pour Eval_Phase2,

soit un total de 3 988 921 images dans la base complète.

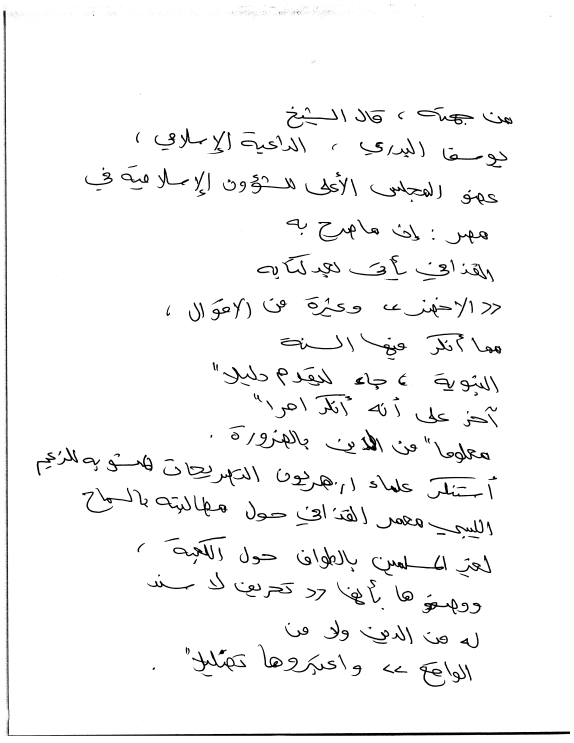
La qualité des images est variable : 25% des données sont écrites sur des pages avec des lignes et 75% sur des pages blanches. De plus, si 90% des pages sont écrites dans des conditions normales, il a été demandé à 5% des scripteurs d'écrire rapidement et sans prendre soin de leur écriture. Quelques exemples de pages de la base OpenHart sont donnés sur la Figure 4.9. La tâche de reconnaissance proposée s'annonce ainsi difficile mais passionnante.

4.3.4 Présentation du système développé pour la compétition OpenHart 2010

Afin de donner un aperçu de nos performances sur l'ensemble de la base OpenHart, nous présentons dans cette section la mise en place du système développé pour



(a) exemple d'écriture sur une page avec des lignes



(b) exemple d'écriture sur une page blanche

FIGURE 4.9 – Exemples de pages manuscrites arabes de la base OpenHart

la compétition OpenHart 2010 de reconnaissance de mots arabes. Pour la compétition, chaque image de page manuscrite a été fournie avec une annotation au format XML donnant le découpage de la page en lignes et le découpage des lignes en mots. Nous avions donc à notre disposition des images de mots isolés accompagnées de leur localisation dans la segmentation de chaque ligne.

Le système élaboré pour la compétition 2010 a été appris sur les 739849 images de mots de Train_Phase1 afin d'accélérer l'apprentissage (nous ne disposions que d'un mois pour élaborer un système et le soumettre). Les caractéristiques extraites sont les caractéristiques statistiques et géométriques présentées dans la Section 2.1. Les caractéristiques directionnelles n'étaient pas encore incluses dans notre extraction. Le nombre d'états par HMM a été fixé à 12 pour l'ensemble des caractères excepté pour les caractères de ponctuation (4 états émetteurs) : l'adaptation de la topologie à la longueur des caractères n'était pas encore incluse dans notre système.

Le dictionnaire utilisé pour le décodage est le lexique constitué des 20000 mots les plus fréquents dans la base OpenHart. Il couvre 80% des mots présents dans le lexique de l'ensemble de validation DevTest_Phase1. Une première évaluation nous a permis de calculer que **le taux de reconnaissance du système générique est de 35.0% sur les** 81948 **images de DevTest_Phase1.**

Nous avons ensuite procédé à la modélisation de caractères en fonction de leur contexte. Nous avons compté 22183 trigraphes dans l'ensemble d'apprentissage de Phase1. L'utilisation des seuils ($\Delta L_{min} = 500, \Gamma_{min} = 200$) nous a permis de générer un total de 4950 clusters d'états. De même que pour le latin, nous avons observé que la diminution du nombre total d'états définis conduit à des modèles de trigraphes identiques car ils partagent pour chaque position d'état exactement les mêmes clusters. Au total 3217 trigraphes différents sont définis à partir des 4950 clusters.

Lors du décodage des trigraphes avec le lexique de 20000 mots, les arbres calculés lors de l'apprentissage sont utilisés pour assigner des clusters aux états des trigraphes non vus lors de l'apprentissage. Au total 15897 trigraphes sont comptés dans ce dictionnaire dont plus de 2000 n'ont pas été appris. On voit bien, dans ce cas, l'utilité du calcul des clusters d'état par arbres de décision. Finalement, **le taux de reconnaissance du système en contexte est de** 38.4% **sur l'ensemble DevTest_Phase1, ce qui représente un gain relatif de 9.7% par rapport au système générique.**

Les taux obtenus montrent clairement que l'utilisation de modèles HMMs de caractères dépendants de leur contexte améliore les performances. Cependant, com-

parativement aux taux obtenus sur d'autres bases de données (Rimes, IAM) et qui oscillent entre 70% et 85% (cf Sections 4.1.6, 4.1.8 et 4.2), le pourcentage final d'erreurs sur cette base semble élevé (plus de 60% des mots de la base de test sont mal reconnus). Nous avons donc proposé d'améliorer nos performances à l'aide de modèles de langage.

4.3.5 De l'utilisation de modèles de langage pour la reconnaissance de lignes manuscrites arabes

Les modèles de langage visent à modéliser statistiquement un langage donné. Ils associent une probabilité à une séquence de mots consécutifs permettant de savoir *a priori* si la séquence est probable ou non. En général, pour la reconnaissance de la parole ou de l'écriture, on utilise des n-grammes, c'est-à-dire des probabilités d'enchaînement de deux, trois, ... ou n mots [156]. Le calcul des probabilités peut se faire entre autres par un simple comptage du nombre d'occurrences de chaque expression dans un ensemble de données. Il existe d'autres techniques d'évaluation des probabilités (avec des réseaux de neurones par exemple [10, 149]) et il est aussi possible de grouper des mots dans des classes et d'évaluer uniquement les transitions entre classes [17, 85] mais une étude comparative des modèles de langage utilisés en reconnaissance de la parole et de l'écrit n'est pas l'objet de ce travail. Nous utiliserons donc dans nos expériences des n-grammes basés sur les comptages d'occurrences d'expressions, réputés pour leur simplicité de mise en oeuvre et leurs bonnes performances. L'outil SRILM [154] a servi a élaborer nos modèles de langage.

L'utilisation des modèles de langage en reconnaissance d'écriture s'inscrit généralement dans le cadre de la reconnaissance de lignes. Afin d'être reconnues, les lignes peuvent être segmentées et plusieurs options de segmentation peuvent être proposées pour chaque ligne. Pour chaque option de segmentation, l'ensemble des pseudo-mots découpés est traité dans le reconnaisseur et les lignes formées sont ensuite évaluées par le modèle de langage. Cette stratégie étant dépendante d'une étape préliminaire de segmentation de ligne, elle est souvent sujette à erreurs. Dans le cas de la base de données MADCAT, nous avons à disposition la véritable segmentation des lignes en mots, annotation effectuée par des opérateurs. Nous souhaitons profiter de cet avantage pour observer directement l'influence des modèles de langage sur notre décodage.

La procédure de décodage avec modèles de langage que nous avons utilisée pour

notre système envoyé à la compétition OpenHart 2010 est décrite maintenant. Dans un premier temps nous avons choisi un dictionnaire pour le décodage. Celui-ci est le lexique contenant les 20000 mots les plus fréquents dans la base OpenHart présenté dans la section précédente. Pour chaque mot du dictionnaire, nous avons ensuite compté tous les unigrammes, bigrammes et trigrammes associés à ce mot sur une base de données prédéfinie. La base peut être soit la base d'apprentissage soit une base externe. Nous avons choisi la base de données Arabic GigaWord [130] pour nos expériences, base composée de documents contenant 850 millions de mots collectés principalement sur des journaux. Lors du comptage, les n-grammes associés à des mots hors vocabulaire n'ont pas été pris en compte (le vocabulaire est dit *fermé*). Une méthode de lissage de type Kneser-Ney est appliquée sur les trigrammes inconnus du test. Une fois le dictionnaire et le modèle de langage fixés, nous avons décodé l'ensemble des mots du test avec notre système à base de HMMs contextuels présenté dans la section précédente. Le décodage est lancé en 50-best, c'est-à-dire que pour chaque image de mot, les 50 meilleures sorties sont données avec un score de reconnaissance. Les scores sont normalisés entre 0 et 1 et somment à 1 pour chaque image de mot. Ligne par ligne et mot par mot, les sorties 50-best sont ensuite enchaînées pour reconstituer des options de reconnaissance de ligne. Finalement, le modèle de langage repondère l'ensemble des scores obtenus selon la probabilité d'occurrence de chaque n-gramme. La pondération est guidée par un *language model scaling factor* qui permet la combinaison efficace des scores de reconnaissance avec les probabilités des n-grammes. Sa valeur optimale est trouvée à l'aide d'une base de validation. La pondération effectuée par le modèle de langage permet de faire remonter en première place des mots dont le score était initialement très bas mais qui correspondent à une expression plus probable que celle induite par la reconnaissance sur le mot isolé.

Nous avons évalué l'influence des modèles de langage sur notre reconnaisseur de mots manuscrits arabes à base de HMMs contextuels. Nous rappelons que, sans modèle de langage, le taux de reconnaissance du système avec modèles en contexte est de 38.4% sur l'ensemble DevTest_Phase1. **Avec le même dictionnaire et un modèle de langage utilisant des trigrammes, nous obtenons 53.4% de mots bien reconnus sur DevTest_Phase1 avec le système original de HMMs contextuels, c'est-à-dire que nous observons un gain de 39% en relatif**, ce qui est très important.

Ce taux se confirme sur l'ensemble de test de la compétiton OpenHart 2010 (Eval_Phase2). Nous avons obtenu un taux de 54.0% de mots bien reconnus avec le

système	ensemble de test	taux de reconnaissance
HMMs non contextuels	DevTest Phase 1	35.0%
HMMs non contextuels + ML trigrammes	DevTest Phase 1	50.0%
HMMs contextuels	DevTest Phase 1	38.4%
HMMs contextuels + ML trigrammes	DevTest Phase 1	53.4%
HMMs non contextuels + ML trigrammes	Eval Phase 2	44.9%
HMMs contextuels + ML trigrammes	Eval Phase 2	54.0%
combinaison CI + CD + hybride + + ML trigrammes	Eval Phase 2	62.3%

TABLE 4.12 – Récapitulatif des performances des systèmes élaborés pour la compétition OpenHart 2010 (ML signifie modèle de langage).

système à base de HMMs en contexte accompagnés du modèle de langage présenté ici. Le meilleur système de la compétition est une combinaison de nos deux systèmes HMMs (générique et contextuel) et d'un système hybride HMM/NN élaboré par A2iA [113]. La combinaison, associée au même modèle de langage à base de trigrammes, a obtenu un taux de reconnaissance de 62.3%.

La Table 4.12 donne le récapitulatif des performances des systèmes élaborés pour la compétition OpenHart 2010.

4.3.6 Evaluation du système sur un sous-ensemble de la base OpenHart

Comme nous l'avons évoqué dans la Section 4.3.4, nous avons manqué de temps au moment de la compétition OpenHart 2010. Nous n'avons donc pas pu élaborer des modèles optimaux pour nos systèmes à base de HMMs (générique et contextuel).

Nous souhaitons cependant explorer les améliorations proposées dans ce manuscrit et évaluer leur influence sur la reconnaissance de l'écriture arabe : utilisation d'un ensemble de caractéristiques plus large et d'une topologie variable des modèles ainsi qu'un calcul précis des seuils de clustering.

Etant donné la taille gigantesque de la base OpenHart, nous proposons dans cette section de travailler sur une sous-partie des données afin d'accélérer les procédures d'apprentissage et de décodage. Nous avons extrait de manière aléatoire 150000 images de mots de Train_Phase1 et Train_Phase2 (75000 sur chacun) pour apprendre nos modèles HMMs et 50000 images de DevTest_Phase1 et DevTest_Phase2 (25000 sur chacun) pour les évaluer : 10000 images sont utilisées pour la validation et 40000 pour le test.

Les caractéristiques extraites sont les caractéristiques statistiques, géométriques et directionnelles présentées au Chapitre 2, Section 2.1, excepté le parcours des fenêtres glissantes qui pour l'arabe est de droite à gauche et non de gauche à droite. De même que pour les bases latines, la taille de fenêtre optimale et le décalage entre deux fenêtres sont optimisés sur la base de validation ($w = 9$ pixels et $\delta = 4$ pixels) et une régression est calculée sur les caractéristiques. Le nombre moyen d'états par HMM est d'abord évalué à 12 puis la procédure d'attribution d'un nombre d'états spécifique pour chaque caractère présentée au Chapitre 2, Section 2.3 est lancée.

Pour le décodage, nous appliquons le lexique utilisé pour la compétition OpenHart 2010 (contenant les 20000 mots les plus fréquents de la base).

Adaptation de la topologie des HMMs aux caractères. Pour un mélange de 20 gaussiennes par caractère, la différence entre les deux types de topologies (constante ou variable) est de **3.4%** : les HMMs de caractère avec 12 états obtiennent 37.7% de mots bien reconnus et ceux à la topologie variable 41.1%. Ces résultats montrent bien l'importance de l'adaptation de la topologie des HMMs pour l'écriture arabe.

Optimisation des seuils de clustering. Nous avons ensuite procédé à la modélisation de caractères en fonction de leur contexte. Nous avons compté 13655 trigraphes dans nos données d'apprentissage. L'optimisation des seuils ΔL_{min} et Γ_{min} pour la construction des arbres a été effectuée à l'aide de la base de validation : le couple optimal ($\Delta L_{min} = 500, \Gamma_{min} = 500$) génère un total de 3706 clusters d'états définissant 1335 trigraphes différents. De même que lors de la compétition, nous

avons pu observer l'utilité du clustering par arbres de décision lors de la phase de test : plus de 3000 nouveaux trigraphes devaient être modélisés. Le taux de reconnaissance des modèles HMM contextuels sur l'ensemble de test proposé est ainsi de 43.6%, soit une augmentation relative de **6.1%** par rapport au système générique sans contexte.

Cette étude sur un sous-ensemble de la base OpenHart nous a permis d'assurer la validité de notre approche. Nous envisageons donc dans un futur proche d'appliquer les améliorations proposées (ajout de caractéristiques directionnelles, adaptation du nombre d'états à la longueur du caractère) à l'ensemble de la base OpenHart. Nous expérons ainsi augmenter nos performances sur la reconnaissance de mots isolés et observer l'impact de cette augmentation sur la reconnaissance de lignes avec modèles de langage.

Conclusion du chapitre 4

Ce chapitre d'expériences a permis d'illustrer sur trois bases de données différentes l'apport de la modélisation de caractères en fonction de leur contexte pour des classifieurs de type HMMs. L'ensemble de nos résultats est résumé dans la Table 4.13.

Nous avons pu constater que, quel que soit l'alphabet utilisé et les caractéristiques de la base de données de travail, la modélisation de caractères en fonction de leur contexte permet d'obtenir de meilleurs performances qu'une modélisation simple de HMMs de caractères. Ainsi, une réduction d'erreurs de près de 4% est observée sur la base de test Rimes 2011 et, pour l'ensemble de test de la base IAM, elle est de 1.6% en absolu. Sur la base de données arabe OpenHart, qui présente une tâche de reconnaissance plus difficile, la modélisation en contexte a permis de gagner 3.4% de mots bien reconnus sur l'ensemble de validation DevTest_Phase1.

Les expériences présentées dans ce chapitre ont aussi permis d'apercevoir des techniques pour améliorer les performances de notre système sans devoir modifier la modélisation. Nous avons ainsi constaté que la combinaison de sorties de classifieurs permet de gagner 5.8% de mots bien reconnus sur la base de test de Rimes 2009 et jusqu'à 15% sur le test 2011. Enfin, l'utilisation de modèles de langage de type n-grammes a augmenté de 15% le taux de reconnaissance de notre système contextuel sur la base OpenHart. Ces deux techniques, brièvement évoquées dans ce manuscrit,

base de données	système	taille de l'ens. d'apprentissage	taille du lexique	ensemble de test	taux de reconnaissance
Rimes	HMMs non contextuels (CI)	51,739	1612	test 2009	81.0%
Rimes	HMMs contextuels (CD)	51,739	1612	test 2009	83.3%
Rimes	combinaison CI + CD + hybride	51,739	1612	test 2009	89.1%
Rimes	HMMs non contextuels	59,203	5744	test 2011	75.4%
Rimes	HMMs contextuels	59,203	5744	test 2011	79.2%
Rimes	combinaison CI + CD + hybride + RNNs	59,203	5744	test 2011	94.4%
IAMdb	HMMs non contextuels	60,404	10212	test	67.9%
IAMdb	HMMs contextuels	60,404	10212	test	69.5%
OpenHart	HMMs non contextuels + trigram LM	739,849	20000	DevTest Phase 1	50.0%
OpenHart	HMMs contextuels + trigram LM	739,849	20000	DevTest Phase 1	53.4%
OpenHart	HMMs non contextuels + trigram LM	739,849	20000	Eval Phase 2	44.9%
OpenHart	HMMs contextuels + trigram LM	739,849	20000	Eval Phase 2	54.0%
OpenHart	combinaison CI + CD + hybride + + trigram LM	739,849	20000	Eval Phase 2	62.2%

TABLE 4.13 – Récapitulatif de l'ensemble de nos résultats sur les trois bases de données étudiées (Rimes, IAM et OpenHart)

font partie intégrante des pistes de recherche pour nos travaux à venir.

