Reconnaissance de textes manuscrits par modèles de Markov cachés et réseaux de neurones récurrents : application à l'écriture latine et arabe

Olivier Morillot

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUT OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0002

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

Télécom ParisTech

Spécialité « Signal et Images »

présentée et soutenue publiquement par

Olivier MORILLOT

le 23 janvier 2014

Reconnaissance de textes manuscrits par Modèles de Markov Cachés et Réseaux de Neurones Récurrents : application à l'écriture latine et arabe

Directeur de thèse : **Laurence LIKFORMAN-SULEM**

Co-encadrement de la thèse : **Emmanuèle GROSICKI**

Jury

Mme Emmanuèle GROSICKI, Docteur, Direction Générale de l'Armement	Examinateur
M. Christopher KERMORVANT, Docteur, A2iA	Examinateur
Mme Laurence LIKFORMAN-SULEM, Maître de conférences HDR, LTCI, Télécom ParisTech	Examinateur
M. Josep LLADÓS, Associate Professor, CVC, Universitat Autonoma de Barcelona	Rapporteur
M. Christian VIARD-GAUDIN, Professeur, IRCCyN, Université de Nantes	Rapporteur
Mme Nicole VINCENT, Professeur, LIPADE, Université de Paris Descartes	Présidente

Télécom ParisTech

école de l'Institut Mines-Télécom - membre de ParisTech

46, rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

Remerciements

Je souhaite tout d'abord remercier Laurence Likforman-Sulem qui m'a permis de faire cette thèse. Je la remercie pour son implication, ses conseils avisés et son soutien tout au long de ces trois années. Je remercie également Emmanuèle Grosicki qui a co-encadré mes travaux. Les nombreuses discussions que nous avons eues tous les trois ont toujours été sources d'idées nouvelles.

Je remercie Josep Lladós et Christian Viard-Gaudin d'avoir acceptés d'être rapporteurs de ma thèse. Mes remerciements vont aussi à Nicole Vincent et à Christopher Kermorvant pour leur examen de mes travaux.

Je suis reconnaissant de l'aide et des conseils que m'ont apporté Marcus Liwicki, Saad Bin Ahmed et Andreas Fischer pour le développement des réseaux récurrents. Un grand merci à eux ainsi qu'à Anne-Laure Bianne-Bernard qui m'a bien aidé dans mes premiers travaux. Je remercie également Cristina Oprean et Chafic Mokbel, avec qui j'ai eu grand plaisir à travailler.

Ensuite, mon expérience de doctorant n'aurait pas été aussi agréable sans la bonne humeur de mes collègues. Entre autres, je tiens à remercier mes amis du bureau DB313 : Cristina, Marc, Paul et Maxime. Ils ont su faire du bureau un lieu très accueillant, convivial et riche en découvertes musicales. Je n'oublierai pas non plus mes autres amis doctorants - Émilie, Amandine, les deux Sylvain, Yao, Andrés, Eric et Mathieu - avec qui j'ai également partagé sorties et voyages.

Je remercie mes parents, Claire et Paul-Éric, qui m'ont toujours encouragé durant mes études, ma sœur Caroline, mon frère Pierre et mes amis pour leur soutien précieux.

Enfin, j'ai évidemment une pensée particulière pour Coline qui a su me soutenir et me supporter durant ces trois années.

Résumé

La reconnaissance d'écriture manuscrite est une composante essentielle de l'analyse de document. Une tendance actuelle de ce domaine est de passer de la reconnaissance de mots isolés à celle d'une séquence de mots. Notre travail consiste donc à proposer un système de reconnaissance de lignes de texte sans segmentation explicite de la ligne en mots. Afin de construire un modèle performant, nous intervenons à plusieurs niveaux du système de reconnaissance. Tout d'abord, nous introduisons deux méthodes de prétraitement originales : un nettoyage des images de lignes de texte et une correction locale de la ligne de base. Ensuite, nous construisons un modèle de langage optimisé pour la reconnaissance de courriers manuscrits. Puis nous proposons deux systèmes de reconnaissance à l'état de l'art fondés sur les HMM (*Hidden Markov Models*) contextuels et les réseaux de neurones récurrents BLSTM (*Bi-directional Long Short-Term Memory*). Nous optimisons nos systèmes afin de proposer une comparaison de ces deux approches. Nos systèmes sont évalués sur l'écriture cursive latine et arabe et ont été soumis à deux compétitions internationales de reconnaissance d'écriture. Enfin, en perspective de notre travail, nous présentons une stratégie de reconnaissance pour certaines chaînes de caractères hors-vocabulaire.

Abstract

Handwriting recognition is an essential component of document analysis. One of the popular trends is to go from isolated word to word sequence recognition. Our work aims to propose a text-line recognition system without explicit word segmentation. In order to build an efficient model, we intervene at different levels of the recognition system. First of all, we introduce two original preprocessing techniques : a cleaning and a local baseline correction for text-lines. Then, a language model is built and optimized for handwritten mails. Afterwards, we propose two state-of-the-art recognition systems based on contextual HMMs (*Hidden Markov Models*) and recurrent neural networks BLSTM (*Bi-directional Long Short-Term Memory*). We optimize our systems in order to give a comparison of those two approaches. Our systems are evaluated on arabic and latin cursive handwritings and have been submitted to two international handwriting recognition competitions. At last, we introduce a strategy for some out-of-vocabulary character strings recognition, as a prospect of future work.

Table des matières

Introduction générale

Nous nous intéressons ici au domaine de la lecture automatisée de l'écriture manuscrite. Son objectif principal est l'obtention d'une transcription numérique d'un texte écrit à la main.

Soixante ans après les premiers travaux, ce domaine constitue toujours un champ de recherche très actif. En effet, si la tâche de lecture peut paraître facile et instantanée pour un individu, c'est qu'elle a nécessité auparavant un long apprentissage et mobilise une très grande variété de facultés (analyse d'image, analyse du contexte, méthodes de reconnaissance globale et locale) et de connaissances (graphie, vocabulaire, linguistique). Ce panel de compétences humaines va constituer tout autant de problématiques de recherche pour la lecture automatique. En outre, la notion d'apprentissage demeure centrale dans le domaine de la reconnaissance automatisée.

Les premiers systèmes de reconnaissance de caractères imprimés (*Optical Character Recognition* - OCR) ont émergé dans les années 50. Au début des années 60, la société IBM proposa les premiers lecteurs optiques capables de retranscrire les chiffres imprimés puis les chiffres écrits à la main. L'identification des adresses postales et la lecture des chèques furent les premières applications de la reconnaissance d'écriture manuscrite. Ainsi en 1965, le service des postes américain utilisa pour la première fois une lecture automatique des adresses (nom de l'état, de la ville et code postal). Ces premières tâches ont pour spécificité de chercher à identifier un contenu très spécifique au sein donc d'un vocabulaire restreint. De plus la reconnaissance des adresses postales et des chèques peut être facilitée par la présence de champs d'information redondants (par exemple, la ville et le code postal pour les adresses). Un historique de la reconnaissance d'écriture peut être trouvé dans [Steinherz *et al.*, 1999, Plamondon et Srihari, 2000, Fujisawa, 2008].

Les travaux sur la reconnaissance d'adresses et de chèques se sont ensuite étendus à la

reconnaissance de mots et la tendance actuelle est à la reconnaissance de séquences de mots. La reconnaissance de l'écriture doit beaucoup au domaine de la reconnaissance de la parole : les techniques de reconnaissance par modèles de Markov cachés et celles de modélisation du langage y sont utilisées depuis le début des années 90. Les progrès effectués ont permis d'aborder de nouvelles problématiques. Actuellement, la numérisation des archives et le traitement automatique du courrier demeurent les deux champs d'application prédominants.

De nombreux projets sont menés pour numériser les archives publiques et privées. En France, le projet Gallica propose notamment une bibliothèque numérique issue des documents de la Bibliothèque nationale de France et de ses partenaires. Disponible en ligne depuis 1997, elle offre aujourd'hui accès à plus de 2 millions de documents. Afin de proposer une version entièrement numérique des documents (image et sa transcription), la reconnaissance du texte est nécessaire. Des travaux comme [Lavrenko *et al.*, 2004, Lladós *et al.*, 2008] sont dédiés à des documents d'archives.

Malgré la croissance des échanges numériques, le volume annuel de documents papier, manuscrits comme imprimés, traités par nombre d'administrations (sécurité sociale, caisses d'allocations, ministère des finances, INSEE) et d'entreprises (banques, assurances, fournisseurs d'énergie, opérateurs de télécommunications) justifie toujours le développement d'un traitement automatisé. Ces documents sont principalement des formulaires, des courriers, des faxes et des factures. Si les premières tâches concernaient la classification des documents, la transcription complète du document apparaît désormais comme l'objectif principal à terme. Cette volonté s'illustre entre autres à travers le projet MAURDOR (Moyens automatisés de reconnaissance de documents écrits) [1]. Initié par la DGA (Direction Générale de l'Armement) et mis en place par le LNE (Laboratoire National de métrologie et d'essais), ce projet s'intéresse au traitement automatisé de documents imprimés et manuscrits et à son évaluation. Outre la création d'une base de documents scannés et annotés, des campagnes d'évaluation sont régulièrement organisées afin de mesurer les progrès sur chacune des tâches successives. Ce projet requiert entre autres la reconnaissance d'éléments textuels. Le contexte de l'étude est plurilingue puisque les documents fournis peuvent être en français, anglais et arabe.

Le développement des surfaces tactiles a fait émerger une nouveau type de lecture

1. http ://www.maurdor-campaign.org

automatique, la reconnaissance "en-ligne" (*online handwriting recognition*), définie par opposition à la reconnaissance "hors-ligne" de documents scannés (*offline*). Un de ses enjeux est de fournir une transcription en temps réel au fil de l'écriture. La reconnaissance en-ligne est une problématique très différente du hors-ligne. Tout d'abord, elle ne nécessite pas l'ensemble des prétraitements de la reconnaissance hors-ligne puisque les coordonnées des traits sont fournies par le dispositif d'acquisition. Ensuite, l'écriture est représentée comme une séquence des coordonnées des points d'écriture, s'affranchissant ainsi des variations d'épaisseur de trait, qui interviennent dans la reconnaissance hors-ligne. Enfin, cette reconnaissance dispose des coordonnées des points d'écriture ordonnées temporellement. Si les problématiques de la lecture en-ligne et hors-ligne sont différentes, les modélisations utilisées pour la reconnaissance sont souvent proches et peuvent faire l'objet d'études croisées [Vinciarelli et Perrone, 2003]. Une base duale nommée Ironoff [Viard-Gaudin *et al.*, 1999] a notamment été créée pour pouvoir comparer les deux approches : pour chaque caractère ou mot de la base, les données en-ligne et hors-ligne sont fournies.

Si la reconnaissance de l'écriture a aujourd'hui plus de 60 ans, elle comprend encore un nombre important de questions non résolues. Actuellement, les bons niveaux de reconnaissance obtenus sur les mots isolés [Grosicki et El-Abed, 2009] conduisent les équipes de recherche à s'intéresser à l'échelle de la ligne.

L'objectif principal de cette thèse est de proposer une reconnaissance de lignes d'écriture cursive, sans segmentation explicite en mots. Face à l'irrégularité des espacements, cette approche permet de s'affranchir des incertitudes inhérentes à la segmentation. Un système de reconnaissance de lignes est une chaîne complexe de traitements. Aucune des composantes de cette chaîne ne doit être négligée pour atteindre une bonne performance de reconnaissance. Les systèmes développés pour la reconnaissance d'écriture présentent un grand nombre de paramètres. Il nous appartient de décider pour quels paramètres une optimisation est essentielle.

Pour disposer d'une chaîne complète de traitement, nous devons proposer un prétraitement adapté aux lignes. En effet, les performances des systèmes de reconnaissance sont très dépendantes de la quantité mais aussi de la qualité des données d'entrée. De plus, le passage du mot à la ligne offre la possibilité de prendre en compte les propriétés du langage en construisant un modèle de langage statistique. Ce modèle doit s'adapter à notre tâche de reconnaissance. Les quelques travaux qui se sont intéressés à la reconnaissance

des lignes utilisent tous des modèles de langage à grands vocabulaires (de 10 000 à 50 000 mots). Cependant, ces modèles nécessitent de grands corpus de texte pour estimer leurs probabilités. Notre système ne repose que sur les transcriptions de la base d'apprentissage (∼7 000 mots). Nous avons cherché à montrer l'efficacité de petits dictionnaires et modèles de langage pour des tâches spécifiques.

Un autre objectif de cette thèse est de proposer une étude comparée des deux grandes classes de systèmes de reconnaissance que sont les Modèles de Markov Cachés (*Hidden Markov Models* - HMM) et les réseaux de neurones récurrents (*Recurrrent Neural Networks* - RNN). Afin d'être en mesure de comparer le plus justement possible un réseau de neurones au reconnaisseur HMM, nous avons choisi de construire une architecture BLSTM (*Bidirectional Long-Short Term Memory*). En effet HMM et BLSTM peuvent partager la même représentation des données en entrée. Dans toutes les étapes de la reconnaissance, nous conservons le souci de développer des méthodes qui ne soient pas spécifiques à un alphabet, une langue ou une base de données. Pour s'en assurer, nous appliquons nos différentes méthodes au français principalement et à l'arabe secondairement.

Nous souhaitons également dresser un état de l'art sur les méthodes de reconnaissance par réseaux de neurones. Notre objectif est d'identifier et de comprendre les cheminements qui ont conduit à l'émergence d'architectures neuronales très complexes - BLSTM et MDLSTM (*Multidimensional Long-Short Term Memory*) - qui ont présenté les meilleurs taux de reconnaissance lors des dernières compétitions internationales [Grosicki et El-Abed, 2009, Grosicki et El-Abed, 2011, NIST, 2010, NIST, 2013a].

Afin de confronter nos résultats à ceux d'autres laboratoires et ainsi d'avoir une meilleure compréhension de nos systèmes, nous avons fait le choix de participer à plusieurs compétitions internationales de reconnaissance d'écriture manuscrite. Ce choix n'est pas anodin puisque la participation à une seule compétition représente plusieurs mois de travail.

Le premier chapitre est consacré à la présentation d'un état de l'art de la reconnaissance hors-ligne de texte manuscrit. Cette présentation suit le déroulement classique d'un système de reconnaissance : prétraitement des images, extraction de caractéristiques depuis les images, modélisations pour la reconnaissance et enfin modélisation du langage. Dans cette présentation, nous insistons sur les différences induites par le passage du mot à la ligne. L'accent est également mis sur les méthodes de reconnaissance par réseaux de

neurones récurrents qui n'ont été que récemment appliquées à la reconnaissance d'écriture manuscrite [Graves *et al.*, 2009] mais avec un grand succès. Enfin, nous effectuons une comparaison des différents systèmes complets de reconnaissance d'écriture manuscrite présentés au cours des dernières années.

Le deuxième chapitre s'intéresse plus précisément au prétraitement des images de lignes de texte. Les bases de données utilisées sont introduites en début de chapitre. Après avoir précisé les spécificités des images de lignes, nous présentons une chaîne complète de prétraitement de ligne de texte. Cette chaîne comprend deux méthodes inédites de nettoyage des images et de correction locale de la ligne de base [Morillot *et al.*, 2013c].

Le troisième chapitre développe notre modélisation HMM appliquée aux lignes de texte. L'approche par modèles HMM en contexte y est détaillée. Nous proposons également la construction d'un modèle de langage pour la reconnaissance de courriers manuscrits et évaluons son impact sur la reconnaissance. Sont données ensuite les différentes expériences d'optimisation des modèles, notamment concernant l'extraction des caractéristiques et la correction de la ligne de base. L'apport de nos prétraitements pour la reconnaissance HMM est ensuite évaluée et discutée. Les résultats obtenus lors d'une compétition internationale sur le français manuscrit sont présentés et commentés à la fin du chapitre.

Le quatrième chapitre aborde la modélisation BLSTM choisie. Nous y détaillons l'architecture du réseau, son paramétrage ainsi que la stratégie d'apprentissage. Comme pour la modélisation HMM, nous optimisons les paramètres d'extraction des caractéristiques du BLSTM. Nous mesurons ensuite les apports respectifs de nos prétraitements sur la reconnaissance par BLSTM. Les résultats obtenus lors d'une compétition internationale sur l'arabe manuscrit sont ensuite présentés et commentés. Ce chapitre propose également une comparaison des deux modélisations choisies selon différents critères : taux d'erreur dans diverses conditions et temps de traitement. Une détection de certaines entités hors-vocabulaire y est également proposée et évaluée.

Enfin, les conclusions sur les apports de nos travaux sont données dans le dernier chapitre. Cette analyse nous permet de proposer des perspectives pour des travaux futurs.

Chapitre 1

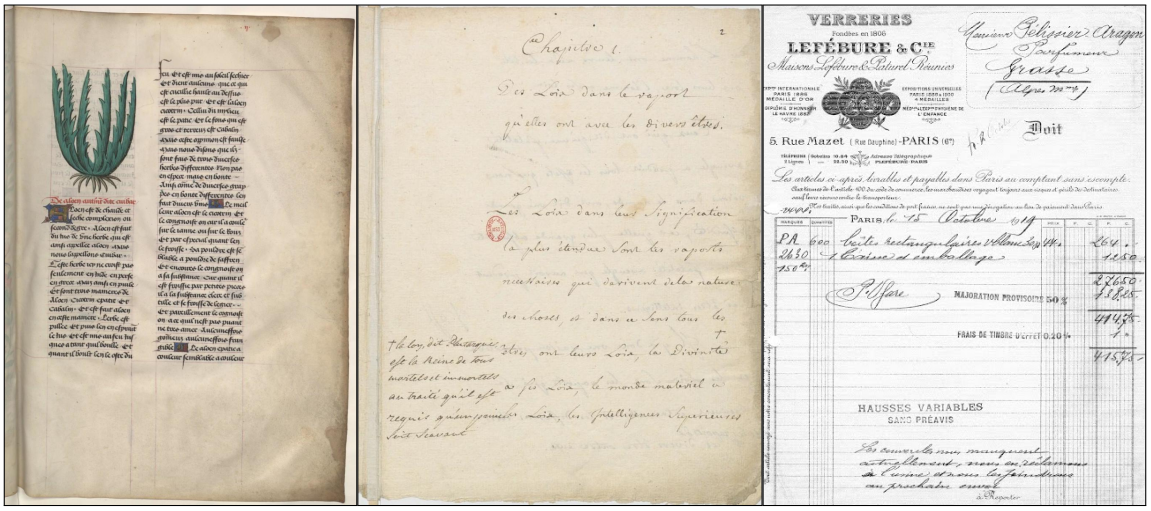
Introduction à la reconnaissance d'écriture manuscrite cursive

1.1 Introduction

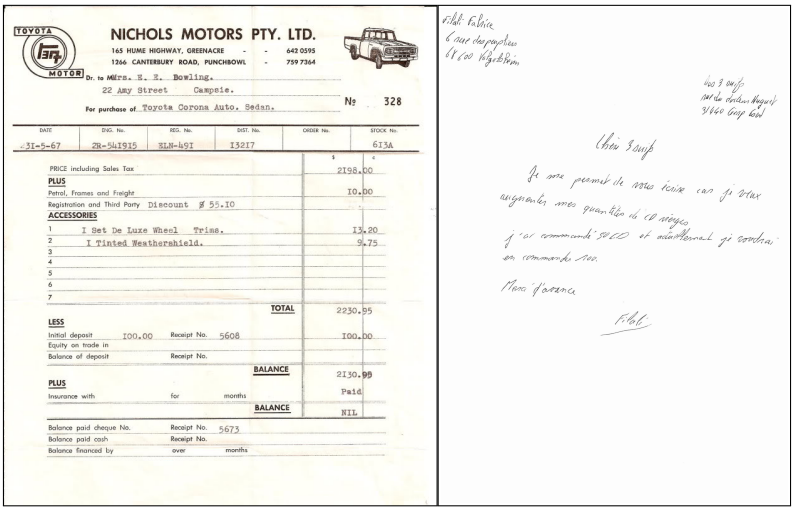
Cette thèse traite de la reconnaissance de lignes de texte manuscrit. Cette tâche se rattache au cadre plus général de l'analyse des documents scannés. L'un des objectifs principaux de ce domaine est la transcription automatique de l'ensemble de l'image du document. Les sources de documents sont très variées (document historique, courrier, factures,...) et peuvent combiner textes imprimés, textes manuscrits et éléments graphiques. Un échantillon de documents scannés est présenté en figure 1.1.

Peuvent être sollicités avant d'aboutir à cette transcription : prétraitements sur l'ensemble de l'image de la page, segmentation de l'image en zones, distinction entre zones graphiques et zones écrites, segmentation des lignes de texte et reconnaissance de la ligne. Ce processus est illustré sur la figure 1.2. Si cette logique en tâches successives est commode, de nombreuses approches attendent pour prendre des décisions définitives d'avoir obtenu des informations de plus bas niveau. La segmentation et l'identification des zones du document peut notamment bénéficier de la reconnaissance du contenu des zones.

Notre travail étant dédié au traitement des lignes de texte, déjà segmentées, nous ne détaillerons pas les méthodes de segmentation du paragraphe en lignes. Un état de l'art de cette question peut être trouvé dans [Likforman-Sulem *et al.*, 2007,Lemaitre *et al.*, 2011].



(a) (b) (c)



(d) (e)

FIGURE 1.1 – Exemples de documents : (a) livre écrit sur parchemin (XVème siècle), (b) manuscrit de Montesquieu (XVIIIème siècle), (c) facture (début du XXème siècle), (d) facture (XXème siècle), (e) courrier manuscrit (XXIème siècle)

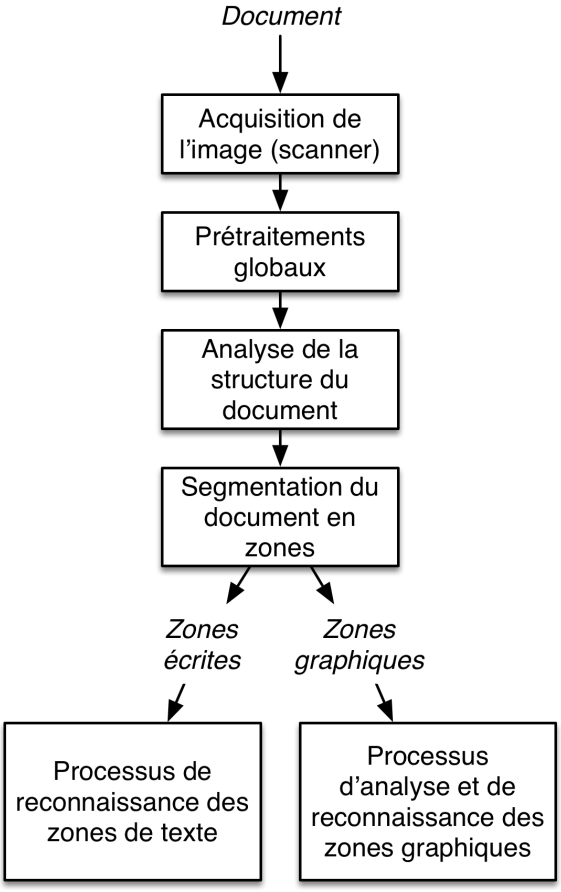


FIGURE 1.2 – Processus général d'analyse du document

Il existe de nombreux travaux sur la reconnaissance des mots manuscrits isolés [Steinherz *et al.*, 1999, Plamondon et Srihari, 2000]. Les différentes études menées sur les mots isolés ont permis d'atteindre des taux de reconnaissance très élevés sur les langue anglaise [Marti et Bunke, 2001], française [Grosicki et El-Abed, 2009] et arabe [NIST, 2010]. Le traitement automatique des lignes s'établit donc comme la poursuite logique des travaux menés sur les mots isolés [Senior et Robinson, 1998, Vinciarelli *et al.*, 2004, Frinken et Bunke, 2010, España-Boquera *et al.*, 2011]. L'intérêt pour la reconnaissance des lignes est exprimé par plusieurs acteurs, à la fois publics et industriels. Les lignes de texte peuvent donc être issues d'une grande variété de documents scannés.

La reconnaissance de l'écriture est principalement abordée comme une tâche d'apprentissage automatique (*machine learning*) où l'objectif est l'étiquetage supervisé des images de texte. Dans cette approche, le système de reconnaissance nécessite une base de données d'apprentissage qui comporte un certain nombre d'images de lignes de texte et leur transcription vérité-terrain. Grâce à cette base, le système de reconnaissance va "apprendre" à associer l'image du texte à sa transcription numérique. Une fois cet apprentissage effectué, une image inédite de texte (i.e. que le système n'a jamais rencontrée pendant l'apprentissage) pourra être présentée au système qui proposera une transcription. Cette phase s'appelle le décodage.

Deux approches peuvent être envisagées pour lire des lignes de texte. Une approche dite "avec segmentation" consiste à découper la ligne en mots et à reconnaître chaque mot comme dans [Senior et Robinson, 1998]. Une autre approche, "sans segmentation", consiste à effectuer un découpage des mots implicitement, au moment de l'apprentissage et de la reconnaissance, de la même manière que l'on peut reconnaître un mot sans le découper au préalable en caractères. Cette approche "sans segmentation" a été notamment mise en oeuvre par [Vinciarelli *et al.*, 2004, Frinken et Bunke, 2010, España-Boquera *et al.*, 2011]. La segmentation est sujette à erreur car les espacements entre les mots manuscrits sont très variables. Nous avons fait le choix de ne pas segmenter les lignes en mots. La segmentation en mots est plus appropriée dans le cas de documents imprimés où l'espacement est régulier et suffisant.

Dans ce chapitre, nous tenterons d'exposer les spécificités du traitement des lignes de texte par rapport celui des mots isolés. Nous détaillons l'ensemble des étapes classiques de la reconnaissance de texte manuscrit comme représenté sur la figure 1.3. Comme pour

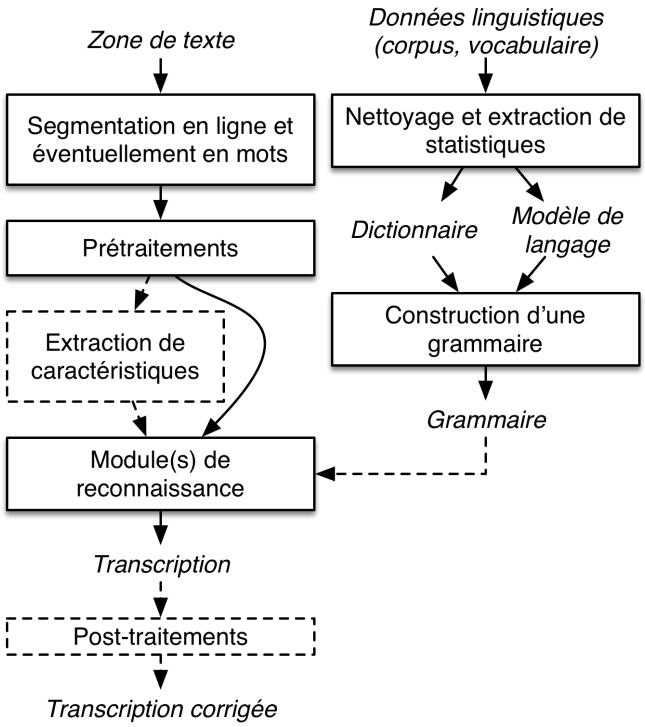


FIGURE 1.3 – Processus général de traitement du texte manuscrit. Les étapes subsidiaires sont indiquées en pointillés.

beaucoup de tâches d'analyse de données réelles, la reconnaissance de l'écriture manuscrite nécessite des étapes de prétraitement des images. Les objectifs du prétraitement sont de nettoyer les données et d'en réduire la variabilité. Nous en dresserons un état de l'art dans la section 1.2 en détaillant précisément les problématiques induites par la ligne. Ensuite, les systèmes de reconnaissance peuvent être appris directement sur les valeurs des pixels ou sur des caractéristiques extraites depuis les images. Nous détaillons donc les différentes méthodes d'extraction ainsi que les différentes caractéristiques possibles dans la section 1.3.

Nous donnons la formalisation mathématique du problème de la reconnaissance des lignes de texte dans la section 1.4, afin de pouvoir évoquer les différentes modélisations possibles de l'écriture dans la section 1.5. La reconnaissance de ligne de texte offre la possibilité d'introduire des connaissances linguistiques dans les systèmes. Ces problématiques de modélisation du langage sont abordées dans la section 1.6. Les métriques d'évaluation de la reconnaissance sont précisées dans la section 1.7. Enfin, nous donnons un état de l'art comparé de systèmes complets de reconnaissance de ligne dans la section 1.8. Ces systèmes ont participé récemment à des compétitions de reconnaissance d'écriture manuscrite.

1.2 Prétraitement des images

Les données d'écriture peuvent inclure des artefacts dus à la qualité du document original, à la numérisation ou aux étapes précédentes de traitement (segmentation en ligne).

Même si certains champs d'application requièrent des approches spécifiques, il existe un panel de prétraitements largement adoptés par la communauté du traitement automatique du document. Lors du développement ou du paramétrage des prétraitements, il est important de s'assurer de leur relative généralité vis-à-vis d'un assez grand échantillon de documents. Établir des prétraitements spécifiques à certaines données peut certes améliorer grandement la reconnaissance mais ces résultats pourront difficilement être étendus dès qu'on s'éloigne du cadre précis de l'étude (autres bases de données, alphabets ou scripteurs).

Nous nous intéressons surtout aux modes d'écriture cursifs tels que les alphabets latins et arabes. L'écriture latine comme l'écriture arabe peut se décomposer en trois zones principales : les hampes, la zone centrale et les jambages. Idéalement des zones peuvent être séparées par des droites (figure 1.4). Les lignes délimitant la zone centrale sont

FIGURE 1.4 – Zones de l'écriture (image extraite de la base RIMES)

appelées "ligne de base basse" et "ligne de base haute". L'extraction des lignes de base est généralement incluse dans le prétraitement car elle permet certaines normalisations (correction de la ligne de base, normalisation de la hauteur de l'écriture).

Une attention particulière doit être portée à la justification des prétraitements. Toute manipulation de l'image modifie le signal et détruit donc de l'information. Une amélioration pour l'œil humain ne trouve pas nécessairement sa justification pour le système de reconnaissance.

Nous présentons maintenant les prétraitements les plus couramment employés dans les systèmes de reconnaissance :

— le nettoyage de l'image
— la binarisation
— la correction de la ligne de base de l'écriture
— la correction de l'inclinaison des caractères (en anglais *slant*)
— l'extraction des lignes de base
— la normalisation de l'écriture

Le nettoyage de l'image est une notion qui rassemble généralement tous les prétraitements conduisant à faire ressortir le signal considéré par rapport à son arrière plan. Le plus souvent, les artefacts à nettoyer sont dus à la dégradation du document original ou au processus d'acquisition de l'image (scanner). Les documents historiques sont particulièrement concernés et nécessitent souvent plusieurs étapes de nettoyage. Les principaux phénomènes observés sont la dégradation du document (trous, changement de teinte, taches de diverses origines) qui se traduit par un fond hétérogène, la présence de structures autres que le texte (guide-lignes, quadrillage, tableaux) et la possibilité de distinguer les caractères du verso par transparence (*bleed-through*). L'élimination des éléments du verso s'effectue généralement par des filtrages [Drira *et al.*, 2006], filtrages à moyenne non locale et filtrage à base de variation totale [Likforman-Sulem *et al.*, 2011]. Un état de l'art

sur les prétraitements de documents anciens peut être trouvé dans [Likforman-Sulem, 2003].

Même si le nettoyage est souvent conduit sur l'ensemble du document, des étapes de nettoyage peuvent être conduites sur les vignettes de lignes. Dans le cadre de nos travaux, nous sommes confrontés à des vignettes présentant des composantes des lignes adjacentes. Ces composantes sont donc assimilables à du bruit. Le fond peut également être hétérogène. Nous abordons ces questions dans 2.3. Des guide-lignes ou autres droites peuvent apparaître également dans les images de texte. Leur suppression a été abordée par [Abd-Almageed *et al.*, 2009, Kumar et Doermann, 2011]. Parallèlement, la suppression d'autres éléments graphiques par filtrage de Kalmann a été proposée par [Likforman-Sulem, 1998]. Dans [Arvind *et al.*, 2007], une analyse des profils de projection permet la suppression des guide-lignes et la restauration des caractères dégradés.

La binarisation consiste à réduire la dynamique en niveaux de gris de l'image à deux classes : les pixels noirs de l'écriture (valeur=0) et l'arrière plan de l'image ramené au blanc (valeur=1). Cette binarisation peut être effectuée sur l'ensemble de l'image, en recherchant un seuil global [Otsu, 1979], ou localement en prenant des décisions locales. En effet, dans le cas d'archives, par exemple, la teinte de l'arrière plan peut varier au sein d'un même document. Les approches locales peuvent reposer sur des filtrages de Wiener [Gatos *et al.*, 2006] ou des approches multi-échelles [Lelore et Bouchara, 2011]. Une approche hybride globale/locale a été proposée par [Kavallieratou et Stamatatos, 2006]. Les dernières compétitions organisées sur les documents historiques (DIBCO - Document Image Binarization Contest) soulignent l'intérêt de combiner plusieurs méthodes [Pratikakis *et al.*, 2011].

Cette étape de binarisation n'est pas toujours nécessaire pour la reconnaissance de l'écriture. En effet, supprimer la dynamique des niveaux de gris fait perdre de l'information. En revanche cette étape peut être utile pour effectuer des analyses en composantes connexes comme il est détaillé dans la section 2.3. Dans le cas de bases de données propres, une binarisation simple est suffisante (voir section 2.2).

Correction de la ligne de base de l'écriture La ligne de base (basse) est la courbe imaginaire sur laquelle repose l'écriture. La correction de cette ligne de base est une étape primordiale dans l'optique d'un système de reconnaissance robuste. En effet, l'extraction

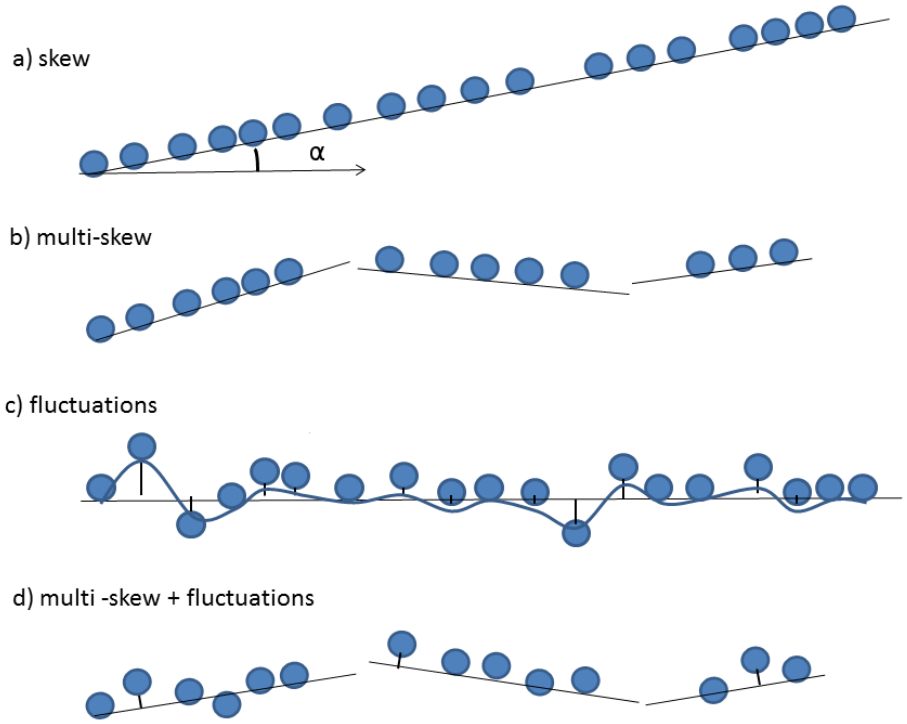


FIGURE 1.5 – (a) ligne de base avec une pente de valeur $\alpha \neq 0$. (b) ligne de base avec plusieurs valeurs de pente. (c) ligne de base à la position fluctuante. (d) plusieurs valeurs de pente et ligne de base fluctuante.

de caractéristiques peut comprendre des caractéristiques de densité estimées dans les trois régions de l'écriture (zone où se situent les hampes, zone centrale, zone où se situent les jambages). Des lignes d'écriture penchées perturbent donc la définition de ces zones et la valeur des caractéristiques extraites. Cet effet est d'autant plus sensible que l'on travaille directement au niveau des lignes de texte. La notion de ligne de base est aussi intrinsèquement liée à la pente de l'écriture (en anglais *slope* ou *skew*) : corriger la ligne de base permet de corriger la pente locale de l'écriture.

La correction de la pente (en anglais *deskew*) a été principalement abordée au niveau du mot : elle consiste à estimer un angle sur l'image de mot et à faire pivoter le mot de cette valeur d'angle. Les estimations de pente au niveau mot s'appuient souvent sur des régressions sur des ensembles des points : ces points peuvent être les centres de masses des composantes du mot [M. Blumenstein *et al.*, 2002, Bozinovic et Srihari, 1989], ou bien les points les plus bas du mot en excluant des jambages [Senior et Robinson, 1998, Vinciarelli

et Luettin, 2001]. Ces approches au niveau mot peuvent être étendues au niveau ligne quand les lignes considérées sont rectilignes avec une unique valeur d'inclinaison (figure 1.5a) [Marti et Bunke, 2000]. Cette hypothèse est notamment valide dans le cas de formulaires ou de documents incluant un guide-lignes. Pour les systèmes découpant les lignes en mots, une estimation et une correction de la pente par morceaux (figure 1.5c) est possible mais elle ne prendra pas en compte les variations internes au mot : des estimations par morceaux de la ligne de base ont notamment été menées sur la base RIMES [Lemaitre *et al.*, 2009] en partant d'une première estimation de la ligne de base sur une image à basse résolution. Dans le cadre de la reconnaissance de texte imprimé (OCR), des méthodes d'estimation par transformée de Hough ont été utilisées avec succès [Amin et Fischer, 2000].

Néanmoins, l'approximation de la ligne de base par morceaux n'est pas toujours valable dans le cas d'une écriture libre. Récemment, une méthode a abordé la correction locale de la ligne de base par morceaux [España-Boquera *et al.*, 2011] en partant des pixels de contours et en utilisant un classement par un perceptron multi-couches. La correction est effectuée par morceaux en ayant découpé au préalable la ligne en blocs d'écriture (éléments d'écritures séparés par des blancs). Nous détaillons cette approche car elle rend compte de la difficulté à estimer précisément la position de la ligne de base tout en prenant en compte les espacements et en éliminant les creux locaux dus à la forme des caractères.

Tout d'abord, cette méthode applique une détection de contours à l'image de la ligne. Ensuite, les points du contour sont regroupés en lignes brisées en suivant un critère de proximité : si deux pixels situés sur des colonnes adjacentes ont des coordonnées verticales proches (< 3 pixels), alors ils sont considérés comme appartenant à la même ligne. Cette étape va lisser les contours et les découper en segments.

Ensuite, les maxima locaux des contours supérieurs et les minima locaux des contours inférieurs sont calculés. Ces pixels sont des candidats potentiels pour les lignes de base basse et haute. Ceci permet d'éviter de sélectionner les différents creux des contours de l'écriture. Puis, un classifieur de type perceptron multi-couches est appliqué sur les pixels sélectionnées afin de décider si les pixels appartiennent à la ligne de base basse ou non. L'entrée du réseau correspond à une fenêtre centrée sur le pixel à classer. Le MLP est entraîné sur une base de 1000 images. L'étiquetage de cette base est semi-automatique.

Après détection des points de la ligne de base (en vert sur la figure 1.6) grâce au MLP,

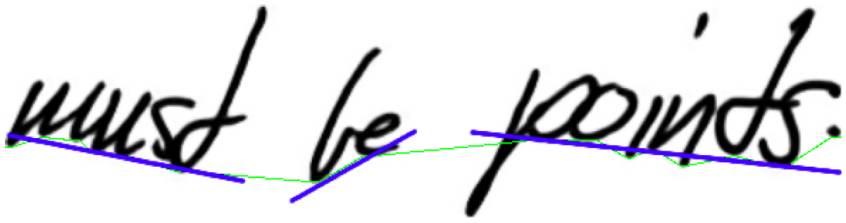


FIGURE 1.6 – Correction de la pente locale par la méthode d'España-Boquera (extrait de [España-Boquera *et al.*, 2011])

l'image de ligne est partagée en segments pour appliquer la correction de la ligne de base par morceaux (en bleu sur la figure 1.6). La segmentation est effectuée en construisant l'histogramme des projections verticales sur l'axe horizontal. Cet histogramme permet d'estimer l'espacement moyen entre les zones d'écriture. Cette valeur moyenne sera choisie comme seuil pour la segmentation de l'écriture en blocs.

La méthode présentée ci-dessus nécessite donc une détection de contours, l'apprentissage d'un MLP sur une base étiquetée et une segmentation des zones d'écriture. Elle permet une correction de la ligne de base par morceaux. Cette correction ne prend donc pas en compte les variations de position de la ligne de base au sein des mots. Nous proposons une méthode inédite de correction locale de la ligne de base dans la section 2.4. Cette nouvelle méthode permet la correction des variations à l'échelle du mot et ne nécessite ni segmentation, ni apprentissage statistique.

Correction de l'inclinaison de l'écriture (en anglais *slant*) Les différents styles d'écriture amènent à avoir des écritures plus ou moins inclinées. Au delà de la variabilité entre les scripteurs, l'inclinaison de l'écriture peut avoir un impact non négligeable sur l'extraction des caractéristiques, détaillée dans la section suivante 1.3.

Principalement, les méthodes de correction de l'inclinaison reposent sur l'estimation d'un angle et sa correction par une rotation [Vinciarelli et Luettin, 2001]. Contrairement à la correction de la ligne de base, les variations de l'inclinaison sont minimes au sein de l'écriture d'un scripteur. Il n'est donc pas impératif d'effectuer des estimations locales de l'angle d'inclinaison. Certaines approches envisagent cependant cette question [Uchida *et al.*, 2001, Bertolami *et al.*, 2007].

L'extraction des lignes de base demande au préalable d'avoir corrigé les fluctuations de la position de la ligne de base basse. L'extraction peut être réalisée par l'analyse de l'histogramme de projection des valeurs de pixel sur l'axe vertical comme présenté dans [M. Blumenstein *et al.*, 2002, Vinciarelli et Luettin, 2001]. D'autres méthodes s'appuient sur l'extraction des contours, la minimisation de l'entropie ou bien l'extraction du squelette de l'écriture. Un récapitulatif et une comparaison des méthodes ont été réalisés par [Maddouri *et al.*, 2008].

La normalisation de l'écriture est un ensemble de méthodes visant à réduire la variabilité entre les scripteurs. Elles abordent principalement les questions de la normalisation de l'épaisseur des traits, de l'intensité des niveaux de gris [Oprean *et al.*, 2013a], de la longueur des caractères et de la hauteur de l'écriture. Certaines approches essaient de normaliser les hauteurs au sein des trois zones principales (la zone où se situent les hampes, la zone centrale et la zone où se situent les jambages) [Bengio *et al.*, 1995, España-Boquera *et al.*, 2011]. Récemment, un système inspiré par la normalisation des chiffres a abordé l'ensemble de ces questions [Kozielski *et al.*, 2012] : les approches reposent sur le calcul des moments de l'image et des outils de morphologie mathématique.

1.3 Extraction des caractéristiques

1.3.1 Stratégies d'extraction

À partir des images de lignes nettoyées, deux représentations de l'information sont possibles. La première, la plus fréquente, consiste à extraire des caractéristiques depuis l'image. Cela permet de réduire la taille des données et de disposer de caractéristiques pertinentes pour la reconnaissance. Néanmoins, il est possible bâtir un système de reconnaissance directement sur les valeurs des pixels [LeCun *et al.*, 1998, Graves et Schmidhuber, 2008]. L'extraction des caractéristiques est alors implicite comme dans le cas des réseaux profonds que nous évoquerons dans la section 1.5.2.d.

Pour extraire les caractéristiques depuis les images, plusieurs stratégies peuvent être appliquées. Il est possible de segmenter l'écriture en graphèmes ou en caractères et d'extraire les caractéristiques de chaque élément. Si cette approche trouvait sa justification

pour la reconnaissance de symboles isolés, elle montre ses limites face aux lignes de texte. La probabilité de faire des erreurs de segmentation augmente avec la taille de la séquence. En revanche, cette approche est beaucoup plus utilisée dans le domaine de la reconnaissance "en-ligne" [Boubaker *et al.*, 2010]. En effet, l'ordre séquentiel des points fournis par l'acquisition en-ligne permet de séparer les traits d'écriture qui se croisent et rend la segmentation en graphèmes plus facile. La deuxième approche, majoritaire, consiste à utiliser des fenêtres glissantes chevauchantes (ou recouvrantes).

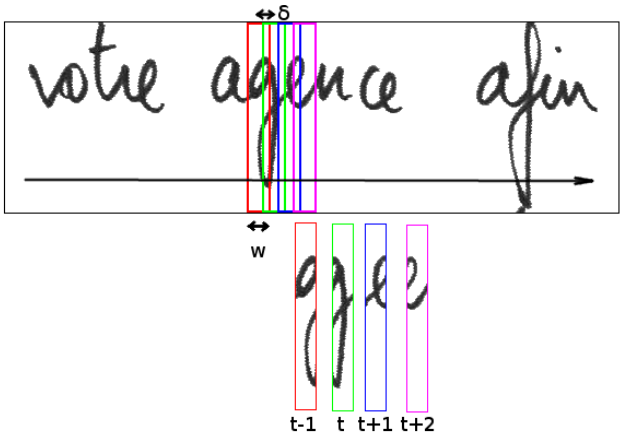


FIGURE 1.7 – Extraction des caractéristiques par fenêtre glissante

L'approche par fenêtres glissantes chevauchantes (*overlapping sliding window*) permet de s'affranchir de la question de la segmentation : la fenêtre rectangulaire, représentée sur la figure 1.7, parcourt l'image de gauche à droite (pour une écriture latine). Ce procédé effectue une segmentation implicite : à chaque instant t, un vecteur de caractéristiques va être extrait au sein de la fenêtre d'analyse (*frame*). L'extraction est définie par deux paramètres : la largeur de fenêtre w et son pas de déplacement δ. Comme nous l'avons précisé dans la section 1.4, la ligne de texte est ainsi représentée par une séquence X de vecteurs x_t : $X = (x_1, x_2, ..., x_T)$.

Certains systèmes utilisent des fenêtres à la hauteur variable [Vinciarelli *et al.*, 2004] pour s'adapter à la taille des caractères. D'autres méthodes proposent des fenêtres penchées pour s'adapter à l'inclinaison de l'écriture [Al-Hajj-Mohamad *et al.*, 2009]. En effet, celle ci peut conduire à avoir plusieurs caractères mélangés au sein de la même fenêtre et ainsi perturber les étapes d'apprentissage et de décodage des systèmes de reconnaissance.

1.3.2 Variété des caractéristiques

La communauté de la reconnaissance de l'écriture manuscrite ne dispose pas de caractéristiques communes uniformisées, comme c'est le cas par exemple pour le domaine de la reconnaissance de la parole avec les MFCC (*Mel-frequency cepstral coefficients*) introduits par [Davis et Mermelstein, 1980]. Chaque système exploite un ensemble de caractéristiques qui lui est propre. Le choix des caractéristiques est lié à celui de la méthode d'extraction. Principalement, ces caractéristiques peuvent être classées en trois catégories :

— les caractéristiques statistiques décrivent des statistiques issues du dénombrement des pixels d'écriture. Ces caractéristiques peuvent décrire notamment les densités de pixel d'écriture dans la fenêtre d'analyse. Dans [Vinciarelli *et al.*, 2004] notamment, la fenêtre est divisée en 16 cellules $(4 * 4)$ et les pixels noirs sont dénombrés en valeur relative, fournissant ainsi un vecteur de 16 valeurs.

— les caractéristiques géométriques décrivent les configurations des pixels. Ce sont des caractéristiques dites de "haut-niveau" car elles ne s'intéressent pas aux valeurs des pixels mais aux motifs formés par ces pixels. Les caractéristiques géométriques sont plus sensibles au bruit que les caractéristiques statistiques. En effet, la présence de pixels de bruit peut changer complètement la configuration géométrique des traits. Les positions des contours de l'écriture et le nombre de transitions entre l'écriture et l'arrière-plan sont utilisées par [Marti et Bunke, 2001]. Dans les travaux de [Al-Hajj-Mohamad *et al.*, 2005,Al-Hajj-Mohamad *et al.*, 2009], des configurations locales de concavité sont estimées.

— les caractéristiques directionnelles rendent compte de l'orientation des traits de l'écriture. L'approche des descripteurs SIFT (*Scale Invariant Feature Transform*) permet notamment d'extraire les différentes orientations dans l'image à partir des points d'intérêt de l'image. L'intérêt de cette approche est que l'extraction est indépendante de l'échelle [Lowe, 1999]. L'extraction des directions peut se faire globalement sur la fenêtre ou localement sur des cellules [Rodríguez-Serrano et Perronnin, 2008].

Face à la grande variété de caractéristiques disponibles, des études ont été menées pour réduire le nombre de caractéristiques en utilisant des analyses en composantes principales

(*Principal Component Analysis* - PCA) [Fischer et Bunke, 2009] ou par des méthodes d'analyse discriminante (*Multiple Discriminant Analysis* - MDA) [Schlapbach *et al.*, 2005]. L'intérêt d'utiliser des ensembles mêlant caractéristiques statistiques et géométriques a été montré par [De Oliveira *et al.*, 2002].

Nous utiliserons les caractéristiques définies par [Al-Hajj-Mohamad *et al.*, 2005,Al-Hajj-Mohamad *et al.*, 2009] qui ont montré leur efficacité sur les écritures arabes, françaises et anglaises [Bianne-Bernard *et al.*, 2011]. Il s'agit de caractéristiques pour certaines statistiques et pour d'autres géométriques. Ces caractéristiques nécessitent l'estimation de lignes de base basse et haute de l'écriture. Chaque fenêtre est divisée en 20 cellules. Ce procédé permet aux caractéristiques d'être indépendantes de la hauteur des images sans effectuer un sous-échantillonnage qui peut induire une perte d'information.

Le nombre des caractéristiques dépend de la largeur de la fenêtre d'extraction w :

— f_1 : densité des pixels d'écriture dans la fenêtre ;
— f_2 : nombre de transitions entre l'écriture et l'arrière plan ;
— f_3 : distance normalisée entre le centre de gravité de la fenêtre courante et de la fenêtre précédente ;
— f_4 : distance normalisée entre le centre de gravité et la ligne de base basse ;
— f_5 : densité des pixels d'écriture au-dessus de la ligne de base haute ;
— f_6 : densité des pixels d'écriture en-dessous de la ligne de base basse ;
— f_7 : nombre de transitions écriture/arrière plan au dessus de la ligne de base basse ;
— f_8 : position du centre de gravité (avant la ligne de base basse, zone centrale, au dessus de la ligne de base haute) ;
— f_9 à f_{11} : caractéristiques de configurations locales (concavités) relatives à l'ensemble de la fenêtre ;
— f_{12} à f_{20} : caractéristiques de configurations locales relatives aux pixels entre les lignes de base haute et basse ;
— f_{21} à f_{21+w-1} : densité des pixels d'écriture dans chaque colonne j de la fenêtre.

Dans certaines expériences détaillées dans le chapitre 4, nous avons également utilisé des caractéristiques directionnelles, en utilisant l'histogramme des gradients. Pour chaque fenêtre d'extraction, la sous-image correspondante est dérivée grâce à un filtre de Sobel selon les deux directions horizontale (h) et verticale (v). Les deux images résultantes

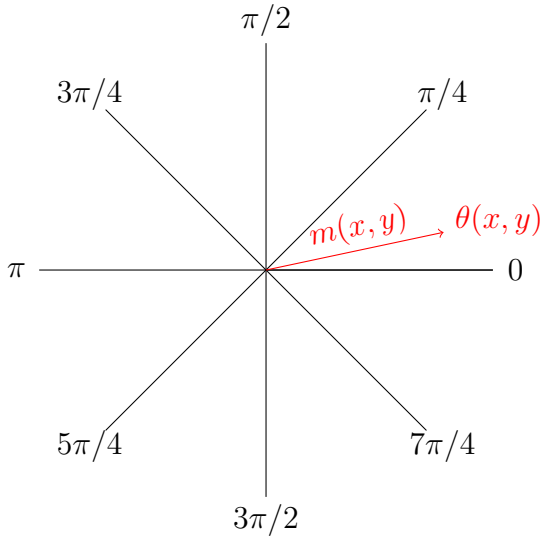


FIGURE 1.8 – Calcul de l'histogramme des valeurs de gradient pour l'extraction des caractéristiques directionnelles

seront notées $G_h(x, y)$ et $G_v(x, y)$. Pour chaque pixel (x, y) de la fenêtre d'extraction, la magnitude du gradient $G(x, y)$ et sa direction $\theta(x, y)$ sont estimées par un filtrage de Sobel :

$$G(x, y) = \sqrt{G_h(x, y)^2 + G_v(x, y)^2}$$
$$\theta(x, y) = \arctan(\frac{G_h(x, y)}{G_v(x, y)})$$

Pour pouvoir construire un histogramme des directions, le continuum des valeurs d'angles est discrétisé en huit classes comme le montre la figure 1.8. Chaque valeur d'angle $\theta(x, y)$ se trouve donc entre deux de ces huit classes. Ainsi pour chaque pixel (x, y) de l'image, la magnitude $m(x, y)$ de l'angle estimé $\theta(x, y)$ est répartie entre les deux valeurs des classes l'encadrant. Cette répartition se fait proportionnellement à la proximité avec les classes. Ainsi un histogramme des huit directions est construit en prenant en compte la magnitude du gradient. Les huit valeurs de cet histogramme constituent les caractéristiques f_{21+w} à f_{29+w}. Le détail du calcul de ces caractéristiques est précisé dans [Bianne-Bernard, 2011, Oprean *et al.*, 2013a].

L'extraction est réalisée à partir d'une fenêtre glissante avec recouvrement qui s'affranchit des différentes hauteurs d'images en utilisant un nombre fixe de cellules d'analyse.

1.4 Formalisme

Notre objectif est donner la transcription en chaînes de symboles numériques d'une ligne de texte manuscrite. La ligne de texte peut être envisagée comme une séquence temporelle orientée selon la direction de l'écriture (de gauche à droite pour l'alphabet latin). Ainsi on peut définir une séquence d'observations $X = (x_1, x_2, ..., x_T)$ extraites depuis une image de ligne. Chaque observation x_t peut correspondre à une colonne de pixels de l'image ou bien à un vecteur de caractéristiques extraites depuis l'image.

Étant donnée cette séquence d'observations X, l'objectif est d'obtenir la séquence de mots $\hat{W} = (w_1, w_2, ..., w_n)$ la plus probable en maximisant la probabilité a posteriori :

$$\hat{W} = \arg\max_{W} P(W|X)$$

où W appartient à l'ensemble des séquences de mots construites sur le vocabulaire V donné. Ce vocabulaire peut comprendre des signes de ponctuation. La loi de Bayes nous permet de séparer le modèle en deux contributions distinctes : celle du modèle optique (équivalente à la probabilité acoustique dans le domaine de la parole) et celle de la probabilité a priori de la chaînes de mots, que nous appellerons modèle linguistique. En résumé, pour une séquence de vecteurs d'observations X donnée, nous cherchons à estimer la séquence de mots la plus probable $\hat{W}$ ainsi :

$$\hat{W} = \arg\max_{W} P_{optique}(X|W) P_{grammaire}(W)$$

La probabilité optique $P_{optique}(X|W)$ pourra être fournie par le système de reconnaissance. Dans le cas particulier d'une reconnaissance de mots isolés, la probabilité grammaticale $P_{grammaire}(W)$ se ramène à la probabilité d'un mot [1]. Un des intérêts du passage de la reconnaissance du niveau mot au niveau ligne est d'offrir la possibilité de modéliser les probabilités de successions des mots. La probabilité grammaticale de la séquence de mots W, $P_{grammaire}(W)$ sera alors estimée grâce à un modèle de langage (*language model* - LM) (Section 1.6).

1. La dénomination "grammatical" est ici un abus de langage car dans nos travaux, nous ne définissons pas de classes grammaticales. Cependant nous l'utilisons pour conserver des notations cohérentes avec l'ensemble de l'état de l'art.

1.5 Méthodes de reconnaissance

1.5.1 Reconnaissance par Modèles de Markov cachés

Les modèles de Markov cachés (HMM : Hidden Markov Models) sont des processus markoviens introduits dans les années 60 par Leonard E. Baum. Ils sont notamment utilisés pour la modélisation et la reconnaissance de séquences temporelles : reconnaissance de la parole et de l'écriture, reconnaissance des gestes [Cappé *et al.*, 2005].

Un modèle de Markov caché est constitué d'un ensemble d'états et de leurs transitions. L'architecture comprend une chaîne séquentielle d'états qui sont nommés "états cachés" car l'observateur n'a pas accès à leurs valeurs. La séquence des observations est générée par les états cachés.

Les HMM peuvent être de type discret ou continu. Un modèle HMM discret est défini par les paramètres suivants :

— les N états de la couche cachée $s_1, s_1, ..., s_N$
— la matrice de probabilités $A = \{a_{i,j}\}$, où $a_{i,j}$ est la probabilité de transition de l'état s_i à s_j
— le vecteur de probabilités $B = \{b_j(k)\}$, où $b_j(k)$ est la probabilité d'émettre le symbole k étant dans l'état s_j
— le vecteur de probabilités $\Pi = \{\pi_i\}$ où π_i est la probabilité d'être initialement à l'état s_i

On notera également x_t et q_t respectivement l'observation et l'état caché au temps t.

Les HMM discrets sont utilisés avec succès dans le domaine de la reconnaissance de l'écriture manuscrite [El-Yacoubi *et al.*, 1999, Plamondon et Srihari, 2000]. Les caractères sont représentés par des séquences d'états HMM. Le nombre d'état par caractère peut être fixé ou être ajusté pour mieux correspondre au caractère considéré [Schambach, 2003]. Généralement, les modèles sont appris à partir de séquences de caractéristiques extraites par une fenêtre glissante sur l'image de texte.

La plupart des systèmes de reconnaissance de séquences temporelles par HMM utilisent une topologie de Bakis : seules les connexions gauche-droite sont autorisées ($a_{i,j} = 0$ si $i > j$). Le modèle peut soit rester dans l'état courant, soit évoluer vers des états suivants mais ne peut pas revenir en arrière. Le fait de pouvoir rester plusieurs instants sur le même état permet de modéliser les distorsion non-linéaires qui apparaissent dans l'écriture manuscrite. Un ordre de HMM supérieur à 2 permet d'introduire des sauts d'états dans la modélisation.

Dans un HMM continu, les probabilités d'émission b_j sont souvent modélisées par des sommes pondérées de distributions gaussiennes (Gaussian Mixture Model - GMM) :

$$b_j = \sum_{m=1}^{N_G} c_{jm} \mathcal{N}(x_t, \mu_{jm}, \Sigma_{jm})$$

où $\mathcal{N}(., \mu_{jm}, \Sigma_{jm})$ est une distribution gaussienne de moyenne μ_{jm}, de matrice de covariance Σ_{jm} et N_G est le nombre de gaussiennes du mélange. c_{jm} est le poids de la m-ième composante du mélange. Ces HMM sont dits continus car la probabilité d'émission est remplacée par une densité continue. Les HMM continus sont les plus largement utilisés pour la reconnaissance d'écriture manuscrite [Bunke *et al.*, 1995].

Face au nombre important de paramètres à estimer, certains privilégient une approche dite "semi-continue" où les états partagent les mêmes gaussiennes. Seuls les poids des composantes diffèrent alors. Certains systèmes préfèrent utiliser des distributions de Bernoulli pour modéliser les émissions [Giménez et Juan, 2009]. Ceci se justifie dans le cas d'images binaires ou bien lorsque les valeurs des caractéristiques extraites sont discrètes.

Principalement, deux stratégies peuvent être employées pour modéliser l'écriture manuscrite à l'aide des HMM : avec ou sans segmentation explicite. Ce choix est lié au type d'extraction de caractéristiques choisi.

La stratégie à segmentation explicite consiste à découper les mots en graphèmes et à extraire les caractéristiques depuis ces parties du mot. Ces graphèmes peuvent être des caractères entiers ou seulement des parties. L'avantage présenté par cette méthode est qu'elle permet d'avoir un étiquetage de l'image pour chaque fenêtre de caractéristiques. Ainsi le HMM effectue un classement, graphème par graphème. En revanche cette mé-

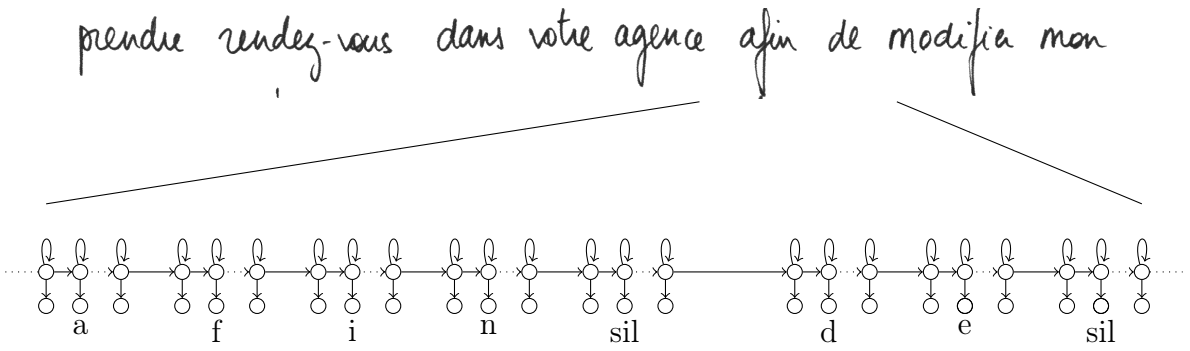


FIGURE 1.9 – Exemple de modélisation HMM d'un ligne de texte (image extraite de la base RIMES)

thode est sensible aux erreurs de segmentation. Or la segmentation des mots de l'écriture manuscrite cursive est une tâche très complexe [Lu et Shridhar, 1996], voire impossible dans certains cas.

Dans le cas d'une stratégie à segmentation implicite, aucune décision de découpe n'intervient. La stratégie d'analyse par fenêtre glissante retenue n'utilise pas de segmentation explicite des lignes en mots. Cette dernière est obtenue implicitement au cours de l'apprentissage comme du décodage. Les modèles de mots sont obtenus par la concaténation des modèles de caractères le composant. Cette stratégie, plus naturelle, est aussi la plus largement employée pour la reconnaissance de mots isolés et obtient les meilleures performances [Grosicki et El-Abed, 2009, NIST, 2010]. Son principal attrait est de ne pas nécessiter de module de segmentation et évite donc les erreurs. De plus, elle exploite pleinement les possibilités des algorithmes forward-backward pour l'étiquetage des séquences temporelles.

Cette distinction entre segmentation explicite et implicite intervient également au niveau ligne. En effet il est possible de segmenter la ligne en mots et d'effectuer une reconnaissance mot par mot. L'autre démarche consiste à considérer la ligne comme la séquence temporelle d'entrée du système. Les lignes sont alors modélisées à partir de la concaténation des modèles de mots séparés par des espaces (modèle du silence) comme illustré par la figure 1.9. La communauté est également partagée sur la question de segmenter ou non la ligne en mots. Nous reviendrons sur cette question dans la section 1.8.

FIGURE 1.10 – Différentes ligatures du 'e' selon le contexte

Des modèles de Markov bi-dimensionnels (*Pseudo-* ou *Planar-HMM* - PHMM) ont également été appliqués à la reconnaissance d'écriture [Gilloux, 1994, Belaïd et Saon, 1997]. Les PHMM diffèrent des HMM du fait que leurs probabilités d'observation dans chaque état sont données par un HMM secondaire. Ainsi, le HMM principal analyse l'image selon la direction horizontale - celle de l'écriture - tandis que le modèle secondaire décrit l'axe vertical.

Apprentissage et décodage Généralement, la probabilité $P_{optique}(X|W)$ d'observer une séquence X de vecteurs de caractéristiques en fonction de la séquence de mots W est estimée lors de l'apprentissage par l'algorithme de Baum-Welch et par l'algorithme de Viterbi lors du décodage. Ces algorithmes reposent tous deux sur l'algorithme forward-backward [Rabiner, 1989].

Modèles contextuels des caractères Pour tenir compte des déformations possibles d'un caractère liées à son contexte c'est-à-dire aux lettres qui l'entourent, des modélisations contextuelles des caractères peuvent être introduites. Ces méthodes sont dérivées de la reconnaissance de la parole où une modélisation en triphones intervient.

Le mot n'est plus vu comme une succession de caractères indépendants, mais comme une succession de caractères en contexte. Les monographes sont ainsi remplacés par des trigraphes où au caractère central est rajouté son caractère de gauche et de droite (contextes gauche et droite). La figure 1.10 montre l'exemple de deux trigraphes "t-e+r" et "v-e+n" partageant la même lettre centrale mais présentant des ligatures différentes. Il n'est cependant pas envisageable de considérer tous les trigraphes possibles car cela augmenterait considérablement le nombre de paramètres à estimer. On passerait ainsi de n monographes (lettres majuscules et minuscules, lettres accentuées, chiffres et caractères spéciaux) à n^3 trigraphes. Heureusement, tous les trigraphes ne sont pas observables. Par exemple, l'analyse d'un dictionnaire de 11000 mots ne fait apparaître que 9400 trigraphes

différents.

Plusieurs stratégies existent afin de réduire le nombre de paramètres à estimer. Principalement, elles consistent en un partage de certains des paramètres ou des états des trigraphes [Natarajan *et al.*, 2008, Fink et Plotz, 2007, Bianne-Bernard *et al.*, 2011].

1.5.2 Reconnaissance par Réseaux de Neurones

1.5.2.a Une approche bio-inspirée

L'étude du fonctionnement du cerveau humain présente un double objectif. Le premier est d'avoir une appréhension de plus en plus précise du fonctionnement neuronal humain. Le second est la construction de modèles de traitement de l'information, le rêve scientifique étant de pouvoir bâtir un cerveau artificiel (*Blue Brain Project* [Markram, 2006]). Les réseaux de neurones artificiels (*Artificial Neural Network - ANN*) sont donc des modélisations mathématiques inspirées des systèmes nerveux biologiques.

Le premier modèle de neurone formel a été introduit par [McCulloch et Pitts, 1943]. Un neurone artificiel est un automate comprenant plusieurs entrées et une sortie. En reprenant l'analogie biologique, les entrées correspondent aux dendrites et la sortie à l'axone. Généralement, le neurone effectue la somme pondérée de ses entrées et applique une fonction d'activation à cette somme donnant ainsi sa valeur de sortie. Les neurones sont reliés les un autres en un réseau, par des connexions pondérées par des poids synaptiques. Un des enjeux de la modélisation des réseaux de neurones est de choisir le nombre de neurones, leurs connexions et les poids synaptiques. L'optimisation des valeurs des poids se fait généralement par un apprentissage statistique. Les principales applications des réseaux de neurones sont le classement automatique, la reconnaissance de formes (*pattern recognition*) [Likforman-Sulem et Barney-Smith, 2012] et la modélisation de fonctions.

Dans cette section, nous donnons un aperçu de l'historique des développements des réseaux de neurones et de leurs méthodes d'apprentissage. En effet la complexité des architectures neuronales utilisées aujourd'hui pour la reconnaissance d'écriture est issue d'un long processus de recherche qu'il est important de préciser. Nous ne cherchons pas à donner un catalogue exhaustif des réseaux proposés mais plutôt à recenser les orientations principales prises par ce domaine de recherche. Partant d'une structure neuronale indivi-

duelle (perceptron) (section 1.5.2.b), nous présentons des structures contenant plusieurs couches de neurones (section 1.5.2.c). Ensuite, nous évoquerons les réseaux convolutionnels (section 1.5.2.d) inspirés par le cortex visuel et les réseaux à retard (section 1.5.2.e). Nous terminerons en évoquant les réseaux de neurones récurrents et leurs dérivés utilisés actuellement pour la reconnaissance d'écriture manuscrite (section 1.5.2.f).

1.5.2.b Le perceptron

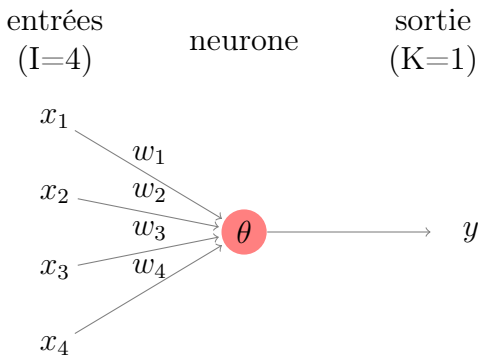


FIGURE 1.11 – Exemple de perceptron simple couche

Un des types de réseaux de neurones parmi les plus simples est le perceptron qui fut introduit par [Rosenblatt, 1958]. Le perceptron est composé d'un seul neurone (figure 1.11). Nous détaillons ce réseau afin de donner un aperçu d'une structure neuronale individuelle. En effet, les structures utilisées pour la reconnaissance d'écriture sont toutes dérivées de cette unité. Pour obtenir la sortie du réseau en fonction de l'entrée, il est tout d'abord nécessaire de calculer l'entrée du neurone. Elle s'obtient en sommant les entrées du réseau x_i pondérées par des poids synaptiques w_i : $\sum_{i=1}^{I} w_i x_i$. La sortie du neurone y est ensuite obtenue en appliquant une fonction d'activation θ à l'entrée du neurone. On réalise ainsi une passe en avant (*Forward pass*) :

$$y = \theta(\sum_{i=1}^{I} w_i x_i) \tag{1.1}$$

$$\text{où} \begin{cases} x_i & \text{valeur du } i\text{-ème neurone d'entrée } (1 \leq i \leq I) \\ w_i & \text{poids synaptique de la } i\text{-ème entrée au neurone} \\ y & \text{valeur de sortie du neurone} \\ \theta & \text{fonction d'activation du neurone} \end{cases}$$

La fonction d'activation utilisée par le perceptron est la fonction de Heaviside : $\theta(u) = 0$ si $u < 0$ et $v = \theta(u) = 1$ si $u \geq 0$. Ainsi, le perceptron peut effectuer une décision binaire à partir de ses I unités d'entrée.

L'apprentissage de l'ensemble des poids w du perceptron simple couche est généralement réalisé grâce à la règle d'apprentissage du perceptron, dérivée de l'algorithme de descente du gradient :

$$w'_i = w_i + \alpha(z - y)x_i \tag{1.2}$$

où w'_i est le poids mis à jour, z la sortie souhaitée, y la sortie effective et α le pas d'apprentissage.

1.5.2.c Le perceptron multi-couches, un approximateur universel

Un des principaux reproches adressés au perceptron est son incapacité à traiter des problèmes de classement où les données ne sont pas linéairement séparables. En effet, la fonction de Heaviside ne permet de modéliser que des décisions binaires. Cette limitation a été vaincue grâce à l'introduction de neurones aux fonctions sigmoïdes : $\theta(u) = \frac{1}{1+e^{-u}}$ ou $\theta(u) = tanh(u)$ par exemple. En effet [Cybenko, 1989] démontre que toute fonction bornée continue peut être approchée par une combinaison linéaire de fonctions sigmoïdes.

Un autre reproche adressé au perceptron est de ne pas pouvoir modéliser des opérations complexes. Par exemple, l'opération "OU EXCLUSIF" (XOR) ne peut pas être effectuée par un simple perceptron mais cette opération est possible avec un réseau comprenant deux niveaux de neurones.

L'introduction de perceptrons multi-couches (*Multi-layer perceptron* - MLP [Rumelhart *et al.*, 1986]) permet de résoudre les deux principaux problèmes sus-cités. Ce réseau est organisé en couches successives, chaque couche prenant pour entrées les sorties de la

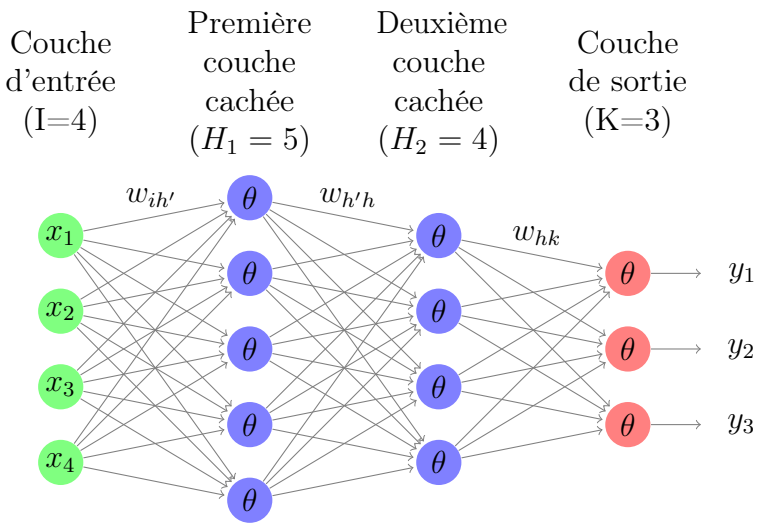


FIGURE 1.12 – Exemple de perceptron à deux couches

couche précédente.

Lorsque l'on considère une tâche de classement (supervisé), la couche d'entrée du réseau correspond alors au vecteur représentant la donnée à classer $x = (x_1, .., x_I)$ et la couche de sortie correspond au vecteur obtenu $y = (y_1, .., y_K)$, où les y_i sont associés à chacune des K classes. L'objectif est d'obtenir un vecteur z où une seule des valeurs est non nulle, correspondant à la classe correcte C_k à associer à la donnée d'entrée. Par exemple, la sortie souhaitée pour une entrée de classe k est $z = (z_1 = 0, .., z_k = 1, ..., z_K = 0)$.

Entre les entrées et les sorties, une succession de couches dites cachées interviennent. Les sorties des couches cachées correspondent à des représentations internes. Les neurones des couches cachées utilisent des fonctions sigmoïdes. Un perceptron multi-couches avec $L = 2$ couches cachées, $I = 4$ unités d'entrée et H_l neurones dans la couche cachée l ($l = 1..L$) est représenté sur la figure 1.12. Pour les neurones de sortie, on considère souvent la fonction *softmax*. En effet, celle-ci permet de normaliser les sorties y_k et elles peuvent ainsi être considérées comme des probabilités a posteriori $P(C_k|x)$ où k est la classe associée au k-ième neurone de sortie :

$$a_k = \sum_{h=1}^{H_2} w_{hk} b_h \tag{1.3}$$

$$y_k = \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}} \text{ (Fonction softmax)} \tag{1.4}$$

où les b_h sont les activations de la deuxième couche cachée (H_2), et w_{hk} les poids synaptiques entre la couche cachée H_2 et la couche de sortie. a_k correspond donc à l'entrée du k-ième neurone de sortie et y_k est sa sortie. Les activations b_h de la couche H_2 sont obtenues classiquement par une passe en avant comme dans l'équation 1.2. Ce réseau est dit *feed-forward* car il ne contient aucun cycle (appelé aussi boucle de rétroaction).

Les possibilités de modélisation du perceptron multi-couches sont très grandes. Dans [Hornik, 1991] il est montré qu'une structure contenant plusieurs couches de neurones peut modéliser n'importe quelle fonction à condition de disposer d'assez de neurones cachés. C'est un approximateur universel.

D'autres structures multi-couches ont émergé dans les années 80. Certaines ont remplacé les sigmoïdes par des fonctions à base radiale (*Radial Basis Function* - RBF) [Broomhead et Lowe, 1988] principalement pour la reconnaissance de caractères isolés [Lemarié, 1993]. Les réseaux de neurones RBF sont comme le perceptron multi-couches des approximateurs universels. Leur principal attrait est de converger rapidement lors de leur apprentissage. D'autres comme les cartes de Kohonen (*Self-Organizing Maps* - SOM) [Kohonen, 1989], dérivées de l'algorithme des k-moyennes, sont fondées sur un apprentissage non-supervisé.

Rétropropagation du gradient L'utilisation de réseaux multi-couches feed-forward n'a été rendue possible que par le développement d'une méthode d'apprentissage des poids w du réseau. La méthode dite de 'Rétropropagation du gradient' (*Gradient backpropagation*) a été introduite par [Werbos, 1974] et appliquée par [Rumelhart *et al.*, 1986] au perceptron multi-couches. Cette méthode repose sur une descente de gradient. Il est nécessaire de disposer d'une base d'apprentissage S constituée d'un certain nombre de couples (x, z) où x est le vecteur d'entrée du réseau et z le vecteur de sortie que l'on souhaite lui voir associer. On notera w l'ensemble des paramètres du modèle (les poids synaptiques).

Grâce à la base d'apprentissage S, nous disposons donc de la probabilité :

$$p(S|w) = \prod_{(x,z)\in S} p(z|x,w) \tag{1.5}$$

La loi de Bayes nous permet d'obtenir la probabilité du paramétrage conditionnellement à la base d'apprentissage :

$$p(w|S) = \frac{p(S|w)p(w)}{p(S)} \tag{1.6}$$

En considérant $p(S)$ constante et la distribution a priori des paramètres $p(w)$ uniforme, cette optimisation se ramène à effectuer l'estimation d'un maximum de vraisemblance :

$$w_{ML} = \arg\max_w p(S|w) = \arg\max_w \prod_{(x,z)\in S} p(z|x,w) \tag{1.7}$$

Il est donc nécessaire de construire une fonction de vraisemblance à maximiser. Ici on considère plutôt la fonction de perte $\mathcal{L}$ qu'il faudra donc minimiser :

$$\mathcal{L}(S) = -\ln \prod_{(x,z)\in S} p(z|x,w) = -\sum_{(x,z)\in S} \ln p(z|x,w) \tag{1.8}$$

On pourra considérer également l'évaluation de cette fonction sur un seul couple (x, z) :

$$\mathcal{L}(x,z) = -\ln p(z|x,w) \tag{1.9}$$

Dans le cas d'un problème à K classes, la probabilité d'obtenir une classe C_k conditionnellement à une entrée x sera notée $y_k = p(C_k|x)$. Ainsi le probabilité d'obtenir une sortie z conditionnellement à l'entrée x et à un paramétrage w pourra être exprimée sous la forme :

$$p(z|x,w) = \prod_{k=1}^{K} y_k^{z_k} \tag{1.10}$$

où $z = (z_1, .., z_K)$ est le vecteur binaire à K éléments, tous nuls, excepté z_k correspondant à la classe correcte C_k. La fonction de perte sur un couple (x, z) s'écrit donc :

$$\mathcal{L}(x,z) = -\ln p(z|x,w) = -\sum_{k=1}^{K} z_k \ln y_k \tag{1.11}$$

La minimisation de cette fonction de perte est l'objectif de chaque étape de l'entraînement du perceptron multi-couches. Nous détaillons ici les étapes de l'apprentissage des poids du réseau :

— Pour chaque vecteur d'entrée x, le vecteur de sortie y du réseau est calculé par une simple passe en avant à travers le réseau.
— La fonction de perte $\mathcal{L}(x, z)$ est calculée
— L'erreur est rétro-propagée à travers le réseau : les dérivées partielles $\frac{\partial \mathcal{L}(x,z)}{\partial w_{ij}}$ sont calculées successivement en partant de la couche de sortie jusqu'à la couche d'entrée
— Les poids du réseau sont mis à jour : $w'_{ij} = w_{ij} - \alpha(z - y)\frac{\partial \mathcal{L}(x,z)}{\partial w_{ij}}$ où w'_{ij} est le poids mis à jour et α le pas de descente.

La dénomination "rétropropagation du gradient" vient du fait que l'on calcule récursivement les dérivées en fonction des poids, couche par couche, en partant de la couche de sortie. Cette méthode peut converger vers des minimums locaux. Généralement, la base d'apprentissage est parcourue plusieurs fois pour apprendre les poids du réseau. En raison de leur nombre important de paramètres, les réseaux de neurones font souvent face au problème du surapprentissage (*over-fitting*) du modèle. Ce phénomène peut être contrôlé en utilisant une base de validation sur laquelle on vérifie régulièrement le taux d'erreur durant l'apprentissage. L'entraînement du réseau sera arrêté quand le taux d'erreur augmente sur la base de validation.

1.5.2.d Réseaux convolutionnels, une architecture profonde

Les réseaux de neurones convolutionnels sont dérivés des réseaux multi-couches. Ces réseaux s'inspirent du système de vision [Hubel et Wiesel, 1968]. Effectivement, les cellules du cortex visuel sont sensibles uniquement à une partie du champ visuel, nommée champ de réception. L'ensemble des cellules de la couche permet de couvrir l'intégralité du champ visuel avec des recouvrements. Deux types de neurones peuvent être distingués : les cellules simples et les cellules complexes. Les cellules simples réagissent à des stimuli précis tels que les bords (*edge*) au sein de leur champ de réception. Quant aux cellules complexes, elles ont des champs de réception plus larges et sont invariants à la position du stimuli. Ces deux types de cellules se présentent en alternance dans le cortex visuel.