Avant de les utiliser, nous tenons toutefois à poursuivre l'amélioration de notre modélisation de caractères. En effet, meilleur est le classifieur, plus efficace sera son utilisation lors d'une combinaison ou d'un décodage avec modèles de langage. A cette fin, nous présentons dans le prochain chapitre l'une des pistes envisagées pour l'amélioration de notre système : l'adaptation des modèles HMM au scripteur.

Chapitre 5

Adaptation des modèles HMMs au scripteur

Introduction

Nous avons présenté dans les chapitres précédents un système original de reconnaissance de mots manuscrits à base de HMMs en contexte et avons prouvé l'efficacité de notre approche sur diverses bases de données et différents langages. Dans ce chapitre, nous introduisons une autre façon d'améliorer les performances de notre reconnaisseur : l'adaptation au scripteur. Adapter un système à un scripteur spécifique signifie modifier les paramètres du système de façon à garder son allure générale tout en l'ajustant aux paramètres spécifiques au scripteur. Cette approche poursuit notre idée générale de construire des modèles de plus en plus précis. La précision ici est représentée par les spécificités du scripteur, qui sont ajoutées au modèle. La différence est que la modification des modèles construits lors de l'apprentissage s'effectue dans ce cas lors de la reconnaissance.

L'adaptation d'un système au scripteur est une alternative très efficace pour répondre au schéma idéal qui serait d'apprendre les paramètres du système sur un grand nombre de données du scripteur considéré. Indermühle *et al.* [77] expliquent qu'il faut un minimum de 1000 mots pour que l'apprentissage de modèles sur les données d'un scripteur spécifique donne les mêmes résultats qu'un apprentissage général suivi d'une adaptation. Ce chiffre équivaut à au moins une dizaine de pages transcrites écrites par le scripteur *pour l'apprentissage*, sans compter les données à tester. Pour une tâche comme la reconnaissance de courrier entrant d'une entreprise,

ce scénario n'est pas concevable mais il peut être utilisé pour la reconnaissance d'archives, comme des tables de recensement, écrites par une même personne au fil des années. Etant donné que nous travaillons sur des bases de données pour lesquelles un scripteur a écrit en moyenne 5 pages de texte manuscrit (cf Section 5.3), nous ne pouvons utiliser ce type d'apprentissage.

Il existe deux manières d'aborder l 'adaptation : soit on modifie les paramètres du système déjà appris pour les adapter au scripteur, soit on modifie les données du scripteur pour qu'elles s'ajustent au système. Dans nos travaux, nous utilisons le premier type. On dit alors qu'on calcule une *transformation* des paramètres du système pour les adapter au scripteur.

Que ce soit dans le domaine de la parole ou de l'écrit, deux techniques d'adaptation au locuteur / scripteur se distinguent : l'adaptation MAP (*Maximum a Posteriori*) et l'adaptation MLLR (*Maximum Likelihood Linear Regression*). Quelle que soit la technique utilisée, il existe deux manières d'adapter un modèle au scripteur : soit on dispose de données étiquetées (transcriptions) pour chaque scripteur à adapter et dans ce cas l'adaptation est dite supervisée, soit on ne dispose pas des transcriptions et dans ce cas on dit que l'adaptation est non supervisée. Nous avons choisi le deuxième scénario, qui est plus conforme aux données sur lesquelles nous travaillons. Dans [115], les deux classes de techniques ont été unifiées et les différents modes d'adaptation ont été explorés pour la reconnaissance de la parole.

L'adaptation au scripteur est une technique encore peu utilisée en reconnaissance d'écriture manuscrite. La raison principale est l'absence de volumes conséquents de données monoscripteur dans les bases de données publiques actuelles. Vinciarelli et Bengio [161] par exemple comparent plusieurs techniques d'adaptation (MAP, MLLR, les deux combinées) et montrent que si l'adaptation de type MLLR obtient avec peu de données des résultats convenables (qui stagnent rapidement), l'adaptation MAP nécessite plusieurs centaines de mots pour atteindre des performances comparables à un système appris directement sur les données du scripteur. Des techniques ont cependant été proposées pour répondre à ce problème et ont été utilisées avec succès. L'utilisation de *styles d'écriture* par exemple permet de supprimer le problème du manque de données par scripteur en regroupant ceux ayant une écriture similaire. L'adaptation se fait ensuite en utilisant le style d'écriture appris le plus proche du scripteur à adapter. Fink et Plötz [49] utilisent ainsi l'apprentissage de modèles de style pour la reconnaissance de lignes. Dans leurs travaux, Nosary *et al.* [123, 122] utilisent des *invariants de scripteurs* (formes élémentaires redondantes

dans l'écriture, spécifiques pour chaque scripteur) pour leur adaptation.

Dans ce chapitre, nous présentons notre travail sur l'adaptation de modèles HMMs en contexte au scripteur. Pour cette étude, nous avons choisi des techniques d'adaptation largement utilisées pour la reconnaissance de la parole. Ces techniques ont prouvé leur efficacité sur des modèles HMMs de triphones dont les états sont partagés, c'est pourquoi nous souhaitons les appliquer à nos modèles de trigraphes : l'adaptation MAP et l'adaptation MLLR. Les principes de ces deux techniques d'adaptation sont expliqués dans les Sections 5.1.1 (MAP) et 5.1.2 (MLLR). Dans la Section 5.2, notre procédure d'adaptation non supervisée est ensuite détaillée, puis les résultats de nos expériences sur la base IAM sont présentés dans la Section 5.3. Dans cette dernière section, nous discuterons aussi autour des améliorations possibles des techniques proposées, qui seront au coeur de nos travaux futurs.

5.1 Techniques d'adaptation des modèles HMM au scripteur

Nous disposons d'un côté d'un modèle HMM avec les paramètres $\boldsymbol{\lambda}$, appris sur des données multi-scripteur (ce sont les modèles de HMMs de caractères en contexte présentés au Chapitre 3) et, de l'autre côté, d'observations $\boldsymbol{O}$ correspondant au scripteur auquel on souhaite s'adapter ($\boldsymbol{O}$ est l'ensemble des vecteurs de caractéristiques extraits des images de mots qu'il a écrits). On cherche à modifier $\boldsymbol{\lambda}$ afin d'obtenir un nouveau modèle adapté avec pour paramètres $\hat{\boldsymbol{\lambda}}$, qui maximisent la probabilité *a posteriori* $p(\hat{\boldsymbol{\lambda}}|\boldsymbol{O})$:

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\ p(\boldsymbol{\lambda}|\boldsymbol{O}) \tag{5.1}$$

En utilisant le théorème de Bayes, $p(\boldsymbol{\lambda}|\boldsymbol{O})$ peut s'écrire :

$$p(\boldsymbol{\lambda}|\boldsymbol{O}) = \frac{p(\boldsymbol{O}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})}{p(\boldsymbol{O})} \tag{5.2}$$

où $p(\boldsymbol{O}|\boldsymbol{\lambda})$ est la vraissemblance du HMM avec les paramètres $\boldsymbol{\lambda}$ et $p(\boldsymbol{\lambda})$ est la distribution *a priori* des paramètres $\boldsymbol{\lambda}$. A partir de cette formule, on peut choisir de procéder à deux techniques d'adaptation différentes. Soit on a un *a priori* sur la distribution de $\boldsymbol{\lambda}$ (on dit alors que $p(\boldsymbol{\lambda})$ est non uniforme) : dans ce cas une adaptation

Bayesienne est effectuée par la technique du Maximum a Posteriori (MAP, Gauvain et Lee [55, 56]), décrite en Section 5.1.1. Cet *a priori* permet d'assurer une stabilité des modèles adaptés et donc de nécessiter moins de données pour calculer correctement $\hat{\boldsymbol{\lambda}}$. Dans l'autre cas, on estime que $p(\boldsymbol{\lambda})$ est uniforme : $p(\boldsymbol{\lambda})$ est constant. On pratique alors une adaptation par maximisation de la vraisemblance avec une régression linéaire, en anglais *maximum likelihood linear regression* (MLLR, Digalakis *et al.* [29], Leggetter et Woodland [92]). A l'inverse de l'adaptation MAP, l'adaptation MLLR nécessite peu de données. De plus, elle ne requiert aucun *a priori* sur $\hat{\boldsymbol{\lambda}}$. Ce type d'adaptation est décrit dans la Section 5.1.2.

Adapter un système à un scripteur, lorsque le système est basé sur des HMMs Gaussiens, se fait en modifiant les moyennes et/ou les variances des distributions gaussiennes. Selon le type d'adaptation, les transformations sont différentes. Dans la suite, nous noterons $\boldsymbol{\mu_m}$ la moyenne de la gaussienne numéro m. Pour plus de clarté, nous ne précisons pas de quel mélange (et donc de quel état) fait partie la gaussienne m mais il est implicite que celle-ci prend sa valeur dans l'ensemble des gaussiennes de tous les états partagés définis dans notre système de HMMs en contexte. Au total, M gaussiennes sont présentes dans le système. De même, $\boldsymbol{\Sigma_m}$ est la matrice de covariance de la gaussienne m. Nous rappelons que $\boldsymbol{\Sigma_m}$ est diagonale. L'ensemble des observations du scripteur à adapter est noté $\boldsymbol{O}$ et est de longueur T : $\boldsymbol{O} = \boldsymbol{o}(1), \boldsymbol{o}(2), \ldots \boldsymbol{o}(T)$. Il correspond à l'ensemble des vecteurs de caractéristiques extraits des images de mots écrits par le scripteur considéré.

5.1.1 Adaptation MAP

Reprenant la formule de Bayes (équation [5.2]), l'adaptation de type *Maximum a Posteriori* [55] cherche une solution à :

$$\frac{\partial}{\partial \boldsymbol{\lambda}} p(\boldsymbol{O}|\boldsymbol{\lambda}) p(\boldsymbol{\lambda}) = 0 \tag{5.3}$$

c'est-à-dire que l'on cherche à minimiser le risque Bayesien sur les données d'adaptation $\boldsymbol{O}$, sachant que l'on connaît la distribution *a priori* de $\boldsymbol{\lambda}$, $p(\boldsymbol{\lambda})$.

Gauvain et Lee [55] proposent que $p(\boldsymbol{\lambda})$ soit le produit d'une loi de Dirichlet [78] – qui modélise la distribution *a priori* des poids des gaussiennes dans les mélanges – et de densités de probabilité Wishart-normales [27] – qui modélisent les paramètres des gaussiennes des mélanges (moyenne, variance ; une loi Wishart-normale

par gaussienne). De par sa définition, cette probabilité permet à l'algorithme EM de calculer les paramètres des modèles adaptés.

En général pour l'adaptation MAP, seule la moyenne des gaussiennes du modèle est adaptée ([161, 77]). La formulation de l'adaptation des moyennes est obtenue en appliquant l'algorithme EM à la distribution *a priori* proposée [55] :

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{m}} = \frac{1}{N_m + \tau} \sum_{t=1}^{T} \gamma_m(t) \boldsymbol{o}(t) \quad + \quad \frac{\tau}{N_m + \tau} \boldsymbol{\mu}_{\boldsymbol{m}} \tag{5.4}$$

τ est un poids que l'on donne à l'*a priori* sur $\boldsymbol{\lambda}$, $\gamma_m(t)$ est la probabilité d'être dans la gaussienne m lorsqu'on observe $\boldsymbol{o}(t)$ et N_m est la vraisemblance d'occupation de la gaussienne m :

$$N_m = \sum_{t=1}^{T} \gamma_m(t) \tag{5.5}$$

Plus τ est fort, plus $\hat{\boldsymbol{\mu}}_{\boldsymbol{m}}$ sera proche de $\boldsymbol{\mu}_{\boldsymbol{m}}$, c'est-à-dire du modèle initial indépendant du scripteur. En revanche, si τ est égal à zéro, alors seules les observations du scripteur sont utilisées pour le calcul de $\hat{\boldsymbol{\mu}}_{\boldsymbol{m}}$. Similairement, si N_m est faible, alors la valeur de $\hat{\boldsymbol{\mu}}_{\boldsymbol{m}}$ sera plus influencée par le modèle initial que par les données du scripteur.

5.1.2 Adaptation MLLR

L'approche de l'adaptation de type *maximum de vraisemblance et régression linéaire* est de dire que $\boldsymbol{\lambda}$ est non informatif et que, en conséquence, $p(\boldsymbol{\lambda})$ est constant [114, 92]. Ainsi la solution cherchée n'est plus celle de l'équation 5.3 mais celle de :

$$\frac{\partial}{\partial \boldsymbol{\lambda}} p(\boldsymbol{O}|\boldsymbol{\lambda}) = 0 \tag{5.6}$$

L'hypothèse suivante est alors faite : il existe une fonction de régression permettant d'aller de l'espace général (modèles indépendants du scripteur) à un espace spécifique au scripteur (modèles adaptés). Les paramètres de cette régression (et $\hat{\boldsymbol{\lambda}}$) sont estimés par maximum de vraisemblance. Dans [116, 114] il a été proposé pour la première fois d'adapter les paramètres d'un HMM par régression. Afin de calculer

les paramètres de l'adaptation, on maximise donc la fonction auxiliaire $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}})$:

$$Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) = \sum_{m=1}^{M} \sum_{t=1}^{T} \gamma_m(t) \left[K_m + \log(|\hat{\boldsymbol{\Sigma}}_{\boldsymbol{m}}|) + (\boldsymbol{o}(t) - \hat{\boldsymbol{\mu}}_{\boldsymbol{m}})^T \hat{\boldsymbol{\Sigma}}_{\boldsymbol{m}}{}^{-1} (\boldsymbol{o}(t) - \hat{\boldsymbol{\mu}}_{\boldsymbol{m}}) \right] \tag{5.7}$$

$\gamma_m(t)$ est la probabilité d'être dans la gaussienne m lorsqu'on observe $\boldsymbol{o}(t)$ et K_m est une constante qui ne dépend que de la gaussienne m.

La fonction auxiliaire $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}})$ est optimisée par plusieurs itérations de l'algorithme EM, jusqu'à ce que la différence entre les paramètres d'un système adapté et ceux de l'itération précédente soit inférieure à un seuil ϵ faible prédéfini. A chaque itération de l'algorithme EM, $\gamma_m(t)$ est recalculé sur les nouvelles valeurs de $\hat{\boldsymbol{\mu}}$ et $\hat{\boldsymbol{\Sigma}}$ puis ceux-ci sont de nouveau optimisés. Les paramètres du modèle $\hat{\boldsymbol{\lambda}}$ final sont ainsi ceux qui maximisent la vraisemblance pour les données d'adaptation.

L'adaptation MLLR est aujourd'hui le type d'adaptation le plus utilisé pour les systèmes à base de HMMs dans le domaine de la parole, notamment grâce à la simplicité et l'efficacité de son approche : les paramètres (moyenne, variance) du modèle adapté sont calculés par une transformation linéaire des paramètres du modèle initial [114]. De plus, l'adaptation MLLR est réputée pour fonctionner avec un nombre réduit de données, ce qui est toujours avantageux pour un système de reconnaissance à utiliser en temps réel.

Il existe plusieurs types de transformation MLLR : la transformation peut être calculée indépendamment sur les moyennes et les matrices de covariance des gaussiennes du modèle (*adaptation avec biais*) mais il est aussi possible de forcer les moyennes et les covariances à partager le calcul de leur transformation (*adaptation contrainte*, Digalakis *et al.* [29], Gales [52]). Ces deux techniques sont présentées ci-dessous.

Adaptation classique avec biais : Dans sa forme la plus simple, l'adaptation MLLR [92] transforme $\boldsymbol{\mu_m}$ de la façon suivante :

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{\boldsymbol{m}} &= \boldsymbol{A}\boldsymbol{\mu}_{\boldsymbol{m}} + \boldsymbol{b} \\ &= \boldsymbol{W}\boldsymbol{\xi}_{\boldsymbol{m}} \end{aligned} \tag{5.8}$$

$\boldsymbol{W}$ est la matrice de transformation, de taille $n(n+1)$ (n est la taille du vecteur de caractéristiques) et $\boldsymbol{b}$ est un biais, qui se matérialise par un 1 dans la moyenne

augmentée $\boldsymbol{\xi_m}$:

$$\boldsymbol{\xi_m} = \begin{bmatrix} 1 \\ \mu_m^1 \\ \mu_m^2 \\ \dots \\ \mu_m^n \end{bmatrix}$$

En ce qui concerne la covariance (diagonale) $\boldsymbol{\Sigma_m}$, on calcule :

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{m}} = \boldsymbol{H}\boldsymbol{\Sigma_m}\boldsymbol{H^T} \tag{5.9}$$

où $\boldsymbol{H}$ est la matrice de transformation.

Notons qu'il est possible de choisir de n'adapter que les moyennes des gaussiennes du système HMM afin de réduire le nombre de calculs à effectuer. On suppose dans ce cas que les observations du scripteur sont décalées par rapport au modèle mais que leur allure est sensiblement identique.

Adaptation contrainte (CMLLR) : Dans le cas de l'adaptation CMLLR [29, 52], la moyenne et la variance partagent une même transformation, c'est-à-dire qu'on retrouve

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{m}} = \boldsymbol{A}\boldsymbol{\mu_m} + \boldsymbol{b} \tag{5.10}$$

mais que le calcul de $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{m}}$ change :

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{m}} = \boldsymbol{A}\boldsymbol{\Sigma_m}\boldsymbol{A^T} \tag{5.11}$$

Les preuves de l'existence de solution pour la maximisation de $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}})$ ont été largement développées et peuvent se trouver dans [92], ainsi que le déroulement précis des opérations mathématiques, dans Gales [51].

5.1.3 Les classes de régression

Dans la Section précédente (5.1.2), nous avons supposé que les moyennes (et/ou variances) des gaussiennes du modèle étaient transformées par une matrice de transformation unique. Etant donné le nombre de gaussiennes différentes définies dans notre système (pour la base IAM : 20 gaussiennes dans chaque mélange * 3,053 états = 61,060 gaussiennes), il paraît difficile de concevoir qu'une seule transforma-

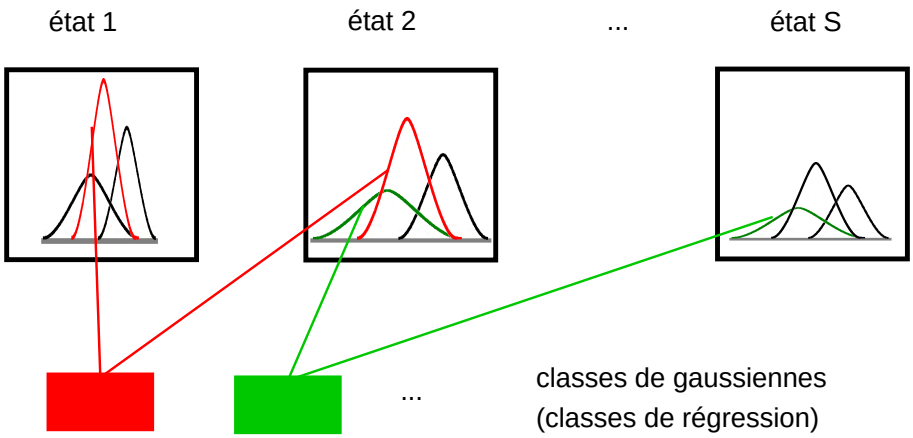


FIGURE 5.1 – Illustration des classes de régression. Les gaussiennes appartenant à des états différents sont regroupées.

tion peut correctement ajuster toutes les gaussiennes. C'est pourquoi des *classes de régression* ont été introduites, qui permettent de classer les gaussiennes du modèle dans différents groupes (R classes) et de calculer une matrice de transformation W_r pour chaque groupe $r \in [1..R]$ (Gales [50]).

Les gaussiennes sont regroupées indépendamment de leur appartenance à un état ou à un modèle de caractère. Ainsi deux gaussiennes d'un même mélange et donc d'un même état peuvent se retrouver dans deux classes différentes, comme illustré sur la Figure 5.1.

Les classes sont calculées sur les données d'apprentissage (indépendantes du scripteur) avec des *arbres de classe de régression* dont la profondeur est choisie à l'avance. Les arbres sont construits de manière à regrouper les gaussiennes les plus proches ensemble. Pour un noeud donné, on calcule ses enfants de la manière suivante :

- la moyenne et la variance globale de l'ensemble des gaussiennes contenues dans le noeud sont calculées,
- le dédoublement est effectué en créant deux nouvelles moyennes égales à la moyenne globale décalée d'une partie de la variance,
- la distance euclidienne de chaque gaussienne du noeud aux nouvelles moyennes est calculée et chaque gaussienne est associée à sa moyenne la plus proche,
- les noeuds enfants sont créés, chacun contenant les gaussiennes lui ayant été associées,
- la moyenne et la variance de l'ensemble des gaussiennes contenues dans chacun

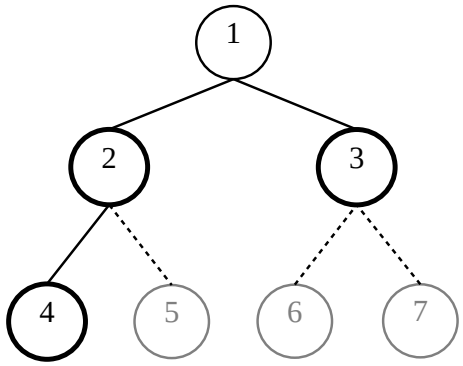


FIGURE 5.2 – Illustration d'un arbre de régression. La profondeur de l'arbre par défaut est 2 et le nombre final de classes est réduit à 3.

des noeuds sont calculées et ainsi de suite jusqu'à l'obtention de la profondeur souhaitée.

L'utilisation d'arbres pour le calcul des classes se justifie par le fait qu'en cas de manque de données du scripteur à adapter il est possible de regrouper des feuilles et des noeuds en remontant l'arbre et que ces regroupements conservent le principe utilisé pour séparer les classes. Ainsi le nombre réel de transformations W_r utilisées pour adapter un scripteur donné varie en fonction des données disponibles pour l'adaptation. Un exemple de réduction d'arbre est illustré sur la Figure 5.2. Pour chaque feuille de l'arbre, le nombre d'observations associées aux gaussiennes contenues dans le cluster est évalué. C'est-à-dire, pour le cluster r :

$$\Gamma_r = \sum_{t=1}^{T} \sum_{m \in \text{cluster}_r} \gamma_m(\boldsymbol{o}(t)). \tag{5.12}$$

($\gamma_m(t)$ est la probabilité d'être dans la gaussienne m lorsqu'on observe $\boldsymbol{o}(t)$). Si Γ_r est inférieur à un seuil η choisi, alors la feuille est rattachée à son parent et celui-ci devient un nouveau cluster. η est un seuil défini de manière unique pour tous les Γ_r et sa valeur optimale est calculée sur un ensemble de validation. La réduction est poursuivie jusqu'à ce que chaque classe définie par l'arbre de régression compte suffisamment de données. Dans l'exemple de la Figure 5.2, les classes 6 et 7 n'ont pas assez de données donc elles sont regroupées dans le noeud 3 et partagent la même matrice de transformation. En revanche, si la classe 4 a suffisamment de données, ce n'est pas le cas de la classe 5, qui est rattachée à son noeud parent, le noeud 2. Les classes finales définies sont donc : 2, 3 et 4.

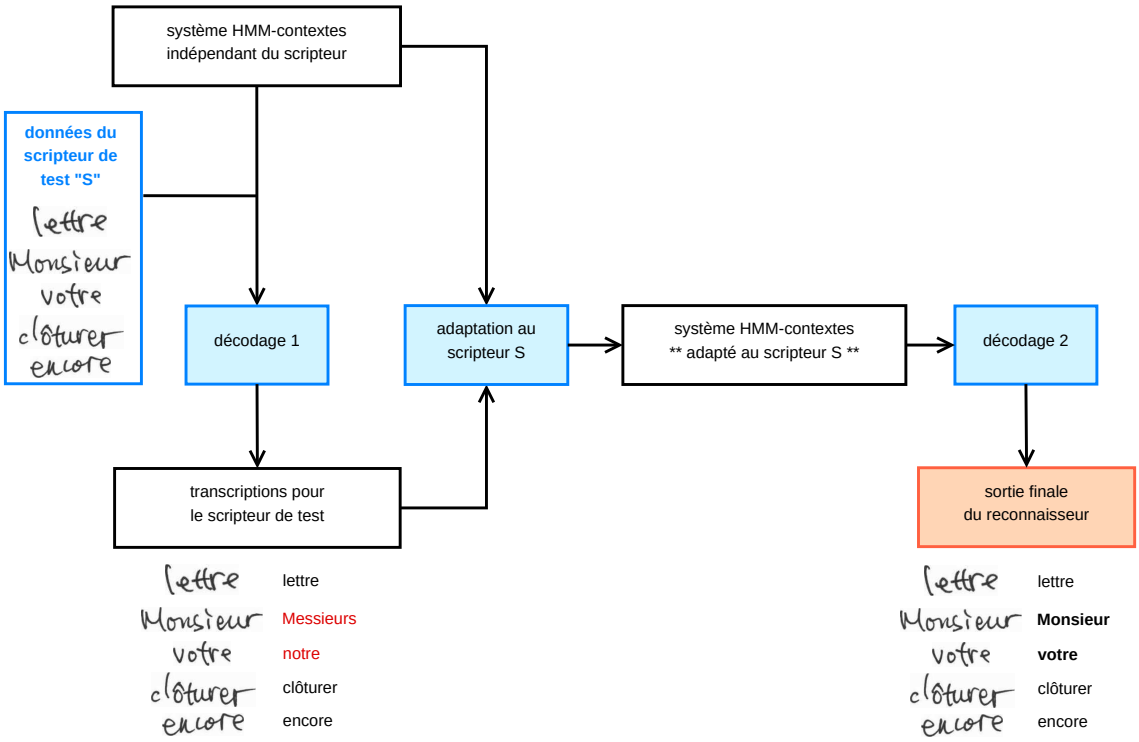


FIGURE 5.3 – Principe de l'adaptation non supervisée utilisée dans notre système.

5.2 Application à la reconnaissance de mots manuscrits : approche non supervisée

Comme nous l'avons précisé dans l'introduction de ce chapitre, nous utiliserons une adaptation non supervisée pour adapter nos modèles aux scripteurs de test. Au vu des données que nous utilisons, cette forme d'adaptation est la plus réaliste : chaque scripteur de test ayant un ensemble restreint de données lui correspondant, nous ne pouvons pas diviser cet ensemble en deux parties, l'une pour adapter et l'autre pour tester. De plus, il est aussi plus cohérent d'envisager que dans une application réelle nous ne disposerons pas d'annotation *vérité-terrain* des scripteurs testés.

Le principe d'adaptation que nous utilisons est schématisé sur la Figure 5.3. La partie la plus à gauche est la partie du système explicitée jusqu'ici : des modèles HMMs contextuels de caractères sont appris et un décodage est effectué sur des données de test, donnant une transcription avec plus ou moins d'erreurs. L'objectif d'adapter le système est donc, dans la mesure du possible, de corriger les éventuelles

erreurs. Pour cela, une fois que l'on dispose de transcriptions pour le scripteur de test à adapter, la transformation du modèle est calculée (case *adaptation au scripteur S*) pour obtenir les modèles modifiés (case *système HMM-contextuels adaptés au scripteur S*). Une fois que le système adapté est en place, les données du scripteur S sont *de nouveau* décodées et de nouvelles transcriptions sont obtenues. Ces dernières sont la sortie finale du système pour le scripteur S.

5.3 Expériences sur la base IAM

Dans cette section, nous évaluons les diverses techniques d'adaptation discutées en Section 5.1 sur la base de données de mots isolés IAM, présentée en Section 4.2. Nous n'avons pu effectuer d'adaptation au scripteur ni sur la base Rimes, étant donné que certains scripteurs de l'ensemble de test ont des données dans l'ensemble d'apprentissage (les résultats seraient faussés) ni sur la base OpenHart.

La base de test IAM comporte 13,750 images de mots, réparties sur 128 scripteurs. La répartition est montrée sur la Figure 5.4. On voit que la grande majorité des scripteurs a moins de 100 images leur correspondant, ce qui est peu pour adapter. Etant donné que l'adaptation MLLR est avantageuse lorsqu'il y a moins de données, nous avons commencé nos expériences avec ce type d'adaptation. Les résultats sont discutés dans la Section 5.3.1. Nous avons ensuite effectué une adaptation de type MAP pour chacun des scripteurs et les résultats obtenus sont discutés en Section 5.3.2. Un résumé de nos expériences et une discussion sur cette étude préliminaire sont ensuite proposés en Section 5.3.3

5.3.1 Adaptation MLLR sur IAM

Nous avons procédé à deux types d'adaptation MLLR [92], présentés dans la section précédente : l'adaptation classique avec biais et l'adaptation contrainte. Nous avons considéré l'adaptation classique dans un premier temps et avons procédé à deux expériences : une première expérience avec une matrice de transformation globale pour toutes les gaussiennes du système et une seconde avec des transformations calculées selon les classes de régression. Nous avons fixé à 64 le nombre maximal de classes de régression possible pour les gaussiennes (correspondant à une profondeur d'arbre égale à 5) et avons calculé l'arbre des classes de régression sur l'ensemble d'apprentissage (composé de 46901 images, cf Section 4.2). Afin de garantir un calcul

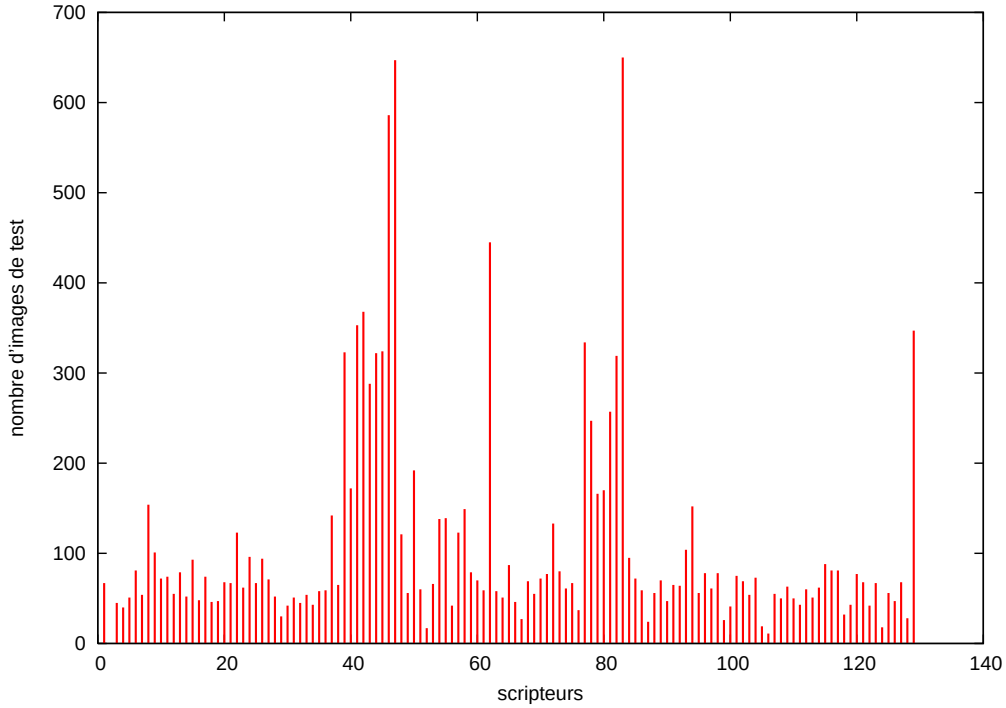


FIGURE 5.4 – Répartition des données de test sur les 128 scripteurs de test (en nombre d'images de mots par scripteur).

robuste pour la matrice de transformation de chaque classe, nous avons fixé le seuil η à $\eta = 1,000$. Dans les deux cas (transformation globale ou transformation par classe de régression), la transformation MLLR n'est calculée que pour les moyennes des gaussiennes : pour cette première expérience, nous avons choisi l'adaptation dans sa forme la plus simple.

D'après nos expériences, l'adaptation MLLR des moyennes (dans le cadre d'une adaptation non supervisée au scripteur) dégrade les performances du reconnaisseur. Le taux de reconnaissance initial du système à base de HMMs en contexte sur la base de test IAM est de 69.5% (cf Table 4.11) alors qu'une adaptation MLLR globale des moyennes donne un taux de reconnaissance de 28.2%. L'utilisation de classes de régression améliore l'adaptation : le taux de mots bien reconnus passe de 28.2% à 50.0%. Mais ce taux reste très inférieur au taux initial.