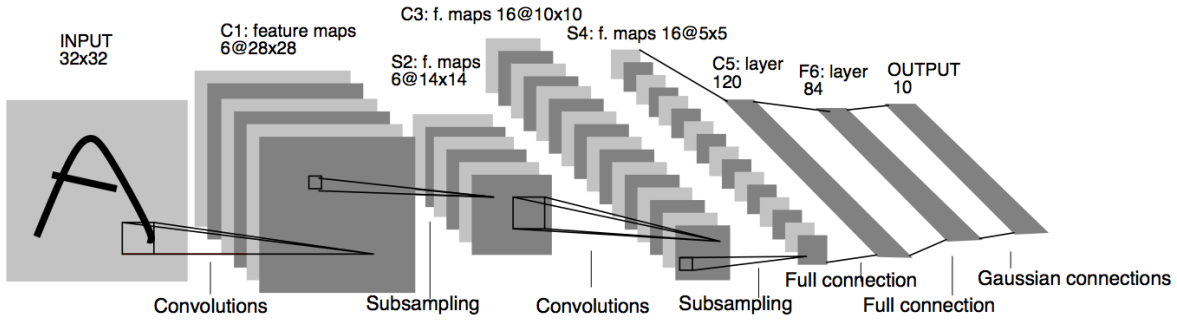


FIGURE 1.13 – Exemple d'un réseau de neurones convolutionnel (d'après [LeCun *et al.*, 1998])

Les premières modélisations inspirées du cortex visuel dont été introduites pour la reconnaissance de caractères manuscrits par [Giebel, 1971] et le Neocognitron [Fukushima, 1980]. Ce système évoluera vers le réseau LeNet-5 [LeCun *et al.*, 1998] très largement repris par l'ensemble de la communauté de la reconnaissance de formes. Ce réseau, représenté sur la figure 1.13 comprend 7 couches cachées, alternant filtres convolutionnels et sous-échantillonnages. Plusieurs filtres convolutionnels avec des noyaux de petites tailles sont appliqués à l'image du caractère. Cette opération permet de disposer d'autant de cartes de caractéristiques que l'on applique de filtres différents. Les caractéristiques extraites par ces filtres sont locales et sont des représentations de bas niveau (essentiellement des contours). Les couches de sous-échantillonnage successives permettent de réduire progressivement la résolution des cartes de caractéristiques au fil des couches. Cela réduit notamment la sensibilité du classifieur aux distorsions et aux décalages. Cette stratégie permet d'obtenir, au fur et à mesure que l'on progresse dans les couches, des représentations de plus haut niveau. En revanche, le nombre des cartes augmente au fil des premières couches pour aboutir à une représentation plus riche des données (plus de caractéristiques). Les dernières couches du réseau vont permettre de réduire le nombre de ces représentations afin d'aboutir aux étiquettes de la couche de sortie. Les poids du réseau sont appris par une rétropropagation du gradient. Les réseaux convolutionnels sont les premiers exemples d'architectures profondes (*deep neural networks*).

1.5.2.e Réseaux à retard, une modélisation spatiale du temps

Face aux problématiques de classement des séquences temporelles, les réseaux de type perceptron multi-couches sont inadaptés puisqu'ils ne peuvent classer que des formes iso-

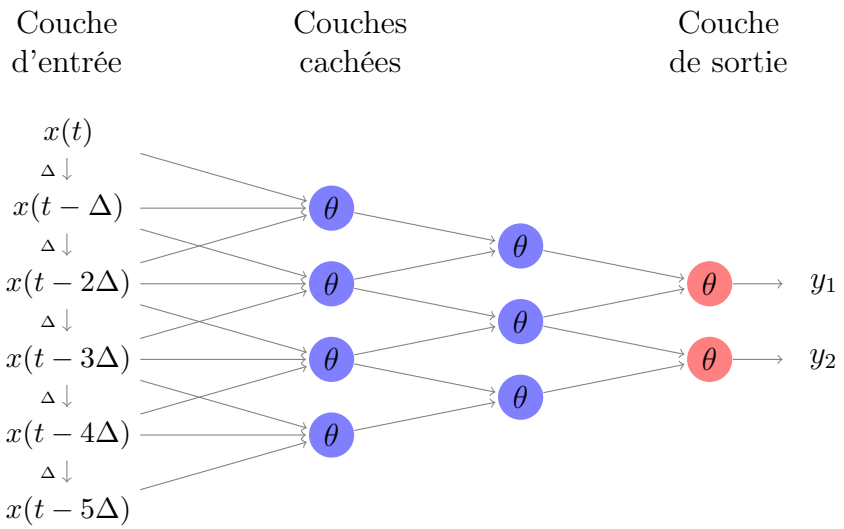


FIGURE 1.14 – Exemple de réseau à retard avec un pas Δ

lées, comme les caractères manuscrits. Une des premières propositions faites pour répondre à cette question sont les réseaux de neurones à retard (*Time-Delay Neural Network* - TDNN). À l'origine développés pour la reconnaissance de phonèmes [Waibel *et al.*, 1989], ils ont été ensuite appliqués à la reconnaissance de caractères manuscrits [Guyon *et al.*, 1991]. Une architecture hybride HMM-TDNN a été proposée pour la reconnaissance de mots cursifs en ligne par [Poisson *et al.*, 2004].

Le principe de ce type de réseau est de considérer la séquence temporelle de la même façon qu'une information spatiale. Ainsi la séquence complète est donnée en entrée du réseau comme schématisé sur la figure 1.14 pour un signal d'entrée défini sur six instants $x = (x(t), x(t-\Delta), x(t-2\Delta), x(t-3\Delta), x(t-4\Delta), x(t-5\Delta))$ où Δ est le pas du retard. Ainsi la séquence temporelle est représentée de la même façon qu'une information spatiale.

La plupart des réseaux à retard ne sont pas à connexions complètes. Chacun des neurones de la première couche cachée est relié à une partie seulement de la séquence. Par exemple sur la figure 1.14, le premier neurone de la première couche cachée n'est relié qu'à trois instants de la séquence d'entrée. Le même procédé est appliqué aux autres couches. Ainsi le réseau conserve l'ordre temporel de la séquence à travers ses différentes couches. La fusion de l'information des différents instants est effectuée progressivement en réduisant le nombre de neurones, couche après couche. Si cette modélisation a connu

un certain succès dans les années 1990 et 2000, elle a été remplacée par les réseaux de neurones récurrents pour la reconnaissance off-line car ceux-ci permettent d'introduire une mémoire à travers le processus de traitement de la séquence.

1.5.2.f Réseaux de neurones récurrents et formes dérivées

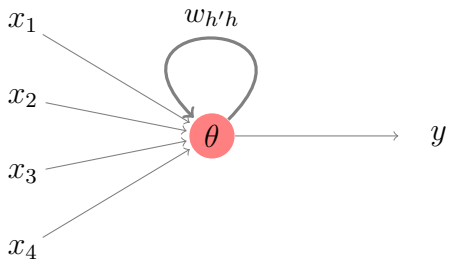


FIGURE 1.15 – Exemple de neurone récurrent

L'idée d'un neurone récurrent est encore une fois inspirée par le cerveau humain et est apparue dès les années 40 [McCulloch et Pitts, 1943]. Le neurone récurrent formel dispose d'une connection synaptique qui va de la sortie d'un neurone vers son entrée, appelée aussi boucle de rétroaction, comme représenté sur la figure 1.15. Ainsi l'état du neurone à l'instant t dépend de son état à l'instant $t-1$:

$$a^t = \sum_{i=1}^{I} w_{ih} x_i^t + w_{h'h} y^{t-1} \tag{1.12}$$

$$y^t = \theta(a^t) \tag{1.13}$$

en reprenant les notations de l'équation 1.2 et où $w_{h'h}$ est le poids synaptique de la boucle de rétroaction.

Cette boucle de rétroaction introduit l'idée d'un traitement neuronal à travers le temps et donc une application possible au classement de séquences temporelles. Cependant, les premiers réseaux récurrents formels, introduits par [Hopfield, 1982], n'étaient pas destinés à traiter des séquences temporelles. L'introduction de la rétroaction permettait alors d'assurer la convergence de l'apprentissage.

Les premiers réseaux récurrents conçus pour traiter des séquences de motifs sont les réseaux de Jordan [Saul et Jordan, 1986] et d'Elman [Elman, 1990]. Ces réseaux comportent une seule couche cachée ainsi qu'une couche dite de "contexte". Dans le cas du réseau de Jordan, cette couche prend pour entrées les sorties du réseau y_k et réinjecte cette connaissance en entrée de la couche cachée. De plus chaque neurone de cette couche dispose d'une liaison récurrente vers lui-même. Le réseau de Elman est légèrement différent. La couche de contexte prend pour entrée les sorties de la couche cachées. Ainsi, pour les deux réseaux, la couche de contexte permet d'établir une mémoire dans le réseau. [Lee et Kim, 1995] propose une reconnaissance de caractères manuscrits par un réseau de neurones avec une couche cachée, et dont les neurones de la couche de sortie ont des connexions récurrentes. Cette structure a renforcé le pouvoir discriminant et la capacité de généralisation du réseau par rapport aux structures de Jordan et de Elman.

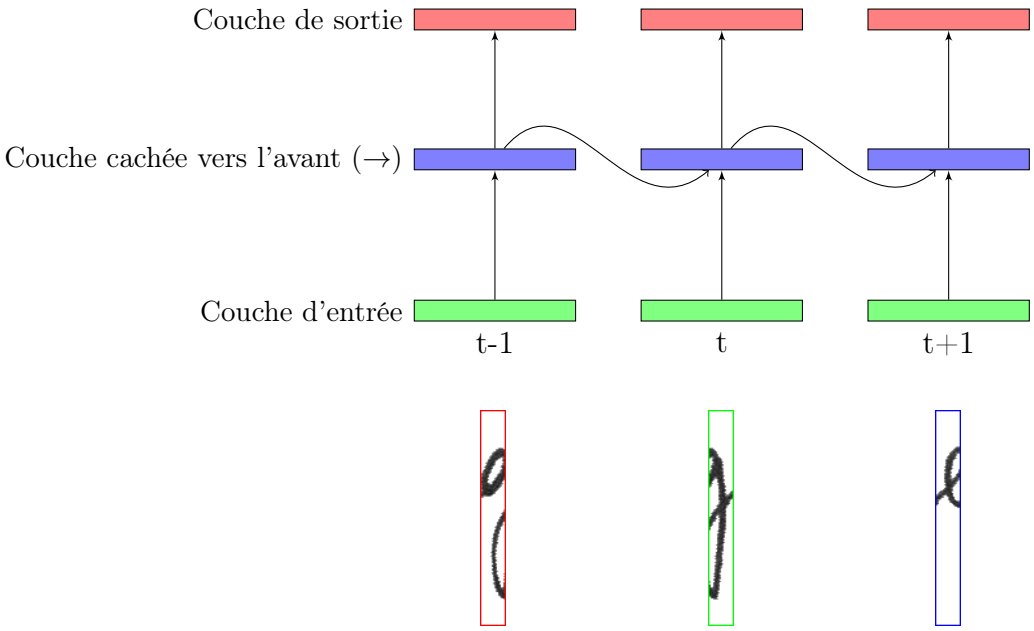
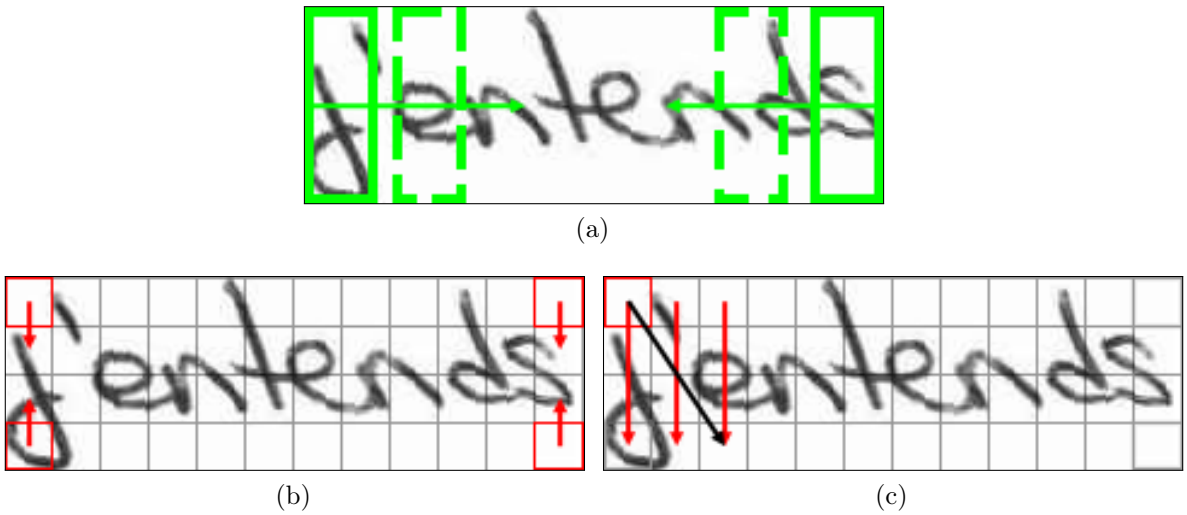


FIGURE 1.16 – Structure d'un réseau de neurones récurrent (RNN) avec une seule couche dépliée dans le temps

Les premières applications de réseaux de neurones récurrents (*Recurrent Neural Network - RNN*) au classement de séquences temporelles ont été rendues possible grâce à l'émergence d'une méthode d'apprentissage adaptée : la rétropropagation du gradient à travers le temps (*Backpropagation Through Time - BPTT*) [Werbos, 1990]. Le principe général de cette méthode est de déplier ce réseau à travers le temps comme représenté sur



(a) (b) (c)

FIGURE 1.17 – (a) Parcours de l'image pour un réseau récurrent bidirectionnel (BRNN). (b) Parcours de l'image pour un réseau récurrent multidimensionnel MDRNN. (c) Orientation résultante pour un des parcours du MDRNN (flèche noire)

la figure 1.16. Ainsi, on se ramène à un réseau feed-forward auquel on peut appliquer une rétropropagation du gradient classique.

Au cours des dix dernières années, les Réseaux de Neurones Récurrents ont été appliqués dans le domaine de la reconnaissance de la parole et plus récemment dans ceux de l'écriture manuscrite en-ligne [Liwicki *et al.*, 2007] et hors-ligne [Graves *et al.*, 2009]. Les réseaux de neurones présentent l'avantage d'être entraînés de manière discriminante, alors que les HMM sont des modèles génératifs en règle générale. À partir de cette structure, plusieurs autres modèles ont été proposés et expérimentés avec succès :

— les BRNN (Bidirectional Recurrent Neural Networks) considèrent également en entrée un signal mono-dimensionnel mais introduisent deux parcours dans l'image (figure 1.17a). Le réseau est alors entraîné simultanément sur les signaux issus des deux parcours, gauche et droite [Schuster et Paliwal, 1997]. Ainsi l'information contextuelle est prise en compte selon les deux directions de l'écriture. Les BRNN disposent de deux couches cachées distinctes qui traitent les deux signaux séparément (figure 1.18). Leurs sorties respectives sont fusionnées au niveau de la couche de sortie.

— les MDRNN (Multidimensional Recurrent Neural Networks) [Graves *et al.*, 2007]

qui étendent la notion des RNN à des données d'entrée de dimension ≥ 2. Une image de texte peut être vue comme un signal mono-dimensionnel (succession de colonnes de pixels ou de vecteurs d'observations) et ainsi être parcourue selon deux directions, comme l'effectue le BRNN. Cependant, elle peut aussi être perçue comme un signal bi-dimensionnel (succession de pixels) et ainsi conduire à $2^2 = 4$ parcours différents (figure 1.17b). Cette approche par quatre parcours renonce, par définition, à une extraction de caractéristiques classique, par fenêtre glissante sur l'axe horizontal. Ici, une fenêtre d'analyse rectangulaire parcourt l'image en partant de ses angles comme présenté sur la figure 1.17c. Le déplacement de cette fenêtre peut donc être perçu comme diagonal. Cette approche reprend une architecture similaire aux réseaux convolutionnels profonds évoqués dans la section 1.5.2.d : des caractéristiques de bas niveau sont extraites par la première couche et des représentations de plus haut niveau apparaissent plus on progresse au sein de la structure.

Cependant, le BRNN comme le MDRNN n'effectuent en réalité qu'un seul parcours de l'image pour générer la séquence d'entrée du réseau. Cette séquence est ensuite parcourue différemment par les couches du réseau. Dans un BRNN, lorsque l'on traite l'entrée au temps t, la couche vers l'avant a déjà examiné les entrées entre 1 et $t-1$ tandis que la couche arrière a examiné les entrées entre T et $t+1$ (pour une séquence d'entrée de longueur T).

Un problème inhérent à l'utilisation de ces réseaux sur des séquences temporelles est la difficulté de conserver l'information contextuelle sur de grands intervalles de temps. Ce phénomène est dû à l'évanouissement du gradient (*vanishing gradient*) [Hochreiter *et al.*, 2001]. Ceci est particulièrement préjudiciable à la reconnaissance de l'écriture. Cette difficulté a été en partie résolue par l'introduction d'un nouveau type de neurones "intelligent" le LSTM (*Long-Short Term Memory*) [Hochreiter et Schmidhuber, 1997] qui contient une mémoire. Grâce à ce type de neurones, les structures BLSTM (*Bidirectional Long-Short Term Memory*) [Graves et Schmidhuber, 2005] et MDLSTM (*Multidimensional Long-Short Term Memory*) [Graves et Schmidhuber, 2008] ont été créées respectivement à partir des BRNN et MDRNN. L'ensemble des développements autour du LSTM et des réseaux de neurones récurrents pour la reconnaissance d'écriture est rassemblé dans [Graves, 2012]. Nous reviendrons plus précisément sur la modélisation BLSTM dans le chapitre 4. En effet, nous avons choisi cette modélisation car les entrées d'un tel réseau sont identiques à celles d'un HMM.

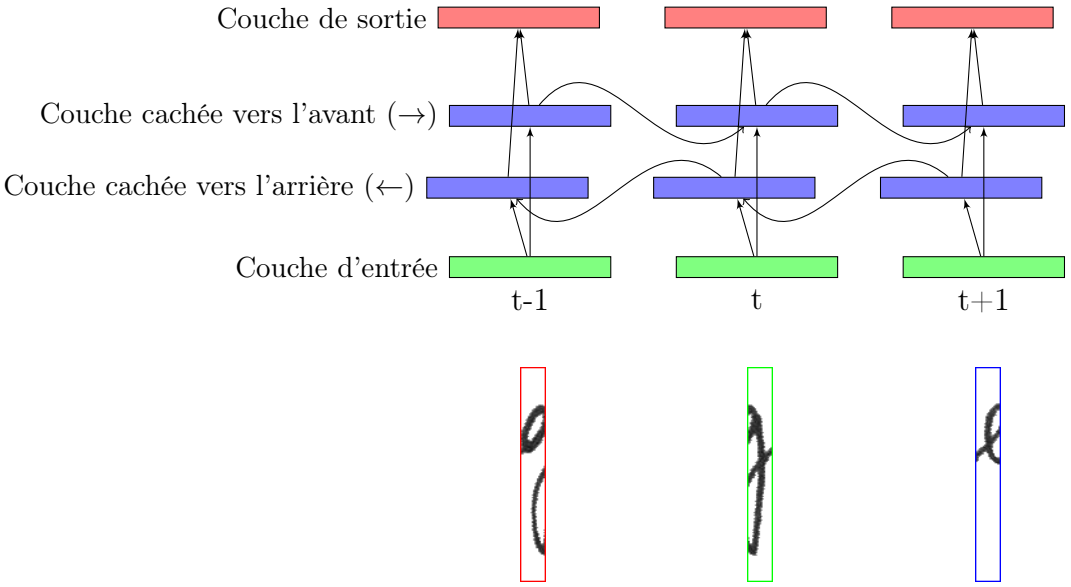


FIGURE 1.18 – Structure d'un réseau de neurones récurrent bi-directionnel (BRNN) simple couche déplié dans les temps

1.5.3 Architectures hybrides HMM / neuronale

Nous avons vu que deux classes de modélisations dominent le domaine de la reconnaissance d'écriture manuscrite : les HMM et les réseaux de neurones artificiels ANN. Les réseaux de neurones présentent l'avantage d'être très discriminants et les modèles de Markov sont reconnus pour leur capacité de modélisation et de résistance au bruit. Certains travaux ont tenté d'associer les atouts des deux méthodes au sein d'architectures hybrides.

Les premiers travaux sur une approche hybride ont été introduits par [Bengio *et al.*, 1995] avec l'outil *LeRec* pour la reconnaissance de mots manuscrits en ligne : la reconnaissance y est effectuée par un réseau convolutionnel multi-couche et les scores fournis en sortie sont considérées comme l'observable d'un HMM. Ainsi le décodage sur le HMM est effectué par une recherche en faisceau (*beam search*). L'apprentissage du réseau de neurones et du HMM sont effectués conjointement. L'apport de la modélisation HMM est de pouvoir introduire des contraintes lexicales lors de la recherche et aussi de pouvoir aboutir à une segmentation précise en caractères. Cette approche a été étendue à la

reconnaissance de mots hors-ligne [Knerr et Augustin, 1998, Tay *et al.*, 2001] et de lignes de texte [España-Boquera *et al.*, 2011]. Un système HMM-réseau profond a également été appliqué à l'extraction de mots-clefs par [Thomas *et al.*, 2013].

1.6 Modélisation du langage

En suivant le formalisme détaillé dans la section 1.4, il s'agit ici d'estimer la probabilité $P_{grammaire}(W)$ d'une séquence de mots W. Cette probabilité permet d'introduire dans notre modèle de reconnaissance les contraintes imposées par les propriétés de la langue. D'une part, les mots d'une langue sont utilisés à des fréquences très différentes. D'autre part, les successions des mots suivent des règles grammaticales et syntaxiques précises. Afin de prendre en compte ces propriétés, des modèles de langages statistiques sont construits pour estimer les successions de mots les plus probables.

Les modèles de langages statistiques sont utilisés dans une grande variété de domaines rattachés au traitement automatique du langage naturel (reconnaissance de la parole, traduction automatique, recherche d'information, ...). Ils sont notamment utilisés dans la reconnaissance de la parole depuis les années 1980 [Bahl *et al.*, 1983] mais ce n'est que relativement récemment que cette approche a été transposée à la reconnaissance d'écriture [Senior et Robinson, 1998, Vinciarelli *et al.*, 2004].

L'attribution de probabilités d'apparition différentes aux séquences de mots peut permettre d'améliorer la reconnaissance faite par le modèle optique. En effet, un mot mal reconnu par la seule modélisation HMM pourra être ainsi corrigé grâce à son contexte. Cependant, le modèle de langage peut également contraindre le système à reconnaître des séquences de mots apprises par le modèle de langage, en dépit de la probabilité optique. Un équilibre doit être trouvé entre la reconnaissance optique et l'apport du modèle de langage. Deux paramètres peuvent venir pondérer l'influence du modèle de langage :

— le *Grammar Scale Factor* (GSF) permet d'attribuer un poids plus ou moins important aux probabilités du modèle de langage (voir Eq. 1.14) ;

— le *Word Insertion Penalty* (WIP) permet d'ajouter une log-probabilité positive ou négative pour respectivement favoriser ou pénaliser l'ajout de mots lors du

décodage (voir Eq.1.14).

Voici l'expression de la probabilité grammaticale d'une séquence W de N mots en fonction du GSF et du WIP :

$$P_{grammaire}(W) = P_{LM}(W)^{GSF} * 10^{N*WIP} \tag{1.14}$$

Le passage au logarithme met en avant les rôles respectifs du GSF et du WIP :

$$log(P_{grammaire}(W)) \quad = GSF * log(P_{LM}(W)) + N * WIP \tag{1.15}$$

Le GSF est un facteur multiplicatif qui vient augmenter le poids pris par le modèle de langage lorsque sa valeur est strictement supérieure à 1. Le WIP est une valeur qui sera ajouté à la log-probabilité grammaticale à chaque fois qu'un mot est ajouté à la chaîne. En prenant une valeur de WIP négative, l'insertion de mots est alors pénalisée.

En supposant que la probabilité d'un mot ne dépend que de ceux qui le précèdent, une chaîne de mots peut être envisagée comme un processus de Markov à temps discret. En suivant donc l'équation de Chapman-Kolmogorov, la probabilité d'une chaîne de mots $W = (w_1, w_2, ..., w_N)$ de longueur N peut s'écrire sous la forme :

$$P_{LM}(W) = P(w_1)P(w_2|w_1)...P(w_N|w_1, w_2, ..., w_{N-1}) = \prod_{i=1}^{N} P(w_i|w_1, ..., w_{i-1}) \tag{1.16}$$

Ce modèle prédit l'occurrence d'un mot en fonction des mots précédents. L'influence des mots suivants n'est donc pas prise en compte.

1.6.1 Modélisation par n-grammes

Les modèles de langage statistiques par n-grammes (n-grams) sont les plus fréquemment utilisés [Katz, 1987] : ils limitent le contexte d'un mots à ces $n-1$ prédécesseurs. Ainsi la probabilité d'un mot est définie en fonction de ces $n-1$ prédécesseurs $w_{i-n+1}, ..., w_{i-1}$: $P(w_i|w_{i-n+1}, ..., w_{i-1})$. L'équation 1.16 devient donc :

$$P_{LM}(W) = \prod_{i=1}^{N} P(w_i|w_{i-n+1}, ..., w_{i-1}) \tag{1.17}$$

Dans le cas de bigrammes, nous obtenons une chaîne de Markov d'ordre 1 :

$$P_{LM}(W) = \prod_{i=1}^{N} P(w_i|w_{i-1}) = P(w_1)P(w_2|w_1)...P(w_N|w_{N-1}) \tag{1.18}$$

Les probabilités de cette chaîne de Markov sont estimées sur un corpus de textes en dénombrant les successions de mots :

$$\hat{P}(w_i|w_{i-n+1}, ..., w_{i-1}) = \frac{C(w_{i-n+1}, ..., w_i)}{C(w_{i-n+1}, ..., w_{i-1})}$$

avec C(.) la fonction de dénombrement. Suivant le cadre probabiliste, nous désignerons l'occurrence d'un n-gramme par le terme d'événement. Le choix de n, la taille du contexte pris en compte, est limité par la taille du corpus d'apprentissage. En effet, plus les séquences de mots considérées sont longues (n grand), plus rares sont leurs apparitions. Pour que leurs estimations de probabilité d'occurrence soient significatives, il est donc nécessaire de disposer de très grands corpus.

Pour cette raison, la plupart des modèles de langage n-grammes choisissent $n \leq 4$. Par exemple, pour un modèle de langage trigramme ($n = 3$) avec un vocabulaire contenant 10000 mots, il est nécessaire d'estimer $10.000^3 = 10^{12}$ probabilités différentes. En pratique, seule une part infime de ces trigrammes apparaissent dans le langage courant et a fortiori dans le corpus. Pour prendre en compte les séquences inédites qui peuvent apparaître dans le décodage, il est possible d'augmenter la taille du corpus d'apprentissage ou bien de limiter la valeur de n. Cependant ces aménagements n'assurent pas la modélisation de tous les n-grammes possibles.

1.6.2 Lissage des probabilités

Afin de ne pas attribuer une probabilité nulle à ces événements, des stratégies de lissage des probabilités (en anglais *smoothing probabilities*) sont mises en œuvre : la masse de probabilité de l'ensemble des n-grammes est redistribuée depuis les événements observés vers ceux qui ne l'ont pas été.

La fréquence d'observation a d'un n-gramme ($a > 0$) va être réduite grâce à un facteur multiplicatif d_a : $a' = d_a * a$. Pour effectuer cette réduction (en anglais *discounting*) des

probabilités des événements appris, il est possible d'employer différentes techniques.

La réduction absolue consiste à prélever une masse constante sur tous les événements, et cela indifféremment à la fréquence de ces événements. Cela constitue son principal défaut : les événements peu fréquents sont relativement plus pénalisés que les fréquents. Le facteur de décompte s'écrit sous la forme :

$$d_a = \frac{a - m}{a}$$

m est fixé selon l'importance que l'on veut apporter aux événements inédits.

La méthode de Good-Turing [Good, 1953] effectue une réduction proportionnelle au nombre d'occurrences de l'événement :

$$d_a = (a+1)\frac{c_{a+1}}{a * c_a}$$

où c_a est le nombre d'événements apparaissant a fois. Un problème de cette méthode est que c_{a+1} peut être égal à 0. Une version améliorée de cette méthode a été proposée par [Katz, 1987] pour prendre en compte ce cas.

Principalement, deux approches existent pour effectuer le lissage des probabilités : une méthode dite de repli (en anglais *back-off*) et une dite d'interpolation.

La méthode du repli [Katz, 1987] consiste à utiliser la probabilité du n-gramme d'ordre inférieur. En effet si un n-gramme est inconnu, le (n-1)-gramme correspondant est plus susceptible d'avoir été rencontré dans le corpus. Dans le cas d'un modèle de bigrammes, les probabilités de ces événements inédits $\hat{P}(w_i|w_{i-1})$ sont calculées à partir de la probabilité du monogramme $\hat{P}(w_i)$:

$$\hat{P}_{repli}(w_i|w_{i-1}) = \alpha(w_{i-1})\hat{P}(w_i) \text{ si } C(w_{i-1}, w_i) = 0$$

où $\alpha(w_{i-1})$ est calculé à partir de la masse de probabilité récupérée.

La méthode de Kneser-Ney repose également sur une stratégie de repli. Cependant,

elle fait le constat qu'un repli sur la probabilité du n-gramme d'ordre inférieur n'est pas toujours le meilleur choix. Elle propose plutôt de se replier sur une autre valeur de probabilité, construite à partir des probabilités des différents historiques conduisant au mot. Le détail de cette méthode est donné dans [Ney *et al.*, 1994].

L'interpolation n'utilise pas au préalable de méthode de réduction. Elle consiste à modifier les probabilités de tous les n-grammes, inédits ou non, en les interpolant avec les probabilités d'ordre inférieur. La réduction et la redistribution de la masse de probabilité est donc faite en une seule étape. Dans le cas du modèle de bigrammes, la probabilité interpolée sera :

$$P_{interpolation}(w_i|w_{i-1}) = \lambda P(w_i|wi-1) + (1-\lambda)P(w_i)$$

avec λ, un paramètre à optimiser. Cette combinaison permet d'avoir des estimations plus robustes pour tous les n-grammes.

Pour évaluer l'adéquation entre un modèle de langage et le texte à reconnaître, une estimation de perplexité (PP) est classiquement effectuée. Cette mesure peut être vue comme une moyenne du nombre de mots possibles pouvant suivre n'importe quel mot. La perplexité est donc liée à la taille du vocabulaire et son interprétation est limitée : une diminution de sa valeur n'est pas toujours signe d'une meilleure reconnaissance. Dans le cas d'un modèle de bigramme, l'évaluation de la perplexité $\hat{PP}$ sur un texte de m mots $(w_1, w_2, ...w_m)$ s'exprime sous la forme :

$$\hat{PP} = 2^{\hat{H}} \text{ où } \hat{H} = \frac{1}{m}\sum_{i=1}^{m} \log P(w_i|w_{i-1})$$

1.6.3 Autres approches de modélisation du langage

Des variantes de ce modèle telles que les n-grammes par classes [Brown *et al.*, 1992, Perraud *et al.*, 2006] ont été introduites. Les mots sont regroupés selon des classes qui peuvent être grammaticales ou statistiques. Ces méthodes permettent notamment de réduire la perplexité du modèle de langage. De plus, elles nécessitent moins de données d'entraînement puisque l'apprentissage des probabilités est mutualisé au sein d'une classe. D'une taille de vocabulaire très grande, on considère désormais un nombre restreint de classes.

D'autres approches plus récentes utilisent des réseaux de neurones NNLM (*Neural Network Language Model*) [Bengio *et al.*, 2001]. À chaque mot du vocabulaire, on associe un vecteur de caractéristiques. Chaque mot va donc correspondre à un point dans l'espace des caractéristiques. L'intérêt de cette approche est que dans cet espace, les mots aux fonctions similaires ont des positions proches. Le réseau de neurones prend en entrée la séquence de vecteurs de caractéristiques du contexte précédant le mot à prédire et donnera en sortie le mot suivant prédit. Cette méthode nécessite un apprentissage préalable des caractéristiques ainsi que des poids du réseau de neurones. Un inconvénient de cette méthode est qu'elle requiert de très longs temps de calcul (plusieurs semaines) mais elle conduit à de meilleurs résultats que les approches par n-grammes.

Ces deux méthodes permettent de réduire l'impact du fléau de la dimension (en anglais *curse of dimensionality*) : face au très grand nombre de probabilités à estimer, elles permettent de regrouper les mots selon différents critères.

1.7 Métrique d'évaluation des performances

Pour la reconnaissance de lignes, les performances de reconnaissance sont données sous la forme de taux d'erreurs au niveau mot (WER : Word Error Rate) et de taux de mots correctement reconnus (WCR : Word Correct Rate) :

$$WER = \frac{\#\,substitutions\ +\ \#\,insertions\ +\ \#\,suppressions}{nombre\,total\,de\,mots}$$

$$WCR = 1 - \frac{\#\,substitutions\ +\ \#\,suppressions}{nombre\,total\,de\,mots}$$

Une substitution pouvant se définir comme la combinaison d'une suppression et d'un insertion, la métrique WER est redondante. Cela peut mener à obtenir des taux d'erreurs supérieurs à 100%. La valeur du WCR en revanche peut aller de 0 à 100%. Dans la compétition OpenHaRT 2013 à laquelle nous avons participé (section 4.10), les performances étaient évaluées selon un taux de reconnaissance $1-WER$. Cette métrique peut prendre des valeurs négatives. Pour les deux métriques, les erreurs de casse ne sont pas prises en compte.

1.8 Systèmes de reconnaissance de lignes

Plusieurs facteurs ont conduit à l'émergence de systèmes complets s'intéressant à la reconnaissance de lignes de texte : principalement l'amélioration des performances de reconnaissance de mots isolés, le développement de nouvelles méthodes de reconnaissance et l'augmentation des puissances de calcul. Nous n'avons pas l'ambition de présenter un catalogue complet de tous les systèmes développés actuellement. Nous souhaitons seulement présenter ici un aperçu des principaux systèmes complets de reconnaissance de lignes de texte et leurs caractéristiques essentielles. Nous avons donc choisi de présenter les systèmes qui ont été récemment soumis à des compétitions de reconnaissance d'écriture.

L'institut IAM de l'université de Berne fut l'un des premiers à proposer une reconnaissance de lignes de texte [Marti et Bunke, 2001]. Ce système s'appuie sur une reconnaissance par HMM et l'utilisation de modèles de langage avec des bigrammes. L'extraction de 9 caractéristiques géométriques est effectué par une fenêtre sans recouvrement. Les étapes de prétraitement comprennent une correction globale de la pente ainsi que de l'inclinaison et une normalisation de l'écriture. Par la suite, l'institut a proposé un système pour intégrer le modèle de langage lors de la combinaison de systèmes de reconnaissance [Bertolami et Bunke, 2008].

L'institut de recherche IDIAP a proposé un système de reconnaissance de lignes à large vocabulaire à base de HMM dérivé d'un système utilisé sur les mots isolés. Ce système utilise également des modèles de langage de type unigramme, bigramme et trigramme [Vinciarelli *et al.*, 2004]. Les opérations de prétraitement consistent à corriger globalement la pente et l'inclinaison de l'écriture. Les caractéristiques utilisées sont des densités de pixels extraites dans des sous-cellules.

L'université technique de Munich (TU München) fut la première à proposer lors d'une compétition une reconnaissance de mots isolés par réseaux MDLSTM [Grosicki et El-Abed, 2009, Graves *et al.*, 2009]. Devant le succès rencontré par cette méthode, nombre d'autres laboratoires ont repris cette structure depuis.

L'université de Valence (Universitat Politècnica de València) propose un système fondé sur un reconnaisseur hybride HMM-RNN [España-Boquera *et al.*, 2011] : si l'écriture est bien modélisée par HMM, un perceptron multicouche permet d'en estimer les probabilités d'émission. Le prétraitement des lignes comprend un nettoyage de l'image, une correction de l'inclinaison, une correction de la pente par morceaux et une détection des lignes de base pour normaliser la taille des zones de l'écriture. Le modèle extrait les caractéristiques en divisant l'image de ligne de texte en cellules. Un autre système présenté à la compétition de reconnaissance d'arabe manuscrit OpenHaRT 2013 utilise des HMM où les probabilités d'émission sont modélisées par des mélanges de distribution de Bernoulli.

La société A2iA a participé à de nombreuses compétitions de reconnaissance d'écriture manuscrite notamment sur l'arabe et le français. Les premiers systèmes proposés étaient fondés sur des HMM contextuels utilisant des séquences de caractéristiques extraites par fenêtre glissante [Bianne-Bernard *et al.*, 2011, Menasri *et al.*, 2012]. Ces systèmes reposaient sur une segmentation préalable de lignes en mots. Plus récemment, ce système a évolué vers une reconnaissance fondée sur les MDLSTM et une architecture profonde ne nécessitant plus d'extraction de caractéristiques. Ce système travaille désormais au niveau ligne et utilise des FST (Finite-State Transducers) pour le décodage. L'approche multi-dimensionnelle semble permettre au système de s'affranchir de tout prétraitement d'image.

L'université d'Aachen (RWTH) a développé un système de reconnaissance par HMM gaussien utilisant une librairie diffusée publiquement RWTH-ASR [Rybach *et al.*, 2011]. L'extraction des caractéristiques est effectué par un réseau LSTM. Lors du prétraitement, les positions centres de gravité de l'écriture sont calculés sur des fenêtres verticales. Les fenêtres sont ensuite translatées verticalement pour corriger les différences de position entre les centres de gravité [Doetsch *et al.*, 2012]. Cela permet de corriger les variations de la position de la ligne de base basse.

Le CITlab de l'université de Rostock (Computational Intelligence Technology Laboratory) est bâti sur une architecture MDLSTM où les cellules LSTM sont modifiées : la fonction de la cellule est remplacée par une combinaison convexe de ses entrées. Le

prétraitement consiste en une correction de l'inclinaison du texte et une normalisation des hauteurs de l'écriture.

Le LITIS de Rouen (Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes) utilise un réseau HMM appris sur des séquences de 133 caractéristiques extraites depuis les images de ligne. Le prétraitement comprend une correction globale de la pente et de l'inclinaison de l'écriture et une normalisation de la hauteur de l'écriture.

1.9 Conclusion du chapitre 1

Dans ce chapitre, nous avons exposé les différentes étapes de la reconnaissance des lignes de texte manuscrites : prétraitement des images, extraction des caractéristiques, méthodes de reconnaissance et modélisation du langage. De ces domaines nous avons pu extraire plusieurs problématiques propres à la ligne de texte.

Tout d'abord, en ce qui concerne le prétraitement des images, nous avons montré que les lignes de texte nécessitent de développer des méthodes spécifiques. En effet, le prétraitement des mots isolés ne répond pas à toutes les problématiques soulevées au niveau ligne. Le découpage en lignes peut notamment introduire des composantes des autres lignes dans la vignette. De plus, le position de la ligne de base peut varier considérablement et perturber l'extraction des caractéristiques. Ces questions de prétraitement seront soulevées dans le chapitre 2.

Ensuite, nous avons exposé les différentes méthodes de reconnaissance automatique de texte manuscrit. Deux grandes classes de méthodes se détachent : d'une part les HMM et d'autre part les ANN. Les HMM ont une grande capacité de modélisation et s'adaptent bien aux séquences de durée variable. Les ANN, quant à eux, présentent l'avantage d'être discriminants. Si on constate l'émergence récente et rapide des méthodes par ANN, les HMM ne sont pas pour autant négligés et continuent à fournir des systèmes très performants. Certains systèmes hybrides HMM/ANN permettent de bénéficier des avantages des deux méthodes. Nous avons recensé les différents systèmes de reconnaissance de lignes de textes. Face à la diversité des méthodes et des systèmes, il serait intéressant de construire plusieurs systèmes de reconnaissance, de les optimiser et d'en comparer les apports et

performances. Afin de pouvoir comparer des méthodes s'appuyant sur les mêmes données, nous choisissons deux méthodes reposant sur une extraction de caractéristiques par fenêtre glissante. Nous allons donc construire une reconnaissance de lignes de texte par HMM (chapitre 3) et par BLSTM et en comparer les apports (chapitre 4).

Enfin, la question du choix du vocabulaire et de la modélisation de langage constitue un problème central de la reconnaissance des lignes de texte. Cette question a une grande influence sur les performances mesurées. Nous aborderons la construction du modèle de langage également dans le chapitre 3.

Chapitre 2

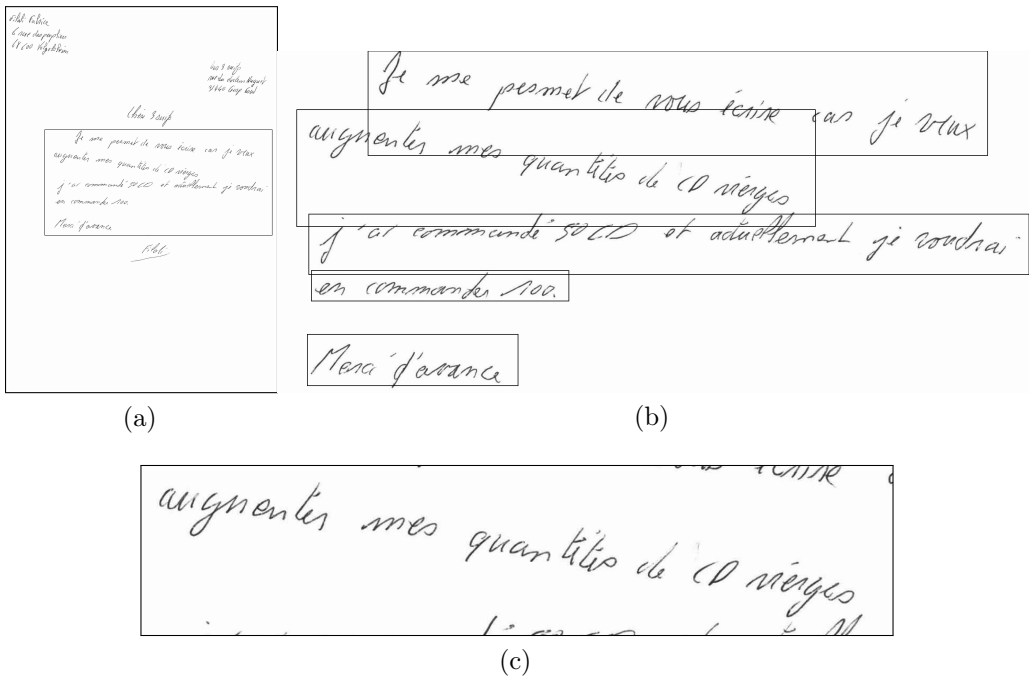
Prétraitement des images de lignes

2.1 Introduction

Les lignes de textes sont en général issues de blocs de texte, eux-mêmes issus du document. Comme les lignes peuvent être penchées, il est difficile de les isoler complètement. Un exemple de découpage des lignes est donné sur la figure 2.1. Même si les lignes sont parfaitement droites, les hampes et les jambages des lignes précédentes et suivantes apparaissent.

Dans un premier temps, nous détaillons les bases de données sur lesquelles nous avons mené nos expériences (section 2.2.2). Nous en fournissons un portrait détaillé afin d'en comparer les spécificités.

Dans un deuxième temps, nous présentons notre contribution sur le prétraitement des images. Nous proposons une chaîne complète de prétraitement au niveau ligne. Cette chaîne comprend deux méthodes originales de nettoyage des lignes de texte et de correction de la position de la ligne de base basse. Ces prétraitements ne sont pas spécifiques à un alphabet, une langue ou une base. Le nettoyage des lignes vise à supprimer les composantes des lignes voisines mais aussi à blanchir l'arrière-plan comme détaillé dans la section 2.3. La correction de la ligne de base permet quant à elle de faciliter l'extraction des caractéristiques par fenêtre glissante. Nous proposons une approche locale pour corriger la ligne de base dans la section 2.4.



(a) (b)

(c)

FIGURE 2.1 – (a) Exemple de courrier et de bloc de texte, (b) rectangle délimitant les lignes du bloc de texte, (c) exemple de ligne découpée (Base RIMES).

2.2 Bases de données

2.2.1 Bases de données et compétitions

La création de bases de documents disponibles à l'ensemble de la communauté scientifique est nécessaire pour évaluer et comparer les systèmes de traitement et de reconnaissance de documents. Disposer d'une base d'une taille assez importante permet également de présenter des résultats statistiquement significatifs. Les documents peuvent être collectés depuis des sources existantes (documents historiques, archives,...) ou être générés en demandant à des personnes (scripteurs) d'écrire de nouveaux documents en suivant un protocole précis. La plupart du temps, ces bases sont créées à l'initiative de laboratoires ou d'instances gouvernementales.

Générer une nouvelle base présente un intérêt particulier, celui de contrôler la difficulté de la tâche. En effet, la reconnaissance d'écriture présente plusieurs types de difficulté :

— la qualité générale du document : résolution, qualité de la numérisation, présence de bruits et d'artefacts divers

— la contrainte sur l'écriture : l'écriture peut être précasée (remplissage de champs),

guidée par des lignes ou complètement libre
— le nombre de scripteurs utilisés : mono-scripteur, multi-scripteur, omni-scripteur
— le vocabulaire : taille, spécificité, ouvert/fermé
— le type d'écriture : scripte, cursive, mixte

En construisant des bases avec un protocole précis, des tâches spécifiques vont pouvoir ainsi être évaluées, par exemple la reconnaissance de courriers manuscrits comme dans le cas de la base RIMES (section 2.2.2).

À partir de ces bases de données, l'évaluation des systèmes peut avoir lieu lors de compétitions de reconnaissance. Devant la variété des systèmes, des conditions expérimentales, la compétition est le meilleur cadre actuellement disponible pour comparer les systèmes. Les compétitions ont pour la plupart d'entre elles le même fonctionnement. Dans un premier temps, des données (base d'apprentissage) sont fournies aux laboratoires pour l'apprentissage de leurs modèles. Dans un deuxième temps, des données inédites (base d'évaluation) sont envoyées aux laboratoires pendant un temps court par rapport à la phase d'apprentissage. Les laboratoires vont appliquer leurs modèles sur ces nouvelles données et renvoyer leurs sorties (transcriptions) à l'institution organisant la compétition. Dans un dernier temps, le laboratoire évalue et publie les résultats de la compétition en suivant un protocole précis. Ce cadre varie selon les compétitions. Une étape dite de test à blanc (*dryrun*) peut notamment être ajoutée avant l'évaluation. Cette phase est similaire à l'évaluation. Elle permet également de s'assurer que tout fonctionne correctement dans la chaîne de traitement afin que rien ne vienne perturber la véritable étape d'évaluation. Une base de validation peut également être fournie lors de la phase d'apprentissage. Elle sert au réglage des paramètres, à leur optimisation. Elle présente des caractéristiques proches de la base d'évaluation.

Dans le cadre de la reconnaissance de textes multilingues, nous avons principalement travaillé sur deux bases de données, l'une écrite en français (RIMES) et l'autre en arabe (OpenHaRT). De façon secondaire, nous avons également testé certaines méthodes sur l'anglais (IAM). Leurs caractéristiques (modalités d'écriture, nature des textes, taille du vocabulaire,...) sont très différentes et sont détaillées dans la suite de cette section.

2.2.2 Base RIMES

La base RIMES (Reconnaissance et Indexation de données Manuscrites et de fac similÉS / Recognition and Indexing of handwritten documents and faxes) regroupe différents types de documents manuscrits écrits en Français : courriers, formulaires et fax. Elle fut créée sous l'impulsion des Ministères de la Défense et de la Recherche dans le cadre du projet TechnoVision auquel ont participé la DGA, Télécom ParisTech, le LITIS, l'INRIA et la société A2iA. La création de cette base a pour objectif d'évaluer les systèmes de reconnaissance automatique et d'indexation de courriers manuscrits. La base des courriers a été composée par des volontaires à qui il a été demandé d'écrire de manière libre des lettres suivant un des neuf scenarii suivants : changement d'information personnelle, demande d'information, ouverture ou fermeture de compte, modification de contrat, commande, plainte, difficultés de paiement, lettre de rappel ou déclaration de sinistre. Les lettres sont écrites sur du papier blanc au stylo noir sans guide-lignes et sont échantillonnées à 300 dpi en niveaux de gris.

Depuis sa création en 2006 [Grosicki *et al.*, 2006], plusieurs compétitions de reconnaissance de mots ont été organisées grâce à cette base de courriers [Grosicki et El-Abed, 2009, Grosicki et El-Abed, 2011] et en 2011 a eu lieu la première compétition de reconnaissance de blocs de mots [Grosicki et El-Abed, 2011].

Les documents de la base RIMES présentent des lignes pentues, inclinées et bruitées (figure 2.2) qui nécessiteront le prétraitement précisé dans la suite du chapitre.

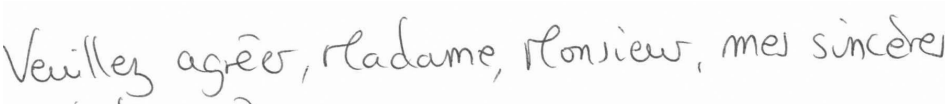
Voici les bases que nous utilisons pour nos expériences :

— La base des mots de Rimes 2011, divisée en trois parties : apprentissage, validation et test regroupant respectivement 51 738, 7 484 et 9 880 images de mots.

— La base des lignes de Rimes 2011 comprenant 11 329 images de lignes de texte. Elles sont extraites de 1500 blocs de textes. Pour nos expériences, nous avons partagé cette base en une base d'apprentissage de 10 329 lignes (1 370 courriers) et de validation de 1 000 lignes (130 courriers). La base de test de Rimes 2011 comprend 778 lignes extraits de 100 blocs de test. Les images de ligne de textes ont une taille d'environ 2000 x 150 pixels. Les images de lignes de texte peuvent

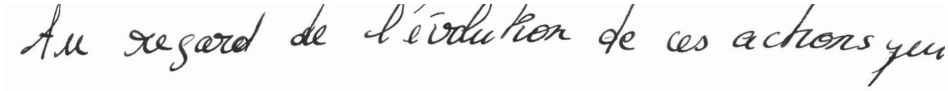
(a)

(b)



(c)

(d)



(e)

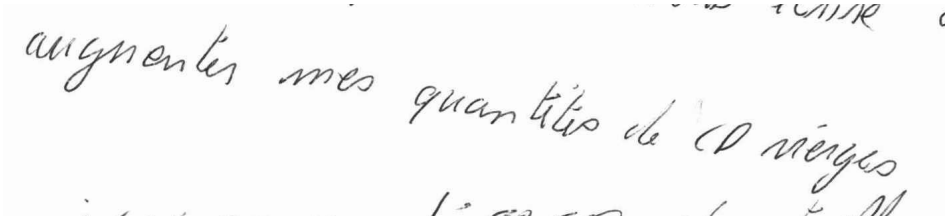


FIGURE 2.2 – Images de lignes de texte de la base RIMES : (a-b-c) ligne de base fluctuante. (d) écriture inclinée. (e) bruit d'arrière plan dû aux lignes environnantes et ligne de base pentue.

être extraites des blocs de texte par des boîtes englobantes des lignes (*bounding boxes*) fournies par les créateurs de la base.

Étant donnée la spécificité des courriers, la taille du vocabulaire de cette base est intermédiaire. Par exemple, la base d'apprentissage de lignes comprend environ 6 000 mots. Les textes comprennent beaucoup de mots considérés comme hors-vocabulaire (*out-of-vocabulary* - OOV). Ce sont essentiellement des entités nommées (noms, lieux, ...), des codes (bancaires, postaux,...). De plus, les courriers présentent des tournures idiomatiques comme des formules de politesse.

2.2.3 Base OpenHaRT

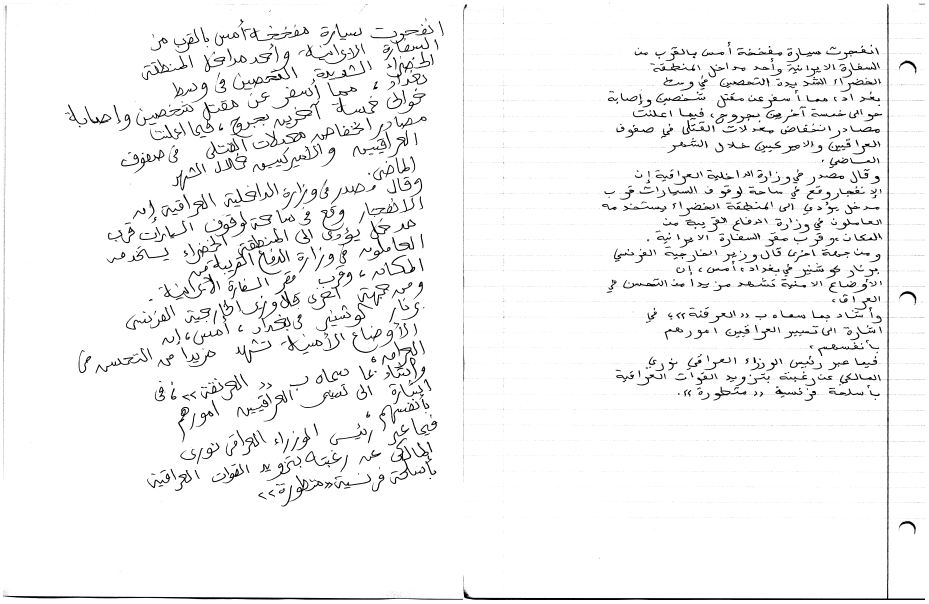
La base OpenHaRT (Open Handwriting Recognition and Translation Evaluation) a été introduite en 2010 par le NIST (*National Institute of Standards and Technology*) afin d'effectuer des évaluations des systèmes de reconnaissance sur une base manuscrite de très grande taille [NIST, 2013b].

L'écriture des documents fut réalisée dans un environnement contrôlé : des personnes dont la langue maternelle est l'arabe ont recopié des extraits de texte en arabe (455 scripteurs au total). Ces extraits leur étaient fournis sous format électronique et provenaient de la presse internet, du contenu de pages web, de blogs et de discussions sur des forums en ligne. Pour l'évaluation, les documents ont été regroupés en deux catégories principales :

— "Newswire" (NW) regroupe les textes formels ou structurés, par exemple la presse internet.

— "Web text" (WB) regroupe les textes informels ou non-structurés, par exemple des discussions sur des forums en ligne.

Chaque extrait était copié par trois scripteurs et scanné à la résolution de 600 dpi en noir et blanc. Un scripteur pouvait recopier plusieurs extraits. Certains scripteurs qui ont participé à l'écriture de la base d'apprentissage ont aussi participé à celle des bases de validation. Les conditions d'écriture sont récapitulées au sein du tableau 2.1. La qualité de la recopie varie énormément avec la vitesse d'écriture du scripteur. Les guide-lignes



(a) (b)

FIGURE 2.3 – Exemple d'un même document recopié par deux scripteurs (extrait de la base OpenHaRT)

peuvent apparaître sur le document scanné (figure 2.3b). Deux exemples de documents sont donnés sur la figure 2.3.

Les évaluations OpenHaRT ont pour origine celles du programme MADCAT (Multilingual Automatic Document Classification Analysis and Translation) initié par le DARPA (Defense Advanced Research Projects Agency). À la différence des évaluations MADCAT réservées aux contractants du DARPA, les évaluations proposées par OpenHaRT sont ouvertes à tous.

TABLE 2.1 – Répartition des différents paramètres d'écriture sur la base OpenHaRT

Instrument d'écriture	Papier	Vitesse d'écriture du scripteur
90% stylo à bille	75% sans guide-lignes	90% normale
10% crayon	25% avec guide-lignes	5% rapide
		5% attentive

La première évaluation OpenHaRT a eu lieu en 2010 sur des mots isolés [NIST, 2010]. Elle comprenait également une tâche de traduction. L'évalution menée en 2013 [NIST, 2013a] comprenait une reconnaissance de lignes de texte sans segmentation en mots.

Nos expériences ont été menées sur la base de lignes d'OpenHaRT de la compétition menée en 2013 :

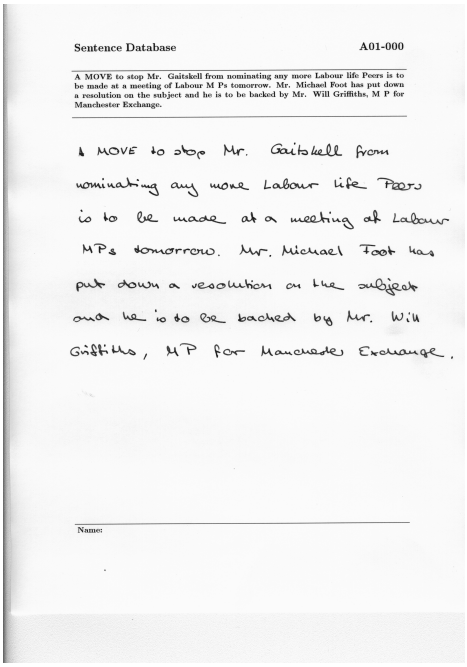
— La base d'apprentissage "MADCAT_Phase1_Training_Data" : 9693 documents, 579360 images de lignes

— La base de validation "DRYRUN" : 534 documents, 10799 lignes

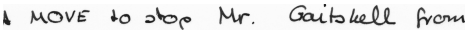
— La base de test "EVAL" : 633 documents, 12644 lignes

Les coordonnées des boîtes englobantes des mots sont fournies au même titre que celles des lignes dans des fichiers xml. Voici un extrait de fichier xml présentant les coordonnées d'une ligne ainsi que les coordonnées de deux mots de cette même ligne :

```
(...)
<zone id="z0" type="line">
        <polygon>
                <point x="3615" y="1377"/>
                <point x="4887" y="1377"/>
                <point x="4887" y="1575"/>
                <point x="3615" y="1575"/>
        </polygon>
        <token-image id="t1">
                <polygon>
                        <point x="4664" y="1460"/>
                        <point x="4887" y="1460"/>
                        <point x="4887" y="1536"/>
                        <point x="4664" y="1536"/>
                </polygon>
        </token-image>
        <token-image id="t2">
                <polygon>
                        <point x="4324" y="1424"/>
                        <point x="4600" y="1424"/>
```



(a)



(b)

FIGURE 2.4 – (a) Exemple de formulaire de la base IAM, (b) ligne de texte extraite

```
                    <point x="4600" y="1571"/>
                    <point x="4324" y="1571"/>
                </polygon>
            </token-image>
(...)
```

Dans cet extrait, les coordonnées correspondent aux quatre coins des boîtes englobantes des mots t1 et t2 ou de la ligne z0. En raison de la variété des documents et de la taille de la base, le vocabulaire est très large (38 000 mots pour la seule base d'apprentissage). En plus des caractères arabes, les extraits comprennent de nombreux chiffres et lettres en caractères latins.

2.2.4 Base IAM

La base d'écriture manuscrite IAM comprend des formulaires écrits en anglais. Elle a été présentée pour la première fois à la conférence ICDAR 1999 [Marti et Bunke, 1999] et était destinée à tester les systèmes de reconnaissance d'écriture manuscrite mais aussi ceux

d'identification de scripteur. La base comprend des mots isolés ainsi que des lignes reconstituées à partir des vignettes de mots. Ceci garantit que la ligne reconstituée est dénuée du bruit des lignes précédentes et suivantes. Les scripteur devant suivre des guide-lignes suffisamment espacés, ce cas se présente rarement. Pour la même raison, les documents présentent peu de lignes pentues. Beaucoup de scripteurs de cette base écrivent en script, plutôt qu'en cursif. Les formulaires ont été scannées à la résolution de 300 dpi en niveaux de gris. La figure 2.4 donne un exemple de formulaire et de ligne segmentée.

Nous avons travaillé sur la base des lignes d'IAM :
— Train : 6 161 lignes de texte
— Validation 1 : 900 lignes de texte
— Validation 2 : 940 lignes de texte
— Test : 1 861 lignes de texte
Le vocabulaire de la base d'apprentissage (Train) comprend environ 10 000 mots.

2.2.5 Comparaison des bases étudiées

Les trois bases sur lesquelles nous avons travaillé présentent des caractéristiques très différentes. Ces caractéristiques vont avoir un effet à différents niveaux du système de reconnaissance.

Tout d'abord, le volume des documents est très différent : deux bases sont petites (RIMES et IAM) et une est grande (OpenHaRT). Le volume de données induit une problématique de temps de calcul pour l'apprentissage et le décodage.

Les vocabulaires sont de tailles différentes et le contenu des bases est très différent. Ces caractéristiques devront être prises en compte lors de la création des dictionnaires et modèles de langage pour le décodage. Les deux bases constituées de recopies (OpenHaRT et IAM) vont contenir moins de fautes d'orthographe que la troisième qui est plus libre (RIMES).

Les conditions physiques de l'écriture vont également demander des prétraitements différents. Si la base IAM permet de disposer de lignes droites et sans composantes des lignes voisines, la base RIMES présente souvent du bruit ainsi que des lignes inclinées. Les lignes d'OpenHaRT, si l'on se contraint à ne pas utiliser les coordonnées des mots,

peuvent aussi inclure des composantes des lignes précédente et suivante. Elles peuvent présenter également des lignes inclinées.

2.3 Nettoyage des images de lignes de texte

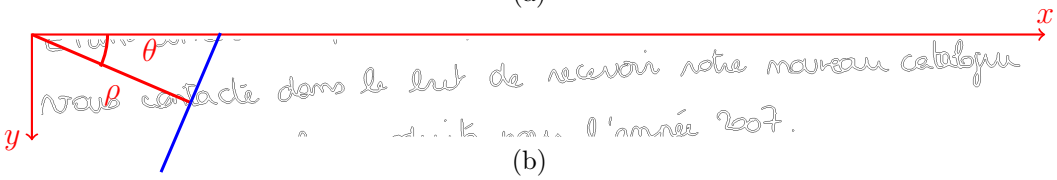
Des méthodes existent pour segmenter directement un document complet en lignes de texte [Likforman-Sulem *et al.*, 2007]. Des compétitions ont été organisées sur ce thème [Gatos *et al.*, 2010]. Principalement ces méthodes reposent sur des approches ascendantes (*bottom-up*) ou descendantes (*top-down*) selon que les composantes de l'image sont respectivement regroupées ou divisées. Parmi ces méthodes, certaines travaillent à différentes résolutions de l'image [Lemaitre *et al.*, 2011].

Notre segmentation en lignes utilise les boîtes rectangulaires englobantes dont les coordonnées sont fournies dans les bases RIMES et OpenHaRT (voir sections 2.2.2 et 2.2.3). Cependant cette segmentation est imparfaite : elle peut introduire des traits, hampes, jambages voire même des mots provenant de lignes voisines dans les régions hautes et basses des images de ligne. En effet lorsque les textes sont écrits sans guide-ligne, nombreuses sont les lignes penchées ou trop serrées les unes aux autres. Ces hampes et jambages correspondent à des composantes connexes excentrées qui perturbent la suite du prétraitement, notamment la correction de la pente des lignes ainsi que l'extraction des lignes de base et des caractéristiques (voir section 1.3). Nous devons donc nettoyer les lignes des composantes parasites.

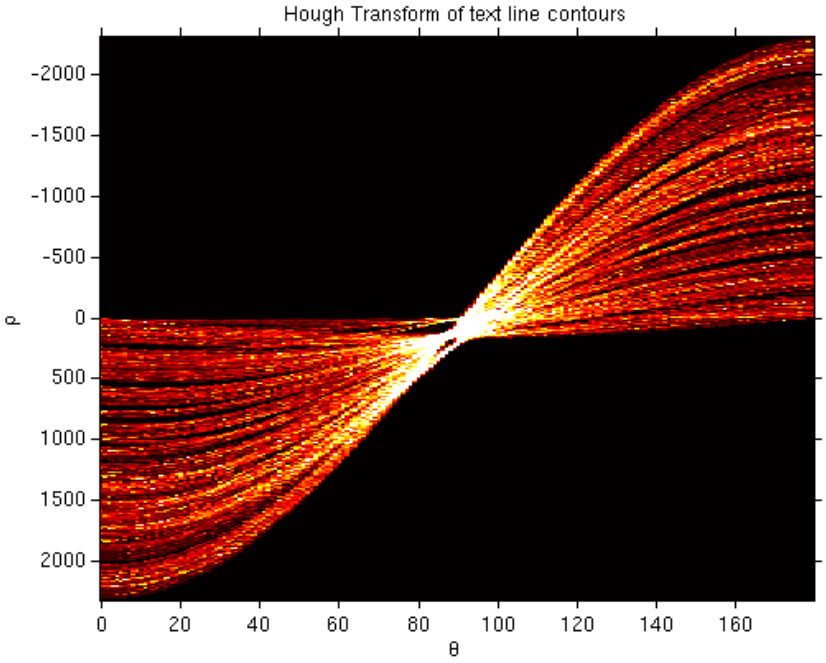
Notre méthode s'appuie sur une extraction de l'axe principal de l'écriture. Cet axe définit la position prise par la ligne centrale d'écriture que l'on souhaite conserver. Ensuite les composantes éloignées de l'axe principal, considérées comme du bruit, sont supprimées. La décision de garder ou non une composante dans la ligne d'écriture est prise à l'aide de seuils estimés pour chaque image de ligne et s'adapte donc à l'écriture de chaque scripteur. Notre méthode comprend les étapes suivantes :

1. L'image des contours est obtenue après une extraction de contours sur l'image binarisée (figure 2.5b). La transformée de Hough sur l'image des contours permet ensuite d'identifier l'axe principal de l'image. En effet, cette ligne correspond au

(a)



(b)



(c)

(d)

(e)

(f)

FIGURE 2.5 – Processus de nettoyage des lignes de texte appliqué à la base RIMES : (a) image de ligne. (b) extraction des contours et repère polaire. (c) matrice (ρ, θ) de la transformée de Hough appliquée aux contours. (d) ligne bleue : axe principal trouvé par Hough ; lignes en pointillés délimitant les seuils de décision ; croix rouges : centres des composantes connexes. (e) retrait des composantes parasites. (f) arrière plan binarisé.

pic le plus important dans le graphe de Hough (ρ,θ) (voir figure 2.5c). Ce pic correspond à la ligne continue sur la figure 2.5d.

2. h_{mean} and h_{max}, respectivement les hauteurs moyennes et maximales des composantes connexes de l'image de la ligne sont estimées sur les composantes qui ne sont pas en contact avec les bords haut et bas de l'image.

3. Un ensemble initial des composantes potentiellement bruitées S_c est créé à partir des composantes connexes éloignées de la ligne de texte principale : leur distance à l'axe de la ligne doit être supérieure au seuil $h_{mean}/2$ calculé précédemment.

4. À partir de l'ensemble S_c constitué, on extrait le sous-ensemble $S_n \subset S_c$ des composantes bruitées à supprimer. Ces composantes en plus d'appartenir à l'ensemble S_c doivent être en contact avec les limites hautes et basses de l'image ou avoir un centre de gravité considéré comme périphérique : la distance de leurs centres de gravité à l'axe principal doit être supérieure à h_{max}, calculé dans l'étape 2.

L'étape 1 permet de s'assurer que l'on trouve l'axe principal et cela malgré la présence éventuelle de fragments des lignes avoisinantes ou d'une ligne inclinée. En effet, l'image contient majoritairement des composantes de la ligne de texte disposées selon un axe précis. L'image est binarisée avec un seuillage d'Otsu [Otsu, 1979] qui sépare l'histogramme des valeurs des pixels en deux classes en minimisant la dispersion intra-classe. Le filtre de Canny permet une extraction des contours selon les deux directions x et y par application de deux filtres de convolution C_x et C_y respectivement de dimension 3×1 et l'autre 1×3 (équation 2.1).

$$C_x = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix};\ C_y = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \tag{2.1}$$

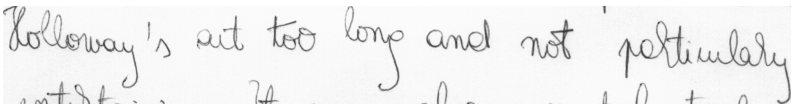
La matrice de la transformée de Hough est une représentation polaire : les axes de l'image peuvent être définis par l'équation $\rho = x\cos(\theta) + y\sin(\theta)$ où x correspond à l'abscisse et y à l'ordonnée (voir figure 2.5b). Le paramètre ρ correspond à la distance entre l'origine du repère et l'axe principal. Le paramètre θ correspond à l'angle entre l'axe des abscisses et la droite orthogonale à la ligne (représentée en bleu). La résolution angulaire de θ est ici de 1 °.

Le pic maximum dans la transformée de Hough (ρ,θ) (figure 2.5c) existe toujours et correspond à l'axe principal de la ligne de texte car l'image contient majoritairement la ligne de texte qui est disposée selon cet axe. Si l'image correspond à un mot, l'axe aura la même pente que le mot. La précision de l'estimation de cet axe est suffisante : l'objectif ici n'est pas d'estimer précisément la pente de la ligne. Cette question sera abordée dans la section suivante (section 2.4). Ici nous souhaitons seulement disposer d'un repère à partir duquel les seuils de décision seront donnés. Dans une version antérieure de notre méthode, l'axe était arbitrairement choisi comme la ligne horizontale séparant l'image en deux aires égales [Morillot *et al.*, 2012a].

Les estimations des seuils de l'étape 2 ne sont effectuées que sur des composantes qui ne sont pas en contact avec les bords afin d'éviter d'introduire un biais dans les valeurs estimées sur des composantes coupées, qu'elles appartiennent ou non à la ligne. Ces estimations faites sur chaque image de ligne permettent d'adapter les seuils de décision à chaque scripteur. Par exemple, la taille des hampes et des jambages est très variable et justifie de s'adapter à chaque nouvelle ligne.

Le premier ensemble S_c des composantes potentiellement bruitées dans l'étape 3 n'inclut pas les composantes de la ligne et ce même si elles n'ont pas d'intersection avec l'axe extrait par la transformée de Hough. On conserve les composantes dont le centre de gravité est situé dans la bande délimitée par les lignes roses de largeur h_{mean} (figure 2.5d). Les composantes à conserver à cette étape correspondent principalement au corps du texte. À ce stade, des accents ou des éléments de ponctuation peuvent ne pas avoir été inclus dans la ligne centrale d'écriture. En effet, la perte d'un accent ou d'une virgule à cette étape est moins préjudiciable que celle des lettres d'un mot.

L'étape 4 permet de considérer comme bruitées non seulement les composantes coupées par les bords supérieurs et inférieurs de l'image mais aussi celles suffisamment éloignées de l'axe principal. Sur la figure 2.5a, par exemple, on distingue la composante "2007" appartenant à la ligne suivante mais ne touchant pas le bord inférieur de l'image. Le seuil choisi de h_{max} permet de conserver les composantes dont la position ne dépasse pas les plus hautes composantes de la ligne. Cette distinction permet d'identifier les signes qui ne s'appuient pas sur la ligne de base de l'écriture comme les accents ou les apostrophes. Conserver les composantes au-delà de ce seuil amène à intégrer notamment les virgules de



(a)

(b)

FIGURE 2.6 – Exemple de nettoyage d'une image de ligne de texte écrite en anglais (IAM)

la ligne de texte précédente.

Les composantes classées comme bruit sont ensuite soustraites directement de l'image originale en niveaux de gris (figure 2.5e).

Notre nettoyage de la ligne de texte peut comprendre des erreurs : des accents appartenant à la ligne de texte peuvent être supprimés. Nous avons choisi de ne pas intégrer des heuristiques de décision plus précises afin de conserver la généralité de notre méthode. Il serait notamment possible de traiter différemment la zone située au dessus de l'axe puisque celle-ci contient potentiellement des accents à la différence de la partie inférieure. Or cette distinction est propre à certaines langues. Notre méthode s'applique généralement à l'écriture cursive : elle a été testée sur le français sur la base RIMES (présentée en section 2.2.2), l'anglais (figure 2.6) sur la base IAM (présentée en section 2.2.4) et l'arabe (figure 2.7) sur la base OpenHaRT (présentée en section 2.2.3). Bien que les coordonnées des mots soient disponibles pour les bases IAM et OpenHaRT, nous avons effectué une nouvelle extraction des lignes à partir de leurs boîtes englobantes afin que les composantes des lignes précédentes et suivantes puissent apparaître.

Nous aurions pu choisir d'optimiser les seuils de décision binaire (distances à l'axe principal) en mesurant un taux de bonnes détections de composantes bruitées et un taux de fausses alarmes. Toutefois cette approche présente plusieurs défauts. D'une part, elle nécessite d'étiqueter les composantes une par une. D'autre part, les seuils auraient été optimisés globalement sur l'ensemble de la base et n'auraient donc pas pris en compte les différences entre les scripteurs. Dans notre approche, ces seuils sont calculés automatiquement pour chaque image.

(a)

(b)

FIGURE 2.7 – Exemple de nettoyage d'une image de ligne de texte écrite en arabe (OpenHaRT)

De par son analyse en composantes connexes de l'image, notre approche ne peut séparer des composantes de deux lignes différentes si celles-ci sont en contact : elles doivent être traitées auparavant lors de la segmentation en lignes de texte.

Nettoyage de l'arrière plan Le processus de numérisation peut induire un bruit dans l'arrière plan des imagettes de ligne. Pour éliminer ce bruit, l'arrière plan est blanchi tout en conservant la dynamique des niveaux de gris des traits d'écriture (figure 2.5f) qui sera utilisée pour l'extraction des caractéristiques (section 1.3). Un seuil de binarisation est estimé sur chaque imagette par la méthode d'Otsu et les valeurs de pixels supérieures à ce seuil sont ramenées à 255 tandis que les autres conservent leur valeur de gris.

L'apport de ce prétraitement dans la chaîne de reconnaissance sera évaluée en section 3.6 pour le système fondé sur les HMM et en section 4.9 pour le système par BLSTM. Notre approche de nettoyage des images de ligne de texte peut être retrouvée dans [Morillot *et al.*, 2013c].

2.4 Correction de la ligne de base : approche locale

Comme nous l'avons détaillé dans le chapitre 1, la correction de la ligne de base est un préalable à la plupart des systèmes de reconnaissance de lignes. Seuls les réseaux de neurones MDRNN qui utilisent plusieurs directions pour analyser les données ne nécessitent pas toujours cette étape. Les autres systèmes de reconnaissance analysent le signal suivant la seule direction horizontale, souvent grâce à une fenêtre glissante. Il est donc important de s'assurer que les composantes de l'écriture soient correctement

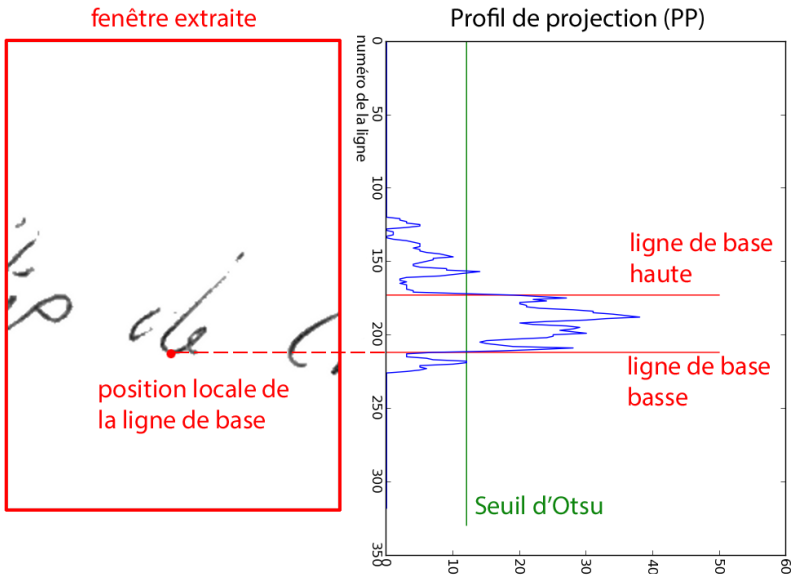
agencées sur une ligne droite. En effet, l'extraction des caractéristiques demande souvent de déterminer les trois régions de l'écriture (hampes, zone centrale et jambage) séparées par les lignes des base basse et haute. En particulier l'ensemble de caractéristiques que nous utilisons comprend des densités, des positions relatives à ces lignes de base.

Le phénomène de l'écriture penchée est d'autant plus sensible que l'on travaille directement au niveau des lignes de texte. Les principales approches de correction au niveau ligne reposaient soit sur une correction globale soit par une correction par morceaux. Ces méthodes ont pour principal défaut de ne pas prendre en compte les variations de position de la ligne de base au sein même des mots. L'approche de correction locale par détection de contour proposée par [España-Boquera *et al.*, 2011] et détaillée dans la section 1.2 ne prend pas en compte ces variations. De plus, elle repose sur l'apprentissage d'un classifieur et nécessite une segmentation de la ligne.

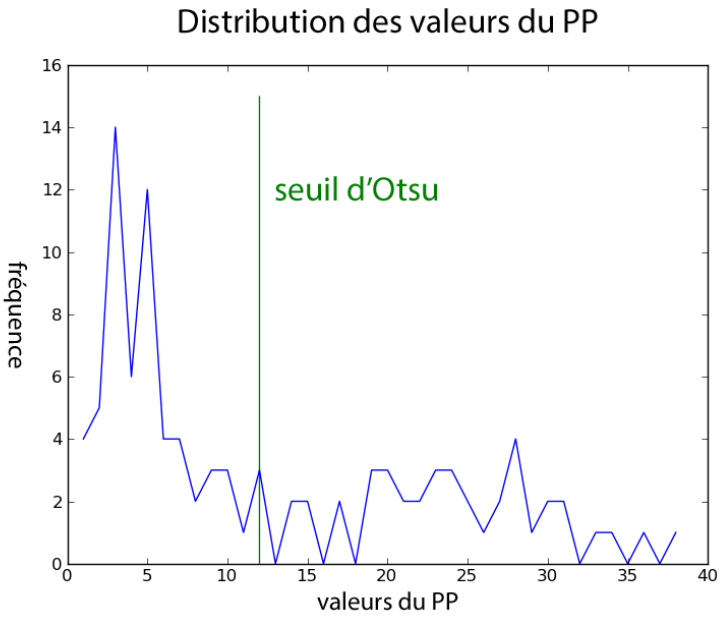
Nous proposons ici une méthode inédite de correction locale de la ligne de base de l'écriture sans segmentation ni détection des composantes connexes. De plus, elle ne repose pas sur un apprentissage statistique. L'objectif est d'estimer la ligne de base basse localement, colonne par colonne par une approche à fenêtre glissante afin de la rectifier. Une fenêtre d'analyse de largeur w_e va parcourir l'image de la ligne de texte de gauche à droite et estimer la position de la ligne de base basse. L'utilisation d'une fenêtre d'analyse nous permet d'avoir un estimateur assez robuste à la présence des jambages ou des espaces (Figure 2.9a). Pour la colonne centrale de pixels de cette fenêtre, la position y de la ligne de base locale est estimée en analysant le profil de projection suivant la direction horizontale sur l'axe vertical [Vinciarelli et Luettin, 2001] de l'ensemble de la fenêtre :

— Premièrement, le profil de projection vertical (PP) est généré en comptant, pour chaque position y, le nombre de pixels noirs selon la direction horizontale (figure 2.8a et algorithme1).

— Ensuite la distribution (histogramme) (figure 2.8b) de ces valeurs de profil de projection est calculée. Cet histogramme doit être bimodal puisque l'on suppose que la zone centrale du texte a une densité de pixels plus importante que les autres zones de l'image. Cette hypothèse est valable car les composantes issues des autres lignes ont été nettoyées auparavant.



(a)



(b)

FIGURE 2.8 – (a) Profil de projection (PP) obtenu par une projection des pixels sur l'axe vertical suivant la direction horizontale. (b) Distribution des valeurs du PP et seuil extrait par la méthode d'Otsu. Le seuil est ensuite reporté sur le PP pour extraire les positions des lignes de base haute et basse.

— Enfin, un seuillage (méthode d'Otsu) est appliqué à la distribution précédemment calculée. La zone centrale du texte correspond à la plus grande des zones centrales continues, au dessus du seuil sur le profil de projection (algorithme 2). On obtient donc, à l'issue de ces étapes, la position de la ligne de base basse pour chaque pixel central d'une fenêtre glissante.

La courbe des positions locales de la ligne de base est ensuite lissée à l'aide d'un filtre gaussien de largeur w_{smooth} afin de se débarrasser de ses discontinuités (figures 2.9b, 2.10d). L'écart type de la distribution gaussienne est défini à partir de la largeur de la fenêtre w_{smooth} : $\sigma = w_{smooth}/(4\sqrt{2})$. Sans cette étape de lissage, des artefacts de cisaillement peuvent apparaître sur les images corrigées. L'étape de correction consiste à translater les pixels noirs de la valeur de la ligne de base estimée pour la colonne de pixels.

Un pseudo-code de l'algorithme de correction de la ligne de base est fourni dans l'algorithme 1. La sous-fonction présentée dans l'algorithme 2 permet quant à elle de déterminer la position de la ligne de base pour un PP donné. Une structure core_regions est créée pour accueillir les positions extrêmes des zones centrales détectées sur le PP. La commande core_regions[i][1] permet d'accéder à la position d'entrée dans la i-ème zone, tandis que core_regions[i][2] en donne la sortie. Ainsi la longueur de la $i$ème zone centrale est donnée par core_regions[i][2]-core_regions[i][1].

Les figures 2.9a et 2.10c montrent des exemples d'échantillons de lignes de texte. La ligne de base fluctuante de la figure 2.9a est corrigée sur la figure 2.9c. La ligne de texte inclinée et fluctuante présentée sur la figure 2.10c est corrigée sur la figure 2.10e : la ligne de texte résultante présente encore des fluctuations qui sont trop faibles pour avoir un effet sur le processus d'extraction des caractéristiques décrit dans la section 1.3.

Afin d'optimiser les valeurs des paramètres w_e et w_{smooth}, nous aurions pu le faire en considérant un critère sur l'image. Ce critère aurait pu être l'étendue moyenne du profil de projection global de l'image de ligne sur l'axe vertical. Cependant, notre objectif est de réduire le taux d'erreur WER et non de fournir une image ayant un meilleur visuel. La qualité visuelle d'une image pour l'œil humain est subjective. Nous avons donc choisi d'optimiser les paramètres w_e et w_{smooth} en fonction du taux d'erreur WER . Cette optimisation est détaillée dans 3.5 pour le reconnaisseur HMM et son apport pour les deux systèmes de reconnaissance, HMM et BLSTM, est évalué respectivement dans les sections

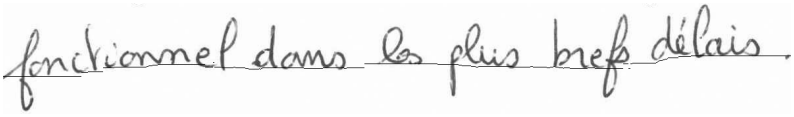
Algorithme 1 Pseudo-code pour la correction de la ligne de base

$I(1:m, 1:n)$ ▷ Image de ligne avec m lignes et n colonnes
Définir w_e, w_{smooth}

Estimation de la ligne de base locale
Soit $baseline(j) = 0$ $(j = 1:n)$
for $j = 1 \rightarrow n$ **do**
  $I_w = I(1:m, j - w_e/2 : j + w_e/2)$
  $PP = projectionprofile(I_w)$
  $hist = histogram(PP)$
  $threshold\_value = Otsu\_method(hist)$
  $baseline(j) =$FINDLOWERBASELINE $(PP, threshold\_value)$
    ▷ Programme détaillé dans Algo. 2
end for

Lissage de la ligne de base par filtrage gaussien
Soit $baseline\_smoothed(j) = 0$ $(j = 1:n)$
Créer gaussian_window de largeur w_smooth et d'écart-type $\sigma = w\_smooth/(4\sqrt{2})$
$baseline\_smoothed = filter(baseline, gaussian\_window)$

Correction
Soit $I_{corr}(1:m, 1:n)$ ▷ Image corrigée
$baseline_{mean} = \frac{1}{n}\sum_{i=1}^{n} baseline\_smoothed(i)$ ▷ Valeur moyenne de la ligne de base locale
for $j = 1 \rightarrow n$ **do**
  for $i = 1 \rightarrow m$ **do**
    if $I(i, j)$ is foreground pixel **then**
      Soit $\Delta = baseline\_smoothed(j) - baseline_{mean}$ ▷ Décalage vertical
      if $0 < i + \Delta < m$ **then**
        $I_{corr}(i + \Delta, j) \leftarrow I(i, j)$
      end if
    end if
  end for
end for



(a)

(b)

(c)

FIGURE 2.9 – Processus de correction de la ligne de base des lignes de texte. (a) Estimations locales de la position y de la ligne de base. (b) Lissage gaussien des valeurs de la ligne de base. (c) Correction de la ligne de base.