Nous avons donc dans un second temps cherché à modifier à la fois la moyenne et la variance des gaussiennes du système. Pour cela, nous avons utilisé l'adaptation de type CMLLR (adaptation contrainte) avec classes de régression. Les classes sont les mêmes 64 classes que celles utilisées pour l'adaptation MLLR classique (elles sont

calculées sur les données d'apprentissage).

Etant donné que nous utilisons des caractéristiques dynamiques (calcul d'une régression sur les caractéristiques), il existe deux possibilités pour le calcul de la matrice de transformation $\boldsymbol{A}$, partagée par les moyennes et variances d'une même classe de régression (cf. Equations 5.10 et 5.11). Soit la matrice $\boldsymbol{A}$ est complète, soit c'est une matrice diagonale par blocs, divisée de façon à séparer la transformation des dimensions originales de celle des dimensions issues de la régression sur les caractéristiques. C'est-à-dire que $\boldsymbol{A}$ est de la forme :

$$\boldsymbol{A} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} & & & \\ \vdots & \ddots & \vdots & & \mathbf{0} & \\ a_{n,1} & \cdots & a_{n,n} & & & \\ & & & a_{n+1,n+1} & \cdots & a_{n+1,2n} \\ & \mathbf{0} & & \vdots & \ddots & \vdots \\ & & & a_{2n,n+1} & \cdots & a_{2n,2n} \end{pmatrix} \tag{5.13}$$

où n est la dimension des vecteurs de caractéristiques avant le calcul des caractéristiques dynamiques ($n = 33$ pour la base IAM).

Les résultats de l'adaptation CMLLR non supervisée sur la base de test IAM sont encourageants. Sur 128 scripteurs, 125 ont pu être adaptés avec la méthode CMLLR : les données des 3 autres scripteurs n'étant pas en nombre suffisant, leur nombre de classes de régression a été réduit à zéro et donc aucune matrice d'adaptation n'a été calculée. En ce qui concerne l'adaptation CMLLR avec matrice de transformation $\boldsymbol{A}$ complète, 85 scripteurs sur 125 ont été adaptés mais sans changement dans leur taux de reconnaissance, 13 ont vu leurs performances se dégrader avec l'adaptation et 27 ont connu une amélioration grâce à l'adaptation. Globalement, le taux de mots bien reconnus pour cette adaptation est de 69.62%, soit une réduction relative du nombre d'erreurs de 0.3%. L'adaptation CMLLR avec matrice de transformation par blocs semble améliorer les performances globales du système : sur les 125 scripteurs, 73 scripteurs ont été adaptés sans changement sur leur sortie, 14 ont vu leurs résultats se dégrader et 38 ont connu une amélioration, donnant un taux de reconnaissance final de 69.93%. Cela représente une réduction d'erreur en relatif de 1.3%.

En conclusion, il semble qu'il est plus avantageux d'effectuer une transformation sur les moyennes et les variances des gaussiennes (adaptation CMLLR) plutôt que sur les moyennes uniquement et que les matrices de transformation diagonales par

bloc donnent de meilleurs résultats que les matrices complètes.

5.3.2 Adaptation MAP sur IAM

Nous avons dans un deuxième temps évalué l'influence d'une adaptation de type MAP [55] sur les scripteurs de test de la base IAM. Comme nous nous y attendions, l'adaptation MAP a très peu influé sur les performances du système : sur les 128 scripteurs de test, seuls 4 ont vu leurs résultats se modifier. Chacun de ces quatre scripteurs possède plus de 300 images de mots. Parmi ces 4 scripteurs, 3 ont vu une amélioration de performance (gain d'une image de mot bien reconnue pour chacun) et 1 a eu un mot en moins bien reconnu. La différence globale sur l'ensemble des $13,750$ images de test des 128 scripteurs est donc de $3 - 1 = 2$ images de mots mieux reconnus. Clairement, ce résultat n'est pas statistiquement significatif (même s'il montre une légère amélioration du système).

5.3.3 Discussion

La Table 5.1 résume nos expériences d'adaptation non supervisée au scripteur sur la base de test IAM. Pour chaque type d'adaptation proposé, nous indiquons le nombre de scripteurs effectivement adaptés, la performance du système adapté par rapport au système initial (nombre de scripteurs pour lequel l'adaptation améliore, détériore ou ne change rien aux résultats), ainsi que le taux de reconnaissance du système adapté sur l'ensemble des scripteurs de test.

On voit clairement sur la Table 5.1 que l'adaptation au scripteur semble être une technique efficace d'amélioration d'un système de reconnaissance à condition de correctement choisir sa méthode. Dans notre cas, le faible nombre de données par scripteur de test élimine l'utilisation de l'adaptation MAP, qui ne change (presque) rien au système. On voit aussi que l'adaptation MLLR des moyennes détériore les performances.

L'adaptation CMLLR quant à elle améliore les résultats. Ainsi la méthode CMLLR avec matrice de transformation diagonale par blocs permet de réduire le taux global d'erreurs du système de 1.3% relativement, ce qui est intéressant.

Les résultats que nous obtenons sont cohérents avec d'autres systèmes de reconnaissance d'écriture manuscrite utilisant une adaptation non supervisée au scripteur. Par exemple Rodríguez-Serrano *et al.* [144] obtiennent une amélioration maximale de 1.2% en utilisant l'adaptation de type MAP pour la détection de mot-clé dans un

type d'adaptation	nb. de scripteurs adaptés	système adapté = initial	système adapté > initial	système adapté < initial	% reconnaissance global
pas d'adaptation	-	-	-	-	69.53%
MLLR sans régression	128/128	0	0	128	28.20%
MLLR avec régression	128/128	0	0	128	50.02%
CMLLR avec régression	125/128	88/125	27/125	13/125	69.62%
CMLLR avec régression diag. par blocs	125/128	76/125	38/125	14/125	**69.93%**
MAP	128/128	124	3	1	69.54%

TABLE 5.1 – Comparaison des techniques d'adaptation MLLR, CMLLR et MAP sur la base de test IAM pour une adaptation au scripteur non supervisée

document (l'adaptation MLLR améliore son système de moins de 1%). De même, Ait-Mohand *et al.* [1] notent une amélioration de moins de 2% avec l'utilisation d'une adaptation à la police de caractère dans une tâche de reconnaissance de document imprimé. Ces deux systèmes ayant un lexique de test de moins de 100 mots pour leurs expériences (soit 100 fois moins que le lexique utilisé pour le test de la base IAM) et un large nombre de données pour l'adaptation, les améliorations qu'ils ont obtenues sont comparables aux nôtres.

Comme nous l'avons indiqué dans l'introduction de ce chapitre, il existe d'autres techniques d'adaptation. Ainsi, Nosary *et al.* [122] utilisent une adaptation itérative basée sur les caractéristiques graphiques des scripteurs (invariants d'un scripteur). Sur un ensemble de 15 scripteurs de test, ils obtiennent une amélioration globale modeste (sic) de leurs performances. Dans [49], Fink et Plötz proposent une estimation de style d'écriture pour améliorer leurs performances sur la base de lignes de IAM. Ils montrent que leur système donne une amélioration d'environ 2%, mais ces résultats restent comparables à ceux obtenus avec une adaptation MAP ou MLLR.

En considérant les données dont nous disposons, nous pensons qu'il serait intéressant de poursuivre notre travail sur l'adaptation au scripteur avec l'adaptation de type CMLLR. Celle-ci est en effet largement utilisée en parole (adaptation ou identification du locuteur) et son utilisation est même qualifiée de standard dans la

plupart des systèmes à base de HMMs. Ces dernières années, plusieurs améliorations ont d'ailleurs été proposées pour l'adaptation au locuteur par CMLLR. Leur but est de faire fonctionner l'adaptation pour un nombre très faible de données (moins de 5 secondes de parole, soit une vingtaine de mots). Ainsi l'utilisation de matrices d'adaptation diagonales par blocs, utilisée dans nos expériences, a été proposée par Gales [52].

Dans [34, 32], Dreuw *et al.* proposent d'utiliser le score de reconnaissance de leur système indépendant du scripteur afin de déterminer quelles données utiliser pour l'adaptation. Ils alignent les transcriptions obtenues par ce système aux données et ne conservent que les mots (et, dans une deuxième étape, que les observations) dont le score est supérieur à un seuil choisi. Ces données sélectionnées sont ensuite utilisées pour le calcul des matrices de transformation. Ceci leur permet de réduire de plus de 5% leur taux d'erreur sur la base IFN-Enit. Nous avons expérimenté cette proposition sur la base de test IAM, avec l'adaptation de type CMLLR dont les transformations sont diagonales par blocs. Notre sélection de données s'effectue au niveau mot. Les résultats de ces expériences sont donnés dans la Table 5.2. Les scores de reconnaissance ont été normalisés entre zéro et un. Les seuils choisis sont limités entre 0.25 et 0.5 : le seuil minimal permet de voir une différence avec une adaptation CMLLR sur toutes les données et le seuil maximal laisse suffisamment de données pour adapter les scripteurs les plus prolifiques. De même que dans la Table 5.1, nous indiquons pour chaque seuil le nombre de scripteurs effectivement adaptés (certains ont trop peu de données sélectionnées pour l'adaptation) et les performances du système adapté par rapport au système initial, indépendant du scripteur. La troisième colonne de la Table permet de voir, sur les 13750 images initiales, combien ont été conservées pour l'adaptation de tous les scripteurs de test en fonction du seuil.

Nous voyons sur la Table 5.2 que la stratégie qui consiste à choisir les données pour l'adaptation permet de garantir que l'adaptation CMLLR non supervisée donne un taux de reconnaissance global au moins égal au taux du système initial. Au mieux, le gain sur l'ensemble des scripteurs (en comptant ceux n'ayant pas été adaptés) est de 0.3%. Ce gain est pourtant inférieur au gain obtenu en utilisant toutes les données (0.4%). Une prochaine étape serait alors de sélectionner les données à adapter au niveau des observations directement, au lieu de sélectionner des images de mots entiers, comme le proposent Dreuw *et al.* [34, 32]. Cependant, quelle que soit la stratégie de sélection, cette méthode implique une diminution du nombre de données

seuil sur le score de reco. init.	nb. de scripteurs adaptés	nb. de mots utilisés pour adapter	système adapté = initial	système adapté > initial	système adapté < initial	% reco. global
0.25	118/128	11260	59/118	35/118	24/118	69.83%
0.3	103/128	9082	46/103	35/103	22/103	69.73%
0.35	75/128	7061	40/75	23/75	12/75	69.70%
0.4	56/128	5390	26/56	16/56	14/56	69.54%
0.45	42/128	4105	16/42	13/42	13/42	69.53%
0.5	28/128	3159	5/28	15/28	8/28	69.73%

TABLE 5.2 – Influence du seuil sur le score de reconnaissance de la sortie du système indépendant du scripteur pour le choix des données d'adaptation sur les performances de l'adaptation CMLLR non supervisée.

disponibles pour l'adaptation. D'autres techniques doivent alors être appliquées en plus afin de répondre à ce problème

Dans Lei *et al.* [93], Huang *et al.* [75] il est proposé d'utiliser une méthode Bayesienne (maximum a posteriori) pour donner une valeur a priori aux matrices de transformation (méthode fMAPLR). Si la méthode fMAPLR a été largement utilisée dans le domaine de la reconnaissance de la parole ces dernières années, une nouvelle méthode nommée CMLLR-*basis* et proposée par Visweswariah *et al.* [163, 162] et Povey et Yao [136] l'a désormais devancée en devenant à ce jour la méthode la plus performante pour l'adaptation au locuteur. Cette stratégie consiste à calculer à l'avance des matrices de transformation CMLLR *de base* (en anglais *basis*) telles que l'adaptation d'un scripteur de test se servira de combinaisons linéaires de ces matrices pour calculer ses transformations. La méthode CMLLR-*basis* permet non seulement un gain en rapidité d'adaptation mais aussi des performances compétitives pour l'adaptation au locuteur avec un nombre de données très limité. Dans nos futurs travaux, nous prévoyons d'inclure les CMLLR-*basis* dans notre stratégie d'adaptation au scripteur.

Conclusion du chapitre 5

Nous avons présenté dans ce chapitre une étude pour l'adaptation d'un système de reconnaissance de mots manuscrits à un scripteur spécifique. Nous avons présenté et expérimenté les deux techniques les plus populaires d'adaptation, qui sont l'adaptation par maximum de vraisemblance (MLLR) et l'adaptation par maximum

a posteriori (MAP). Pour nos expériences, nous avons choisi une approche non supervisée de l'adaptation au scripteur car cela concorde avec les données que nous possédons et car cette approche est réaliste. Nous nous sommes donc servis des transcriptions des données des scripteurs de test fournies par le décodage du système à base de HMMs en contexte pour apprendre les transformations de nos modèles.

Les résultats que nous avons obtenus sur la base de test IAM sont encourageants. Si l'adaptation MAP ne change pas (ou très peu) les résultats et si l'adaptation MLLR des moyennes détériore les performances du système, l'adaptation de type CMLLR permet un gain relatif de 1.3% de mots bien reconnus.

Cette amélioration de performances est intéressante, pourtant elle reste limitée. Cela peut s'expliquer par plusieurs raisons. D'abord, le système initial, à base de modèles en contexte, génère des transcriptions contenant environ 30% d'erreurs. Etant donné que ces transcriptions sont utilisées pour l'adaptation, on peut supposer que les erreurs empêchent l'algorithme de donner les meilleurs résultats possible. Ensuite, le nombre de données disponibles par scripteur est faible ; il ne permet pas un calcul robuste de transformations des modèles, surtout avec les transformations classiques que nous utilisons.

Afin de répondre à ces problèmes, nous pouvons dans un premier temps *choisir* les données d'adaptation pour chaque scripteur, avec un critère prédéfini. Ainsi Dreuw *et al.* [34, 32] appliquent un seuil sur le score de reconnaissance de chaque mot (voire chaque état) de leur système indépendant du scripteur. Ce seuil permet de discerner les données à utiliser pour l'adaptation non supervisée en choisissant celles au score de confiance le plus haut. Ceci leur permet de réduire de plus de 5% leur taux d'erreur. Cette méthode pourtant implique une diminution du nombre de données disponibles pour l'adaptation. D'autres techniques doivent alors être appliquées en plus afin de répondre à ce deuxième problème. Dernièrement, des techniques utilisant des matrices d'adaptation CMLLR prédéfinies ont prouvé leur efficacité pour l'adaptation au locuteur avec peu de données (Visweswariah *et al.* [163, 162], Povey et Yao [136]). Ces deux axes de recherche constituent notre travail à venir sur l'adaptation scripteur.

Conclusion

Nous avons présenté dans ce document notre travail sur la reconnaissance de mots manuscrits cursifs à base de modèles de Markov cachés et d'une modélisation originale de caractères en fonction de leur contexte. Nous avons montré que cette modélisation peut se généraliser à plusieurs types de langages cursifs et s'adapter aux données de développement.

Pour construire notre système, nous nous sommes basés sur les HMMs. Largement utilisés pour la reconnaissance de la parole depuis les années 1980 [139], ils sont aujourd'hui une méthode incontournable pour la reconnaissance hors-ligne de l'écriture. Cet attrait s'explique par plusieurs raisons, développées dans le premier chapitre de ce manuscrit. D'abord, l'utilisation de modèles de Markov permet la prise en compte de l'aspect séquentiel des données. Pour cela, les images de mots en entrée du système sont transformées en séquence de vecteurs de caractéristiques. Les caractéristiques extraites peuvent être de haut niveau (par exemple le nombre de caractères contenus dans l'image, sa hauteur ou son ratio largeur/hauteur) ou de bas niveau (par exemple le nombre de pixels d'écriture) et l'extraction peut s'effectuer de manière holistique ou alors à la suite d'une segmentation de l'image. Pour notre extraction de caractéristiques, nous avons choisi une approche sans segmentation de l'image, qui parcourt celle-ci dans le sens de lecture avec des fenêtres glissantes au lieu de la découper explicitement en caractères ou sous-parties de caractères. L'approche par fenêtres glissantes permet d'éviter les problèmes de segmentation et le balayage séquentiel de l'image qu'elle implique est parfaitement adapté à l'utilisation de HMMs. L'approche séquentielle est d'autant plus intéressante pour la reconnaissance d'écriture manuscrite que, de par leurs propriétés, les HMMs permettent d'absorber les disparités entre les données. Ainsi, les HMMs de type Bakis utilisés dans nos travaux autorisent depuis un état donné de boucler sur l'état, de passer à l'état suivant ou encore de sauter un état. Ceci nous permet de gérer la géométrie variable des caractères présents dans les ensembles d'apprentissage et de test.

Enfin, des algorithmes performants pour l'apprentissage et le décodage des HMMs ont été développés et utilisés avec succès au fil des années. Les HMMs sont donc une approche à la fois simple et robuste pour la reconnaissance d'écriture manuscrite. Notre approche vise à modéliser les caractères d'un alphabet avec des HMMs. Elle diffère d'une approche holistique dans le sens où elle peut permettre la construction et le décodage de n'importe quel mot, même non appris, à partir du moment où il utilise les caractères de l'aphabet modélisé.

Nous avons construit dans un premier temps un système générique à base de HMMs où un caractère est modélisé par un HMM et avons exploré quelques pistes pour le rendre le plus robuste possible. Nous avons dans un premier temps travaillé sur les caractéristiques. Comme nous l'avons présenté en Section 1.2, dans la littérature beaucoup de caractéritiques différentes existent (à vrai dire il en existe autant que de systèmes HMMs) et une uniformisation telle qu'elle existe en parole avec les MFCCs ne semble pas être à l'ordre du jour. Nous avons donc fait en sorte de ne pas, nous aussi, ajouter un nouvel ensemble de caractéristiques à la littérature mais plutôt d'exploiter au mieux l'existant dans notre système [1]. A cette fin, nous avons distingué trois courants principaux parmi les caractéristiques bas niveau existantes : les caractéristiques statistiques, les caractéristiques géométriques et les caractéristiques directionnelles (dérivées des SIFT [99]). Nous proposons dans notre système la combinaison de ces trois types de caractéristiques (voir Chapitre 2, Section 2.1.1). Cette combinaison permet de prendre en compte toutes les spécificités des caractères contenus dans l'image et d'extraire le maximum d'informations dans la fenêtre courante.

Une seconde amélioration a été proposée au niveau des caractéristiques : l'ajout d'une régression du premier ordre (voir Chapitre 2, Section 2.1.2). Dans ce cas, on cherche à récupérer des informations sur les fenêtres avoisinantes. La régression permet de représenter la dynamique des données et la corrélation existante entre les pixels de la fenêtre courante et ceux des fenêtres voisines. L'ajout d'une régression permet par exemple de gagner 5.2% de mots bien reconnus sur la base Rimes par rapport à un système sans régression.

Nous avons proposé une troisième et dernière amélioration pour notre système générique à base de HMMs : la construction de modèles HMMs dont la longueur varie selon le caractère à modéliser. Cette approche, utilisée dans d'autres systèmes de l'état de l'art, permet de gagner en performance et surtout de construire des modèles plus robustes. Elle est surtout utile en arabe, où la longueur des caractères

1. http://xkcd.com/927/

de l'alphabet est très variable (le caractère أ isolé et la forme centrale de ن sont courts alors que les formes finales de س et ض peuvent être très longues) et où les ligatures entre caractères font varier fortement leur longueur mais doivent être absorbées par les modèles. Cette modélisation variable permet de gagner 4% de mots bien reconnus sur une base de test de la base Openhart par rapport à un système générique où tous les caractères sont modélisés par des HMMs à topologie identique.

En résumé, nous avons élaboré dans la première partie de ce manuscrit un système générique de reconnaissance de mots manuscrits à base de HMMs de caractères et l'avons optimisé sur plusieurs aspects (caractéristiques extraites et topologie des modèles). Ce système obtient des performances comparables à l'état de l'art (voir Chapitre 4, Sections 4.1.5, 4.1.7, 4.1.8, 4.2 et 4.3).

Nous avons par la suite souhaité poursuivre nos recherches avec une stratégie de calcul de modèles de plus en plus précis. En observant les différentes formes qu'un caractère peut prendre en fonction de sa place dans un mot ou des caractères qui l'entourent, nous avons observé que les variations pouvaient être très grandes même au sein d'un ensemble de mots écrits par un même scripteur. Par exemple, il est aisément observable que le commencement du caractère u est différent selon qu'il est précédé d'un v ou d'un l. Nous avons alors proposé une modélisation inédite à notre connaissance dans les systèmes de reconnaissance d'écriture manuscrite : la modélisation d'un caractère en fonction de son contexte. Cela se traduit par l'apprentissage de modèles de caractères dépendants des caractères le précédant et le suivant : les trigraphes. Cette modélisation, présentée au Chapitre 3, est au coeur de nos travaux.

L'apprentissage de modèles de trigraphes se traduit par un très grand nombre de paramètres à calculer : pour chaque caractère, toutes les combinaisons de contextes présentes dans l'ensemble d'apprentissage doivent être modélisées. Or, certains trigraphes n'ont que très peu d'exemples et leur apprentissage ne pourrait pas s'effectuer correctement. Nous avons alors proposé d'effectuer un partage de paramètres HMMs afin de réduire le nombre final de paramètres à calculer. Il est possible de

partager les paramètres à plusieurs niveaux : au niveau des gaussiennes (HMMs semi-continus), au niveau des modèles (partage de modèles entiers) ou, à mi-chemin, au niveau des états. C'est ce dernier type de partage que nous avons choisi. Nous procédons à un regroupement d'états tel que les états d'une même position dans le HMM et correspondant à un même caractère central sont regroupés dans un ou plusieurs clusters mais séparés des autres états.

Afin de calculer nos clusters, nous avons proposé d'utiliser des arbres binaires de décision. Ceux-ci sont basés sur des questions originales portant sur la morphologie des caractères. Un arbre est défini pour chaque position d'état de chaque HMM de caractère présent dans le système générique (sans modélisation contextuelle). Les feuilles d'un arbre définissent les clusters pour une position d'état d'un caractère central donné. Les arbres sont calculés sur les données d'apprentissage et sont contrôlés par deux paramètres (le nombre minimal de données pour calculer le modèle d'état d'un cluster et l'augmentation minimale de vraisemblance entre un cluster et lui-même divisé en deux sous-clusters), assurant l'élaboration de clusters d'états robustes et correctement modélisés. Nous avons choisi les arbres de décision pour notre clustering car ceux-ci permettent l'utilisation de lexiques indépendants du lexique d'apprentissage lors de la phase de test, contrairement à un clustering basé sur les données. Pour un nouveau trigraphe (non appris), une simple descente d'arbre pour chacun de ses états permet de lui allouer des clusters et donc de créer un modèle HMM pour le trigraphe.

Nous avons évalué notre approche à base de HMMs contextuels sur trois bases de données différentes. Une base en français (Rimes [4, 70]), une en anglais (IAM [110]) et une base de mots manuscrits arabes (OpenHart [74]). Chacune de ces bases a ses propres spécificités. La base Rimes et la base IAM ont un nombre de données d'apprentissage et de test comparables (60000 / 10000 mots environ) mais leurs dictionnaires diffèrent (le lexique de test de IAM est 2 à 5 fois plus grand que celui de Rimes). La base OpenHart quant à elle propose un très grand nombre de données (plusieurs centaines de milliers d'images de mots). Nous avons restreint son vocabulaire de test à 20000 mots.

Pour chacune de ces bases de données, nous avons d'abord élaboré un système HMM générique optimal afin d'obtenir l'extraction de caractéristiques et la topologie des modèles les mieux adaptées. Celles-ci étant fixées pour chaque base, elles sont ensuite utilisées pour la construction du système avec HMMs contextuels. Au coeur de l'élaboration des HMMs dépendants du contexte se trouve le calcul des arbres

binaires pour la répartition des états des trigraphes dans les clusters. Ce calcul est optimisé sur une base de validation. Les arbres sont alors fixés pour chaque base de données et, par la suite, ni l'ajout de données supplémentaires pour l'apprentissage des modèles ni le changement de lexique lors de la phase de test ne les modifient. Au contraire, les clusters calculés par les arbres sont la base qui permet de procéder à ces changements sans dégrader les performances du système.

Nous avons ainsi vu au cours d'expériences développées au Chapitre 4 que la modélisation de caractères en fonction de leur contexte permet d'obtenir des résultats nettement supérieurs à des HMMs génériques. Ainsi sur la base de test de Rimes 2011, la modélisation de trigraphes permet de réduire de 15.4% relativement le nombre de mots mal reconnus. Sur la base IAM, la réduction relative est de 5.2% et, sur l'ensemble de test DevTest1 de la base OpenHart, 9.7% de mots bien reconnus sont gagnés relativement par rapport à un système générique à base de HMMs de caractères indépendants de leur contexte.

La comparaison de notre système à l'état de l'art des systèmes HMMs nous montre qu'il est le meilleur système présenté seul et qu'il est compétitif face à d'autres systèmes issus de combinaisons de classifieurs. Afin d'ailleurs de nous comparer à ces combinaisons nous avons proposé notre propre combinaison pour la compétition 2009 de reconnaissance de mots sur la base Rimes. Notre combinaison allie les sorties des deux systèmes HMMs présentés danc ce manuscrit (un générique et un contextuel) et un système hybride HMM/NN développé par A2iA. En 2009, cette combinaison a permis à notre système de devenir le meilleur système à base de HMMs [64, 80, 12]. Il reste cependant en deuxième place derrière le système à base de réseaux de neurones récurrents (RNNs) proposé par l'université de Munich [62, 61]. Il se présente alors à nous deux possibilités. Une première consiste à poursuivre la combinaison de systèmes en intégrant les RNNs à notre système combiné. La combinaison de systèmes est en effet à ce jour le moyen le plus sûr d'obtenir les meilleurs résultats possibles pour une tâche de reconnaissance. Appliquée astucieusement, la combinaison de sorties de classifieurs garantit l'augmentation des performances par rapport à n'importe quel système présenté seul [37]. Nous avons considéré cette option pour notre participation à la compétition Rimes 2011 de reconnaissance de mots. Grâce à cela, nous avons atteint un taux exceptionnel de 94.4% de mots bien reconnus sur la base de test avec un vocabulaire de plus de 5000 mots. L'autre scénario possible consiste à chercher à améliorer notre système à base de HMMs afin que, seul, il gagne en performance et soit comparable aux RNNs entre autres.

De fait, plusieurs améliorations sont possibles pour les systèmes HMMs de reconnaissance d'écriture. Un état de l'art des dernières avancées dans la recherche sur la reconnaissance de la parole avec des systèmes HMMs permet d'apercevoir la plupart de ces améliorations. (La reconnaissance d'écriture est en effet en retard par rapport à la parole dans l'utilisation efficace des HMMs.) Une de ces améliorations est par exemple l'adaptation des modèles HMMs au scripteur. Celle-ci a été introduite dans le dernier chapitre de notre manuscrit (Chapitre 5). L'étude effectuée dans ce chapitre a permis de constater que des améliorations étaient possibles même dans un contexte difficile (adaptation non supervisée avec un nombre de données restreint et à partir d'un système ayant plus de 30% d'erreurs). Les expériences effectuées sur la base IAM ont permis de diminuer relativement le taux d'erreurs de 1.2% avec une adaptation de type CMLLR alors que le nombre de données pour chaque scripteur de test est faible. L'utilisation de techniques d'adaptation pour des tâches de reconnaissance d'écriture sur des archives (principe de scripteur unique pour un grand nombre de données) pourrait ainsi permettre d'obtenir un système très performant. Ce premier type d'amélioration est l'une des pistes de recherche que nous suivons dans nos travaux actuels. D'ailleurs, l'utilisation massive de techniques d'adaptation dans le monde de la parole nous rend optimiste quant à l'avenir de ces méthodes pour la reconnaissance d'écriture manuscrite. Dernièrement, des méthodes d'adaptation au locuteur ont atteint des performances inégalées avec pourtant des données en faibles quantités et bruitées [93, 75, 163, 162, 136]. L'utilisation de ces méthodes dans nos systèmes et l'élaboration de méthodes spécifiques à l'écriture est un axe d'amélioration prometteur.

L'observation des autres techniques de l'état de l'art et l'analyse de leurs avantages ont toujours été un moyen judicieux pour comprendre comment améliorer un système. Ainsi, étant donné leurs très bons résultats sur la base Rimes, il serait judicieux d'observer de plus près le fonctionnement des RNNs afin de comprendre leur réussite [62, 61]. En effet, si le réseau LSTM permet la modélisation de séquences pour les réseaux de neurones, leur utilisation n'est pas la seule explication des performances obtenues. Par exemple, l'extraction des caractéristiques utilisée dans le système à base de RNNs est différente de la nôtre. Ils n'utilisent pas une fenêtre glissante mais parcourent l'image selon quatre directions différentes (les quatre diagonales) et utilisent des perceptrons multi-couches pour leur extraction. Cela leur permet entre autres de ne pas avoir de paramètres à optimiser pour chaque base de données et d'automatiser l'extraction de caractéristiques. Ces multiples parcours

leur permettent aussi d'utiliser des réseaux LSTM bidirectionnels. Cela pourrait être traduit par des HMMs en deux dimensions ou bien par une adaptation de l'algorithme d'apprentissage des modèles à un parcours multidirectionnel dans les états des HMMs.

Enfin, une dernière possibilité pour améliorer notre système HMM réside dans l'utilisation de modèles de langage pour la reconnaissance de lignes et de paragraphes. Quels qu'ils soient, les classifieurs ont tous des performances très bonnes en 2, 5 ou 10-best. L'utilisation de modèles de langage permet de faire remonter à la première place des mots initialement mal reconnus et de diminuer drastiquement le nombre d'erreurs. Nous avons présenté brièvement l'intérêt des modèles de langage pour la reconnaissance de lignes manuscrites arabes sur la base Openhart dans le Chapitre 4, Section 4.3.5. Pour nos expériences nous avons utilisé les très répandus n-grams [156] et ceux-ci nous ont permis de gagner relativement jusqu'à 39% de mots bien reconnus sur cette tâche difficile. Le calcul que nous avons effectué pour élaborer notre modèle de langage se base sur des comptages d'occurrences d'expressions dans une base de données. Cette méthode très simple permet donc de gagner en performance. Mais elle présente de nombreux problèmes dus à la rareté (en anglais, *sparsity*) de certains n-grammes. Récemment, de nouveaux types de calculs de probabilité qui résolvent ce problème de sparsité ont été proposés pour l'élaboration de modèles de langage : les NN-LM (modèles de langage à base de réseaux de neurones [10, 149]) et les modèles M (modèles de langage exponentiels basés sur des classes de mots [23, 17]). Ils ont été utilisés avec succès dans le traitement de la parole [42, 148, 24, 129] et nous projetons avec enthousiasme de les utiliser pour nos futurs travaux de reconnaissance de lignes et de paragraphes manuscrits.

Il existe donc encore beaucoup de possibilités pour rendre les systèmes HMMs de reconnaissance d'écriture plus performants. Dans notre manuscrit, nous avons proposé une amélioration originale pour la reconnaissance de mots manuscrits : la modélisation de caractères en fonction de leur contexte (les trigraphes). Nous envisageons pour la suite de nos recherches d'explorer d'autres pistes, telle l'adaptation au scripteur introduite dans le dernier chapitre de ce manuscrit ou encore l'utilisation de modèles de langages pour la reconnaissance de lignes et de paragraphes. Ces améliorations permettront d'obtenir un système robuste dans le cadre du traitement et de la classification de documents.

Liste de Publications

Les travaux effectués durant cette thèse ont permis l'élaboration d'articles de journaux [a,b] et de conférence [c,d,e,f] ainsi qu'un poster [g]. Deux articles de conférence supplémentaires ont été écrits avec la société A2iA [h,i]. Leur contenu scientifique est externe à cette thèse.

Le système original de reconnaissance d'écriture manuscrite à base de HMMs contextuels a été soumis à plusieurs compétitions de reconnaissance de mots et sa combinaison avec d'autres systèmes à base de HMMs a permis d'obtenir de bons résultats :

- seconde place pour la compétition ICDAR 2009 de reconnaissance de mots manuscrits français [12, 64],
- seconde place pour la compétition ICDAR 2009 de reconnaissance de mots manuscrits arabes [38],
- première place pour la compétition OpenHart 2010 de reconnaissance de mots manuscrits arabes [74],
- première place pour la compétition ICDAR 2011 de reconnaissance de mots manuscrits français [65],
- première place pour la compétition ICDAR 2011 de reconnaissance de lignes manuscrites françaises [65].

[a] A-L. Bianne-Bernard, F. Menasri, R. Al-Hajj Mohamad, C. Mokbel, C. Kermorvant and L. Likforman-Sulem. Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10) : 2066 – 2080, 2011.

[b] A-L. Bianne-Bernard, C. Kermorvant, L. Likforman-Sulem and C. Mokbel. Modélisation de HMMs en contexte avec des arbres de décision pour la reconnaissance de mots manuscrits. *Document Numérique*, 14(2) :29 – 52, 2011.

[c] A-L. Bianne-Bernard, F. Menasri, L. Likforman-Sulem, C. Mokbel and C. Kermorvant. Variable length and context-dependent HMM letter form models for Arabic handwritten word recognition. In *Proccedings of the 19th Document Recognition and Retrieval Conference, part of the IS&T-SPIE Electronic Imaging Symposium - DRR2012*, vol. 8297 : *pages to appear*, 2012.