Algorithme 2 Pseudo-code pour l'extraction de la ligne de base basse

function FINDLOWERBASELINE(PP,threshold_value)
  $inside\_region = 0$ ▷ Booléen
  for $i = 1 \rightarrow length(PP)$ **do**
    if $PP(i) \geq threshold\_value$ and $inside\_region = 0$ **then**
      ▷ Début d'une région centrale
      $inside\_region \leftarrow 1$ ▷ Créer nouvelle région centrale
      ajouter [i,.] à core_regions ▷ Ligne de base haute
    elseif $PP(i) \leq threshold\_value$ and $inside\_region = 1$
      $inside\_region \leftarrow 0$ ▷ Fin d'une région centrale
      $core\_regions[length(core\_regions)][2] \leftarrow i$
      ▷ Ligne de base basse de la région
    end if
  end for
  if $length(core\_regions[length(core\_regions) - 1]) = 1$ **then**
    ajouter length(PP)-1 à core_regions[length(core_regions)-1]
  end if
  for $k = 1 \rightarrow length(core\_regions)$ **do**
    ajouter core_regions[k][2]-core_regions[k][1] à size_regions
    ▷ Hauteur de la k-ième région centrale
  end for
  Soit i_max l'indice de max(size_regions) ▷ Indice de la plus grande zone centrale
  return $core\_regions[i\_max][2]$ ▷ Ligne de base basse
end function

3.6 et 4.9. Le détail de notre méthode a été publié dans [Morillot *et al.*, 2013c, Morillot *et al.*, 2013a].

2.5 Correction de l'inclinaison de l'écriture

L'approche d'extraction des caractéristiques par fenêtre glissante est sensible à l'inclinaison de l'écriture (en anglais *slant*). En effet une écriture inclinée peut induire une superposition de traits appartenant à des caractères différents à l'intérieur de la fenêtre verticale. Plutôt que d'extraire les caractéristiques avec des fenêtres inclinées [Al-Hajj-Mohamad *et al.*, 2009], nous avons préféré redresser l'écriture. Nous utilisons une méthode fondée sur les profils de projection introduite par [Buse *et al.*, 1997, Vinciarelli et Luettin, 2001].

Une rotation est appliqué à l'image avec différents angles α ($-45° < \alpha < 45°$). Pour chaque image transformée, le profil de projection vertical sur l'axe horizontal est calculé. Ce profil permet d'estimer la valeur moyenne p_i des niveaux de gris, pour chaque colonne i. Pour chaque angle α, nous calculons une mesure proche d'une entropie $H_\alpha = -\sum_i^n p_i log(p_i)$ (où n est le nombre de colonnes de l'image). Les images avec les traits les plus verticaux auront une valeur H_α faible. En effet, p_i est alors proche soit de 0 soit de 1 et le produit $p_i log(p_i)$ est alors proche de 0. L'image subit ensuite une rotation de la valeur d'angle $\hat{\alpha}$ qui minimise H_α. Comme la ligne de base, l'inclinaison de l'écriture pourrait être estimée localement. Cependant, les variations de l'inclinaison sont beaucoup moins prononcées au sein de la ligne que les variations de la position de la ligne de base. Un exemple de correction de l'inclinaison est présenté sur la figure 2.10f.

2.6 Conclusion du chapitre 2

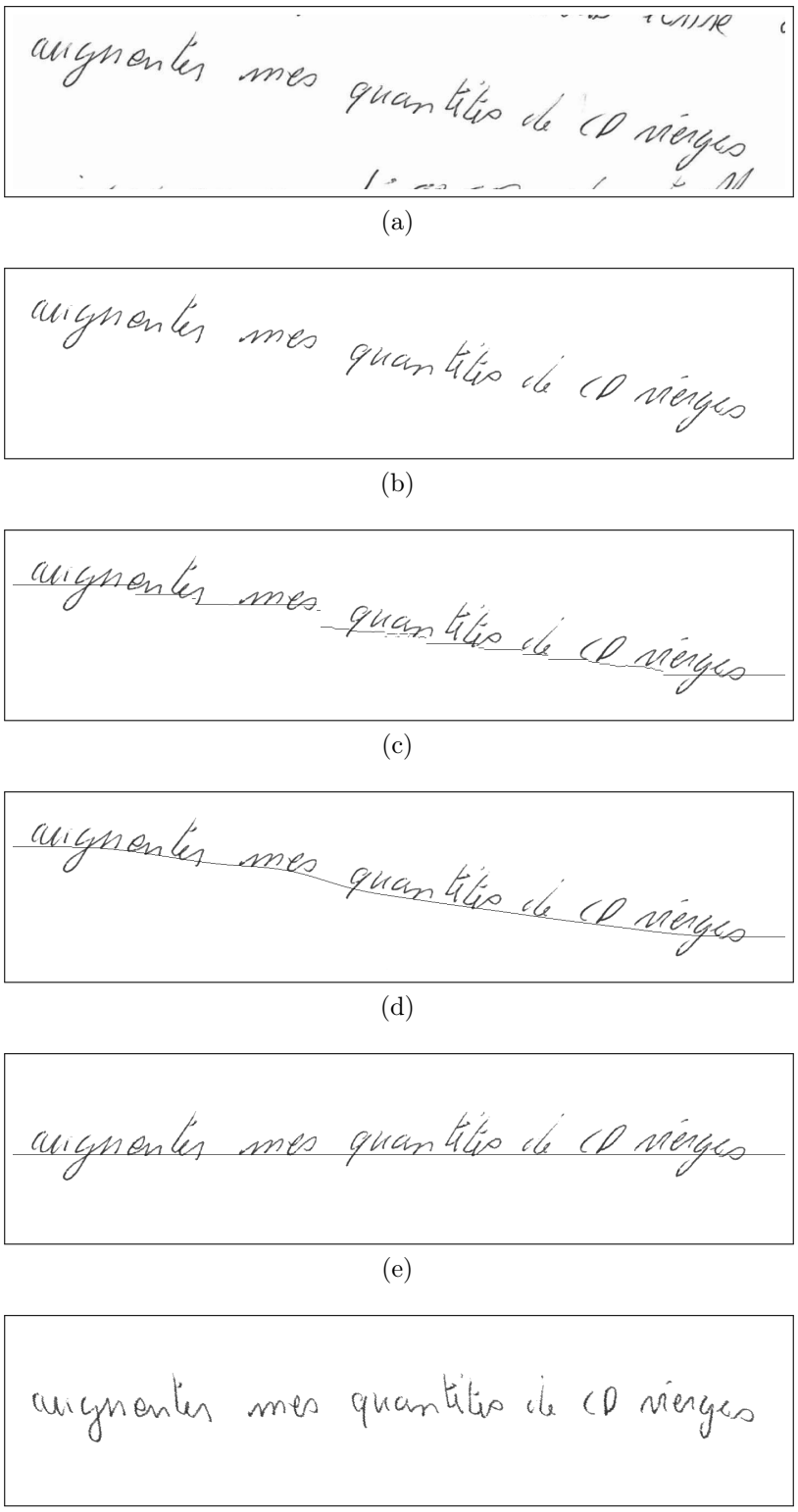
Dans ce deuxième chapitre, nous avons présenté une chaîne complète de prétraitement des lignes de texte. Avant de nous intéresser à l'apprentissage et au décodage des modèles, il est nécessaire de disposer de données propres et d'en réduire la variabilité. Pour ce faire, nous sommes intervenus sur le bruit dans la ligne de l'écriture (composantes des autres lignes et bruit dans l'arrière plan), la ligne de base et l'inclinaison de l'écriture.

Nous avons commencé par analyser les différents artefacts qui peuvent apparaître dans l'image d'une ligne d'écriture : la présence de composantes provenant d'autres lignes et la position fluctuante de la ligne de base. Nous avons proposé deux méthodes originales, une de nettoyage et une de correction, adaptées à la configuration des lignes de texte et qui ne nécessitent pas de segmentation en mots.

Pour nettoyer les lignes de texte bruitées par les lignes environnantes, nous avons effectué une transformée de Hough sur les contours de l'image afin d'extraire l'axe principal de l'écriture. À partir de cet axe, nous avons défini des seuils de décision permettant de classer les composantes connexes selon qu'elles appartiennent ou non à la ligne d'écriture centrale. Ces seuils présentent l'avantage d'être estimés automatiquement pour chaque image de ligne et sont donc adaptés à la morphologie de l'écriture de chaque scripteur. Notre méthode de nettoyage permet d'identifier les composantes parasites même quand elles ne sont pas en contact avec le bord de l'image ou quand l'écriture a une forte pente.

Ensuite, nous avons abordé le problème des fluctuations de la ligne de base. Plutôt que de corriger la pente de l'écriture globalement ou par morceaux, nous avons estimé et corrigé la ligne de base basse pour chaque colonne de pixels. Notre approche repose sur l'utilisation de fenêtres glissantes. Les valeurs des largeurs de ces fenêtres sont optimisées dans le chapitre suivant pour la base RIMES sur une sous-base indépendante. Cette méthode présente l'avantage d'être robuste au bruit et aux espaces puisque les valeurs estimées sont moyennées sur de très larges fenêtres.

L'ensemble du prétraitement est représenté dans la figure 2.10 et est intégralement détaillé dans [Morillot *et al.*, 2013c]. Une fois les images nettoyées et corrigées, l'extraction des caractéristiques et la reconnaissance des lignes de texte peut être réalisée (chapitres 3 et 4). Notre premier système, à base de HMM est présenté dans le chapitre suivant.



(a)

(b)

(c)

(d)

(e)

(f)

FIGURE 2.10 – Étapes de prétraitement : (a) recadrement de la ligne. (b) nettoyage de l'arrière plan. (c) estimation de la ligne de base locale. (d) lissage de la ligne de base. (e) correction de la ligne de base. (f) correction de l'inclinaison.

Chapitre 3

Système de reconnaissance de lignes par Modèles de Markov cachés

3.1 Introduction

Ce chapitre est consacré à la présentation de notre modèle de Markov caché (HMM) et à son optimisation pour la reconnaissance de lignes de texte. Nous avons adopté un HMM de type Bakis avec une modélisation contextuelle des caractères par trigraphes [Bianne-Bernard *et al.*, 2011] que nous présentons dans la section 3.2.

Ensuite, les paramètres d'extraction de caractéristiques sont optimisés pour les lignes de texte dans la section 3.3. Puis nous évoquons l'optimisation de la correction de la ligne de base (section 3.5).

Puis, nous présentons notre approche pour la création du modèle de langage adapté à la reconnaissance de courriers manuscrits dans la section 3.4. Nous poursuivons l'étude en optimisant son paramétrage dans la section 3.4.3.

Dans un troisième temps, nous évoquons l'optimisation de notre méthode de correction de la ligne de base (section 3.5). Les apports des deux prétraitements que nous proposons (nettoyage de l'arrière plan et correction de la ligne de base) pour la reconnaissance par HMM sont mesurés dans la section 3.6.

Enfin, nous présentons et discutons les résultats que nous avons obtenus lors d'une

compétition internationale de reconnaissance de français manuscrit (section 3.7).

3.2 Modélisation contextuelle des caractères

3.2.1 Motivation et démarche générale

La modélisation contextuelle des caractères vise à distinguer les graphies des caractères en fonction de leur contexte. Comme nous l'avons présenté dans la section 1.5.1, l'écriture cursive présente une grande variété de graphies pour un même caractère. Cette variété est essentiellement due aux ligatures qui relient le caractère à ses voisins.

La modélisation des caractères en contexte consiste donc à remplacer les modèles de caractère par des trigraphes. Ces trigraphes seront composés du caractère précédent (signalé par un "-"), du caractère dit central et du caractère suivant (signalé par un "+"). Par exemple dans le mot "Retour", le modèle "u" sera remplacé par le trigraphe "o-u+r". La principale difficulté que soulève cette modélisation est la multiplication des modèles : de n monographes, on passe à n^3 trigraphes possibles. Pour réduire le nombre des paramètres à estimer, nous avons suivi la démarche proposée par [Bianne-Bernard *et al.*, 2011] sur les mots isolés :

— Tout d'abord, une phase d'apprentissage permet d'apprendre les monographes du lexique d'apprentissage. À ce stade, on ne considère qu'une distribution gaussienne par état.

— Ensuite, les modèles de trigraphes sont initialisés en copiant simplement les modèles des caractères centraux appris à l'étape précédente. Seuls les trigraphes présents dans la base d'apprentissage sont créés. Les matrices de transition entre trigraphes sont partagées par les trigraphes qui ont le même caractère central. Les modèles de trigraphes sont ensuite réappris en considérant toujours une gaussienne par état.

— La seconde étape consiste à regrouper les états des trigraphes qui partagent le même caractère central et de les répartir ensuite dans différents ensembles (ou clusters). L'utilisation de clusters d'états va ainsi permettre de réduire le nombre de modèles à calculer. Cette étape est précisée dans la section 3.2.2.

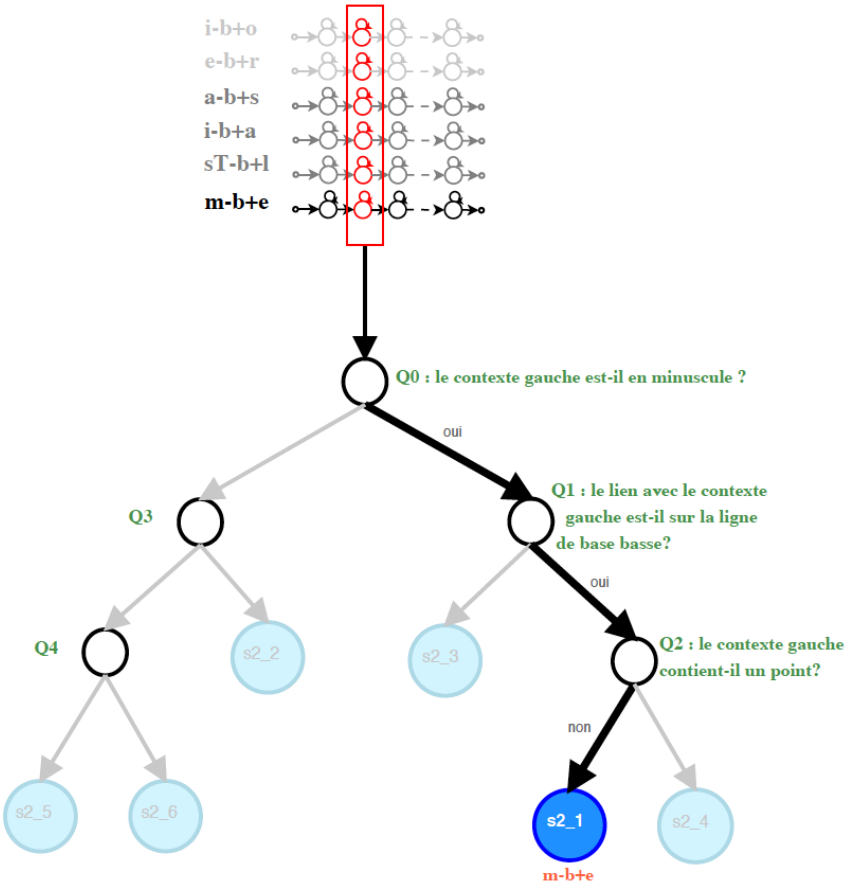


FIGURE 3.1 – Modélisation contextuelle des caractères : arbres binaires de décision (d'après [Bianne-Bernard, 2011])

— L'étape précédente conduit à obtenir certains trigraphes qui partagent exactement les mêmes états. Ils sont alors regroupés.

— Le système est ensuite réappris en utilisant toujours l'algorithme de Baum-Welch mais en augmentant le nombre de gaussiennes progressivement jusqu'à atteindre sa valeur finale N_G.

3.2.2 Clustering d'états par arbres de décision

Un modèle de Markov est composé d'une succession d'états. Les modèles de type "*-b+*" (trigraphes partageant le même caractère central "b") ont tous le même nombre

d'états. Pour chacun des états le composant, un processus de clustering est engagé.

Un cluster "racine" est créé en rassemblant tous les états dans la même position pour un même caractère central. Un exemple est donné sur la figure 3.1 pour le 2ème état des trigraphes partageant le caractère central b. Puis ce cluster est divisé en deux sous-ensembles. Cette séparation est fondée sur une question sur la morphologie du contexte du caractère central (par exemple "le contexte gauche est-il en minuscule ?"). En répétant l'opération, un arbre binaire est créé dont les feuilles correspondent aux sous-ensembles comprenant les états regroupés et les nœuds aux questions binaires. La construction des arbres est contrôlée par deux paramètres : la vraisemblance des clusters et le taux d'occupation des clusters.

On définit pour chaque cluster une vraisemblance $L(P)$. Cette valeur correspond à la somme des log-probabilités de générer les observations pour tous les états présents dans le cluster. Ces états partagent les mêmes probabilités d'émission car ils appartiennent au même cluster. Le détail du calcul de cette vraisemblance peut être trouvé dans les travaux d'Anne-Laure Bianne-Bernard. La séparation d'un cluster en deux sous-ensembles (feuilles dans l'arbre), P_{q+} et P_{q-}, est régulée par un seuil ΔL_{min} sur la vraisemblance. La différence entre la vraisemblance des deux ensembles créés (P_{q+} et P_{q-}) avec la vraisemblance du cluster initial doit être supérieure au seuil ΔL_{min}, comme indiqué dans l'équation 3.1.

Le taux d'occupation d'un cluster P, $\Gamma(P)$, correspond à la probabilité cumulée d'occupation des états des trigraphes qu'il rassemble. Ce taux d'occupation doit être supérieur à un taux minimal Γ_{min}. Ce critère permet d'éviter la multiplication des clusters, l'objectif étant de réduire le nombre de paramètres du modèle.

$$\Delta L_q = (L(P_{q+}) + L(P_{q-})) - L(P) \tag{3.1}$$

si $\Delta L_q > \Delta L_{min}$ et $\Gamma(P) > \Gamma(P)_{min}$, on créé les deux sous ensembles P_{q+} et $Pq-$

La division d'un cluster en deux sous-ensembles n'est effectuée que s'ils remplissent les deux critères que nous venons d'énoncer. Les questions binaires, que nous utilisons reprennent celles présentées par [Bianne-Bernard, 2011] pour le français. Nous avons seulement enrichi cet ensemble en ajoutant certains caractères, notamment des caractères accentuées et des signes de ponctuation, qui n'apparaissaient pas dans la base des mots de

W	δ	nombre d'états par caractère	WER
8	4	12	25.28%
9	3	12	26.13%
9	3	14	23.87%
9	2	14	34.58%
9	2	16	23.86%
9	**2**	**18**	**22.31%**

TABLE 3.1 – Taux WER obtenus sur la base de validation des mots RIMES 2011 avec différentes valeurs de largeur de fenêtre d'extraction (W) et de pas de déplacement (δ) pour une reconnaissance par HMM (Taille du dictionnaire : 5334)

RIMES 2011. Pour ce faire, nous avons ajouté ces caractères dans les questions existantes.

L'optimisation du seuil de vraisemblance et du taux minimum d'occupation des clusters est effectué sur la base de validation des lignes de RIMES 2011. Nous avons réutilisé les travaux d'Anne-Laure Bianne-Bernard, qui avait optimisé ces valeurs pour les mots isolés, afin de ne pas à avoir à explorer à nouveau l'ensemble de l'espace des paramètres (ΔL_{min},Γ_{min}). Les paramètres qui ont présenté un taux d'erreur WER minimal sont $(\Delta L_{min}, \Gamma_{min}) = (650, 450)$.

Cette approche par arbres de décisions présente l'avantage de pouvoir intégrer des nouveaux trigraphes qui n'étaient pas apparus dans la base d'apprentissage, à condition toutefois que son caractère central lui existe dans la base d'apprentissage. En effet, pour chaque état du nouveau trigraphe, il suffira de le faire percoler à travers l'arbre binaire correspondant au même état du même caractère central en suivant les questions binaires, pour trouver à quel cluster il appartient.

3.3 Optimisation des paramètres d'extraction des caractéristiques

Nous avons fait le choix d'effectuer une optimisation de paramètres d'extraction des caractéristiques (largeur w et pas δ de la fenêtre) afin de disposer d'un système de reconnaissance robuste aux différences d'étirement horizontal de l'écriture des différents

scripteurs et adapté à la résolution de la base de données. Le fait d'avoir une fenêtre recouvrante permet d'analyser finement chaque caractère et son contexte.

L'optimisation des paramètres d'extraction des caractéristiques δ et w pour le système de reconnaissance HMM (tableau 3.1) est réalisée sur la base de validation des mots isolés de RIMES. En effet les mots isolés partagent la même résolution que les images de blocs de texte. Pour le HMM, nous effectuons les étapes d'extraction des caractéristiques, d'apprentissage et de décodage avec différents paramètres d'extraction mais aussi différents nombres d'états par caractère. En effet, le pas de déplacement δ et le nombre d'états émetteurs par caractère sont liés : avec des petits pas de déplacement (grand recouvrement des fenêtres), les séquences de vecteurs de caractéristiques sont plus longues et conséquemment, le nombre d'états par caractère doit être augmenté. Cependant, ce lien nous évite de tester arbitrairement toutes les valeurs de nombre d'états. De plus, en suivant les travaux antérieurs de [Bianne-Bernard, 2011], nous avons restreint la recherche du paramètre de pas à l'intervalle $[2, 4]$ et celle de la largeur de la fenêtre d'extraction à l'intervalle $[8, 9]$. Nous avons attribué un nombre d'états inférieur (égal à 6) pour les caractères de ponctuation. En effet ceux-ci ont généralement une graphie plus courte que les lettres. Le minimum de taux d'erreur au niveau mot est atteint pour une fenêtre de largeur $w = 9$ et de pas $\delta = 2$. Le nombre de caractéristiques extraites sur chaque fenêtre est donc égal à 29 tel qu'établi dans la section 1.3.2. De plus, pour tenir compte de la dynamique des fenêtres glissantes entourant la fenêtre courante, les caractéristiques extraites sont également dérivées et ces valeurs sont ajoutées à l'ensemble des caractéristiques. Le nombre total de caractéristiques est donc égal à 58.

3.4 Modélisation du langage

3.4.1 Choix du corpus et du modèle

Comme nous l'avons vu dans les sections 1.4 et 1.6, la reconnaissance des lignes de texte permet d'introduire des connaissances sous la forme de modèles de langage statistiques (*Language Model* - LM). Or pour générer ces modèles, il est nécessaire de disposer d'un corpus de textes sur lequel vont être estimées les probabilités d'apparition des séquences de mots.

Cet apprentissage du modèle de langage doit être rattaché à la tâche de reconnaissance désirée. Son vocabulaire et la structure de ses phrases doivent être en adéquation avec le contenu des documents à reconnaître. De nombreux corpus de texte de grande taille sont disponibles pour la construction de modèles de langage généraux, fondés notamment sur des transcriptions de journaux. Cependant ceux-ci ne correspondent pas forcément à tous les types de langage et de vocabulaire. Si l'on considère un courrier comme ceux de la base RIMES (section 2.2.2) par exemple, on constate l'apparition de phrases commençant par "Je". De plus, de nombreuses formules idiomatiques y apparaissent comme les phrases de politesse (par exemple, "Veuillez agréer l'expression de mes sentiments distingués"). Ce genre de phrases n'apparait que très rarement dans les journaux.

Aucun des corpus librement disponibles n'est adapté à notre tâche de modélisation de la langue présente dans les courriers. Nous avons donc choisi de construire notre modèle de langage uniquement à partir des transcriptions des courriers de la base RIMES. La quantité restreinte de textes disponibles pour l'apprentissage nous a donc orientés vers un modèle de bigramme que nous avons construit grâce à la librairie *SRILM Toolkit* [Stolcke, 2002]. Nous détaillons la création du corpus de texte dans la section 3.4.2 puis son optimisation dans la section 3.4.3.

3.4.2 Construction du corpus

Découpage du corpus L'apprentissage du modèle de langage sur l'intégralité des transcriptions est maladroit. En effet, ce sont des lignes que nous cherchons à reconnaître et non des textes complets. En construisant globalement le LM sur l'ensemble des transcriptions du corpus, un seul bigramme de début et un seul bigramme de fin de texte apparaissent. De même, l'estimation des probabilités sur les phrases complètes ne modélise que des débuts de séquences correspondant à des débuts de phrases, et des fins de séquences correspondant à des fins de phrase. Or les lignes d'un courrier commencent rarement par le premier mot de la phrase, hormis dans le début d'un paragraphe.

Nous avons donc choisi d'effectuer l'apprentissage du modèle de langage sur les transcriptions des lignes séparément. Ainsi, les lignes peuvent commencer et finir par n'importe quel type de mot. Cependant, ce choix rejette de la modélisation tous les bigrammes formés par le mot finissant une ligne et celui débutant la suivante. Ce rejet n'est pas négligeable.

Transcriptions des lignes de texte originales
<s> Je vous adresse ce courrier afin </s> <s> de connaitre la marche à suivre </s> <s> pour devenir client . Veuillez me </s>
Redécoupage 1
<s> Je vous adresse </s> <s> ce courrier afin de connaitre la </s> <s> marche à suivre pour devenir client . </s>
Redécoupage 2
<s> Je vous adresse ce courrier </s> <s> afin de connaitre la marche à </s> <s> suivre pour devenir client . Veuillez </s>

TABLE 3.2 – Exemple de redécoupages des transcriptions des lignes de la base RIMES afin de prendre en compte les bigrammes à la jonction des lignes.

En effet, nous avons mesuré qu'en moyenne les lignes de la base RIMES comprennent 6,3 mots. En choisissant d'apprendre le modèle de langage sur les transcriptions des lignes, on exclut en moyenne 16% des transitions entre mots. Afin de prendre en compte les bigrammes situés à la jonction entre deux lignes, nous avons effectué un redécoupage du corpus en plaçant les retours à la ligne à des endroits différents. Deux redécoupages sont effectués afin que globalement tous les bigrammes de mot aient été vus le même nombre de fois. Un exemple de transcriptions redécoupées est présenté dans le tableau 3.2. Des balises (<s> et </s>) sont ajoutées pour modéliser l'entrée et la sortie de la ligne. Cette approche permet également de modéliser un plus grand nombre de débuts et de fins de lignes arbitraires. L'idée d'utiliser des fragments de texte pour construire le modèle de langage a été également utilisée par [España-Boquera *et al.*, 2011] sur la base IAM.

Correction des erreurs syntaxiques Écrits sans contrainte, les courriers de la base Rimes (voir section 2.2.2) présentent la particularité d'être assez riches en fautes d'orthographe. Ici, nous nous intéressons uniquement aux erreurs lexicales et non grammaticales. Des exemples en sont donnés sur la figure 3.2. Ce type d'erreur représente environ 3 % des mots du dictionnaire. Dans certains cas rares, ces erreurs peuvent même être aussi fréquentes que la véritable orthographe. Ces erreurs lexicales peuvent provoquer deux

FIGURE 3.2 – Exemples d'erreurs d'orthographe dans la base RIMES : à gauche, le mot correctement orthographié et à droite le mot comprenant une erreur)

types d'erreurs de décodage. Si l'erreur apparaît dans le texte d'apprentissage, elle est ajoutée au dictionnaire et peut conduire à ajouter des erreurs aux mots décodés. Si l'erreur apparaît dans le texte à décoder et n'est pas dans le dictionnaire, une erreur de décodage peut intervenir.

Pour répondre à cette problématique, nous avons mis en œuvre la stratégie suivante : on autorise plusieurs orthographes pour un mot en les regroupant sous la même étiquette dans le dictionnaire. Le processus de reconnaissance pourra ainsi sortir le mot correctement orthographié (l'étiquette), voire même corriger des erreurs. Pour ce faire, nous avons ajouté les orthographes correctes au dictionnaire lorsqu'elles manquaient. Le corpus d'apprentissage du modèle de langage a aussi été modifié de manière à ce que son vocabulaire corresponde à celui du dictionnaire. Cette procédure a pour effet d'augmenter le nombre d'orthographes possibles (taille du dictionnaire) tout en limitant le nombre de sorties possibles (nombre de monogrammes).

3.4.3 Optimisation du modèle

L'expérience montre que l'ajout de lignes redécoupées au corpus a un effet positif sur la reconnaissance en diminuant le taux d'erreur au niveau mot WER (tableau 3.3). Le modèle de langage est construit en n'utilisant que les transcriptions de la base d'apprentissage. La valeur de la perplexité (PP) augmente car le redécoupage introduit des nouveaux bigrammes.

Corpus	Nombre de lignes	WER	PP
Transcriptions	15172	42,3 %	48,0
Transcriptions+redécoupages	43766	41,5 %	146,9

TABLE 3.3 – Taux d'erreur WER et perplexité (PP) obtenus par une augmentation artificielle de la taille du corpus par redécoupages (Décodage de la base de validation avec LM, GSF=1 et WIP=0. Dictionnaire et LM sont construits sur la base d'apprentissage)

		WIP													
		0	**-100**	**-200**	**-250**	**-300**	**-350**	**-400**	**-450**	**-500**	**-550**	**-600**	**-650**	**-700**	**-750**	**-1000**
	15	33.9	33.2	33.0	32.8	32.6	32.4	32.2	32.3	32.3	32.3	32.4	32.8	33.2	33.6	37.2
	20	32.6	32.2	31.9	31.8	31.5	31.4	31.4	31.4	31.4	31.5	31.7	31.9	32.2	32.7	36.2
	25	31.8	31.5	31.1	30.9	30.9	30.8	30.7	30.7	30.7	30.8	30.8	31.2	31.5	32.1	35.4
G	**30**	31.3	30.9	30.5	30.4	30.2	30.2	30.2	30.2	30.1	30.4	30.5	30.7	31.1	31.6	34.7
S	**35**	31.1	30.8	30.3	30.2	30.0	30.0	30.0	30.0	30.1	30.2	30.4	30.6	31.0	31.3	34.6
F	**40**	31.0	30.7	30.3	30.1	30.0	**29.9**	30.0	30.0	30.1	30.3	30.4	30.6	31.0	31.3	34.5
	45	31.0	30.7	30.4	30.1	30.1	30.1	30.2	30.2	30.2	30.4	30.5	30.7	31.1	31.3	34.2
	50	31.1	30.8	30.4	30.3	30.2	30.3	30.3	30.3	30.5	30.6	30.7	31.0	31.2	31.4	34.2
	55	31.1	30.8	30.5	30.4	30.4	30.4	30.3	30.5	30.6	30.7	31.0	31.2	31.4	31.6	34.4

TABLE 3.4 – Évolution du taux d'erreur WER (%) en fonction des valeurs du Grammar Scale Factor GSF et de la Word Insertion Penalty WIP sur la base de validation des lignes de Rimes 2011

Nous avons optimisé le poids de la grammaire (GSF) et de la pénalité d'insertion (WIP) en évaluant les performances de la reconnaissance sur la base de validation de Rimes 2011. Le modèle de langage construit utilise une stratégie de discounting de Good-Turing avec une stratégie de repli (*back-off*) pour modéliser les bigrammes qui n'apparaissent pas dans le corpus d'apprentissage (voir section 1.6.2). Les valeurs optimales trouvées sont $(GSF, WIP) = (40, -350)$ pour un taux $WER = 29.9\%$. La paramètre WIP permet de faire chuter le taux d'erreur WER en réduisant le nombre d'insertions. À titre de comparaison, le taux d'erreur WER obtenu dans les mêmes conditions sans modèle de langage est de 48.6%. Le modèle de langage correctement paramétré permet donc de réduire le taux WER de 18.7% en valeur absolue. Les valeurs optimales de GSF et de WIP trouvées dépendent des valeurs de vraisemblance en sortie du système de reconnaissance. Il est donc important d'optimiser ces valeurs pour chaque nouveau système.

Dans un système précédent [Morillot *et al.*, 2012b] où les paramètres de la fenêtre

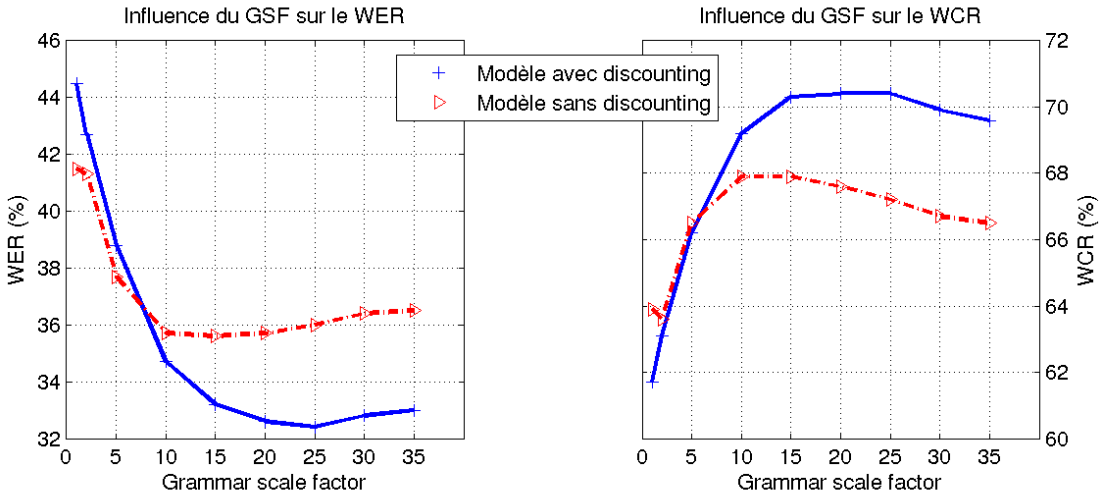


FIGURE 3.3 – Optimisation du GSF sur la base de validation

d'extraction de caractéristiques étaient différents, la valeur optimale trouvée du GSF était égale à 25, bien différente de celle trouvée plus haut. D'où l'importance d'optimiser ces valeurs pour chaque système construit. Nous avions à cette occasion comparé l'évolution du GSF (figure 3.3) pour un modèle de bigramme avec une stratégie de discounting absolu pour le lissage des probabilités et une stratégie sans discounting et donc sans modélisation des bigrammes inédits. Dans une première phase, pour des valeurs de GSF faibles (GSF<10) nous pouvons observer que la stratégie avec discounting dégrade les performances. Cette tendance s'explique par le fait que le discounting affaiblit la probabilité des bigrammes correctement appris pour la redistribuer sur les bigrammes inédits. Ainsi la probabilité des bigrammes est trop faible pour avoir un impact suffisant sur la reconnaissance.

Passé une certaine valeur du GSF (GSF>10), le poids donné au modèle de langage est suffisant pour que le modèle avec discounting obtienne de meilleurs résultats. Une trop grande valeur de GSF ($GSF > 25$) conduit à une diminution des performances. En effet, en donnant un poids trop important au modèle de langage, le module de reconnaissance finit par imposer les successions de mots les plus probables en dépit de la réalité optique.

Dictionnaire	WER	PP normalisée
sans ajouts	41,5 %	0,029
avec ajouts	41,2 %	0,022

TABLE 3.5 – Influence de la correction des erreurs syntaxiques par un dictionnaire enrichi avec les mots mal orthographiés sur le taux d'erreur WER (Décodage de la base de validation avec LM, GSF=1 et WIP=0, dictionnaire et LM construits sur la base d'apprentissage)

Le raffinement consistant à corriger les erreurs syntaxiques conduit à un gain de 0,3 % (table 3.5). Ce gain doit être mis en regard du taux de mots incorrects dans le dictionnaire (∼3 %). Les corrections conduisent également à une diminution de la perplexité normalisée (perplexité divisée par le nombre de monogrammes) de 0,029 à 0,022.

3.5 Optimisation de la correction de la ligne de base

L'approche de correction de la ligne de base que nous avons proposée repose sur deux paramètres : w_e et w_{smooth}. Ils doivent tous deux être optimisés pour la base des lignes de RIMES. L'optimisation est faite sur une partie de la base de validation : afin de réduire les temps de calculs nous n'avons utilisé que 500 images de lignes de la base de validation des lignes RIMES 2011.

L'optimisation est effectuée en explorant la grille formée par les deux paramètres w_e et w_{smooth}, utilisés pour calculer le taux d'erreur WER (tableau 3.6) : nous trouvons pour valeurs optimales $w_e = 225$ pixels et $w_{smooth} = 350$ pixels. On constate sur la figure 3.4 l'apparition d'un plateau pour les grandes largeurs de fenêtre, montrant une certaine stabilité de l'estimation. Les deux valeurs sont assez élevées afin que l'extraction soit robuste à la présence de longues séquences d'espacement entre les mots, et aux éventuelles composantes connexes d'autres lignes qui n'auraient pas été retirées lors du nettoyage (section 2.3). Sans correction de la ligne de base, un $WER = 62,7\%$ est trouvé. Une amélioration de $5,1\%$ est donc obtenue par la correction de la ligne de base. Il est intéressant de remarquer également que de trop petites fenêtres d'extraction peuvent conduire à une dégradation importante des performances. Les estimations des lignes de base ne sont alors plus robustes à la présence de longs espaces et d'accents par exemple.

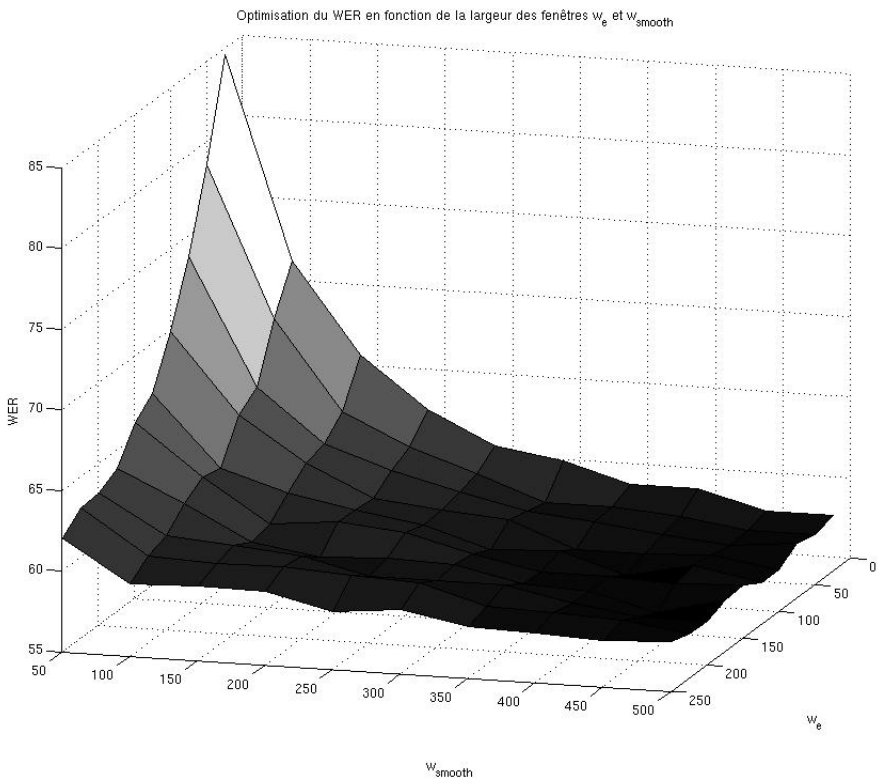
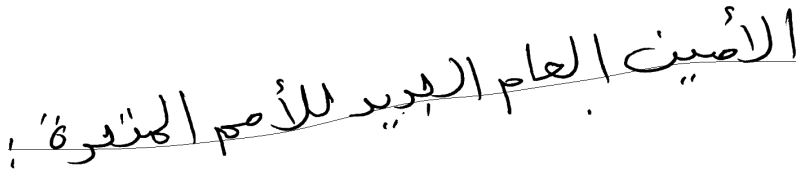
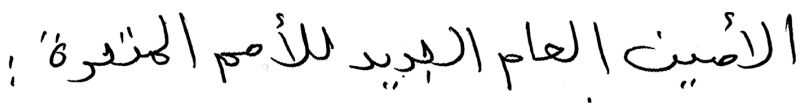


FIGURE 3.4 – Valeurs du WER obtenues sur 500 fichiers de lignes de la base de validation Rimes 2011 en fonction des paramètres w_e et w_{smooth} de correction de la ligne de base



(a)



(b)

FIGURE 3.5 – Exemple de correction de la ligne de base sur la base OpenHaRT

Les tailles de fenêtre optimales sont liées à la résolution de l'image et devront donc être adaptées pour d'autres bases que RIMES. Cependant, la correction de la ligne de base fournit des bons résultats sur les lignes de la base OpenHaRT sans changer les paramètres d'extraction ($w_e = 225$ pixels et $w_{smooth} = 350$ pixels) (figure 3.5). Les images de la base OpenHaRT en 600 dpi (*dot per inch* - point par pouce) ont uniquement été sous-échantillonnées au préalable pour être à la même échelle que celle de RIMES (300 dpi). En ce qui concerne la base IAM (section 2.2.4), les lignes ayant été écrites avec un guide-ligne et sans contrainte de temps, une correction de la ligne de base ne s'impose pas.