[d] C. Kermorvant, F. Menasri, A-L. Bianne, R. Al-Hajj Mohamad, C. Mokbel and L. Likforman-Sulem. The A2iA-Télécom ParisTech-UOB system for the ICDAR 2009 handwriting recognition competition. In *Proceedings of the 12th International Workshop on Frontiers of Handwriting Recognition - IWFHR2010*, pages 247-252, 2010.

[e] A-L. Bianne, C. Kermorvant and L. Likforman-Sulem. Modélisation de HMMs en contexte avec des arbres de décision pour la reconnaissance de mots manuscrits. In *Proccedings of the Colloque International Francophone sur l'Ecrit et le Document - CIFED2010*, 2010.

[f] A-L. Bianne, C. Kermorvant and L. Likforman-Sulem. Context-dependent HMM modeling using tree-based clustering for the recognition of handwritten words. In *Proccedings of the 17th Document Recognition and Retrieval Conference, part of the IS&T-SPIE Electronic Imaging Symposium - DRR2010*, vol. 7534, 2010.

[g] F. Menasri, J. Louradour, A-L. Bianne-Bernard, C. Kermorvant. The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition. In *Proccedings of the 19th Document Recognition and Retrieval Conference, part of the IS&T-SPIE Electronic Imaging Symposium - DRR2012*, vol. 8297 : *pages to appear*, 2012.

[h] C. Kermorvant, A-L. Bianne, P. Marty and F. Menasri. From isolated handwritten characters to fields recognition : There's many a slip twixt cup and lip. In *Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, pages 1031-1035, 2009.

[i] A-L. Bianne, C. Kermorvant, P. Marty and F. Menasri. Les caractères ne sont pas la clef des champs. In *Proceedings of the 11th Conférence Francophone sur l'Apprentissage Artificiel - CAP2009*, 2009.

Annexe A

Annexe : questions rhétoriques pour la construction des arbres de décision

Les questions présentées dans cette annexe sont toutes originales et ont été créées en prenant en compte la morphologie des caractères de l'alphabet. Nous présentons d'abord les questions élaborées pour le latin et ensuite celles élaborées pour l'écriture arabe.

Chaque question binaire est de la forme :
`"NomDeLaQuestion" {contexte1, ..., contexteN}`

Le logiciel utilisé pour l'apprentissage des modèles (HTK [169]) ayant un comportement non robuste face aux caractères accentués présents dans le français et ne pouvant prendre en compte des caractères arabes, nous avons translitéré au format ASCII les caractères des alphabets latin et arabe pour éviter des erreurs de transcodage. Pour l'arabe, la translitération utilisée est celle proposée pour la base de données IFN-Enit [131]. Pour le latin, les correspondances sont les suivantes :

- `A1...Z1` réfèrent à a...z
- `A0...Z0` réfèrent à A...Z
- `E2, E3, E4, E5, E6` correspondent à é, è, ê, ë et $\acute{E}$
- `n0...n9` referent to `0...9`
- `sA, sB, sT, sN` correspondent respectivement à `', /, -, %`

Les contextes situés entre les accolades sont ceux pour lesquels la réponse à la question posée (`NomDeLaQuestion`) est oui. Par exemple, les contextes droits b, f, h, k, l et 8 (`*+B1`, `*+F1`, `*+H1`, `*+K1`, `*+L1` et `*+n8`) sont les seuls qui répondent positivement à la question `"R_LC_loopascender"` signifiant *est-ce que le contexte*

droit, qui est un caractère minuscule, inclut un ascendant en forme de boucle ?. Le nom des questions définies pour le latin est donné selon les contextes auxquels la question se réferre :

- `R_` et `L_` correspondent respectivement à une question sur un contexte droit et gauche,
- `LC_` et `UC_` réfèrent à des contextes minuscule et majuscule respectivement,
- `NUM_` correspond à des questions portant sur des contextes numériques.

Pour l'arabe, les questions sont simplement numérotées.

A.1 Questions pour l'écriture latine

```
nom de la                       contextes pour lesquels
question                        la réponse est oui
-------------------------------------------------------------------------
QS "R_isnotchar"                {*+sA,*+sB,*+sT,*+sN}

QS "R_lowercase"                {*+n1,*+n8,*+A1,*+B1,*+C1,*+D1,*+E1,*+F1,
      *+G1,*+H1,*+I1,*+J1,*+K1,*+L1,*+M1,*+N1,*+O1,*+P1,*+Q1,*+R1,*+S1,*+T1,
      *+U1,*+V1,*+W1,*+X1,*+Y1,*+Z1,*+C2}
QS "R_uppercase"                {*+n2,*+n3,*+n4,*+n6,*+n7,*+n9,*+A0,*+B0,
      *+C0,*+D0,*+E0,*+F0,*+G0,*+H0,*+I0,*+J0,*+K0,*+L0,*+M0,*+N0,*+O0,*+P0,
      *+Q0,*+R0,*+S0,*+T0,*+U0,*+V0,*+W0,*+X0,*+Y0,*+Z0,*+A6,*+E6}

QS "R_LC_descender"             {*+F1,*+G1,*+J1,*+P1,*+Q1,*+Y1,*+Z1,*+C2}
QS "R_LC_ascender"              {*+n1,*+n8,*+B1,*+D1,*+F1,*+H1,*+K1,*+L1,
      *+T1}
QS "R_LC_small"                 {*+A1,*+C1,*+D1,*+E1,*+I1,*+M1,*+N1,*+O1,
      *+Q1,*+R1,*+S1,*+U1,*+V1,*+W1,*+X1,*+Z1}
QS "R_LC_accent"                {*+A3,*+A4,*+E2,*+E3,*+E4,*+E5,*+I4,*+I5,
      *+O4,*+U3,*+U4}
QS "R_LC_loopsmall"             {*+n8,*+A1,*+C1,*+D1,*+G1,*+E1,*+O1,*+Q1,
      *+A3,*+A4,*+E2,*+E3,*+E4,*+E5,*+O4,*+C2}
QS "R_LC_loopascender"          {*+n8,*+B1,*+F1,*+H1,*+K1,*+L1}
QS "R_LC_loopdescender"         {*+F1,*+G1,*+J1,*+Y1,*+Z1}
QS "R_LC_barascender"           {*+n1,*+B1,*+F1,*+H1,*+K1,*+L1,*+T1}
QS "R_LC_bardescender"          {*+F1,*+P1}
```

```
QS "R_LC_nushape"                {*+M1,*+N1,*+R1,*+U1,*+V1,*+W1,*+Y1}
QS "R_LC_complexwithinbaseline"  {*+A1,*+B1,*+E1,*+K1,*+O1,*+R1,*+S1,*+X1,
      *+Z1,*+A3,*+A4,*+E2,*+E3,*+E4,*+E5,*+O4}
QS "R_LC_starthigh"              {*+n1,*+V1,*+W1,*+X1,*+Z1}
QS "R_LC_point"                  {*+I1,*+J1}
QS "R_LC_accentaigu"             {*+E2}
QS "R_LC_accentgrave"            {*+A3,*+E3,*+U3}
QS "R_LC_accentcirconflexe"      {*+A4,*+E4,*+I4,*+O4,*+U4}
QS "R_LC_accenttrema"            {*+E5,*+I5}

QS "R_UC_verticalbar"            {*+A0,*+B0,*+E0,*+F0,*+H0,*+K0,*+L0,*+P0,
      *+R0,*+U0,*+A6,*+E6}
QS "R_UC_middlehorizontalbar"    {*+n6,*+n7,*+A0,*+B0,*+E0,*+F0,*+H0,*+K0,
      *+P0,*+R0,*+S0,*+A6,*+E6}
QS "R_UC_uphorizontalbar"        {*+n7,*+E0,*+F0,*+I0,*+J0,*+M0,*+T0,*+Z0,
      *+E6}
QS "R_UC_bottomhorizontalbar"    {*+n4,*+n2,*+E0,*+I0,*+L0,*+Z0,*+E6}
QS "R_UC_upcurve"                {*+n3,*+n2,*+F0,*+H0,*+K0,*+P0,*+R0,*+U0,
      *+V0,*+W0,*+X0,*+Y0}
QS "R_UC_bottomcurve"            {*+n3,*+A0,*+F0,*+H0,*+I0,*+J0,*+K0,*+M0,
      *+N0,*+P0,*+R0,*+S0,*+A6}
QS "R_UC_bottomloop"             {*+n2,*+B0,*+D0,*+L0,*+Q0,*+Z0}
QS "R_UC_bigloop"                {*+C0,*+G0,*+O0,*+Q0}
QS "R_UC_descender"              {*+G0,*+J0,*+Y0}
QS "R_UC_uplefttobottomrightbar" {*+V0,*+W0,*+X0,*+Y0}
QS "R_UC_bottomlefttouprightbar" {*+n7,*+n4,*+n2,*+A0,*+X0,*+Z0,*+A6}
QS "R_UC_accent" {*+A6,*+E6}

QS "R_NUM_isn3"                  {*+n3}
QS "R_NUM_isn6"                  {*+n6}
QS "R_NUM_isn9"                  {*+n9}

QS "L_isnotchar"                 {sA-*,sB-*,sT-*,sN-*}

QS "L_lowercase"                 {n8-*,n9-*,A1-*,B1-*,C1-*,D1-*,E1-*,F1-*,
      G1-*,H1-*,I1-*,J1-*,K1-*,L1-*,M1-*,N1-*,O1-*,P1-*,Q1-*,R1-*,S1-*,T1-*,
      U1-*,V1-*,W1-*,X1-*,Y1-*,Z1-*,A3-*,A4-*,C2-*,E2-*,E3-*,E4-*,E5-*,I4-*,
```

```
      I5-*,O4-*,U3-*,U4-*}
QS "L_uppercase"                {n1-*,n2-*,n3-*,n4-*,n6-*,n7-*,A0-*,B0-*,
      C0-*,D0-*,E0-*,F0-*,G0-*,H0-*,I0-*,J0-*,K0-*,L0-*,M0-*,N0-*,O0-*,P0-*,
      Q0-*,R0-*,S0-*,T0-*,U0-*,V0-*,W0-*,X0-*,Y0-*,Z0-*,A6-*,E6-*}

QS "L_LC_descender"             {F1-*,G1-*,J1-*,P1-*,Q1-*,Y1-*,Z1-*,C2-*}
QS "L_LC_ascender"              {n8-*,n9-*,B1-*,D1-*,F1-*,H1-*,K1-*,L1-*,
      T1-*}
QS "L_LC_small"                 {A1-*,B1-*,C1-*,E1-*,H1-*,I1-*,K1-*,M1-*,
      N1-*,O1-*,P1-*,R1-*,S1-*,U1-*,V1-*,W1-*,X1-*,Z1-*}
QS "L_LC_accent"                {A3-*,A4-*,E2-*,E3-*,E4-*,E5-*,I4-*,I5-*,
      O4-*,U3-*,U4-*}
QS "L_LC_loopsmall"             {n8-*,A1-*,B1-*,O1-*,P1-*,A3-*,A4-*,E2-*,
      E3-*,E4-*,E5-*,O4-*}
QS "L_LC_loopascender"          {n8-*,B1-*,F1-*,H1-*,K1-*,L1-*}
QS "L_LC_loopdescender"         {F1-*,G1-*,J1-*,Y1-*,Z1-*}
QS "L_LC_barascender"           {n9-*,D1-*,F1-*,L1-*}
QS "L_LC_bardescender"          {F1-*,J1-*,Q1-*,Y1-*}
QS "L_LC_nushape"               {M1-*,N1-*,U1-*}
QS "L_LC_complexwithinbaseline" {A1-*,B1-*,E1-*,K1-*,O1-*,S1-*,R1-*,X1-*,
      Z1-*,A3-*,A4-*,E2-*,E3-*,E4-*,E5-*,O4-*}
QS "L_LC_linklow"               {A1-*,C1-*,D1-*,E1-*,F1-*,G1-*,H1-*,I1-*,
      J1-*,K1-*,L1-*,M1-*,N1-*,P1-*,Q1-*,R1-*,S1-*,T1-*,U1-*,X1-*,Y1-*,A3-*,
      A4-*,E2-*,E3-*,E4-*,E5-*,I4-*,I5-*,U3-*,U4-*}
QS "L_LC_linkhigh"              {B1-*,O1-*,R1-*,S1-*,T1-*,V1-*,W1-*,X1-*,
      Z1-*,O4-*}
QS "L_LC_point"                 {I1-*,J1-*,I4-*,I5-*}
QS "L_LC_accentaigu"            {E2-*}
QS "L_LC_accentgrave"           {A3-*,E3-*,U3-*}
QS "L_LC_accentcirconflexe"     {A4-*,E4-*,I4-*,O4-*,U4-*}
QS "L_LC_accenttrema"           {E5-*,I5-*}

QS "L_UC_middlehorizontalbar"   {n6-*,n7-*,A0-*,n3-*,B0-*,E0-*,F0-*,G0-*,
      H0-*,P0-*,S0-*,A6-*,E6-*}
QS "L_UC_uphorizontalbar"       {n7-*,E0-*,F0-*,I0-*,J0-*,n1-*,M0-*,T0-*,
      Z0-*,E6-*}
QS "L_UC_bottomhorizontalbar"   {n2-*,n4-*,E0-*,I0-*,L0-*,Z0-*,E6-*}
```

```
QS "L_UC_bottomcurve"            {A0-*,H0-*,K0-*,L0-*,n1-*,M0-*,R0-*,U0-*,
      Y0-*,Z0-*,A6-*}
QS "L_UC_upcurve"                {n6-*,F0-*,H0-*,K0-*,N0-*,T0-*,X0-*}
QS "L_UC_uploopfromleft"         {C0-*,E0-*,n1-*,M0-*,S0-*,Z0-*,E6-*}
QS "L_UC_uploopfrombottom"       {O0-*,V0-*,W0-*}
QS "L_UC_halfcircleup"           {n2-*,n3-*,B0-*,P0-*,R0-*}
QS "L_UC_halfcirclebottom"       {n6-*,n3-*,B0-*,S0-*}
QS "L_UC_uplefttobottomrightbar" {A0-*,K0-*,Q0-*,R0-*,X0-*,A6-*}
QS "L_UC_bottomlefttouprightbar" {n7-*,K0-*,V0-*,W0-*,X0-*,Y0-*,Z0-*}
QS "L_UC_bigloop"                {D0-*,O0-*,Q0-*}
QS "L_UC_descender"              {n4-*,G0-*,J0-*,Y0-*}
QS "L_UC_accent"                 {A6-*,E6-*}

QS "L_NUM_isn2"                  {n2-*}
QS "L_NUM_isn9"                  {n9-*}
```

A.2 Questions pour l'écriture arabe

```
nom de la     contextes pour lesquels
question      la réponse est oui
-------------------------------------------------------------------------
QS "Q1"       {*+seM,*+seE,*+naM,*+taM,*+taE,*+baM,*+baE,*+thM,*+thE,*+shM,
               *+shE}
QS "Q2"       {*+seB,*+seA,*+naB,*+taB,*+taA,*+baB,*+baA,*+thB,*+thA,*+shB,
               *+shA}
QS "Q3"       {*+teE,*+heE}
QS "Q4"       {*+teA,*+heA}

QS "Q5"       {*+alE,*+eeE,*+yaE}
QS "Q6"       {*+alA,*+eeA,*+yaA}

QS "Q7"       {*+laM,*+keE}
QS "Q8"       {*+laB,*+keA}

QS "Q9"       {*+ayB,*+ghB,*+hhA}
QS "Q10"      {*+ayM,*+ghM}
```

```
QS "Q11"      {*+ayE,*+ghE}
QS "Q12"      {*+ayA,*+ghA}

QS "Q13"      {*+aaE,*+amE,*+aeE,*+ahE}
QS "Q14"      {*+aaA,*+amA,*+aeA,*+ahA}

QS "Q15"      {*+toM,*+dzM,*+saM,*+deM,*+toE,*+dzE,*+saE,*+deE}
QS "Q16"      {*+toB,*+dzB,*+saB,*+deB,*+toA,*+dzA,*+saA,*+deA}


QS "Q17"      {*+raE,*+zaE}
QS "Q18"      {*+raM,*+zaM}

QS "Q19"      {*+waE,*+whE}
QS "Q20"      {*+waA,*+whA}

QS "Q21"      {*+faM,*+kaM,*+faE,*+kaE}
QS "Q22"      {*+faB,*+kaB,*+faA,*+kaA}


QS "Q23"      {*+jaB,*+haB,*+khB}
QS "Q24"      {*+jaM,*+haM,*+khM}
QS "Q25"      {*+jaE,*+haE,*+khE}
QS "Q26"      {*+jaA,*+haA,*+khA}

QS "Q27"      {*+daA,*+dhA}
QS "Q28"      {*+daE,*+dhE}


QS "Q29"      {waE-*,whE-*,raE-*,zaE-*,waA-*,whA-*,raA-*,zaA-*}
QS "Q30"      {waE-*,whE-*,raE-*,zaE-*,waA-*,whA-*,raA-*,zaA-*,naE-*,naA-*,
               laE-*,laA-*}


QS "Q31"      {seB-*,seM-*,naB-*,naM-*,taB-*,taM-*,baB-*,baM-*,thB-*,thM-*,
               shB-*,shM-*}
```

```
QS "Q32"      {aaE-*,amE-*,aeE-*,ahE-*}
QS "Q33"      {aaA-*,amA-*,aeA-*,ahA-*}

QS "Q34"      {alA-*,eeA-*,yaA-*,alE-*,eeE-*,yaE-*}


QS "Q35"      {jaB-*,haB-*,khB-*}
QS "Q36"      {jaM-*,haM-*,khM-*}
QS "Q37"      {jaE-*,haE-*,khE-*}
QS "Q38"      {jaA-*,haA-*,khA-*}


QS "Q39"      {ayB-*,ghB-*}
QS "Q40"      {ayM-*,ghM-*}
QS "Q41"      {ayE-*,ghE-*}
QS "Q42"      {ayA-*,ghA-*}


QS "Q43"      {daA-*,dhA-*,daE-*,dhE-*}

QS "Q44"      {saM-*,deM-*,saB-*,deB-*}
QS "Q45"      {toM-*,dzM-*,toB-*,dzB-*}


QS "Q46"      {naE-*,naA-*,seE-*,seA-*,shE-*,shA-*,saE-*,saA-*,deE-*,deA-*,
               laE-*,laA-*}
```

Bibliographie

[1] K. Ait-Mohand, T. Paquet, N. Ragot et L. Heutte : Structure adaptation of HMM applied to ocr. *In Proceedings of the 20th International Conference on Pattern Recognition - ICPR2010*, p. 2877–2880, 2010.

[2] R. Al-Hajj Mohamad : *Reconnaissance hors ligne de mots manuscrits cursifs par l'utilisation de systemes hybrides et de techniques d'apprentissage automatique*. Thèse de doctorat, École Nationale Supérieure des Télécommunications, at École Doctorale d'Informatique, Télécommunications et Electronique de Paris, 2007.

[3] N. B. Amara et F. Bouslama : Classification of arabic script using multiple sources of information : state of the art and perspectives. *International Journal on Document Analysis and Recognition - IJDAR*, 5:195–212, 2003.

[4] E. Augustin, M. Carre, E. Grosicki, J.-M. Brodin, E. Geoffrois et F. Preteux : RIMES evaluation campaign for handwritten mail processing. *In Proceedings of the Tenth International Workshop on Frontiers of Handwriting Recognition - IWFHR2006*, p. 231–235, 2006.

[5] E. Augustin : *Handwritten Word Recognition using Hybrid Neural Network and Hidden Markov Systems*. Thèse de doctorat, René Descartes University - Paris V, Mathematics and Computer Science Department, 2001.

[6] P. Baldi, Y. Chauvin, T. Hunkapiller et M. A. McClure : Hidden Markov Models in molecular biology : new algorithms and applications. *Advances in Neural Processing Systems*, 5:747–754, 1993.

[7] L. E. Baum, T. Petrie, G. Soules et N. Weiss : A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[8] I. Bazzi, R. Schwartz et J. Makhoul : An omnifont open-vocabulary OCR system for english and arabic. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:495–504, 1999.

[9] J. Benesty, M. M. Sondhi et Y. Huang : *Springer Handbook of Speech Processing*, p. 177–179. Springer, 2008.

[10] Y. Bengio, R. Ducharme et P. Vincent : A neural probabilistic language model. *Neural Information Processing Systems*, 13:933–938, 2001.

[11] R. Bertolami, S. Uchida, M. Zimmermann et H. Bunke : Non-uniform slant correction for handwritten text line recognition. *In Proceedings of the Ninth International Conference on Document Analysis and Recognition - ICDAR2007*, p. 18–22, 2007.

[12] A.-L. Bianne-Bernard, F. Menasri, R. Al-Hajj Mohamad, C. Mokbel, C. Kermorvant et L. Likforman-Sulem : Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. to appear, 2011.

[13] M. Blumenstein, C. K. Cheng et X. Y. Liu : New preprocessing techniques for handwritten word recognition. *In Proceedings of the Second IASTED International Conference on Visualization, Imaging and Image Processing*, p. 480–484, 2002.

[14] R. M. Bozinovic et S. N. Srihari : Off-line cursive script word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:68–83, 1989.

[15] A. Brakensiek et G. Rigoll : Handwritten address recognition using Hidden Markov Models. *In* A. Dengel, M. Junker et A. Weisbecker, éds : *Reading and Learning*, vol. 2956 de *Lecture Notes in Computer Science*, p. 103–122. Springer Berlin Heidelberg, 2004.

[16] A. Brakensiek, J. Rottland et G. Rigoll : Handwritten address recognition with open vocabulary using character n-grams. *In Proceedings of the Eighth International Workshop on Frontiers of Handwriting Recognition - IWFHR2002*, p. 357, 2002.

[17] P. F. Brown, V. J. Della Pietra, P. V. de Souza, J. C. Lai et R. L. Mercer : Class based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[18] R. Buse, Z.-Q. Liu et T. Caelli : A structural and relational approach to handwritten word recognition. *IEEE Transactions on Systems, Man and Cybernetics*, B:847–861, 1997.

[19] T. Caesar, J. Gloger et E. Mandler : Preprocessing and feature extraction for a handwriting recognition system. *In Proceedings of the Second International Conference on Document Analysis and Recognition - ICDAR1993*, p. 408–411, 1993.

[20] H. Cao, R. Prasad et P. Natarajan : Improvements in HMM adaptation for handwriting recognition using writer identification and duration adaptation. *In Proceedings of the 12th International Conference on Frontiers of Handwriting Recognition - ICFHR2010*, p. 154–159, 2010.

[21] R. Cattoni, T. Coianiz, S. Messelodi et C. M. Modena : Geometric layout analysis techniques for document image understanding : a review. Rap. tech., TC-IRST Technical Report, 1998.

[22] M.-Y. Chen, A. Kundu et S. N. Srihari : Variable duration Hidden Markov Model and morphological segmentation for handwritten word recognition. *IEEE Transactions on Image Processing*, 4:1675–1688, 1995.

[23] S. F. Chen : Shrinking exponential language models. *In Proceedings of Human Language Technologies*, p. 468–476, 2009.

[24] S. F. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya et A. Sethy : Scaling shrinkage-based language models. *In Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding - ASRU2009*, p. 299–302, 2009.

[25] C. Choisy : Dynamic handwritten keyword spotting based on the NSHP-HMM. *In Proceedings of the Ninth International Conference on Document Analysis and Recognition - ICDAR2007*, p. 242–246, 2007.

[26] M. Côté, E. Lecolinet, M. Cheriet et C. Y. Suen : Automatic reading of cursive scripts using a reading model and perceptual concepts. the perpecto system. *International Journal on Document Analysis and Recognition - IJDAR*, 1(1):3–17, 1998.

[27] M. De Groot : *Optimal statistical decisions*. McGraw-Hill, New York, 1970.

[28] J. J. De Oliveira, J. de Carvalho, C. Freitas et R. Sabourin : Feature sets evaluation for handwritten word recognition. *In Proceedings of the Eigth International Workshop on Frontiers of Handwriting Recognition - IWFHR2002*, p. 446–451, 2002.

[29] V. Digalakis, D. Rtischev, L. Neumeyer et E. Sa : Speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Transactions on Speech and Audio Processing*, 3:357–366, 1995.

[30] W. Ding, C. Suen et A. Krzyzak : A new courtesy amount recognition module of a check reading system. *In Proceedings of the 19th International Conference on Pattern Recognition - ICPR2008*, p. 1–4, 2008.

[31] J. Domenech, A. H. Toselli, A. Juan, E. Vidal et F. Casacuberta : An off-line HTK-based OCR system for isolated handwritten lowecase letters. *Proceedings of the Ninth Spanish Symposium on Pattern Recognition and Image Analysis*, 2:49–54, 2001.

[32] P. Dreuw, G. Heigold et H. Ney : Confidence-based discriminative training for model adaptation in offline arabic handwriting recognition. *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, p. 596–600, 2009.

[33] P. Dreuw, S. Jonas et H. Ney : White-space models for offline arabic handwriting recognition. *In Proceedings of the 19th International Conference on Pattern Recognition - ICPR2008*, 2008.

[34] P. Dreuw, D. Rybach, C. Gollan et H. Ney : Writer adaptive training and writing variant model refinement for offline arabic handwriting recognition. *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, p. 21–25, 2009.

[35] X. Dupré : *Contributions à la reconnaissance de l'écriture cursive à l'aide de modèles de Markov cachés*. Thèse de doctorat, René Descartes University - Paris V, Mathematics and Computer Science Department, 2004.

[36] D. Eck, A. Graves et J. Schmidhuber : A new approach to continuous speech recognition using lstm recurrent neural networks. Rap. tech., IDSIA, 2003.

[37] H. El Abed et M. V. : Reject rules and combination methods to improve arabic handwritten word recognizers. *In Proceedings of the 11th International Conference on Frontiers of Handwriting Recognition - ICFHR2008*, p. 6, 2008.

[38] H. El Abed et V. Märgner : ICDAR2009 arabic handwriting recognition competition. *International Journal on Document Analysis and Recognition - IJDAR*, 14:3–13, 2011.

[39] R. El-Hajj, L. Likforman-Sulem et C. Mokbel : Arabic handwriting recognition using baseline dependant features and Hidden Markov Modeling. *In Proceedings of the Eigth International Conference on Document Analysis and Recognition - ICDAR2005*, p. 893–897, 2005.

[40] R. M. El-Hajj, L. Likforman-Sulem et C. Mokbel : Combining slanted-frame classifiers for improved HMM-based arabic handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1165–1177, 2009.

[41] A. El-Yacoubi, R. Sabourin, C. Y. Suen et M. Gilloux : An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:752–760, 1999. ISSN 0162-8828.

[42] A. Emami, S. F. Chen, A. Ittycheriah, H. Soltau et B. Zhao : Decoding with shrinkage-based language models. *In Proceedings of Interspeech*, p. 1033–1036, 2010.

[43] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya et F. Zamora-Martínez : Improving offline handwritten text recognition with

hybrid HMM/ANN models. Rap. tech., Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valence, Espagne, 2009.

[44] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya et F. Zamora-Martínez : Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:767–779, 2011.

[45] S. España-Boquera, J. Gorbe-Moya, M. J. Castro-Bleda et F. Zamora-Martínez : Hybrid HMM/ann models for bimodal online and offline cursive word recognition. *In Proceedings of the 20th International Conference on Pattern Recognition - ICPR2010*, 2010.

[46] J. Favata : Character model word recognition. *In Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition - IWFHR1996*, p. 437–440, 1996.

[47] M. Feldbach : *Generierung einer semantischen reprasentation aus abbildungen handschritlicher kirchenbuchaufzeichnungen.* Thèse de doctorat, Universitat Magdeburg, 2000.

[48] G. Fink et T. Plötz : On the use of context-dependent modeling units for HMM-based offline handwriting recognition. *In Proceedings of the Ninth International Conference on Document Analysis and Recognition - ICDAR2007*, p. 729–733, 2007.

[49] G. A. Fink et T. Plötz : Unsupervised estimation of writing style models for improved unconstrained off-line handwriting recognition. *In Proceedings of the 10th International Workshop on Frontiers of Handwriting Recognition - IWFHR2006*, p. 429–434, 2006.

[50] M. Gales : The generation and use of regression class trees for mllr adaptation. Rap. tech., Technical Report CUED/F-INFENG/TR263, Cambridge University, 1996.

[51] M. Gales : Maximum likelihood linear transformations for HMM-based speech recognition. Rap. tech., Technical Report CUED/F-INFENG/TR291, Cambridge University, 1997.

[52] M. Gales : Maximum likelihood linear transformations for HMM-based speech recognition. *Computer, Speech and Language*, 12:75–98, 1998.

[53] B. Gatos, N. Stamatopoulos et G. Louloudis : ICDAR2009 handwriting segmentation contest. *International Journal on Document Analysis and Recognition - IJDAR*, 14:25–33, 2011.

[54] B. Gatos, K. Ntirogiannis et I. Pratikakis : ICDAR 2009 document image binarization contest (DIBCO 2009). *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, p. 1375–1382, 2009.

[55] J.-L. Gauvain et C.-H. Lee : MAP estimation of continuous density HMM : Theory and applications. *In Proceedings of DARPA Speech and Natural Language Workshop*, p. 185–190, 1992.

[56] J.-L. Gauvain et C.-H. Lee : Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 1994.

[57] A. Giménez et A. Juan : Embedded bernoulli mixture HMMs for handwritten word recognition. *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, p. 896–900, 2009.

[58] A. Giménez, I. Khoury et A. Juan : Windowed bernoulli mixture HMMs for arabic handwritten word recognition. *In Proceedings of the 12th International Conference on Frontiers of Handwriting Recognition - ICFHR2010*, p. 533–538, 2010.

[59] A. Giménez-Pastor et A. Juan-Císcar : Bernoulli HMMs for off-line handwriting recognition. *In Proceedings of the 8th International Workshop on Pattern Recognition in Information Systems - PRIS2008*, p. 86–91, 2008.

[60] N. Gorski, V. Anisimov, E. Augustin, O. Baret et S. Maximov : Industrial bank check processing : the a2ia checkreader. *International Journal on Document Analysis and Recognition - IJDAR*, 3(4):196–206, 2001.

[61] A. Graves, S. Fernàndez et J. Schmidhuber : Multidimensional recurrent neural networks. *In Proceedings of the International Conference on Artificial Neural Networks - ICANN2007*, 2007.

[62] A. Graves et J. Schmidhuber : Offline handwriting recognition with multi-dimensional recurrent neural networks. *Advances in Neural Information Processing Systems*, 21, 2009.

[63] E. Grosicki, M. Carre, J.-M. Brodin et E. Geoffrois : Results of the second rimes evaluation campaign for handwritten mail processing. *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, 2009.

[64] E. Grosicki et H. El-Abed : ICDAR 2009 handwriting recognition competition. *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, p. 1398–1402, 2009.

[65] E. Grosicki et H. El-Abed : ICDAR 2011 – french handwriting recognition competition. *In Proceedings of the Eleventh International Conference on Document Analysis and Recognition - ICDAR2011*, p. 1459–1463, 2011.

[66] L. Guichard, A. Toselli et B. Coasnon : Handwritten word verification by svm-based hypotheses re-scoring and multiple thresholds rejection. *In Proceedings of the 12th International Conference on Frontiers of Handwriting Recognition - ICFHR2010*, p. 57–62, 2010.

[67] S. Hochreiter et J. Schmidhuber : Long short-term memory. *Neural Computer*, 9:1735–1780, 1997.

[68] P. V. C. Hough : Methods and means for recognizing complex patterns, 1962.

[69] Gallica project website : http ://gallica.bnf.fr.

[70] Rimes database website : http ://rimes.itsudparis.eu/.

[71] RNN-Lib downloading website : https ://github.com/meierue/RNNLIB/.

[72] sites web de généalogistes utilisant la reconnaissance automatique de manuscrits anciens : https ://www.familysearch.org et www.coutot-roehrig.com.

[73] Europeana project website : http ://www.europeana.eu/portal.

[74] OpenHart project website : http ://www.nist.gov/itl/iad/mig/hart2010.cfm, 2010.

[75] J. Huang, E. Marcheret et K. Visweswariah : Rapid feature space speaker adaptation for multi-stream HMM-based audio-visual speech recognition. *In IEEE International Conference on International Multimedia and Expo*, p. 338–341, 2005.

[76] ICDAR, éd. *10th International Conference on Document Analysis and Recognition*, 2009.

[77] E. Indermühle, M. Eichenberger-Liwicki et H. Bunke : Recognition of handwritten historical documents : HMM-adaptation vs. writer specific training. *In Proceedings of the 11th International Conference on Frontiers of Handwriting Recognition - ICFHR2008*, p. 186–191, 2008.

[78] N. L. Johnson et S. Kotz : *Distribution in Statistics*. Wiley, New York, 1972.

[79] A. Kaltenmeier, T. Caesar, J. Gloger et E. Mandler : Sophisticated topology of Hidden Markov Models for cursive script recognition. *In Proceedings of the Second International Conference on Document Analysis and Recognition - ICDAR1993*, p. 139–142, 1993.

[80] C. Kermorvant, F. Menasri, A.-L. Bianne, R. Al-Hajj, C. Mokbel et L. Likforman-Sulem : The A2iA-Telecom ParisTech-UOB system for the ICDAR 2009 handwriting recognition competition. *In Proceedings of the 12th International Workshop on Frontiers of Handwriting Recognition - IWFHR2010*, p. 247–252, 2010.