3.6 Apport des prétraitements

La table 3.7 montre l'apport des prétraitements pour le système de reconnaissance par HMM. Nous avons évalué la reconnaissance par HMM avec et sans modèle de langage afin de mieux mesurer l'impact réel du prétraitement sur la performance. Cependant, nous n'avons pas effectué de reconnaissance HMM au niveau caractère. En effet, les sorties du HMM au niveau caractère étaient de trop mauvaise qualité pour être exploitées. Les deux prétraitements que nous proposons permettent de réduire le taux d'erreur de 8.5% en valeur absolue sans utilisation d'un modèle de langage et de 6.1% avec l'utilisation d'un modèle de langage. À elle seule, la correction de la ligne de base intervient pour 4.1% dans cette amélioration. L'apport des prétraitements est moins important quand on utilise un modèle de langage car celui-ci permet de compenser la faiblesse du modèle optique. L'ensemble de ces résultats montrent l'intérêt de nos prétraitements, et en particulier celui

	w_{smooth}									
w_e	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
25	84,6	78,6	73,7	69,9	66,9	65,9	63,9	63,2	63,1	62,1
50	72,1	69,3	65,9	65,0	62,5	62,8	62,0	60,8	60,4	59,5
75	66,5	63,8	62,7	62,3	61,3	60,2	59,9	60,5	60,2	59,6
100	63,4	61,9	61,3	60,4	60,5	60,6	59,2	60,1	60,3	59,6
125	61,4	60,5	60,2	59,9	59,8	60,2	59,4	59,1	60,0	58,6
150	60,8	59,0	59,3	59,6	58,6	59,1	59,3	59,1	59,8	59,0
175	59,6	59,0	58,9	58,0	58,8	58,3	58,4	59,1	58,6	58,2
200	59,6	58,7	58,4	58,0	57,5	58,3	58,2	58,3	58,5	58,1
225	58,5	58,1	58,0	58,2	58,4	58,1	**57,1**	57,7	58,3	57,9
250	58,5	58,1	58,4	57,6	57,6	58,3	58,2	57,7	57,7	58,1

TABLE 3.6 – Valeurs en pourcentage du WER obtenues sur 500 fichiers de la base de validation de Rimes 2011 en fonction des paramètres w_e et w_{smooth} de correction de la ligne de base

Prétraitement	WER (dic./sans LM)	WER (dic.&LM)
Sans nettoyage & sans correction ligne de base	48.6%	29.9%
Avec nettoyage & sans correction ligne de base	44.5%	27.9%
Avec nettoyage & correction ligne de base	40.1%	23.8%

TABLE 3.7 – Impact de la correction de la ligne de base sur la reconnaissance HMM (Décodage de la base de validation avec et sans LM, dictionnaire construit sur la base d'apprentissage)

de la correction de la ligne de base pour le modèle optique HMM.

3.7 Compétition ICDAR - RIMES 2011

Nous avons participé aux compétitions de reconnaissance d'écriture en Français sur la base Rimes (ICDAR 2011 French Handwriting Recognition Competition [Grosicki et

El-Abed, 2011]). Deux tâches étaient proposées : une reconnaissance de mots isolés et une reconnaissance de blocs de texte. Cette dernière était inédite pour une compétition internationale. Les coordonnées des lignes étaient fournies.

3.7.1 Reconnaissance de mots isolés

La tâche consistait à reconnaître 7776 images de mots. Un dictionnaire de 5743 mots, comprenant ceux du test, était fourni. Les résultats obtenus par les différents systèmes sont recensés dans le tableau 3.8. Un taux d'erreur sur la meilleure sortie proposée est donné (Top1) ainsi qu'un taux d'erreur sur les 10 meilleures sorties (Top10). Dans notre cas, ce dernier correspond aux 10 mots ayant la vraisemblance la plus grande en sortie du décodage. L'évolution du taux d'erreur en fonction du nombre de sorties considérées est représenté sur la figure 3.6. Notre système a obtenu un taux d'erreur au niveau mot de 24.88% et de 6.85% si l'on considère le Top10 des sorties.

Équipe	type de système	Taux d'erreur Top1	Taux d'erreur Top10
A2iA	Combinaison MLP-HMM GMM-HMM & MDLSTM	5.13%	0.44%
Jouve	Combinaison HMM/MDRNN	12.53%	2.04%
IRISA 1	HMM rescoring par SVM	21.41%	11.51%
Télécom ParisTech	HMM	24.88%	6.85%
IRISA 2	HMM rescoring par SVM	25.46%	16.08%

TABLE 3.8 – Taux d'erreurs obtenus sur les mots isolés

3.7.2 Reconnaissance de blocs de textes

Les images en niveau de gris des paragraphes ainsi que les coordonnées des paragraphes nous étaient fournies. En revanche, le dictionnaire de la base de test ne l'était pas. Cette base comprenait 100 courriers (778 lignes de texte). Les résultats obtenus par les systèmes participants sont recensés dans le tableau 3.9. Notre système évalué sur les lignes a obtenu

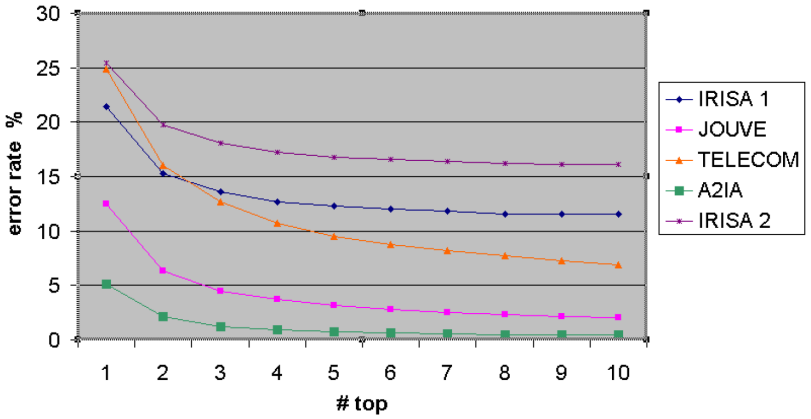


FIGURE 3.6 – Taux d'erreur obtenus sur les mots isolés

un taux WER de 31,2% ainsi qu'un taux WRC de 73,2%.

Équipe	type de système	segmentation	WCR	WER	sub	del	ins
A2iA	Combinaison MLP-HMM	lignes	86.1%	15.2%	10.0%	3.9%	1.3%
Télécom ParisTech	GMM-HMM & MDLSTM HMM	mots	73.2%	31.2%	24.4%	2.4%	4.4%

TABLE 3.9 – Taux de reconnaissance (WRC) et d'erreur (WER) obtenus sur les lignes lors de la compétition RIMES 2011

3.7.3 Analyse des résultats

Si l'on considère les systèmes fondés sur des HMM de reconnaissance de mots, on constate que les performances qu'ils obtiennent sont proches. En revanche, les réseaux de neurones s'avèrent plus performants aussi bien au niveau mot qu'au niveau ligne. D'autre part, la quasi-totalité des systèmes proposés aux deux compétitions utilisent des combinaisons de méthodes (SVM, réseaux de neurones MDLSTM,...) alors que nous proposons un système non combiné.

Le taux d'erreur que nous avons obtenu sur l'ensemble du Top 10 pour la reconnaissance de mots est assez bas (figure 3.6). Ceci peut justifier en partie le gain important en taux de reconnaissance généré par le modèle de langage pour la reconnaissance des lignes.

Les bonnes réponses du top10 au niveau mot peuvent ainsi être restituées par l'utilisation d'un modèle de langage. Le fait que le système IRISA1 obtienne un taux d'erreur plus faible que nous sur la reconnaissance de mots Top1 alors que nous obtenons un taux d'erreur plus faible sur le Top10 peut s'expliquer par la stratégie de rescoring par SVM sur les sorties Top-n entreprise par l'IRISA [Grosicki et El-Abed, 2011].

Le système présenté à cette compétition ne comprenait pas l'étape de correction de la ligne de base. À elle seule, la correction de la ligne de base a permis d'atteindre atteint un nouveau taux WER de 26.2% ainsi qu'un taux WRC de 77.7%. Le gain en valeur absolue du WER est donc de 5%. Les systèmes que nous avons soumis à cette compétition sont décrits dans [Morillot *et al.*, 2012b, Morillot *et al.*, 2012a].

Depuis la compétition, nous avons fait évoluer notre système de reconnaissance par HMM en intégrant nos nouveaux prétraitements ainsi qu'une extraction de caractéristiques optimisée (taille et pas le la fenêtre). Nous avons également développé une reconnaissance par BLSTM comme détaillé dans le chapitre 4. L'évaluation finale de nos systèmes sur la base de la compétition peut être retrouvée dans la section 4.11 du chapitre suivant.

3.8 Conclusion du chapitre 3

Dans ce chapitre, nous avons présenté notre système de reconnaissance par HMM contextuels. La stratégie pour réduire le nombre de paramètres du modèle a été exposée. Elle consiste à utiliser des arbres de décision binaires pour regrouper dans des clusters les états des trigraphes à une position donnée.

Dans un deuxième temps, nous nous sommes intéressés à l'optimisation de l'extraction des caractéristiques. Cette tâche est essentielle car, d'un jeu de paramètres d'extraction à un autre, le taux d'erreur WER obtenu après décodage peut présenter de grands écarts. Cette optimisation est faite en modifiant également le nombre d'états émetteurs HMM par caractère.

Dans un troisième temps, nous avons introduit un modèle de langage pour la reconnaissance de courriers manuscrits. Le corpus d'apprentissage de ce modèle de langage ne repose que sur les transcriptions des courriers. Les différentes étapes de la construction du

corpus sont détaillées ainsi que l'optimisation des paramètres GSF et WIP du modèle de langage. Les différentes stratégies d'amélioration proposées ainsi que l'optimisation des paramètres de ce modèle conduisent à une diminution globale de 18.7% du WER en valeur absolue par rapport à une reconnaissance sans modèle de langage. Ces travaux sont détaillés dans le chapitre suivant.

Dans un quatrième temps, nous avons présenté l'optimisation de notre approche locale de correction de la ligne de base. Afin de réduire le temps de calcul de cette optimisation, nous avons considéré une sous-base de validation sur laquelle nous avons mesuré le WER pour différentes valeurs de paramètres. Notre méthode présente un minimum d'erreur local assez stable. Nous avons mesuré l'apport final de nos deux méthodes inédites de prétraitement : on constate un gain absolu de 6.1% sur le taux d'erreur avec l'utilisation d'un modèle de langage optimisé.

Dans un dernier temps, les résultats de la compétition de reconnaissance de français manuscrit ont été analysés. C'est cette compétition qui nous a décidé à développer un système de reconnaissance par réseaux de neurones récurrents afin d'améliorer les performances, effectuer éventuellement une combinaison avec le HMM et surtout le comparer au HMM.

Chapitre 4

Système de reconnaissance de lignes par réseau de neurones bi-directionnel BLSTM

4.1 Introduction

Dans ce chapitre, nous détaillons la modélisation du BLSTM que nous utilisons pour la reconnaissance de lignes d'écriture manuscrite. Les réseaux de type BLSTM sont des réseaux de neurones bidirectionnels qui utilisent des cellules de type LSTM.

Afin de bien dissocier la question de l'architecture bi-directionnelle de la cellule à mémoire LSTM, nous détaillons tout d'abord la structure d'un réseau de neurones bidirectionnel dans la section 4.2 avant de présenter celle de la cellule LSTM dans la section 4.3. Le réseau CTC (*Connectionist Temporal Classification*) permettant le classement en sortie du réseau récurrent est présenté dans la section 4.5. Les algorithmes d'apprentissage et de décodage pour la couche CTC sont donnés dans la section 4.6.

L'extraction des caractéristiques est ensuite optimisée pour ce système de reconnaissance par BLSTM (section 4.8). En effet, rien n'indique que ces paramètres doivent identiques à ceux trouvés pour le HMM (voir section 3.3). Les apports des deux prétraitements que nous proposons (nettoyage de l'arrière plan et correction de la ligne de base) sont mesurés dans la section 4.9.

Les résultats de la compétition de reconnaissance de l'arabe manuscrit sont détaillés dans la section 4.10.

Après avoir construit et optimisé notre modélisation par BLSTM, nous la comparons avec le HMM suivant le critère du taux de reconnaissance et celui du temps de traitement (section 4.11). Enfin, comme ouverture de notre travail, nous proposons un module de détection des codes hors-vocabulaire et nous l'évaluons dans la section 4.12. Cette approche, fondée sur la reconnaissance au niveau caractère du BLSTM, met en avant le pouvoir discriminant du réseau de neurones.

4.2 Structure d'un réseau bi-directionnel BRNN

Les BRNN (*Bidirectional Recurrent Neural Networks*) considèrent en entrée un signal mono-dimensionnel mais introduisent deux contextes dans l'image, l'un passé et l'autre futur. Ainsi l'information contextuelle est prise en compte selon les deux directions de l'écriture (figure 4.1).

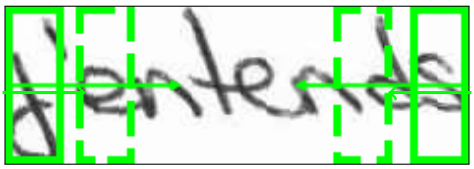


FIGURE 4.1 – Parcours de l'image pour un BRNN

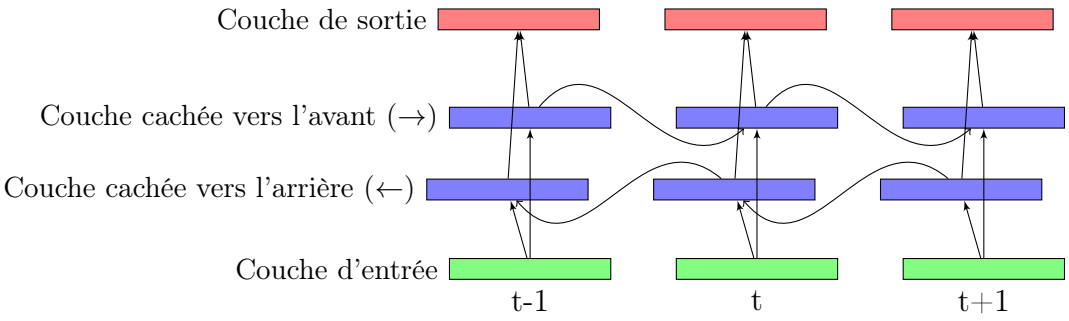


FIGURE 4.2 – Réseau de neurones BRNN déplié sur trois instants

Ici, on considère un réseau de neurones récurrents bidirectionnel mono-couche comprenant I unités d'entrée et H neurones dans chaque couche cachée : la couche orientée vers l'avant (*forward layer*) et celle orientée vers l'arrière (*backward layer*). Les deux couches cachées sont calculées séparément et sont identiques à celle d'un réseau récurrent RNN en suivant un parcours en avant et un en arrière. Ensuite une somme pondérée des activations des deux couches cachées est donnée en entrée de la couche de sortie. Une représentation de ce réseau est donnée sur la figure 4.2.

Équations du modèle Nous adoptons la notation $\rightarrow$ pour indiquer les paramètres concernant la couche cachée vers l'avant, et $\leftarrow$ pour indiquer les paramètres concernant la couche vers l'arrière. Voici les équations de la passe en avant pour la couche cachée vers l'avant :

$$a_h^{t,\rightarrow} = \sum_{i=1}^{I} w_{ih}^{\rightarrow} x_i^t + \sum_{h'=1}^{H} w_{h'h}^{\rightarrow} b_{h'}^{t-1,\rightarrow} \tag{4.1}$$

$$b_h^{t,\rightarrow} = \theta(a_h^{t,\rightarrow}) \tag{4.2}$$

où

- $a_h^{t,\rightarrow}$: entrée du h-ième neurone de la couche cachée vers l'avant à l'instant t ($1 \leq h \leq H$)
- x_i^t : valeur du i-ème neurone d'entrée à l'instant t ($1 \leq i \leq I$)
- $w_{ih}^{\rightarrow}$: poids synaptique du i-ème neurone d'entrée au h-ième neurone de la couche cachée vers l'avant
- $w_{h'h}^{\rightarrow}$: poids synaptique de la boucle de rétroaction (de l'instant $t-1$ à t) entre les neurones h' et h de la couche cachée vers l'avant
- $b_h^{t,\rightarrow}$: activation du h-ième neurone de la couche cachée vers l'avant à l'instant t ($1 \leq h \leq H$)
- θ : fonction d'activation du neurone

Pour obtenir les équations de la couche vers l'arrière, il suffit de fournir en entrée la séquence d'entrée retournée. Voici donc les équations de la passe en avant pour la couche cachée vers l'arrière :

$$a_h^{t,\leftarrow} = \sum_{i=1}^{I} w_{ih}^{\leftarrow} x_i^t + \sum_{h'=1}^{H} w_{h'h}^{\leftarrow} b_{h'}^{t+1,\leftarrow} \tag{4.3}$$

$$b_h^{t,\leftarrow} = \theta(a_h^{t,\leftarrow}) \tag{4.4}$$

où
- $a_h^{t,\leftarrow}$ entrée du h-ième neurone de la couche cachée vers l'arrière à l'instant t ($1 \leq h \leq H$)
- x_i^t valeur du i-ème neurone d'entrée à l'instant t ($1 \leq i \leq I$)
- $w_{ih}^{\leftarrow}$ poids synaptique du i-ème neurone d'entrée au h-ième neurone de la couche cachée vers l'arrière
- $w_{h'h}^{\leftarrow}$ poids synaptique de la boucle de rétroaction (de l'instant $t+1$ à t) entre les neurones h' et h de la couche cachée vers l'arrière
- $b_h^{t,\leftarrow}$ activation du h-ième neurone de la couche cachée vers l'arrière à l'instant t ($1 \leq h \leq H$)
- θ fonction d'activation du neurone

Les sorties des couches cachées en avant et en arrière sont sommées en entrée de la couche de sortie :

$$a_k^t = \sum_{h=1}^{H} w_{hk}^{\rightarrow} b_h^{t,\rightarrow} + \sum_{l=1}^{H} w_{lk}^{\leftarrow} b_l^{t,\leftarrow} \tag{4.5}$$

$$y_k^t = \theta(a_k^t) \tag{4.6}$$

où
- a_k^t entrée du k-ième neurone de sortie à l'instant t ($1 \leq k \leq K$)
- $b_h^{t,\rightarrow}$ activation du h-ième neurone de la couche cachée en avant à l'instant t ($1 \leq h \leq H$)
- $w_{hk}^{\rightarrow}$ poids synaptique du h-ième neurone caché en avant au k-ième neurone de sortie
- $b_h^{t,\leftarrow}$ activation du h-ième neurone de la couche cachée en avant à l'instant t ($1 \leq h \leq H$
- $w_{lk}^{\leftarrow}$ poids synaptique du l-ième neurone caché en avant au k-ième neurone de sortie
- y_k^t valeur du k-ième neurone de sortie à l'instant t ($1 \leq k \leq K$)

Comme pour la passe en avant, les équations de la passe en arrière sont identiques à celles d'un simple réseau de neurones récurrents classique (RNN) présenté dans la section 1.5.2.f. En effet, les deux couches cachées peuvent être traitées séparément.

Entraînement et décodage Puisqu'il partage les même passes avant et arrière que le RNN, le BRNN peut donc être décodé simplement (algorithme 3) et être appris par l'algorithme de rétropropagation du gradient à travers le temps comme présenté dans l'algorithme 4.

Algorithme 3 Pseudo-code de décodage du BRNN

```
Pour chaque exemple x de la base de test
  Faire pour t allant de 1 à T :
    Passe en avant pour la couche cachée vers l'avant
          (parcours gauche droite)
  Faire pour t allant de T à 1 :
    Passe en avant pour la couche cachée vers l'arrière
          (parcours droite gauche)
  Faire pour t allant de 1 à T : (ordre indifférent)
    Passe en avant pour calculer les sorties à partir des couches cachées
```

Algorithme 4 Pseudo-code d'apprentissage du BRNN

```
Initialisation des poids du réseau (aléatoire)
Faire jusque à un critère d'arrêt :
  Faire pour chaque exemple (x,z) de l'ensemble S :
    Faire pour t allant de 1 à T :
      Passe en avant pour calculer les sorties
        y^t à partir des entrées x^t
    Calcul de l'erreur réalisée z^t- y^t en sortie
    Faire pour t allant de T à 1 :
      Passe en arrière pour pour calculer les gradients de la couche
                cachée vers l'avant
    Faire pour t allant de 1 à T :
      Passe en arrière pour pour calculer les gradients de la couche
                cachée vers l'arrière
    Mise à jour des poids du réseau
```

4.3 Le bloc mémoire LSTM

Un problème inhérent à l'utilisation de ces réseaux récurrents sur des séquences temporelles est la difficulté de conserver l'information contextuelle sur de grands intervalles de temps [Hochreiter *et al.*, 2001]. Or en reconnaissance d'écriture, conserver l'information pendant plusieurs instants avant et après renseigne sur la forme courante. Cette difficulté a été en partie résolue par l'introduction d'un bloc mémoire, le LSTM (Long Short-Term Memory) [Hochreiter et Schmidhuber, 1997] qui contient une cellule pouvant conserver son état courant sur de longues durées (plus de 1000 échantillons temporels) et se réinitialiser en un instant. Ce bloc vient remplacer le neurone récurrent présenté dans la section 1.5.2.f. Les LSTM sont particulièrement adaptés aux tâches où de longues périodes séparent les informations pertinentes dans les séquences d'entrée. Un bloc LSTM est présenté sur la figure 4.3 et son intégration au sein d'un réseau BLSTM est schématisé sur la figure 4.4. Les notations des figures sont expliquées dans la section 4.4.

Description et rôle des différents éléments du LSTM

— La cellule : C'est l'élément central du bloc qui peut conserver sa valeur à travers les instants. Le bloc LSTM peut contenir plusieurs cellules. Cependant, afin de simplifier le descriptif de la structure, nous considérons ici uniquement des blocs avec une seule cellule.

— Porte d'entrée (*Input gate*) : Cette unité intervient en entrée de la cellule du bloc LSTM. Elle permet de protéger l'état de mémoire de la cellule de nouvelles entrées non pertinentes, grâce à une valeur multiplicative qui vient bloquer l'entrée de nouvelles données.

— Porte de sortie (*Output gate*) : Cette unité intervient en aval de la cellule du bloc LSTM. De façon similaire à la porte d'entrée, la porte de sortie protège la couche de sortie et les autres unités de l'information courante contenue dans la cellule si celle-ci n'est pas pertinente.

— Porte d'oubli (*Forget gate*) : Cette porte multiplicative fut ajoutée afin de mieux gérer les séquences temporelles continues. En effet, elle permet de réinitialiser la

valeur de la cellule du neurone et d'ainsi éviter que celle-ci ne croisse sans fin (de même pour l'erreur lors de la passe en arrière) [Gers *et al.*, 2000]. De plus, une porte d'oubli va apprendre à réinitialiser la mémoire de la cellule lorsque son contenu est périmé. Cette unité possède les mêmes entrées et le même fonctionnement que les portes d'entrée et de sortie.

— Peepholes : Les peepholes sont des connexions entre la cellule et les portes (entrée, sortie et oubli) permettant aux portes d'avoir accès aux états courants de la cellule. Ce perfectionnement du réseau est issu de la nécessité de synchroniser les sorties aux entrées [Gers *et al.*, 2002]. Dans le cas d'une porte de sortie bloquée, la dite porte de sortie n'a plus accès à l'information de la cellule dont elle est censée contrôler l'état. Les connexions peepholes sont des chaînes de retard ; à un instant t, elle reçoivent une information pondérée sur l'état de la cellule à l'instant $t-1$. Les connexions peepholes ont permis d'améliorer la synchronisation entre entrées et sorties.

Les premières structures ne comprenaient qu'une porte d'entrée et de sortie [Hochreiter et Schmidhuber, 1997]. Les portes d'oubli et les liaisons peepholes ont été ajoutées par la suite [Gers *et al.*, 2000, Gers *et al.*, 2002].

4.4 Equations de la passe avant d'un bloc BLSTM

Ici, nous considérons un réseau de neurones récurrents mono-couche comprenant I unités d'entrée et H blocs LSTM dans la couche cachée et K sorties. Chaque bloc mémoire LSTM de la couche cachée comprend une cellule. Chaque cellule LSTM se compose d'une porte d'entrée (Input gate), d'une porte d'oubli (Forget gate), et d'une porte de sortie (Output gate) et de la cellule à proprement parler. Les indices ι, ϕ, ω se réfèrent respectivement à la porte d'entrée, la porte d'oubli et la porte de sortie. Nous présentons ici les équations de la passe en avant d'un bloc LSTM de la couche cachée orientée vers l'avant d'un réseau BLSTM. Les équations pour la couche en arrière sont identiques. Les paramètres des équations du bloc peuvent être retrouvés sur la figure 4.3.

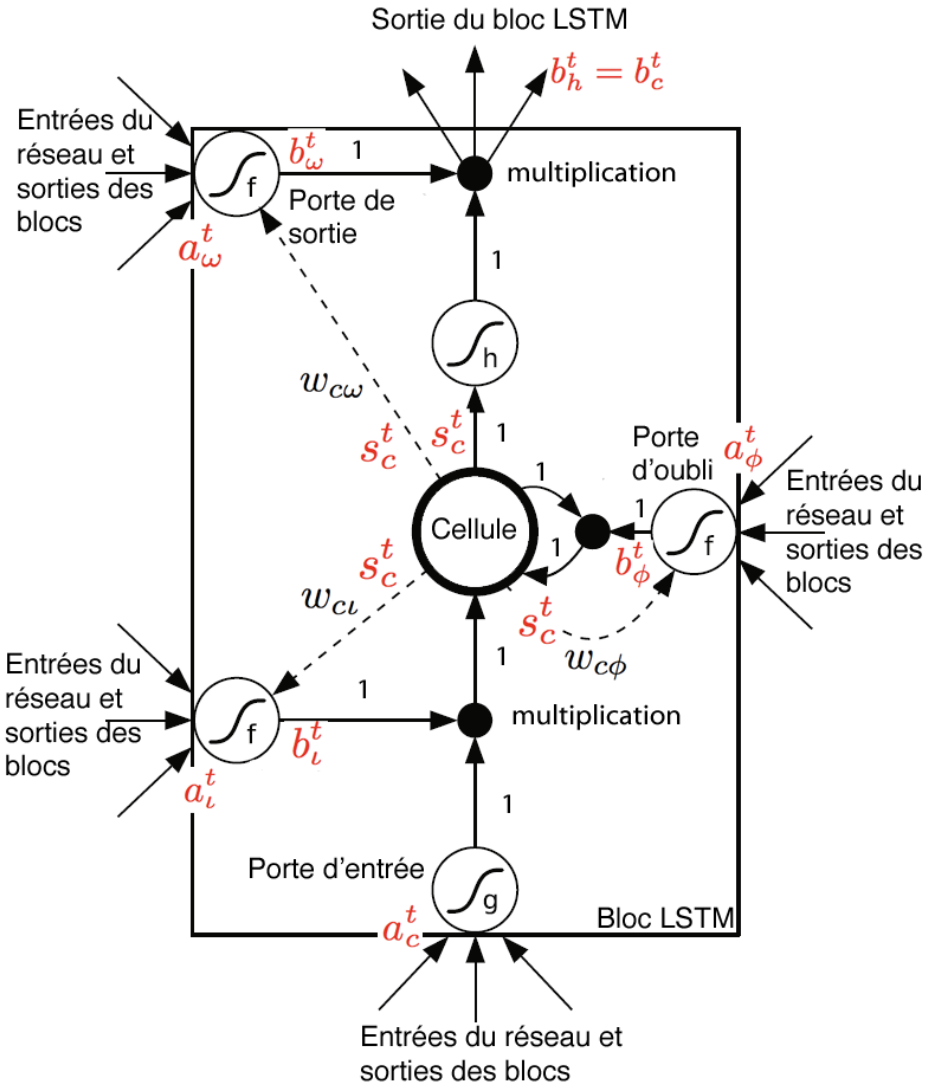


FIGURE 4.3 – Bloc mémoire LSTM avec une seule cellule (poids des connexions indiquées en noir, valeurs en rouge). Les flèches en pointillés indiquent les connexions peepholes. (adapté de [Graves, 2012])

Porte d'entrée

$$a_\iota^t = \sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} b_h^{t-1} + w_{c\iota} s_c^{t-1} \tag{4.7}$$

$$b_\iota^t = f(a_\iota^t) \tag{4.8}$$

où

x_i^t	valeur i-ième neurone d'entrée à l'instant $t(1 \leq I \leq I)$
$w_{i\iota}$	poids synaptique entre le i-ième neurone de la couche d'entrée et la porte d'entrée
b_h^{t-1}	sortie du h-ième bloc de la couche cachée à l'instant $t-1(1 \leq h \leq H)$
$w_{h\iota}$	poids synaptique de la sortie du h-ème bloc de la couche cachée à la porte d'entrée (liaison récurrente)
s_c^{t-1}	sortie de la cellule du bloc de la couche cachée à l'instant $t-1$
$w_{c\iota}$	poids synaptique de la liaison peephole entre la cellule du bloc et la porte d'entrée
b_ι^t	sortie de la porte d'entrée à l'instant t
f	fonction d'activation

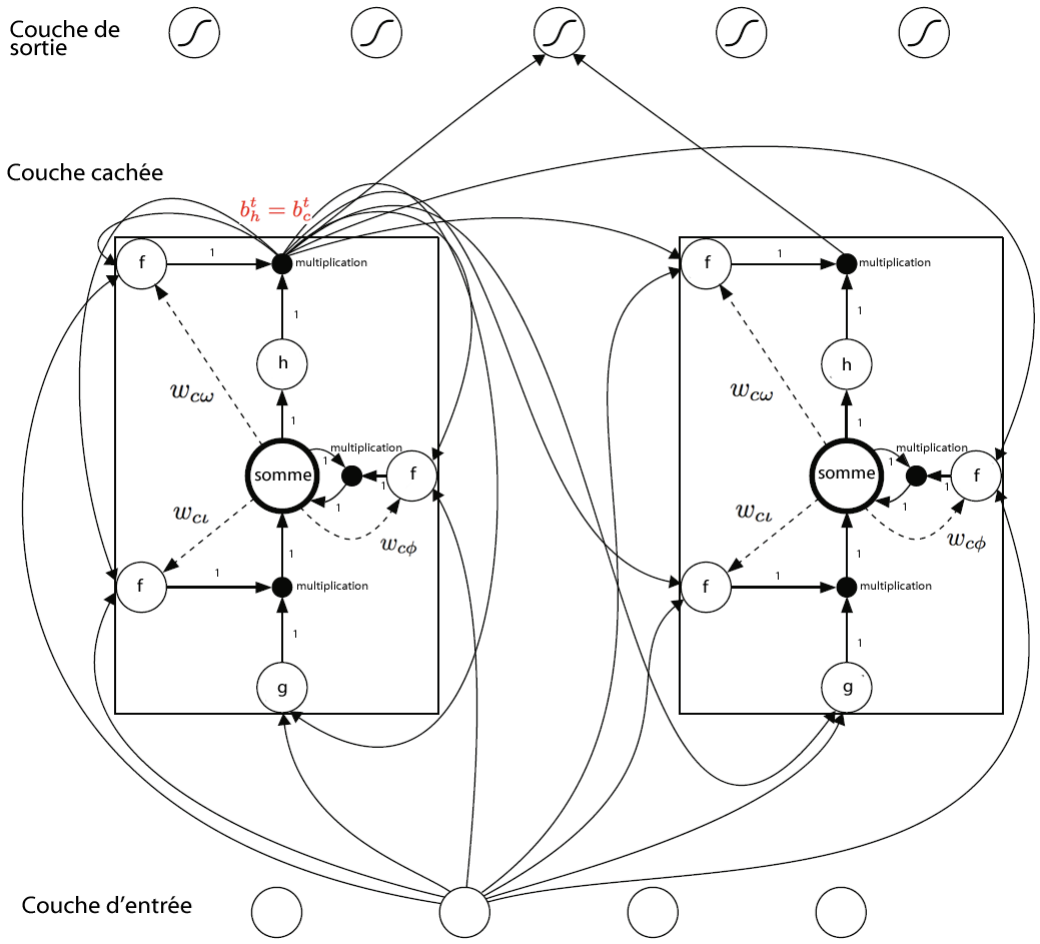


FIGURE 4.4 – Exemple de réseau LSTM à une couche cachée comprenant deux blocs LSTM à une seule cellule (Toutes les connexions ne sont pas indiquées. Les annotations sur les connexions correspondent aux poids. Les flèches en pointillés indiquent les connexions peepholes. (adapté de [Graves, 2012])

Porte d'oubli

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + w_{c\phi} s_c^{t-1} \tag{4.9}$$

$$b_\phi^t = f(a_\phi^t) \tag{4.10}$$

où

x_i^t	valeur i-ième neurone d'entrée à l'instant $t(1 \leq I \leq I)$
$w_{i\phi}$	poids synaptique entre le i-ième neurone de la couche d'entrée et la porte d'oubli
b_h^{t-1}	sortie du h-ième bloc de la couche cachée à l'instant $t-1(1 \leq h \leq H)$
$w_{h\phi}$	poids synaptique de la sortie du h-ème bloc de la couche cachée à la porte d'oubli (liaison récurrente)
s_c^{t-1}	sortie de la cellule du bloc de la couche cachée à l'instant $t-1$
$w_{c\phi}$	poids synaptique de la liaison peephole entre la cellule du bloc et la porte d'oubli
b_ϕ^t	sortie de la porte d'oubli à l'instant t
f	fonction d'activation

Cellule

$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1} \tag{4.11}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_\iota^t g(a_c^t) \tag{4.12}$$

où

x_i^t	valeur i-ième neurone d'entrée à l'instant $t(1 \leq I \leq I)$
w_{ic}	poids synaptique entre le i-ième neurone de la couche d'entrée et la c-ième cellule de la couche cachée
w_{hc}	poids synaptique entre la sortie du h-ème bloc de la couche cachée et la cellule du bloc (liaison récurrente)
b_h^{t-1}	sortie du h-ième bloc de la couche cachée à l'instant $t-1(1 \leq h \leq H)$
s_c^{t-1}	sortie de la cellule du bloc de la couche cachée à l'instant $t-1$
s_c^t	sortie de la cellule du bloc de la couche cachée à l'instant t
b_ϕ^t	sortie de la porte d'oubli à l'instant t
g	fonction d'activation

Porte de sortie

$$a_\omega^t = \sum_{i=1}^{I} w_{i\omega} x_i^t + \sum_{h=1}^{H} w_{h\omega} b_h^{t-1} + w_{c\omega} s_c^t \tag{4.13}$$

$$b_\omega^t = f(a_\omega^t) \tag{4.14}$$

où

x_i^t	valeur i-ième neurone d'entrée à l'instant $t(1 \leq I \leq I)$
$w_{i\omega}$	poids synaptique entre le i-ième neurone de la couche d'entrée et la porte de sortie
b_h^{t-1}	sortie du h-ième bloc de la couche cachée à l'instant $t-1(1 \leq h \leq H)$
$w_{h\omega}$	poids synaptique de la sortie du h-ème bloc de la couche cachée et la porte de sortie à l'instant t (liaison récurrente)
s_c^{t-1}	sortie de la cellule du bloc de la couche cachée à l'instant $t-1$
$w_{c\omega}$	poids synaptique de la liaison peephole entre la cellule du bloc et la porte de sortie
b_ω^t	sortie de la porte de sortie à l'instant t
f	fonction d'activation

Sortie du bloc LSTM

$$b_c^t = b_\omega^t h(s_c^t) \tag{4.15}$$

où

s_c^t	sortie de la cellule du bloc de la couche cachée à l'instant t
b_ω^t	sortie de la porte de sortie à l'instant t
b_c^t	sortie du bloc à l'instant t
h	fonction d'activation

Les sorties b_h^t de la couche cachée correspondent aux sorties b_c^t des différentes cellules.

4.5 Réseau CTC - Connectionist Temporal Classification

Une des difficultés rencontrées dans l'apprentissage des réseaux récurrents est de disposer d'un alignement entre la séquence d'entrée et celle des étiquettes de sortie. Dans cette optique, une première approche consiste à effectuer une pré-segmentation des séquences

d'entrées en caractères. Or nous avons vu que la segmentation en caractères est très incertaine et source d'erreurs (section 1.3.1). L'autre approche consiste à disposer d'une méthode automatique de décodage qui effectue implicitement cet alignement. C'est le cas notamment de l'algorithme *forward-backward* "Baum-Welch" dans les cas des reconnaissances HMM et hybrides HMM/RNN (section 1.5.3.

Le réseau CTC a été introduit dans le cas de tâches de classement de séquences temporelles par réseaux de neurones récurrents par Alex Graves [Graves *et al.*, 2006] (dont nous reprenons les notations). Il intervient en sortie d'un réseau de neurones récurrents et permet d'aligner automatiquement la séquence d'entrée avec les étiquettes de sortie correspondantes. Ce modèle CTC s'est avéré, selon Alex Graves [Graves, 2012], plus performant que les approches hybrides HMM/RNN, où la couche CTC est remplacée par un HMM. Le réseau CTC repose également sur un algorithme de type *forward/backward*.

Le CTC prend en entrée le tableau bi-dimensionnel des probabilités a posteriori données en sorties du réseau RNN. Ce tableau a donc pour dimensions, d'une part le nombre d'instants de la séquence d'entrée, et d'autre part, le nombre d'étiquettes soit la taille de la couche de sortie du RNN. Le CTC est une couche de sortie composée neurones de type *soft-max*. Elle comprend $|A'|$ unités où $A' = A \cup \{blanc\}$ est l'ensemble des étiquettes A (caractères, chiffres, silence...) auquel on ajoute un modèle appelé "blanc", pouvant correspondre à une absence d'étiquettes ou à un espacement. L'étiquette "blanc" est à distinguer du silence "sil" : cette nouvelle étiquette est introduite pour mieux modéliser les sorties du réseau de neurones. Elle évite d'imposer au réseau de sortir une étiquette de l'ensemble A à chaque instant et désigne d'un état rebut.

Ainsi on définit la probabilité d'une séquences d'étiquettes $\pi = (\pi_1, \pi_2, .., \pi_T) \in A'^T$ (appelé aussi chemin ou en anglais *path*) à partir d'une séquence d'entrée $x = (x_1, ..x_T)$ de longueur T :

$$p(\pi|x) = \prod_{t=1}^{T} y^t_{\pi_t} \tag{4.16}$$

où $y^t_{\pi_t}$ est la probabilité d'observer l'étiquette π à l'instant t.

Cette notation de chemin est à distinguer de celle de séquence d'étiquettes finales l sur A. Le chemin correspond aux sorties effectives du réseau : un caractère peut y apparaître avec la plus grande probabilité plusieurs instants t consécutifs sans que pour autant cela

corresponde à une répétition du caractère dans l'image de texte. De même, des *blancs* peuvent apparaître entre deux caractères alors que l'image ne présente pas d'espacement.

Pour passer des chemins aux étiquetages, on définit une fonction surjective F associant à plusieurs chemins un étiquetage : $F : A'^T \mapsto A^T$. Cette fonction va tout d'abord retirer les doublements de caractère qui ne sont pas séparés par un *blanc*, puis retirer les *blancs*. L'intérêt de l'étiquette *blanc* apparaît ici puisque sans elle, on ne pourrait pas distinguer un doublement de lettre d'un caractère étalé sur plusieurs instants. Voici deux exemples de chemins qui donnent le même étiquetage après application de la fonction F : $F(a, blanc, a, b, b) = a, a, b$ et $F(a, a, blanc, a, blanc, b) = a, a, b$.

Ainsi on définit la probabilité d'un étiquetage comme la somme des probabilités de différents chemins :

$$p(l|x) = \sum_{\pi \in F^{-1}(l)} p(\pi|x) \tag{4.17}$$

L'estimation des probabilités conditionnelles $p(l|x)$ peut être effectuée grâce à une programmation dynamique (algorithme *forward/backward*).

4.6 Apprentissage et décodage de la couche CTC

Algorithme Forward-backward Afin de pouvoir générer des sorties comprenant des silences, on définit une nouvelle séquence d'étiquettes l' à partir de l qui comprend des *blancs* en début et en fin de séquence et entre chaque paire d'étiquettes. Si l était de longueur U, l' a pour longueur $U' = 2U + 1$. L'algorithme va permettre de déterminer le chemin associant les sorties du réseau aux étiquettes.

Pour un étiquetage l, on définit une variable en avant (forward) $\alpha(t, u)$ comme la somme des probabilités de tous les chemins préfixes de longueur t arrivant sur le symbole u :

$$\alpha(t, u) = \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} y^i_{\pi_i} \tag{4.18}$$

où $V(t, u)$ est l'ensemble des chemins préfixes $\{\pi \in A'^t : F(\pi) = l_{1:u/2}, \pi_t = l'_u\}$. t est la variable correspondant à l'instant dans la séquence des sorties ($t = 1...T$). u est la variable

correspondant à la position dans la séquence d'étiquettes l'.

La variable forward se calcule itérativement à partir des conditions initiales :

$$\begin{aligned}
\alpha(t,0) &= 0 \quad \forall t \quad \text{(condition aux limites imposée)} \\
\alpha(1,1) &= y^1_{blanc} \\
\alpha(1,2) &= y^1_{l_1} \\
\alpha(1,u) &= 0 \quad \forall u > 2 \\
\alpha(t,u) &= y^t_{l'_u} \sum_{i=f(u)}^{u} \alpha(t-1,i)
\end{aligned} \tag{4.19}$$

où :

$$f(u) = \begin{cases} u-1 \text{ si } l'_u = blanc \text{ ou } l'_{u-2} = l'_u \\ u-2 \text{ sinon} \end{cases}$$

Ces conditions permettent d'autoriser certaines transitions à travers le treillis des symboles (figure 4.5) :

— Si le u-ième symbole de l' à l'instant t est un blanc alors le symbole à l'instant $t-1$ peut être soit le u-ième (blanc), soit le $(u-1)$-ième (caractère précédent).

— Si le u-ième symbole est identique au $(u-2)$-ième symbole de l', cela correspond à une lettre doublée. Dans ce cas le symbole à l'instant $t-1$ peut être le u-ième (le même caractère) ou le $(u-1)$-ième (un blanc). Cela ne peut être le $(u-2)$-ième (la première occurrence de la lettre) car sans blanc entre les deux lettres identiques, le doublement serait réduit après application de la fonction F.

— Si le u-ième symbole de l' n'est pas un blanc alors le symbole précédent à l'instant $t-1$ peut être soit u (le même caractère), le $(u-1)$-ème (un blanc) ou le $(u-2)$-ième (le caractère précédent).

Dans le cas où la séquence des sorties y_t est plus petite que la séquence des étiquettes l, certains caractères ne peuvent être atteints :

$$\alpha(t,u) = 0 \quad \forall u < U' - 2(T-t) - 1 \tag{4.20}$$

On définit la variable en arrière (backward) $\beta(t,u)$ comme la somme des probabilités

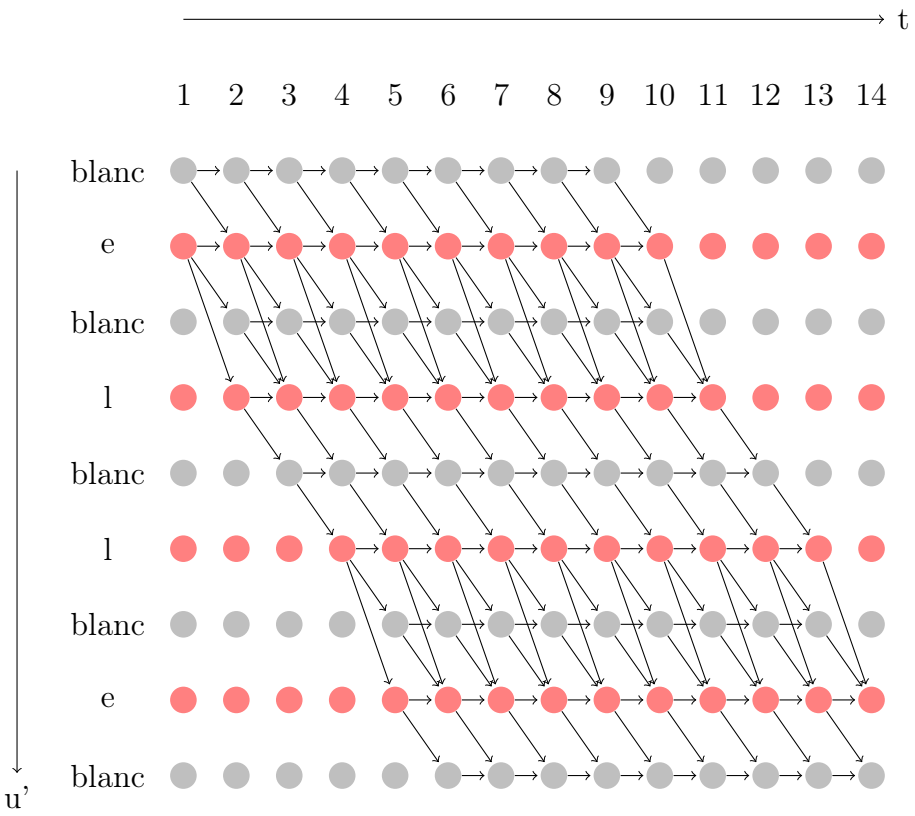


FIGURE 4.5 – Exemple des transitions possibles pour l'algorithme *forward/backward* de la couche CTC à travers le treillis des étiquettes pour le mot "elle" sur une séquence temporelle de 14 instants

de tous les chemins suffixes allant de $t+1$ à T de longueur $T-t$ partant du symbole u :

$$\beta(t,u) = \sum_{\pi \in W(t,u)} \prod_{i=1}^{T-t} y_{\pi_i}^{t+i} \tag{4.21}$$

où $W(t,u)$ est l'ensemble des chemins suffixes $\{\pi \in A'^{T-t} : F(\hat{\pi}+\pi) = l \ \forall \hat{\pi} \in V(t,u)\}$.

La variable backward se calcule récursivement à partir des conditions finales :

$$\begin{aligned}
\beta(t,U'+1) &= 0 \quad \forall t \quad \text{(condition aux limites imposée)} && (4.22)\\
\beta(T,U') &= \beta(T,U'-1) = 1 && (4.23)\\
\beta(T,u) &= 0 \quad \forall u < U'-1 && (4.24)\\
\beta(t,u) &= \sum_{i=u}^{g(u)} \beta(t+1,i) y_{l'_i}^{t+1} && (4.25)
\end{aligned}$$

où :

$$g(u) = \begin{cases} u+1 \text{ si } l'_u = blanc \text{ ou } l'_{u+2} = l'_u \\ u+2 \text{ sinon} \end{cases}$$

De même que pour la passe forward, par construction :

$$\beta(t,u) = 0 \quad \forall u > 2t \tag{4.26}$$

Fonction de perte et apprentissage De même que pour les réseaux précédemment évoqués, on définit une fonction de perte $\mathcal{L}(S) = -\sum_{(x,z)\in S} \ln p(z|x)$ sur l'ensemble d'apprentissage S. En posant $l = z$, on définit l'ensemble des chemins passant à travers z'_u à l'instant t : $X(t,u) = \{\pi \in A'^T : F(\pi) = z, \pi_t = z'_u\}$. En multipliant les variables forward α et backward β définies pour n'importe quel u et à n'importe quel instant t, on obtient la probabilité sommée de tous les chemins passant à travers z'_u à l'instant t :

$$\alpha(t,u)\beta(t,u) = \sum_{\pi\in X(t,u)} \prod_{t=1}^{T} y^t_{\pi_t} = \sum_{\pi\in X(t,u)} \ln p(\pi|x) \tag{4.27}$$

Ce coût décrit tous les chemins correspondant à la vérité, passant par u à l'instant t. En sommant cette quantité sur toutes les étiquettes u de la chaîne de caractères $l' = z'$ à un instant t donné, et en prenant l'opposé de cette valeur, on obtient la fonction de perte :

$$\mathcal{L}(x,z) = -\sum_{u=1}^{|z'|} \alpha(t,u)\beta(t,u) \tag{4.28}$$

Cette évaluation est indépendante de l'instant t.

Les dérivées par rapport aux poids du réseau s'obtiennent avec la même démarche que pour un réseau classique. Le détail de ces dérivées est donné dans l'annexe A.

Décodage L'objectif est de déterminer le meilleur étiquetage l^* étant donné la séquence d'entrées x. En supposant que le meilleur étiquetage l^* correspond au meilleur chemin π^* ($l^* = F(\pi^*)$), on peut se ramener à chercher le meilleur chemin au sein des activations des différentes sorties à travers le temps. Choisir la plus forte activation pour chaque échantillon temporel n'est pas suffisant puisque la valeur de l'activation du caractère qu'on souhaite sortir n'est pas nécessairement la plus forte à un instant donné. Il est essentiel de

prendre la décision en observant plusieurs instants. Il est donc préférable d'effectuer un décodage par recherche de préfixe (*prefix search decoding*) ou par passage de jeton (*token passing*).

Le décodage par recherche de préfixe est très similaire à l'apprentissage : là où l'on définissait une variable forward α, on définit ici la probabilité à un instant t d'avoir le préfixe p.

On distingue la probabilité à un instant t de sortir le *blanc* et d'avoir le préfixe p : $\gamma(p_{blanc}, t)$ et la probabilité à un instant t de sortir une étiquette différente de *blanc* et d'avoir le préfixe p : $\gamma(p_n, t)$. On définit l'ensemble des chemins préfixes $Y = \{\pi \in A'^t : F(\pi) = p\}$. Les probabilités γ sont estimées par récurrence :

$$\gamma(p_n, t) = \sum_{\pi \in Y : \pi_t = p_{|p|}} p(\pi|x) \tag{4.29}$$

$$\gamma(p_{blanc}, t) = \sum_{\pi \in Y : \pi_t = blanc} p(\pi|x) \tag{4.30}$$

Le décodage par passage du jeton (*token passing*) a été appliqué au CTC pour la reconnaissance de texte par [Graves et Schmidhuber, 2008]. Cet algorithme va permettre de calculer pour chaque colonne du treillis de sortie les transitions à la colonne suivante sous la forme de jetons (*token*). Nous avons choisi cette méthode car elle permet d'intégrer facilement des probabilités de bigrammes. Le détail de cette méthode est fourni dans l'annexe B.

4.7 Paramétrage du BLSTM

Les paramètres principaux du système de reconnaissance par BLSTM peuvent être rassemblés en deux classes : les paramètres structurels du réseau et les paramètres d'apprentissage. Nous les détaillons ci-dessous et justifions le choix de leurs valeurs. En effet, la durée des apprentissages des RNN peuvent atteindre plusieurs semaines, voire plusieurs mois. Il est donc préférable de s'appuyer sur les travaux antérieurs pour le choix des paramètres de l'architecture.

La structure du réseau BLSTM est définie par :

— le nombre de neurones de la couche d'entrée : cette valeur correspond au nombre de caractéristiques extraites par fenêtre. Or ce nombre varie en fonction de la taille de la fenêtre comme précisé dans la section 1.3. Nous fixerons donc la taille de la couche d'entrée dans la section 4.8.
— le nombre de neurones de la couche de sortie : cette valeur dépend seulement du nombre de classes que l'on souhaite donner. Dans le cas de la reconnaissance d'écriture latine comme le français, ce chiffre peut atteindre 91 en considérant les 26 lettres en minuscule et majuscule, certaines variantes accentuées des lettres, les chiffres, le silence "sil", certains caractères de ponctuation et un symbole pour le *blanc*.
— le nombre de niveaux de couches cachées : dans notre cas, nous travaillons sur des caractéristiques extraites au préalable grâce à une fenêtre glissante. Rien ne justifie donc l'usage d'une architecture profonde. Nous considérons donc un seul niveau de couches cachées.
— le nombre de blocs LSTM : nous avons choisi le nombre de 100 blocs pour la couche vers l'avant et celle vers l'arrière. Chaque bloc contient une cellule LSTM. Cette valeur de 100 blocs est assez largement adoptée pour la reconnaissance de l'écriture manuscrite cursive [Liwicki *et al.*, 2007, Wöllmer *et al.*, 2010].

L'apprentissage du réseau est contrôlé par :
— le nombre de tests sans amélioration sur la base de validation avant l'arrêt de l'apprentissage : nous avons choisi ce nombre égal à 20. Prendre une valeur importante permet de s'assurer que l'on a bien atteint la meilleure performance possible à partir des données d'apprentissage. Ce choix se fait évidemment au détriment du temps de convergence.
— les paramètres de la descente de gradient, l'inertie et le pas, prennent respectivement pour valeurs 0.9 et 10^{-5}. Ces valeurs sont reprises des travaux de [Wöllmer *et al.*, 2010, Graves, 2012].
— le nombre total d'itérations est décidé automatiquement à partir des paramètres précédents. Après chaque parcours de la base d'apprentissage, le taux d'erreur est testé sur la base de validation. Si celui-ci ne progresse plus pendant 20 parcours de la base, alors l'apprentissage s'arrête.

Notre BLSTM a été développé sur une version évoluée de la librairie en C++ RNNLIB [Graves, 2013] distribuée sous la licence GPL 3. Cette version nous a été fournie par

Marcus Liwicki et Saad Bin Ahmed du laboratoire DFKI (Deutsche Forschungszentrum für Künstliche Intelligenz) à Kaiserslautern en Allemagne. Cette librairie est très proche de celle proposée publiquement. Elle permettait l'intégration des modèles de langage de type bigramme. Mais si cet outil est d'un abord assez simple (un seul exécutable), il n'est pas aussi abouti que les librairies de modèles de Markov cachés telles que HTK ou Julius. L'absence d'une documentation complète ainsi que la programmation en langage orienté objet rendent difficile la modification du code de cette librairie. De plus, cette librairie repose sur la création de fichiers binaires *netcdf* où est placé l'ensemble des observations (caractéristiques ou valeurs des pixels). Ce fonctionnement permet d'accélérer le traitement des données mais demande de charger en mémoire un fichier de plusieurs gigaoctets. La taille des bases d'apprentissage est ainsi limitée par la mémoire disponible pour charger ce fichier. Cette limite apparait notamment lorsque l'on considère de très grandes bases, comme lors de la compétition OpenHaRT (section 4.10).

4.8 Optimisation des paramètres d'extraction des caractéristiques

L'optimisation des paramètres d'extraction des caractéristiques δ et w pour le système de reconnaissance par BLSTM (tableau 4.1) est réalisée sur la base de validation des mots isolés de RIMES.

Nous obtenons les mêmes paramètres d'extraction optimaux pour les reconnaisseurs HMM et BLSTM : $w = 9$, $\delta = 2$. Le fait que les systèmes partagent la même valeur optimale de largeur de fenêtre est assez logique car la taille de la fenêtre est intrinsèquement liée à la résolution de l'image et non à la méthode de reconnaissance. Cette valeur de $w = 9$ correspond à une distance de $0,76$ mm à la résolution de 300 dpi. La largeur de la fenêtre est corrélée à la largeur en pixels des caractères et à la nature des caractéristiques extraites. En revanche, le fait d'avoir une valeur de pas optimal identique $\delta = 2$ est plus riche d'enseignements. Un pas de 2 pixels correspond à un recouvrement important. Cela génère donc des séquences de vecteurs caractéristiques à l'information très redondante à travers les instants. Le fait que le BLSTM bénéficie de cette redondance n'était pas évident car ce réseau est déjà spécialement conçu pour conserver l'information à travers les instants, grâce à son architecture récurrente et à ses blocs LSTM. On constate cependant

W	δ	WER
4	4	19.64%
8	4	18.77%
8	8	24.61%
9	4	18.53%
9	3	17.74%
9	**2**	**15.72%**
9	1	20.71%
10	3	19.35%

TABLE 4.1 – Taux WER obtenus sur la base de validation des mots RIMES 2011 avec différentes valeurs de largeur de fenêtre d'extraction (W) et de pas de déplacement (δ) pour une reconnaissance par BLSTM (Taille du dictionnaire : 5334)

dans le tableau 4.1 que pour une valeur inférieure de pas ($\delta = 1$) la performance chute drastiquement. L'extraction de caractéristiques par des fenêtres glissantes présente un inconvénient majeur : elle crée de plus longues séquences de vecteurs et demande ainsi des phases plus longues d'apprentissage et de décodage. Le choix de $w = 9$ conduit à un total de 29 caractéristiques par fenêtre. La couche d'entrée du BLSTM comprendra donc 29 neurones.

Le fait que les systèmes partagent les mêmes valeurs optimales, $w = 9$ et $\delta = 2$, est pratique dans l'optique de les comparer : ainsi, ils partagent exactement les mêmes séquences d'entrée. Les valeurs optimales trouvées ici sont adaptées à la base de données RIMES. Cependant, il est raisonnable de penser que ces valeurs puissent être appliquées à d'autres écritures manuscrites telles que l'arabe, à condition que les images partagent la même résolution.

4.9 Apport des prétraitements

Nous évaluons ici l'impact de nos prétraitements pour la reconnaissance de lignes de texte par BLSTM.

Pour évaluer nos algorithmes de nettoyage de l'arrière plan et de correction de la ligne de base, nous avons mené plusieurs expériences de reconnaissance de lignes de texte

avec les différentes combinaisons de ces prétraitements. Afin de distinguer l'amélioration issue des éléments linguistiques (dictionnaire et LM) de celle de la performance brute du système de reconnaissance par BLSTM, nous fournissons tous les résultats avec ou sans dictionnaire et avec ou sans modèle de langage. En effet, le système de reconnaissance peut fournir les chaînes de caractères, issues du treillis des activations, sans l'aide d'un dictionnaire. Nous nous référons à ce cas par la dénomination "sans dic./ sans LM". L'ensemble des performances est rassemblé dans le tableau 4.2.

Prétraitement	WER (sans dic./sans LM)	WER (dic./sans LM)	WER (dic./LM)
Sans nettoyage & sans correction ligne de base	64.4%	31.6%	19.9%
Avec nettoyage & sans correction ligne de base	60.7%	27.9%	18.0%
Avec nettoyage & correction ligne de base	56.8%	26.0%	15.8%

TABLE 4.2 – Impact du prétraitement sur le taux d'erreur au niveau mot WER obtenu avec un BLSTM (Décodage de la base de validation des lignes de RIMES 2011)

Le taux d'erreur WER obtenu sans prétraitement et sans apport de données linguistiques (ni dictionnaire, ni modèle de langage) s'établit à 64,4%. Grâce à nos méthodes de prétraitement, le taux d'erreur est réduit de 7,6% en valeur absolue. La réduction du WER est de 4.1% en valeur absolue lorsqu'un dictionnaire et un modèle de langage sont utilisés. Ce résultat est inférieur au cas sans aide linguistique, car le dictionnaire et le modèle de langage vont compenser certaines erreurs de reconnaissance. Toutefois, cette valeur correspond à une réduction relative de 20% du taux d'erreur, dont 11% pour le seul nettoyage de l'arrière plan. Nous pouvons donc conclure que l'amélioration apportée par nos méthodes de nettoyage de l'arrière plan et de correction de la ligne de base a un impact significatif sur la reconnaissance de lignes de texte par BLSTM.

4.10 Compétition ICDAR - OpenHaRT 2013

Cette compétition a été organisée par le NIST sur la base de documents manuscrits en arabe OpenHaRT (section 2.2.3). Elle revendique comme objectif d'étalonner les capacités techniques actuelles dans la reconnaissance de documents manuscrits. La compétition comprenait également un atelier où les participants et les personnes intéressées ont pu discuter des problématiques soulevées lors de cette compétition.

Équipe	type de système	WER
A2iA	combinaison de 11 MDLSTM	20.10%
RWTH	hybride HMM/LSTM	23.76%
CITLAB	dérivé du MDLSTM	26.27%
UPV	Bernoulli HMM	29.32%
UOB - Télécom ParisTech (UOB-TPT)	BLSTM	47.93%
LITIS	HMM	77.59%

TABLE 4.3 – Compétition ICDAR OpenHaRT 2013 : taux d'erreur (WER) obtenus sur les paragraphes

Pour cette compétition, nous nous sommes associés à l'équipe de Chafic Mokbel de l'Université de Balamand au Liban (UOB). Cette compétition comprenait une reconnaissance de lignes de texte (rassemblés en paragraphes) ainsi que des tâches de traduction que nous ne détaillerons pas ici. L'ensemble de nos systèmes (transcription et traduction) sont présentés dans [Morillot *et al.*, 2013d]. Les conditions de l'évaluation sont détaillées dans [Tong *et al.*, 2013]. Pour cette compétition, nous avons ajouté les caractéristiques géométriques définies dans la section 1.3 aux 29 autres caractéristiques, formant ainsi un ensemble de 37 caractéristiques. Le réseau BLSTM que nous avons construit comprenait donc 37 neurones d'entrée. La couche de sortie, quant à elle, comprend 120 neurones, correspondant aux caractères arabes, à des signes de ponctuation, aux chiffres (1,2,3,...) et à certains caractères latins. Le reste du paramétrage de notre système est identique à celui détaillé dans la section 4.7. Le modèle de langage que nous avons utilisé est construit sur les transcriptions de la base d'apprentissage.

Les taux d'erreurs obtenus par les différents systèmes sont recensés dans le tableau 4.3 et l'évolution du taux d'erreur en fonction du pourcentage de la base de test considéré est présenté sur la figure 4.6.

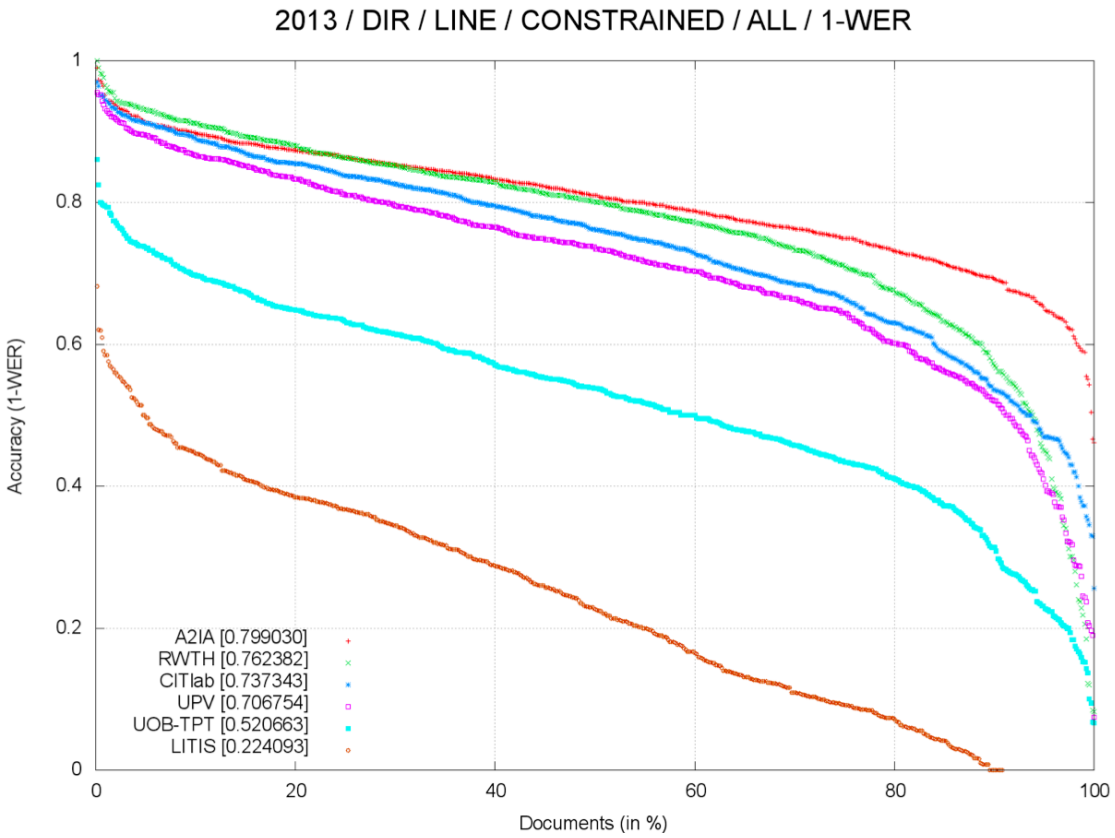


FIGURE 4.6 – Compétition ICDAR OpenHaRT 2013 : taux (1-WER) minimum obtenu en fonction du pourcentage de documents de la base (Les documents sont préalablement triés du mieux au moins bien reconnu.)

La taille des données à traiter est sans commune mesure avec la compétition ICDAR - RIMES 2011 : trois bases d'apprentissage étaient à disposition (MADCAT Phase1 Training Data, MADCAT Phase2 Training Data et MADCAT Phase3 Training Data) pour un total de 41 747 pages écrites. La base de test, quant à elle, comprenait 633 formulaires soit 12 644 lignes de texte.

Ce système de reconnaissance des lignes de texte est le premier que nous avons développé pour l'arabe manuscrit. Son résultat ne présente donc pas un aboutissement mais plutôt une première étape. L'écart entre nos performances et celles des meilleurs systèmes peut trouver plusieurs justifications. D'une part, nous avions limité le dictionnaire de décodage aux 22 000 mots les plus fréquents pour des raisons de temps de traitement. D'autre part, nous n'avons pu traiter l'ensemble des données d'apprentissage mais nous

nous sommes limités à 11% de la base "MADCAT Phase1 Training Data".

Toutefois, afin de couvrir la plus grande variété de documents possibles, notre sous-base d'apprentissage rassemblait un échantillon de tous les types de documents (voir 2.2.3). Ainsi notre système se comporte plutôt uniformément à travers la base de test, comme l'indique la forme relativement plate de la courbe (cyan sur la figure 4.6).

Depuis la compétition, nous avons ré-entraîné le modèle appris pour la compétition sur une autre partie de la base d'apprentissage (à nouveau 11% d'une des 3 bases d'apprentissage). Cette amélioration a permis d'atteindre un taux $WER = 45.57\%$ soit une diminution de 2.36%. En utilisant un vocabulaire de décodage de 30 000 mots, nous avons atteint un taux $WER = 44.44\%$. Au total, notre système a connu une diminution de 3.49% de son taux d'erreur WER depuis la compétition.

4.11 Comparaison des modèles HMM et BLSTM

Notre principale motivation pour comparer des approches de reconnaissance est l'émergence de systèmes de reconnaissance combinés ou hybrides. En effet, il est souvent difficile de mesures l'influence de chacune des composantes d'un système combiné. Nous proposons ici de comparer deux approches parmi les plus populaires dans le domaine de la reconnaissance d'écriture manuscrite : les HMM et les BLSTM. Cette comparaison a fait l'objet d'une publication [Morillot *et al.*, 2013b]. Nous n'abordons pas les structures profondes telle que les réseaux MDLSTM. En effet, nous souhaitons que les systèmes de reconnaissance soient jugés sur les mêmes données d'entrée, en l'occurrence des séquences de vecteurs de caractéristiques, extraites par une fenêtre glissante. Les paramètres d'extraction ont été optimisés séparément pour les deux méthodes (voir sections 3.3 et 4.8).

HMM comme BLSTM peuvent utiliser un dictionnaire et un modèle de langage : le dictionnaire est construit à partir des transcriptions des lignes de textes et comprend 6000 éléments. Un modèle de langage bigramme est aussi construit à partir des mêmes transcriptions comme détaillé dans la section 3.4. HMM comme BLSTM peuvent décoder des séquences de vecteurs de caractéristiques sans l'usage d'un dictionnaire, on parle alors de reconnaissance au niveau caractère. Dans le cas de la reconnaissance HMM au niveau caractère, on parle alors de modèle de remplissage (*filler model*).

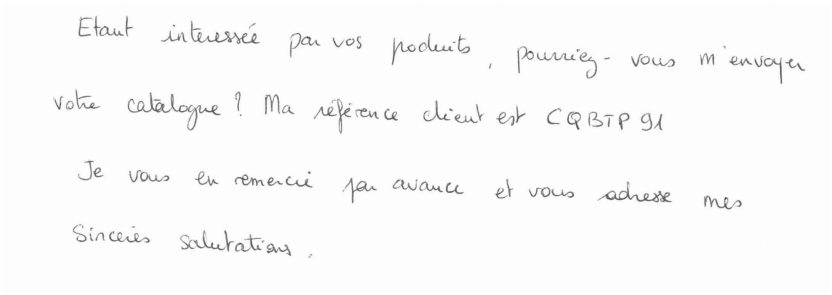
Performances obtenues Les performances de chaque approche, HMM et BLSTM, sont évaluées sur la base d'évaluation RIMES 2011, la même que celle utilisée pour la compétition présentée dans la section 3.7. Plusieurs configurations de dictionnaire et de modèle de langage sont présentées dans le tableau 4.4. Le BLSTM surpasse le HMM dans tous les cas. Il atteint un taux WER de 16.1% lorsqu'il bénéficie du dictionnaire et du modèle de langage. Ce taux de 16.1% est proche de celui atteint par le meilleur système lors de la compétition Rimes 2011 sur les blocs de texte : 15.2% (section 3.7). Notre performance est d'autant plus notable que notre approche ne requiert aucune segmentation, repose sur un seul système de reconnaissance et ne requiert aucune donnée linguistique extérieure. Le meilleur système de la compétition reposait quant à lui sur une segmentation en mots et combinait une reconnaissance par HMM gaussiens, une méthode hybride MLP-HMM et un MDLSTM.

Système de reconnaissance	Dictionnaire	LM	WER
HMM	oui	non	40.3 %
HMM	oui	oui	24.1%
BLSTM	non	non	57.1%
BLSTM	oui	non	26.5%
BLSTM	**oui**	**oui**	**16.1**%

TABLE 4.4 – Taux d'erreurs WER obtenus sur la base d'évaluation de RIMES 2011 par HMM et BLSTM

Notre nouveau système de reconnaissance par BLSTM, associé aux nouveaux prétraitements que nous avons proposés, a permis de faire chuter notre taux d'erreur WER de 31.2% (précédemment obtenu avec un HMM à la compétition RIMES 2011, voir tab. 3.9) à 16.1% (obtenu avec un BLSTM). Cela représente une diminution de 15.1% du taux WER en valeur absolue.

Il est intéressant de noter que le BLSTM sans aucune aide linguistique atteint déjà un taux d'erreur de 57.1%. En revanche, le taux d'erreur WER du modèle de remplissage HMM n'est pas présenté dans le tableau. En effet celui ci obtenait un taux WER supérieur à 100% en raison d'un très grand nombre d'insertions.



Transcription
Etant intéressée par vos produits , pourriez vous m' envoyer votre catalogue ? Ma référence client est CQBTP91 Je vous en remercie par avance et vous adresse mes sincères salutations .
HMM (dictionnaire+LM)
Etant intéressée par vos produits. pourriez-vous m'envoyer Votre catalogue ? Ma référence client est **CE site si 'a** vous en remercie par avance et vous adresse mes sincères salutations
HMM (dictionnaire seulement)
Etant intéressée par vos produits. pourriez-vous **M renvoyer** Votre catalogue **l ?** Ma référence client est **ici BMP si C** vous en remercie par avance et vous **situé** mes sincères salutations
BLSTM (dictionnaire+LM)
Etant intéressée par vos produits , pourriez vous m'envoyer votre catalogue ? Ma référence client est **CCP 21** Je vous en remercie par avance et vous adresse mes Sincères salutations .
BLSTM (sans dictionnaire, ni LM)
Etant interessée par vos **poduits**, pourriez vous **m'envyer** votre **catalgue l lMMa** référence **dclient** est C**RQHR**P 91 **e** vous en **remercié** par **avauce** et vous adresse mes Sinceres **ssalutations**.

TABLE 4.5 – Exemple de bloc de texte et sorties des systèmes de reconnaissance HMM et BLSTM avec différentes combinaisons de dictionnaire et de modèle de langage (**Erreurs en gras**)

	BLSTM incorrect	BLSTM correct
HMM incorrect	8.2%	10.9%
HMM correct	5.5%	75.4%

TABLE 4.6 – Comparison entre le HMM et le BLSTM en termes de pourcentage de mots correctement reconnus (nombre total de mots : 5938 words).

Un exemple des séquences de sortie produites par les systèmes de reconnaissance est fourni dans le tableau 4.5. Les deux systèmes de reconnaissance avec données linguistiques sont en échec face aux mots hors-vocabulaire (*out-of-vocabulary* - OOV). Dans l'exemple présenté, il y a une chaîne de caractères hors-vocabulaire correspondant à un code. Plus généralement, les OOV peuvent être des codes ou des entités nommées. La sortie de reconnaissance illustre bien le pouvoir discriminant du BLSTM. En effet, même lors du décodage au niveau caractère, le BLSTM produit des séquences intelligibles. Les mots y sont souvent correctement orthographiés et les erreurs sont principalement des lettres oubliées ou bien des doublement de lettre inopportuns.

Afin de déterminer dans quelle mesure le BLSTM surpasse le HMM, nous avons comparé les deux systèmes en observant leurs taux de reconnaissance conjoint. Au lieu de s'intéresser au WER, nous choisissons ici de mesurer le nombre de mots correctement et incorrectement reconnus. Les deux systèmes utilisent dictionnaire et modèle de langage. Le tableau 4.6 montre que 75.4% des mots sont correctement reconnus par les deux systèmes. Les systèmes ne sont pas d'accord que pour 16.4% des mots. De surcroît, seuls 5.5% des mots sont correctement reconnus par le HMM et non par le BLSTM.

Nous avons essayé d'effectuer une combinaison des scores avec la méthode *Rover* [Ficus, 1997]. Rover permet tout d'abord d'aligner les séquences de mots alternatives au sein d'un réseau et d'effectuer un vote à partir des scores des mots et/ou des fréquences d'occurrence des mots dans les sorties. Cette méthode s'avère ne pas améliorer les performances de notre BLSTM. Ceci s'explique par le fait que le BLSTM surpasse le HMM la plupart du temps. La similarité des sorties des deux systèmes peut être attribuée à l'entraînement qui a été effectué sur les mêmes données avec le même jeu de caractéristiques. Le modèle de langage a également pour effet d'uniformiser les sorties du HMM et du BLSTM.

Pour obtenir un gain sur la combinaison de lignes, plusieurs stratégies peuvent être mises en œuvre. Les systèmes peuvent être entraînés sur des ensembles de données différents et utiliser des jeux de caractéristiques différents comme dans [El Abed et Margner, 2009, Hamdani *et al.*, 2009]. Les systèmes qui observent un gain grâce à la combinaison utilisent souvent un nombre important de systèmes. Ainsi il est possible d'effectuer une combinaison non seulement sur les scores mais aussi sur les rangs d'apparition des différentes sorties N-best. Construire des modèles de langage différents ou tout du moins utiliser des vocabulaires différents parait aussi être une stratégie intéressante. Des schémas de combinaison [Bertolami *et al.*, 2006] proposent une stratégie récursive dite du rejet de l'erreur (*error reject*) : les sorties pour lesquelles les différents reconnaisseurs sont d'accord (consensus) sont conservées. Les mots ou les séquences de mots où il n'y a pas de consensus sont extraites des images de lignes et le décodage est ensuite relancé sur ces images. Cette méthode permet donc d'améliorer la reconnaissance des zones indécises. Ces stratégies pourront être envisagées pour des travaux futurs.

	Temps CPU d'apprentissge	Temps CPU moyen par cycle pour l'apprentissage	Temps CPU de décodage	Temps CPU de décodage par fichier
HMM	21 :05 :41	00 :19 :11	09 :41 :40	00 :00 :44,86
BLSTM	351 :27 :00	01 :11 :00	00 :19 :38	00 :00 :01,51

TABLE 4.7 – Temps CPU (aux formats HH :MM :SS or HH :MM :SS,00) pour les phases d'apprentissage et de décodage des systèmes HMM et BLSTM

Temps de traitement Les expériences ont été réalisées sur un Intel Xeon E5620 avec 8 coeurs cadencés à 2.40GHz et 24 GB de mémoire vive. Le système d'exploitation est une distribution Linux Debian 6.0.6. Les processus ont été lancés séparément sans aucune autre tâche en parallèle. Les deux systèmes parcourent la même quantité de données et sont tous deux implémentés en C pour les HMM (HTK) et en C++ pour le BLSTM (RNNLIB). Les procédures d'apprentissage différent entre le HMM et le BLSTM. Nous fournissons donc un temps d'apprentissage par cycle (en anglais *epoch*). Un cycle correspond au parcours complet de la base d'apprentissage.

Des différences significatives de temps de traitement peuvent être observées entre les deux approches. Si l'on regarde le temps CPU par cycle, on remarque que les HMM sont

3.7 fois plus rapides à entraîner (tableau 4.7). Les deux semaines d'entraînement pour le BLSTM peuvent contrebalancer ces bonnes performances. En revanche le décodage avec BLSTM apparaît être 30 fois plus rapide que pour le HMM. Cette différence peut s'expliquer par le fait que le décodage d'un réseau de neurones consiste simplement à propager des poids à travers un réseau et appliquer un dictionnaire et un modèle de langage sur le treillis des sorties. Les caractères du HMM sont quant à eux composés d'un certains nombre d'états par caractère, conduisant à la création d'un treillis plus grand et à un décodage à la complexité plus importante.

4.12 Détection de codes dans les courriers

4.12.1 Motivations et principe de l'approche

L'utilisation d'un dictionnaire par un système de reconnaissance présente des avantages mais aussi des inconvénients. En effet, contraindre les sorties d'un système avec un dictionnaire permet d'améliorer très sensiblement ses performances car même si la reconnaissance au niveau caractère n'est pas parfaite, le dictionnaire va avoir pour effet de corriger les sorties. Cependant, les mots qui ne sont pas présents dans le dictionnaire ne pourront jamais être reconnus. On parle alors de mots hors-vocabulaire (*Out Of Vocabulary* - OOV).

Face à ce problème, une approche est d'augmenter la taille du dictionnaire afin de couvrir au mieux le vocabulaire. Cette approche présente deux faiblesses. D'une part, l'augmentation du vocabulaire peut finir par dégrader les performances de reconnaissance puisqu'elle augmente le nombre des alternatives de décodage. En outre, elle ralentit sensiblement l'étape de décodage en augmentant la taille de l'arbre de recherche. D'autre part, rien n'assure qu'en augmentant le dictionnaire, toutes les chaînes de caractères soient couvertes. Nous avons vu dans la section 4.11 que la base RIMES présentait nombres d'entités nommées et autres codes qu'il est impossible de prévoir. Or ces éléments contiennent souvent des informations cruciales (noms, adresses, codes, numéros de compte, de téléphone...) dans le cadre d'une application industrielle.

Certains systèmes ont proposé un modèle de remplissage (*Filler model*) qui permet de laisser au système de reconnaissance la possibilité de générer des séquences de caractères, en plus de mots du dictionnaire. Cependant cette approche n'a été couronnée de succès

que dans l'optique de détection de mots-clefs (*word spotting*) avec des HMM [Rodríguez-Serrano et Perronnin, 2009, Fischer *et al.*, 2012]. Cette stratégie ne semble pas améliorer les performances de reconnaissance sur des lignes de texte.

Nous proposons ici une nouvelle approche pour la détection de certains mots hors-vocabulaire dans les lignes de texte. Nous nous restreignons à l'identification des chaînes de caractère contenant des chiffres hors-vocaulaire, tels que les codes. Le principe de cette méthode est de combiner la reconnaissance avec dictionnaire (et modèle de langage) à une reconnaissance au niveau caractère. Cette approche est fondée sur les BLSTM car ceux-ci sont très discriminants et obtiennent donc de relativement bons résultats sans dictionnaire.

La reconnaissance au niveau caractère est obtenue simplement en ne contraignant pas les sorties de la couche CTC ni avec un dictionnaire, ni avec un modèle de langage. À partir de la séquence obtenue, on va chercher à détecter les codes hors-vocabulaire. Pour cela, on va comparer les séquences de caractères aux mots du dictionnaire. Nous formulons l'hypothèse que les chaînes de caractères correspondant à des mots du vocabulaire "ressembleront" sensiblement à un mot du dictionnaire tandis que les codes en seront très éloignés. Il nous appartient de définir cette ressemblance.

4.12.2 Choix de la distance

La comparaison des chaînes de caractères peut être faite grâce à des métriques estimant la distance ou la similarité entre deux chaînes.

La distance de Hamming donne le nombre de positions où les chaînes partagent le même caractère. Cette approche est inadaptée à notre problème puisque des insertions et des suppressions peuvent intervenir et doivent être prises en considération.

La distance de Levenshtein (appelée aussi distance d'édition) correspond au nombre minimum de transformations (insertions, suppressions et substitutions) nécessaires pour passer d'une chaîne à l'autre. Plus la distance est faible plus les chaînes sont proches.

La méthode de Ratcliff-Obershelp estime la similarité entre deux chaînes de caractères. Elle commence par rechercher la plus grande sous-chaîne commune aux deux

chaînes qu'elle définit comme un point d'ancrage. À partir de là elle agit récursivement : la recherche des portions communes continue sur les deux côtés de la sous-chaîne trouvée. La récursion s'arrête quand l'algorithme ne trouve plus de sous-chaînes identiques. Ainsi, un certain nombre de caractères en commun sont identifiés. La similarité des chaînes est ensuite définie comme le double du nombre de caractères commun divisé par le nombre additionné des caractères dans les deux chaînes.

Pour décider quelle métrique choisir entre les métriques de Levenshtein et de Ratcliff-Obershelp, nous avons observé la distribution de leurs valeurs sur la figure 4.7 afin de déterminer laquelle est la plus adaptée à notre tâche de classement. Pour chaque ligne de texte de la base de validation des lignes de Rimes 2011 décodée sans dictionnaire, nous avons mesuré la distance minimum et la similarité maximum entre les chaînes de caractères et les mots du dictionnaire.