[81] Y. Kessentini, T. Paquet et A. Benhamadou : A multi-stream HMM-based approach for off-line multi-script handwritten word recognition. *In Proceedings of the 11th International Conference on Frontiers of Handwriting Recognition - ICFHR2008*, 2008.

[82] D. Keysers, T. Deselaers, C. Gollan et H. Ney : Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1422–1435, 2007.

[83] G. Kim et V. Govindaraju : A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:366–379, 1997.

[84] I.-K. Kim, D.-W. Jung et R.-H. Park : Document image binarization based on topographic analysis using a water flow model. *Pattern Recognition*, 35:265–277, 2002.

[85] R. Kneser et H. Ney : Improved clustering techniques for class-based statistical language modelling. *In Proceedings of Eurospeech*, p. 973–976, 1993.

[86] A. L. Koerich, A. S. Britto Jr., L. E. S. D. Oliveira et R. Sabourin : Fusing High- and Low-Level Features for Handwritten Word Recognition. *In Proceedings of the Tenth International Workshop on Frontiers of Handwriting Recognition - IWFHR2006*, 2006.

[87] A. Kundu, Y. He et M.-Y. Chen : Alternatives to variable duration HMM in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1275–1280, 1998.

[88] I. Lamouche et C. Bellissant : Séparation rectoverso d'images de manuscrits anciens. *In Colloque National sur l'Ecrit et le Document - CNED1996*, p. 199–206, 1996.

[89] V. Lavrenko, T. M. Rath et R. Manmatha : Holistic word recognition for handwritten historical documents. *In First International Workshop on Document Image Analysis for Libraries - DIAL2004*, p. 278–287, 2004.

[90] F. LeBourgeois, H. Emptoz, E. Trinh et J. Duong : Networking digital document images. *In Proceedings of the Sixth International Conference on Document Analysis and Recognition - ICDAR2001*, p. 379–383, 2001.

[91] K.-F. Lee : Context-dependent phonetic Hidden Markov Models for speaker-independent continuous speech recognition. *Readings in Speech Recognition*, 1:347–366, 1990.

[92] C. J. Leggetter et P. C. Woodland : Maximum likelihood linear regression for speaker adaptation of continuous density Hidden Markov Models. *Computer, Speech and Language*, 9:171–185, 1995.

[93] X. Lei, J. Hamaker et X. He : Robust feature space adaptation for telephony speech recognition. *In Proceedings of the Ninth International Conference on Spoken Language Processing - ICSLP2006 - INTERSPEECH*, 2006.

[94] Y. Li, Y. Zheng, D. Doermann et S. Jaeger : Script-independent text line segmentation in freestyle handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1313–1329, 2008.

[95] L. Likforman-Sulem : Extraction d'éléments graphiques dans les images de manuscrits. *In Proccedings of the Colloque International Francophone sur l'Ecrit et le Document - CIFED1998*, p. 223–232, 1998.

[96] L. Likforman-Sulem et C. Faure : Extracting lines on handwritten documents by perceptual grouping. *In* C. Faure, P. Keuss, G. Lorette et A. Winter, éds : *Advances in Handwiting and drawing : a multidisciplinary approach*, p. 21–38. Europia, Paris, 1994.

[97] L. Likforman-Sulem, A. Hanimyan et C. Faure : A hough based algorithm for extracting text lines in handwritten document. *In Proceedings of the Third International Conference on Document Analysis and Recognition - ICDAR1995*, p. 774–777, 1995.

[98] L. Likforman-Sulem, A. Zahour et B. Taconet : Text line segmentation of historical documents : a survey. *International Journal on Document Analysis and Recognition - IJDAR*, 9:123–138, 2007.

[99] D. G. Lowe : Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[100] E. MacRostie, R. Prasad, S. Rawls, M. Kamali, H. Cao, K. Subramanian et P. Natarajan : The bbn document analysis service : a platform for multilingual document translation. *In Proceedings of the Ninth IAPR International Workshop on Document Analysis Systems - DAS2010*, p. 447–454, 2010.

[101] S. Madhvanath et V. Govindaraju : Local reference lines for handwritten phrase recognition. *Pattern Recognition*, 32(12):2021–2028, 1999.

[102] R. Manmatha et N. Srimal : Scale space technique for word segmentation in handwritten manuscripts. *In Proceedings of the Second International Conference on Scale-Space Theories Computer Vision - ScaleSpace1999*, p. 22–33, 1999.

[103] S. Mao, A. Rosenfeld et T. Kanungo : Document structure analysis algorithms : a litterature survey. *In Proccedings of the 10th Document Recognition and Retrieval Conference, part of the IS&T-SPIE Electronic Imaging Symposium - DRR2003*, p. 197–207, 2003.

[104] V. Märgner, et H. El Abed : *Arabic Handwriting Recognition Competition*, p. 1274–1278. 2007.

[105] V. Märgner et H. E. Abed : ICFHR 2010 arabic handwriting recognition competition. *In Proceedings of the 12th International Conference on Frontiers of Handwriting Recognition - ICFHR2010*, p. 709–714, 2010.

[106] V. Märgner, M. Pechwitz et H. E. Abed : ICDAR 2005 arabic handwriting recognition competition. *In Proceedings of the Eigth International Conference on Document Analysis and Recognition - ICDAR2005*, 2005.

[107] U.-V. Marti et H. Bunke. : A full english sentence database for off-line handwriting recognition. *In Proceedings of the Fifth International Conference on Document Analysis and Recognition - ICDAR1999*, p. 705–708, 1999.

[108] U.-V. Marti et H. Bunke : On the influence of vocabulary size and language models in unconstrained handwritten text recognition. *In Proceedings of the Sixth International Conference on Document Analysis and Recognition - ICDAR2001*, p. 260–265, 2001.

[109] U.-V. Marti et H. Bunke : Text line segmentation and word recognition in a system for general writer independent handwriting recognition. *In Proceedings of the Sixth International Conference on Document Analysis and Recognition - ICDAR2001*, p. 159–163, 2001.

[110] U.-V. Marti et H. Bunke : The IAM-database : An English sentence database for off-line handwriting recognition. *International Journal on Document Analysis and Recognition - IJDAR*, 5:39–46, 2002.

[111] U.-V. Marti et H. Bunke : *Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems*, p. 65–90. World Scientific Publishing Co., Inc., 2002.

[112] F. Menasri, N. Vincent, M. Cheriet et E. Augustin : Shape-based alphabet for off-line arabic handwriting recognition. *In Proceedings of the Ninth International Conference on Document Analysis and Recognition - ICDAR2007*, p. 969–973, 2007.

[113] F. Menasri : *Contributions à la reconnaissance de l'ecriture arabe manuscrite*. Thèse de doctorat, René Descartes University - Paris V, Mathematics and Computer Science Department, 2008.

[114] C. Mokbel : *Reconnaissance de la parole dans le bruit : bruitagedébruitage*. Thèse de doctorat, École Nationale Supérieure des Télécommunications, 1992.

[115] C. Mokbel : On-line adaptation of HMMs to real-life conditions : A unified framework. *IEEE Transactions on Speech and Audio Processing*, 9:342–357, 2001.

[116] C. Mokbel et G. Chollet : Word recognition in the car : Speech enhancement spectral transformation. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing - ICASSP1991*, p. 925–928, 1991.

[117] G. Nagy, S. Seth et M. Viswanathan : A prototype document image analysis system for technical journals. *Computer*, 25:10–22, 1992.

[118] P. Natarajan, S. Saleem, R. Prasad, E. MacRostie et K. Subramanian : Multi-lingual offline handwriting recognition using Hidden Markov Models : A script-independent approach. *In Summit on Arabic and Chinese Handwriting - SACH2006*, 2006.

[119] P. Natarajan, Z. Lu, R. M. Schwartz, I. Bazzi et J. Makhoul : Multilingual machine printed OCR. *International Journal of Pattern Recognition and Artificial Intelligence - IJPRAI*, 15(1):43–63, 2001.

[120] navidomass.univ-lr.fr : NAVIgation into DOcument MASSes - navigation dans des masses de documents, 2007.

[121] W. Niblack : *An introduction to digital image processing*, p. 115–116. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[122] A. Nosary, L. Heutte et T. Paquet : Unsupervised writer adaptation applied to handwritten text recognition. *Pattern Recognition*, 37(2):385–388, 2004.

[123] A. Nosary, L. Heutte, T. Paquet et Y. Lecourtier : Defining writer's invariants to adapt the recognition task. *In Proceedings of the Fifth International Conference on Document Analysis and Recognition - ICDAR1999*, p. 765–768, 1999.

[124] J. J. Odell : *The Use of Decision Trees with Context Sensitive Phoneme Modelling*. Thèse de doctorat, Cambridge University Engineering Department, 1992.

[125] L. O'Gorman : The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1162–1173, 1993.

[126] N. Otsu : A threshold selection method from grey-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[127] R. Palacios, A. Gupta et P. Wang : Handwritten bank check recognition of courtesy amounts. *International Journal of Image and Graphics - IJIG*, 4 (2):203–222, 2004.

[128] T. Paquet et Y. Lecourtier : Automatic reading of the literal amount of bank checks. *Machine Vision and Applications - MVA*, 6(2-3):151–162, 1993.

[129] J. Park, X. Liu, M. J. F. Gales et P. C. Woodland : Improved neural network based language modeling and adaptation. *In Proceedings of Interspeech*, p. 1041–1044, 2010.

[130] R. Parker : Arabic gigaword fourth edition, 2009.

[131] M. Pechwitz, S. S. Maddouri, V. Maergner, N. Ellouze et H. Amiri : Ifn/enit database of handwritten arabic words. *In Proccedings of the Colloque International Francophone sur l'Ecrit et le Document - CIFED2002*, 2002.

[132] R. Plamondon et S. Srihari : Online and offline handwriting recognition : A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:63–84, 2000.

[133] T. Plötz, C. Thurau et G. A. Fink : Camera-based whiteboard reading : New approaches to a challenging task. *In Proceedings of the 11th International Conference on Frontiers of Handwriting Recognition - ICFHR2008*, p. 385–390, 2008.

[134] T. Plötz et G. Fink : Markov models for offline handwriting recognition : a survey. *International Journal on Document Analysis and Recognition - IJDAR*, 12:269–298, 2009.

[135] E. Poisson, C. Viard-Gaudin et P.-M. Lallican : Système TDNN / HMM de reconnaissance de mots cursifs en ligne à apprentissage simplifié. *In Proccedings of the Colloque International Francophone sur l'Ecrit et le Document - CIFED2004*, 2004.

[136] D. Povey et K. Yao : A basis representation of constrained MLLR transforms for robust adaptation. *Computer, Speech and Language*, 26(1):35–51, 2012.

[137] I. Pratikakis, B. Gatos et K. Ntirogiannis : H-DIBCO 2010 - handwritten document image binarization competition. *In Proceedings of the 12th International Conference on Frontiers of Handwriting Recognition - ICFHR2010*, p. 727–732, 2010.

[138] projet DIGIDOC : Document Image diGitisation with Interactive Description Capability, 2011.

[139] L. R. Rabiner : A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[140] A. F. R. Rahman et M. C. Fairhurst : Multiple classifier decision combination strategies for character recognition : A review. *International Journal on Document Analysis and Recognition - IJDAR*, 5(4):166–194, 2003.

[141] G. Rigoll, A. Kosmala et D. Willett : An investigation of context-dependent and hybrid modeling techniques for very large vocabulary on-line cursive handwriting recognition, 1998.

[142] J. A. Rodriguez et F. Perronnin : Local gradient histogram features for word spotting in unconstrained handwritten documents. *In Proceedings of the 11th International Conference on Frontiers of Handwriting Recognition - ICFHR2008*, 2008.

[143] J. A. Rodríguez et F. Perronnin : Handwritten word-spotting using Hidden Markov Models and universal vocabularies. *Pattern Recognition*, 42 (9):2106–2116, 2009.

[144] J. A. Rodríguez-Serrano, F. Perronnin, G. Sánchez et J. Lladós : Unsupervised writer adaptation of whole-word HMMs with application to word-spotting. *Pattern Recognition Letters*, 31:742–749, 2010.

[145] J. Sauvola et M. Pietikäinen : Adaptive document image binarization. *Pattern Recognition*, 33:225–236, 2000.

[146] M.-P. Schambach : Model length adaptation of an HMM-based cursive word recognition system. *In Proceedings of the Seventh International Conference on Document Analysis and Recognition - ICDAR2003*, p. 109–113, 2003.

[147] M. Schussler et H. Niemann : A HMM-based system for recognition of handwritten address words. *In Proceedings of the Sixth International Workshop on Frontiers of Handwriting Recognition - IWFHR1998*, p. 505–514, 1998.

[148] H. Schwenk : Continuous space language models. *Computer, Speech and Language*, 21:492–518, 2007.

[149] H. Schwenk et J.-L. Gauvain : Connectionist language modeling for large vocabulary continuous speech recognition. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing - ICASSP2002*, p. 765–768, 2002.

[150] A. W. Senior et A. J. Robinson : An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:309–321, 1998.

[151] V. Shapiro, G. Gluchev et V. Sgurev : Handwritten document image segmentation and analysis. *Pattern Recognition Letters*, 14:71–78, 1993.

[152] Z. Shi, S. Setlur et V. Govindaraju : Text extraction from gray scale historical document images using adaptive local connectivity map. *In Proceedings of the Eighth International Conference on Document Analysis and Recognition - ICDAR2005*, p. 794–798, 2005.

[153] Z. Shi, S. Setlur et V. Govindaraju : A steerable directional local profile technique for extraction of handwritten arabic text lines. *In Proceedings of the Tenth International Conference on Document Analysis and Recognition - ICDAR2009*, p. 176–180, 2009.

[154] A. Stolcke : SRILM - an extensible language modeling toolkit, 2002.

[155] B. Su, S. Lu et C. L. Tan : Binarization of historical document images using the local maximum and minimum. *In Proceedings of the Ninth IAPR International Workshop on Document Analysis Systems - DAS2010*, p. 159–166, 2010.

[156] C. Y. Suen : N-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:164–172, 1979.

[157] C. Suen, L. Lam, D. Guillevic, N. Strathy, M. Cheriet, J. Said et R. Fan : Bank check processing system. *International Journal of Imaging Systems and Technology - IJIST*, 7(4):392–403, 1996.

[158] C. L. Tann, R. Cao et P. Shen : Restoration of archival documents using a wavelet technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1399–1404, 2002.

[159] A. Vinciarelli, S. Bengio et H. Bunke : offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:709–720, 2004.

[160] A. Vinciarelli et J. Luettin : A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9):1043–1050, 2001.

[161] A. Vinciarelli et S. Bengio : Writer adaptation techniques in off-line cursive word recognition. *In Proceedings of the Eighth International Workshop on Frontiers of Handwriting Recognition - IWFHR2002*, p. 287, 2002.

[162] K. Visweswariah, V. Goel et R. Gopinath : Maximum likelihood training of bases for rapid adaptation. *In Proceedings of the Seventh International Conference on Spoken Language Processing - ICSLP2002 - INTERSPEECH*, 2002.

[163] K. Visweswariah, V. Goel et R. Gopinath : Structuring linear transforms for adaptation using training time information. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing - ICASSP2002*, 2002.

[164] A. Viterbi : Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 (2):260–269, 1967.

[165] M. Wienecke, G. A. Fink et G. Sagerer : Toward automatic video-based whiteboard reading. *International Journal on Document Analysis and Recognition - IJDAR*, 7(2–3):188–200, 2005.

[166] K. Wong, R. Casey et F. Wahl : Document analysis systems. *IBM Journal of Research and Development*, 26, 1982.

[167] OpenHart 2010 competition description : www.nist.gov/itl/iad/mig/upload/openhart2010_EvalPlan_v2-8.pdf, 2010.

[168] S. J. Young, J. J. Odell et P. C. Woodland : Tree-based state tying for high accuracy acoustic modelling. *In Proceedings of the workshop on Human Language Technology - HLT1994*, p. 307–312, 1994.

[169] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev et P. Woodland : *The HTK Book V3.4*. Cambridge University Press, Cambridge, UK, 2006.

[170] F. Zamora-Martínez, M. J. Castro-Bleda, S. España-Boquera et J. Gorbe-Moya : Improving isolated handwritten word recognition using a specialized classifier for short words. *In 10.1007978-3-642-14264-2_ 7*, vol. 5988 de *LNCS*, p. 61–70. Springer, 2010.

[171] M. Zimmermann et H. Bunke : Hidden Markov Model length optimization for handwriting recognition systems. *In Proceedings of the Eighth International Workshop on Frontiers of Handwriting Recognition - IWFHR2002*, p. 369–375, 2002.