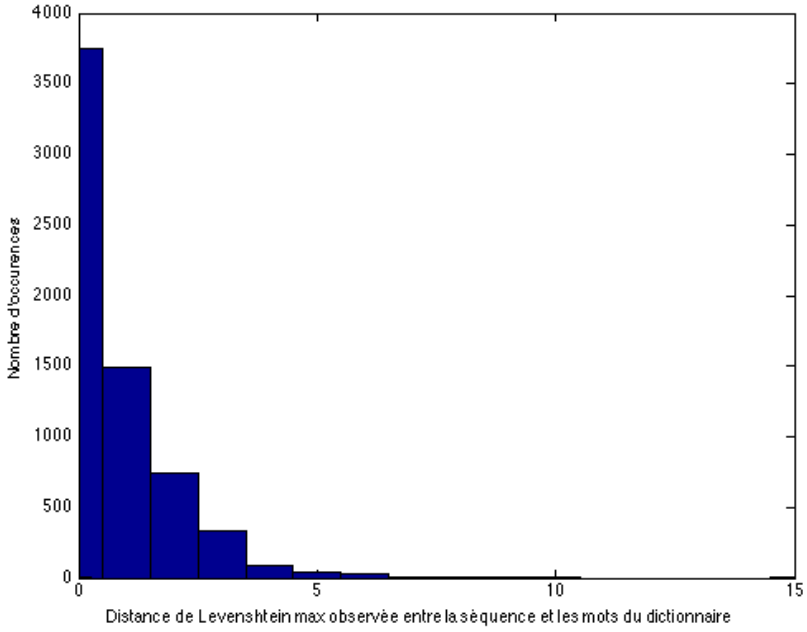
Pour les deux métriques, on constate un pic important correspondant aux mots correctement reconnus (distance=0 et similarité=1). La distance de Levenshtein étant un cardinal d'opérations élémentaires, elle ne prend que des valeurs discrètes. De plus, cette métrique recherche le nombre minimal d'opérations pour passer d'une chaîne à une autre. Ainsi, on constate que l'essentiel de la distribution est répartie sur les premières valeurs de distance : pour 92.4% des chaînes de caractères, il y a un mot dans le dictionnaire identique à 3 modifications près ou moins. Nous privilégierons donc la similarité de Ratcliff-Obershelp qui présente une distribution de valeurs plus étalée.

4.12.3 Formalisation du problème

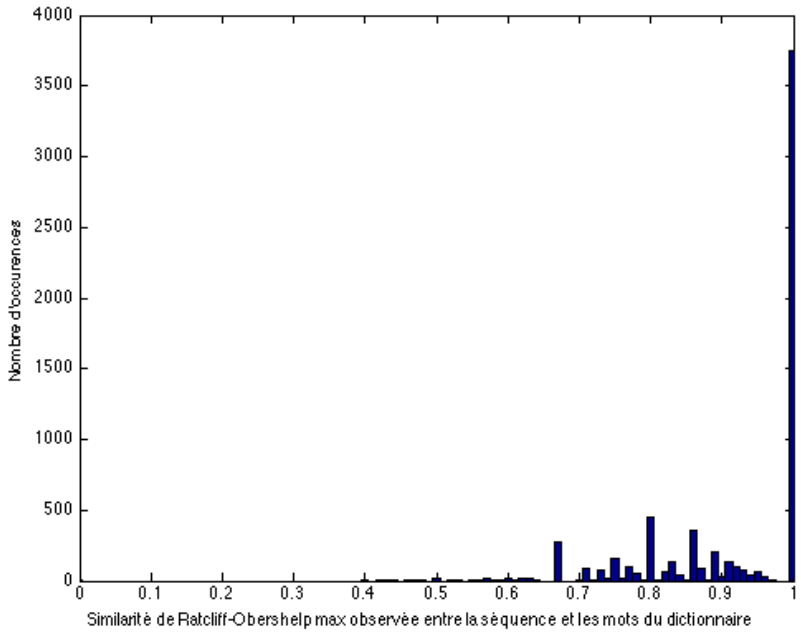
Nous cherchons à classer les chaînes de caractères reconnues sans dictionnaire comprenant au moins un chiffre (codes) en deux classes :

— Les codes hors-vocabulaire (codes OOV), c'est à dire n'apparaissant pas dans la base d'apprentissage et donc absents du dictionnaire.

— Le reste des codes appartenant au vocabulaire



(a)



(b)

FIGURE 4.7 – Distributions des valeurs minimales de la distance de Levenshtein (a) et des valeurs maximales de la similarité de Ratcliff-Obershelp (b) évaluées entre les séquences décodées sans dictionnaire et les mots du dictionnaire

Les étapes de notre méthode sont les suivantes :

- Une reconnaissance des lignes de texte est effectuée sans dictionnaire, fournissant ainsi des chaînes de caractères séparées par des espaces.
- Pour chaque chaîne de caractères de la ligne ainsi reconnue comprenant au moins un chiffre :
  - La similarité de Ratcliff-Obershelp est calculée entre la chaîne de caractères et chaque mot du dictionnaire.
  - Le mot du dictionnaire ayant la similarité maximale avec la chaîne est choisi.
  - La décision binaire du classement de la chaîne en code OOV est effectuée grâce au seuil sur la valeur de la similarité maximum précédemment calculée : si la similarité est trop faible, la chaîne est classée comme un code OOV.

4.12.4 Résultats

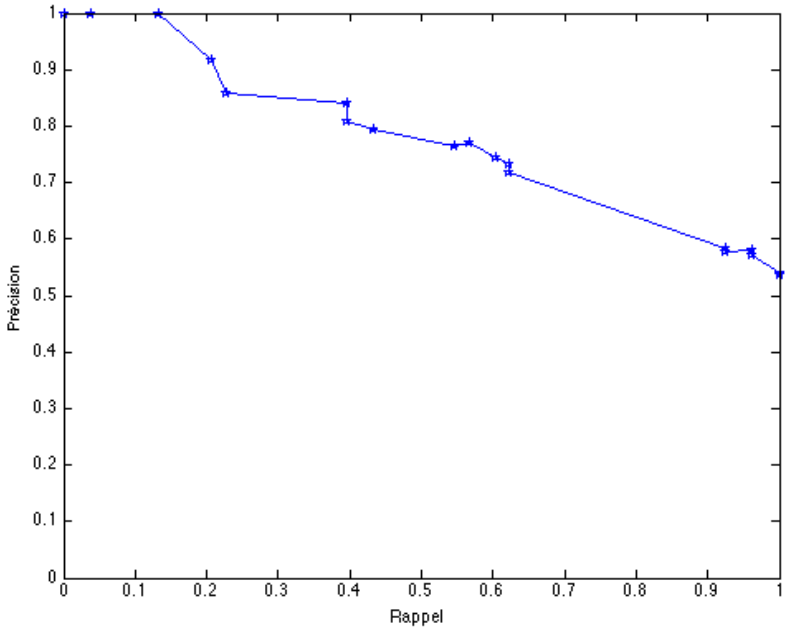
Le choix du seuil de décision peut être effectué en observant les courbes du taux de précision de la détection (*precision*) en fonction du rappel (*recall*) (figure 4.8a) et celle du rappel en fonction du taux de fausse alarme (*false alarm*) (figure 4.8b). Ces taux sont définis ainsi :

$$\text{Précision} = \frac{\#\text{codes OOV correctement détectés}}{\#\text{codes OOV détectés}}$$

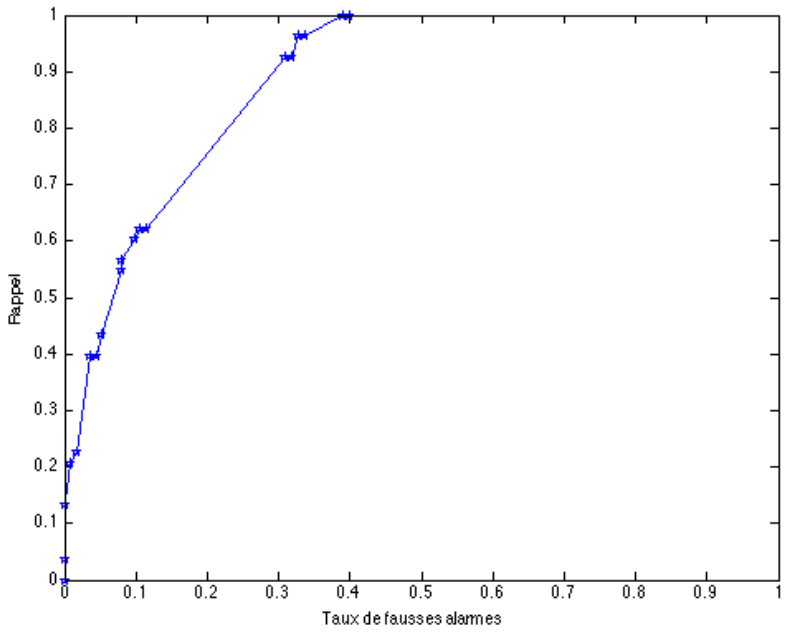
$$\text{Rappel} = \frac{\#\text{codes OOV correctement détectés}}{\#\text{codes OOV correctement détectés} + \#\text{codes OOV non détectés}}$$

$$\text{Fausse alarme} = \frac{\#\text{codes OOV incorrectement détectés}}{\#\text{séquences de caractères comprenant au moins 1 chiffre}}$$

Même si les valeurs de similarité ne sont pas continues, elles sont loin de former un continuum. Ceci explique la répartition non-homogène des points sur les courbes. Les graphiques montrent que 100% des codes OOV peuvent être détectés en ayant un taux de fausses alarmes de 39% et une précision de 54%. Le choix du couple (précision,rappel) est un compromis qui dépend de l'application souhaitée. Un seuil de similarité établi à 0.7 permet d'atteindre un taux de détection de 92.5% avec 31% de taux de fausses alarmes.



(a)



(b)

FIGURE 4.8 – Évaluation de la détection de codes OOV : (a) précision de la détection en fonction du rappel, (a) rappel en fonction du taux de fausses alarmes (base de validation de lignes de RIMES 2011)

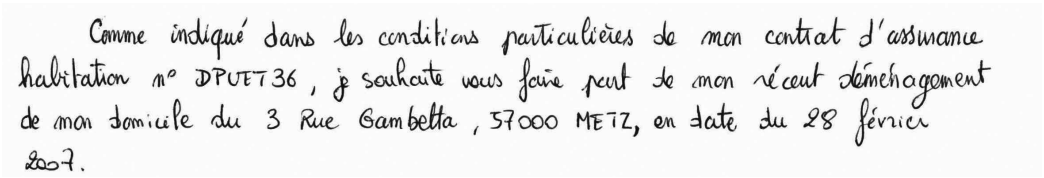

Vérité-terrain
Comme indiqué dans les conditions particulières de mon contrat d'assurance habitation n° DPUET36, je souhaite vous faire part de mon récent déménagement de mon domicile du 3 Rue Gambetta, 57000 METZ, en date du 28 février 2007.
Reconnaissance avec dictionnaire et modèle de langage
Comme indiqué dans les conditions particulières de mon contrat d' assurance habitation n° 06 , je souhaite vous faire part de mon récent changement de mon domicile du 3 le bas 1000 , en date du 28 février 2007.
Reconnaissance au niveau caractère et détection des codes OOV
Cmnme ndqué dans les conditions paitceuliores de mon contiat d'assmrance Sabtaition n°e **D0PVr36**, je souhaite vous faie pert e mon récent échiageemrent de mon domile du 3 ue Gambea, **57000** N, en date du 28 fénier 2007.
Fusion des deux sorties de reconnaissance
Comme indiqué dans les conditions particulières de mon constat d' assurance habitation n° **D0PVr36**, je souhaite vous faire part de mon récent changement de mon domicile du 3 le bas **57000** N, en date du 28 février 2007.

TABLE 4.8 – Exemple de détection de codes pour la reconnaissance de courriers manuscrits (codes détectés en gras et bleu) avec un système de reconnaissance par BLSTM sur la base d'évaluation de RIMES 2011

4.12.5 Reconnaissance des codes OOV

Une fois les codes OOV détectés, ils peuvent être remplacés au sein des autres séquences de mots préalablement reconnues avec un dictionnaire et un modèle de langage comme présenté dans le tableau 4.8. Cette fusion n'améliore pas toujours le taux de reconnaissance WER des lignes car la reconnaissance au niveau caractère n'assure pas que le code sera parfaitement retranscrit. Cependant, le remplacement des codes peut améliorer la lisibilité générale du texte. De surcroît, un post-traitement spécifique peut-être

appliqué aux codes une fois détectés (reconnaissance avec un nombre de modèles réduit ou un dictionnaire spécifique par exemple). L'exemple donné dans le tableau 4.8 permet également de constater que la reconnaissance au niveau caractère pourrait permettre d'améliorer la lecture des entités nommées.

4.13 Conclusion du chapitre 4

Dans ce chapitre, l'approche de reconnaissance par réseau récurrent bi-directionnel utilisant des blocs mémoires LSTM a été détaillée. Les entrées de ces réseaux sont des séquences d'observations extraites par une fenêtre glissante. Ces séquences sont analysées de gauche à droite et de droite à gauche par deux couches cachées distinctes. Les sorties de ces couches sont ensuite combinées au niveau de la sortie à tous les instants t. Ainsi les contextes passé et futur d'une observation permettent d'aider la prédiction de son étiquette. Les blocs mémoires permettent de conserver l'information à travers la séquence et de la réinitialiser instantanément. Un réseau CTC intervient en sortie du réseau récurrent. Son rôle est d'aligner les séquences de vecteurs en entrée et les étiquettes en sortie.

Nous avons optimisé les paramètres d'extraction des caractéristiques pour le système de reconnaissance par BLSTM. Les paramètres trouvés sont identiques à ceux du HMM : $w = 9$ et $\delta = 2$. Cette optimisation a permis de montrer que la reconnaissance par BLSTM est améliorée en utilisant des séquences d'entrée à l'information redondante. Le BLSTM bénéficie aussi de la correction de la ligne de base locale et du nettoyage de l'arrière-plan. Ces deux prétraitements conduisent à une réduction de 4.1% du taux WER avec utilisation d'un modèle de langage.

Nous avons soumis un système de reconnaissance à la compétition de reconnaissance d'écriture manuscrite arabe OpenHaRT'13. Ce système était le premier que nous avons construit pour la reconnaissance de lignes de texte en arabe. Ses résultats sont encourageants si l'on considère la faible part de données d'apprentissage que nous avons utilisée. Depuis la compétition, nous avons amélioré les performances de $WER = 47.93\%$ à $WER = 44.44\%$. Un des enseignements que nous tirons de cette compétition est la nécessité de développer des outils pour traiter efficacement les gigantesques bases de données.

La comparaison menée entre les systèmes de reconnaissance souligne les meilleures

performances atteintes par le BLSTM ($WER = 16.1\%$) en comparaison avec le HMM ($WER = 24.1\%$). Cependant les phases d'apprentissage du réseau de neurones peuvent prendre plusieurs semaines alors qu'elle ne nécessitent que quelques jours au maximum pour le HMM. La construction et l'optimisation d'un modèle BLSTM seront donc beaucoup plus longues. Nous avons constaté que la proximité entre les sorties des deux systèmes rend inutile la combinaison. Un nouveau système différent du BLSTM, comme un MDLSTM, pourrait être construit pour améliorer le taux d'erreur. Nos systèmes HMM et BLSTM sont tous deux mis en difficulté par les mots hors-vocabulaire.

Un autre des enseignements tirés de la comparaison des systèmes de reconnaissance est que le BLSTM produit des sorties intelligibles même lorsqu'il n'est pas contraint par des données linguistiques (dictionnaire et modèle de langage). Son caractère discriminant permet d'atteindre un taux $WER = 57.1\%$. Nous avons donc construit un système de détection des mots OOV en comparant les sorties du BLSTM avec et sans contraintes linguistiques. Cette détection repose sur le calcul de la similarité de Ratcliff-Obershelp pour comparer les chaînes de caractères. Pour l'instant, nous nous sommes limités à la détection de codes (chaînes de caractères comprenant au moins un chiffre) mais nous pensons étendre notre étude aux entités nommées (noms propres). Nous sommes parvenus à un taux de bonne détection de 92.5% avec 31% de taux de fausse alarme. Dans une perspective applicative, la détection de ces champs permettrait de leur réserver un traitement spécifique, comme une nouvelle passe de reconnaissance avec un dictionnaire spécifique de codes.

Conclusions et perspectives

Nous avons exposé dans ce document les différents travaux que nous avons menés sur la reconnaissance de lignes de texte manuscrit. Nous avons adopté avec succès une stratégie de reconnaissance sans segmentation explicite en mots. En renonçant à cette découpe, nous nous affranchissons des erreurs qu'elle peut induire. Nous avons construit deux systèmes de reconnaissance de lignes de texte à l'état de l'art : une reconnaissance par HMM contextuels et une reconnaissance par réseaux de neurones bi-directionnels BLSTM. Nos contributions interviennent à plusieurs niveaux du système de reconnaissance de ligne de texte manuscrit : prétraitements, modélisation et post-traitements.

Tout d'abord, nous avons proposé une chaîne de prétraitement adaptée aux lignes de texte. Cette chaîne prend comme entrée l'image rectangulaire de la ligne. Nous avons proposé une méthode de nettoyage des composantes des lignes voisines fondée sur une transformée de Hough qui estime la position de l'axe de l'écriture. Cet axe donne une approximation de la direction principale suivant laquelle les composantes de l'écriture de la ligne sont disposées. Une décision binaire est ensuite effectuée sur les composantes connexes de l'image en fonction de leur distance à cet axe. Cette décision ne nécessite pas d'optimisation car les seuils de décision sont calculés automatiquement sur chaque image de ligne. Grâce à ce prétraitement, nous diminuons le WER de 2% pour la reconnaissance HMM et de 1.9% pour la reconnaissance BLSTM évaluée sur la base de validation de RIMES 2011.

Nous avons également abordé une des principales difficultés du traitement des lignes de texte : la variation de la position de la ligne de base dans le cas d'une écriture libre. Nous avons proposé une estimation et une correction de la position de la ligne de base pour chaque colonne de pixels grâce au parcours d'une fenêtre glissante. Cette contribution est majeure à notre connaissance puisque, jusque là, les méthodes avaient abordé la correction au niveau mot en estimant une seule valeur d'angle ou au niveau ligne en effectuant une correction par morceaux. Notre méthode consiste à estimer la ligne de base locale en

analysant le profil de projection horizontal sur l'axe vertical. En utilisant des fenêtres d'analyse assez grandes, ainsi qu'un lissage gaussien des positions de la ligne de base, nous sommes parvenus à proposer une méthode suffisamment robuste aux espacements, de sorte qu'elle fonctionne avec une grande variété de scripteurs, pour une écriture cursive comme scripte. Les largeurs des fenêtres d'analyse et de lissage ont fait l'objet d'une optimisation sur la base RIMES. Cette correction présente un gain de WER de 4.1% pour la reconnaissance HMM et de 2.2% pour la reconnaissance BLSTM évaluée sur la base publique de validation de RIMES 2011. Elle a également montré de bons résultats visuels sur les base OpenHaRT et IAM. Nos deux méthodes inédites de prétraitement ont fait l'objet d'un article de revue [Morillot *et al.*, 2013c].

Nous avons également proposé les étapes de construction d'un modèle de langage de type bigramme adapté à la reconnaissance de courriers manuscrits d'entreprises. Le corpus d'apprentissage du modèle ne repose que sur les transcriptions des courriers. En effet, à notre connaissance, aucun corpus librement disponible ne permet de reproduire les structures de phrase rencontrées dans ces courriers. Le corpus est constitué de chaînes de mots qui reproduisent les lignes telles qu'elles apparaissent dans les courriers. Nous avons mis en œuvre une stratégie de recoupe des transcriptions qui a permis de diminuer le taux d'erreur WER de 0.8% en valeur absolue. Notre modèle de langage regroupe les différentes orthographes d'un même mot sous une même étiquette. Ainsi la perplexité du modèle de langage est réduit et les erreurs lexicales peuvent être corrigées. Afin de modéliser les bigrammes qui n'apparaissent pas dans le corpus d'apprentissage, une stratégie de décompte est appliquée aux bigrammes rencontrés dans le corpus et la masse de probabilité ainsi récupérée est redistribuée sur les événements inédits. Nous avons également optimisé les poids du modèle de langage (poids de la grammaire GSF et pénalité d'insertion de mots WIP) pour la reconnaissance HMM sur notre base de validation de RIMES 2011. Les différentes stratégies d'amélioration proposées ainsi que l'optimisation des paramètres de ce modèle conduisent à une diminution globale de 18.7% du WER en valeur absolue par rapport à une reconnaissance sans modèle de langage.

Afin de nous confronter aux autres laboratoires et aux entreprises, nous avons participé à deux compétitions internationales de reconnaissance d'écriture manuscrite, l'une pour une écriture latine et l'autre pour l'écriture arabe. Nous avons soumis un système de reconnaissance de mots isolés par HMM contextuels et un autre pour la reconnaissance de blocs de texte lors de la compétition RIMES 2011. Nous avons également soumis un

système de reconnaissance de paragraphes de texte lors de la compétition OpenHaRT 2013 organisée par le NIST (*National Institute of Standards and Technology*). Cette dernière compétition a souligné l'importance des ressources informatiques. En effet, le système de reconnaissance doit être conçu pour pouvoir traiter dans un temps limité une quantité impressionnante de documents (41 747 documents au total pour les bases d'apprentissage). Si la participation à des compétitions est coûteuse en temps pour mettre au point le système, elle est cependant source de beaucoup d'enseignements.

Nous avons comparé les approches BLSTM et HMM, dont les systèmes correspondant ont été appris sur les mêmes caractéristiques. Nous avons optimisé l'extraction de caractéristiques pour les deux systèmes de reconnaissance et constaté que les deux systèmes partagent les mêmes paramètres optimaux : une largeur de fenêtre glissante $w = 9$ déplacée avec un pas $\delta = 2$. Nous avons donc constaté que le BLSTM bénéficie de séquences de vecteurs de caractéristiques à l'information redondante extraites par une fenêtre glissante à recouvrement. Cette conclusion n'était pas évidente en soi, car la structure du bloc LSTM permet déjà de conserver l'information à travers le temps. Le BLSTM atteint un taux d'erreur de 16.1% et le HMM $24,1\%$ sur la base d'évaluation RIMES 2011. La performance obtenue par le BLSTM est proche de celle obtenue par le meilleur système lors de la compétition de reconnaissance de blocs de texte RIMES 2011 ($WER = 15.2\%$). Notre performance est donc à l'état de l'art pour un système de type BLSTM. Cette performance est d'autant plus intéressante que notre système ne requiert aucune segmentation, repose sur un seul module de reconnaissance et ne requiert aucune donnée linguistique extérieure. Le meilleur système de la compétition reposait quant à lui sur une segmentation en mots et combinait une reconnaissance par HMM gaussiens, une méthode hybride MLP-HMM et un MDLSTM. Si le BLSTM présente de meilleurs taux de reconnaissance que le HMM, son temps d'apprentissage de plusieurs semaines est un désavantage certain. Nous avons également constaté que la proximité entre les sorties de nos systèmes HMM et BLSTM rend la combinaison de leurs sorties inutile.

La dernière contribution de nos travaux concerne la détection de codes hors-vocabulaire à partir des sorties du BLSTM sans dictionnaire ni modèle de langage. En effet, un taux d'erreur au niveau mot $WER = 57.1\%$ peut être atteint sans utiliser de données linguistiques sur la base d'évaluation de RIMES 2011. Nous avons donc construit un système de détection des mots OOV dans les courriers manuscrits de RIMES en comparant les sorties du BLSTM avec et sans contraintes linguistiques. Cette détection repose sur le calcul de

la similarité de Ratcliff-Obershelp pour comparer les chaînes de caractères. Pour l'instant, nous nous sommes limités à la détection de codes (chaînes de caractères comprenant des chiffres) mais nous pensons étendre notre étude aux entités nommées (noms propres). Nous sommes parvenus à taux de détection de codes OOV de 92.5% avec 31% de taux de fausses alarmes. L'utilisation des sorties du BLSTM sans apport linguistique renforce l'intérêt présenté par notre prétraitement. En effet, nos deux méthodes de prétraitement permettent de diminuer le taux d'erreur de $7,6\%$ pour un BLSTM sans dictionnaire, ni modèle de langage.

Les développements futurs possibles sont multiples. Tout d'abord, ils pourront concerner la poursuite du traitement dédié aux mots hors-vocabulaire. Cette question peut également être abordée en utilisant des dictionnaires dynamiques [Oprean *et al.*, 2013b]. Une interpolation de différents modèles de langage peut aussi permettre de réduire le nombre de OOV. Ensuite, en vue d'une combinaison de modèles, il apparait essentiel de développer une approche suffisamment éloignée pour que ses sorties soient différentes de celles de nos systèmes actuels. Pour ce faire, nous pouvons envisager d'utiliser les approches d'extraction de caractéristiques par des architectures profondes telles que les MDLSTM. Les approches par modèles de Markov cachés ne doivent pas être abandonnés pour autant. La compétition OpenHaRT 2013 [Tong *et al.*, 2013] a montré qu'un des meilleurs systèmes actuels était fondé sur des HMM utilisant des distributions de Bernoulli (BHMM). Enfin, dans l'optique de participer à de nouvelles compétitions de reconnaissance et de certaines applications industrielles, il parait indispensable de développer de nouvelles stratégies pour réduire le temps d'apprentissage des modèles.

Liste des publications

Les travaux de cette thèse ont donné lieu à des publications d'articles de revue et d'articles publiés dans les actes de conférences à comité de lecture :

[Morillot et al., 2012b] Morillot, O., Likforman-Sulem, L., and Grosicki, E. (2012). Construction of language models for an handwritten mail reading system. In *Proceedings of Electronic Imaging - Document Recognition and Retrieval (DRR) XIX*. Burlingame (USA). (**Best student paper award DRR 2012**)

[Morillot et al., 2012a] Morillot, O., Grosicki, E., and Likforman-Sulem, L. (2012). Reconnaissance de courriers manuscrits par HMM contextuels et modèle de langage. In *Actes de CIFED 2012*. Bordeaux.

[Morillot et al., 2013c] Morillot, O., Likforman-Sulem, L., and Grosicki, E. (2013). New baseline correction algorithm for text-line recognition with bidirectional recurrent neural networks. *SPIE Journal of Electronic Imaging*, 22(2).

[Morillot et al., 2013d] Morillot, O., Oprean, C., Likforman-Sulem, L., Mokbel, C., Chammas, E., and Grosicki, E. (2013). The UOB-Telecom ParisTech Arabic handwriting recognition and translation systems for the OpenHaRT 2013 competition. In *NIST-OpenHaRT Workshop*. Washington DC (USA).

[Morillot et al., 2013b] Morillot, O., Likforman-Sulem, L., and Grosicki, E. (2013). Comparative study of HMM and BLSTM segmentation-free approaches for the recognition of handwritten text-lines. In *Proceedings of the 12th International Conference on Document Analysis and Recognition*, ICDAR '13. Washington DC (USA).

[Morillot et al., 2013a] Morillot, O., Grosicki, E., and Likforman-Sulem, L. (2013). Reconnaissance de courriers manuscrits par HMM contextuels et modèle de langage. *Document Numérique*, 16(2/2013) :69–90.

Glossaire et notations

Acronymes

— **HMM** Hidden Markov Model

— **GMM** Gaussian Mixture Model

— **HTK** Hidden markov model ToolKit

— **ANN** Artificial Neural Network

— **MLP** Multi-Layer Perceptron

— **TDNN** Time-Delay Neural Network

— **RNN** Recurrent Neural Network

— **BRNN** Bidirectional Recurrent Neural Network

— **BLSTM** Bidirectional Long Short-Term Memory

— **MDRNN** Multi-directional Recurrent Neural Network

— **MDLSTM** Multi-directional Long Short-Term Memory

— **LM** Language Model

Principales notations

— w largeur de la fenêtre d'extraction des caractéristiques

— δ pas de la fenêtre glissante d'extraction des caractéristiques

— w_e largeur de la fenêtre d'estimation de la position locale de la ligne de base basse

— w_{smooth} largeur de la fenêtre de lissage gaussien pour la correction locale de la ligne de base

— h_{mean} and h_{max}, respectivement les hauteurs moyennes et maximales des caractères, estimées pour le nettoyage des images de lignes de texte.

— GSF, le Grammar Scale Factor permettant de pondérer la probabilité de la séquence de mots.

— WIP, le Word Insertion Penalty permettant de pénaliser les insertions de mots lors du décodage.

Annexes

Annexe A

Apprentissage des réseaux de neurones

A.1 Rétropropagation du gradient

Comme nous l'avons détaillé dans le paragraphe 1.5.2.c, la mise à jour des poids du réseau peut être effectuée lors d'un apprentissage par une méthode de rétropropagation du gradient (*Gradient backpropagation*). Il est nécessaire de disposer d'une base d'apprentissage S constituée d'un certain nombre de couples (x, z) où x est le vecteur d'entrée du réseau et z le vecteur de sortie de longueur K que l'on souhaite lui voir associer. y est la sortie donnée par le réseau en l'état. On notera w l'ensemble des paramètres du modèle (les poids synaptiques).

Pour ce faire, nous avons construit une fonction de perte $\mathcal{L}$ à minimiser :

$$\mathcal{L}(S) = -\ln \prod_{(x,z)\in S} p(z|x,w) = -\sum_{(x,z)\in S} \ln p(z|x,w) \tag{A.1}$$

L'évaluation de cette fonction sur un seul couple (x, z) dans le case d'une classification à

K classes s'exprime sous la forme :

$$\mathcal{L}(x,z) = -\ln p(z|x,w) = -\sum_{k=1}^{K} z_k \ln y_k \tag{A.2}$$

car les sorties y_k s'interprètent comme des probabilités. Les poids du réseau sont ainsi mis à jour : $w'_{ij} = w_{ij} - \alpha(z-y)\dfrac{\partial \mathcal{L}(x,z)}{\partial w_{ij}}$ où w'_{ij} est le poids mis à jour et α le pas de descente. L'objectif est de déterminer ces dérivées partielles de la fonction de perte en fonction des poids du réseau. Ces dérivées vont être déterminées grâce à une passe en arrière : elles sont calculées successivement en partant de la couche de sortie.

A.1.1 Équations de la passe en arrière

Nous considérons ici un perceptron à une seule couche cachée comme décrit dans la section 1.5.2.c avec des neurones de sortie disposant d'une fonction d'activation 'softmax'. On considère la notation suivante :

$$\delta_k = \frac{\partial \mathcal{L}(x,z)}{\partial a_k} \tag{A.3}$$

Voici donc les équations de la passe en arrière (Backward pass) pour la couche cachée :

$$\frac{\partial \mathcal{L}(x,z)}{\partial y_k} = -\sum_{k'=1}^{K} \frac{\partial}{\partial y_k}(z_{k'} \ln y_{k'}) = -\frac{z_k}{y_k} \tag{A.4}$$

$$\delta_k = \frac{\partial \mathcal{L}(x,z)}{\partial a_k} = \sum_{k'=1}^{K} \frac{\partial \mathcal{L}(x,z)}{\partial y_{k'}} \frac{\partial y_{k'}}{\partial a_k} = y_k - z_k \tag{A.5}$$

$$\frac{\partial \mathcal{L}(x,z)}{\partial w_{hk}} = \frac{\partial \mathcal{L}(x,z)}{\partial a_k} \frac{\partial a_k}{\partial w_{hk}} = \delta_k b_h \tag{A.6}$$

où
- y_k valeur du k-ième neurone de sortie ($k \in K$)
- a_k somme des activations de la couche cachée en entrée du k-ième neurone de sortie ($k \in K$)
- w_{hk} poids synaptique du h-ième neurone caché au k-ième neurone de sortie

Pour obtenir les dérivées en fonction des poids allant de la couche d'entrée à la couche cachée w_{ih}, l'équation reste identique :

$$\frac{\partial \mathcal{L}(x,z)}{\partial w_{ih}} = \frac{\partial \mathcal{L}(x,z)}{\partial a_h}\frac{\partial a_h}{\partial w_{ih}} = \delta_h x_i \tag{A.7}$$

Le terme δ_h est calculé récursivement :

$$\delta_h = \frac{\partial \mathcal{L}(x,z)}{\partial a_h} = \theta'(a_h)\sum_{k=1}^{K}\delta_k w_{hk} \tag{A.8}$$

Le détail de ces calculs peut être retrouvé dans [Graves, 2012].

A.2 Rétropropagation du gradient à travers le temps

Dans le cas de la classification de séquences temporelles par un réseau récurrent tel que le BLSTM, on utilisera une méthode d'apprentissage dérivée de la précédente : la rétropropagation du gradient à travers le temps (*Backpropagation Through Time - BPTT*) [Werbos, 1990]. Le principe général de cette méthode est de déplier ce réseau à travers le temps. Ainsi, on se ramène à un réseau feed-forward auquel on peut appliquer une rétropropagation du gradient classique.

A.2.1 Équations de la passe en arrière pour un réseau récurrent

De façon similaire au perceptron, on adopte la notation suivante pour un réseau récurrent :

$$\delta_k^t = \frac{\partial \mathcal{L}(x,z)}{\partial a_k^t} \tag{A.9}$$

Voici donc les équations de la passe en arrière (Backward pass) pour la couche cachée :

$$\frac{\partial \mathcal{L}(x,z)}{\partial y_k^t} = -\sum_{k'=1}^{K}\frac{\partial}{\partial y_k^t}(z_{k'}^t \ln y_{k'}^t) = -\frac{z_k^t}{y_k^t} \tag{A.10}$$

$$\frac{\partial \mathcal{L}(x,z)}{\partial a_k^t} = \sum_{k'=1}^{K}\frac{\partial \mathcal{L}(x,z)}{\partial y_{k'}^t}\frac{\partial y_{k'}^t}{\partial a_k^t} = y_k^t - z_k^t \tag{A.11}$$

$$\frac{\partial \mathcal{L}(x,z)}{\partial w_{hk}} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}(x,z)}{\partial a_k^t}\frac{\partial a_k^t}{\partial w_{hk}} = \sum_{t=1}^{T}\delta_k^t b_h^t \tag{A.12}$$

où
- y_k^t valeur du k-ième neurone de sortie à l'instant t($k \in K$)
- a_k^t somme des activations de la couche cachée en entrée du k-ième neurone de sortie à l'instant t ($k \in K$)
- w_{hk} poids synaptique du h-ième neurone caché au k-ième neurone de sortie

Pour obtenir les dérivées en fonction des poids allant de la couche d'entrée à la couche cachée w_{ih}, l'équation reste identique :

$$\frac{\partial \mathcal{L}(x,z)}{\partial w_{ih}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}(x,z)}{\partial a_h^t} \frac{\partial a_h^t}{\partial w_{ih}} = \sum_{t=1}^{T} \delta_h^t x_i^t \tag{A.13}$$

Les dérivées en fonction des poids de récurrences $w_{hh'}$ sont :

$$\frac{\partial \mathcal{L}(x,z)}{\partial w_{hh'}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}(x,z)}{\partial a_h^t} \frac{\partial a_h^t}{\partial w_{hh'}} = \sum_{t=1}^{T} \delta_h^t b_h^{t-1} = \sum_{t=1}^{T} \delta_h^{t+1} b_h^t \tag{A.14}$$

Le terme δ_h^t est calculé récursivement en fonction de la couche de sortie K à l'instant t et de la couche cachée H à l'instant $t+1$:

$$\delta_h^t = \theta'(a_h^t)(\sum_{k=1}^{K} \delta_k^t w_{hk} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} w_{hh'}) \tag{A.15}$$

A.2.2 Équations de la passe en arrière pour un réseau récurrent à cellules LSTM

De façon similaire au perceptron, on adopte la notation suivante :

$$\epsilon_c^t = \frac{\partial \mathcal{L}}{\partial b_c^t} \qquad \epsilon_s^t = \frac{\partial \mathcal{L}(x,z)}{\partial s_c^t} \tag{A.16}$$

Sorties des cellules :

$$\epsilon_c^t = \sum_{k=1}^{K} w_{ck} \delta_k^t + \sum_{h=1}^{H} w_{ch} \delta_h^{t+1} \tag{A.17}$$

Portes de sortie :

$$\delta_\omega^t = f'(a_\omega^t) h(s_c^t) \epsilon_c^t \tag{A.18}$$

Cellules :

$$\epsilon_s^t = b_\omega^t h'(s_c^t) \epsilon_c^t + b_\phi^{t+1} \epsilon_s^{t+1} + w_{c\iota} \delta_\iota^{t+1} + w_{c\phi} \delta_\phi^{t+1} + w_{c\omega} \delta_\omega^t \tag{A.19}$$

$$\delta_c^t = b_\iota^t g'(a_c^t) \epsilon_s^t \tag{A.20}$$

Portes d'oubli :

$$\delta_\phi^t = f'(a_\phi^t) s_c^{t-1} \epsilon_s^t \tag{A.21}$$

Portes d'entrée :

$$\delta_\iota^t = f'(a_\iota^t) g(a_c^t) \epsilon_s^t \tag{A.22}$$

Annexe B

Décodage CTC par token passing

De même que pour l'apprentissage (section 4.6), on définit le mot w' comme le mot w auquel on ajout des *blancs* en début et en fin de séquence et entre chaque paire d'étiquettes. Un jeton $tok = (score, historique)$ comprend deux attributs : son score et son historique. L'historique correspond au chemin parcouru par le jeton pour arriver là et la log-probabilité de ce chemin fournit son score.

À chaque instant t de la séquence de sortie de longueur T et pour chaque segment s de chaque mot w' correspond un jeton $tok(w, s, t)$. Ce jeton est celui ayant le score le plus élevé arrivant au segment s au temps t.

Le décodage contraint par dictionnaire et modèle de langage est présenté dans l'algorithme 5. On note y_c^t l'activation obtenue en sortie du réseau pour l'étiquette c à l'instant t et $P(w|w')$ la probabilité du bigramme (w', w). $tok(w, 0, t)$ correspond au jeton arrivant au mot w au temps t et sera nommé jeton d'entrée. $tok(w, -1, t)$ correspond au jeton repartant du mot w au temps t et sera nommé jeton de sortie.

Les jetons sont tout d'abord initialisés avec les premières activations obtenues au temps $t = 1$ en sortie du réseau. Ensuite, les jetons sont propagés au fil des instants t. À chaque fin de mot, la log-probabilité du bigramme correspondant est ajoutée au score du jeton. Enfin, lorsque tous les instants ont été parcourus, il suffit de sélectionner le jeton au score maximal. Son historique donne la séquence de mots ainsi décodée.

Algorithme 5 Pseudo-code pour le décodage CTC par passage du jeton (*token passing*) avec un modèle de langage bigramme

Soit D le dictionnaire et $\{p(w_|w')\}$ l'ensemble des probabilités bigrammes
Initialisation des jetons :
for words $w \in D$ **do**
  $tok(w, 1, 1) = (lny_b^1, (w))$
  $tok(w, 2, 1) = (lny_{w_1}^1, (w))$ où w_1 est la première lettre de w
  if $|w| = 1$ **then**
    $tok(w, -1, 1) = tok(w, 2, 1)$
  else
    $tok(w, -1, 1) = (-\infty, \emptyset)$
    $tok(w, s, 1) = (-\infty, \emptyset)$ pour tous les autres s
  end if
end for

Propagation :
for $t = 2 \rightarrow T$ **do**
  tri des jetons de sortie $tok(w, -1, t-1)$ par score croissant
  for words $w \in D$ **do**
    $w^* = \arg\max_{\hat{w}}[tok(\hat{w}, -1, t-1).score + \ln P(w|\hat{w})]$
    $tok(w, 0, t).score = tok(w^*, -1, t-1).score + \ln P(w|w^*)]$
    $tok(w, 0, t).historique = (tok(w^*, -1, t-1).historique, w)$
    for segment $s = 1 \rightarrow |w'|$ **do**
      P={tok(w,s,t-1),tok(w,s-1,t-1)}
      if then$w'_s \neq blank$ and $s > 2$ and $w_{s'-2} \neq w'_s$
        ajouter $tok(w, s-2, t-1)$ à P
      end if
      $tok(w, s, t)$ = jeton de P avec le plus haut score
      $tok(w, s, t).score+ = \ln y_{w'_s}^t$
    end for
    $tok(w, -1, t)$ = token de plus haut score de $\{tok(w, |w'|, t), tok(w, |w'|-1, t)\}$
  end for
end for

Terminaison :
$w^* = \arg\max_w tok(w, -1, T).score$
Renvoyer comme séquence de mots $tok(w^*, -1, T).history$

Bibliographie

[Abd-Almageed *et al.*, 2009] Abd-Almageed, W., Kumar, J. et Doermann, D. (2009). Page rule-line removal using linear subspaces in monochromatic handwritten arabic documents. *In Proceedings of the 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 768–772.

[Al-Hajj-Mohamad *et al.*, 2005] Al-Hajj-Mohamad, R., Likforman-Sulem, L. et Mokbel, C. (2005). Arabic handwriting recognition using baseline dependant features and Hidden Markov Modeling. *In Proceedings of the 8th International Conference on Document Analysis and Recognition*, ICDAR '05, pages 893–897.

[Al-Hajj-Mohamad *et al.*, 2009] Al-Hajj-Mohamad, R., Likforman-Sulem, L. et Mokbel, C. (2009). Combining slanted-frame classifiers for improved HMM-based arabic handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(7) :1165–1177.

[Amin et Fischer, 2000] Amin, A. et Fischer, S. (2000). A document skew detection method using the Hough transform. *Pattern Analysis and Applications*, 3(3) :243–253.

[Arvind *et al.*, 2007] Arvind, K., Kumar, J. et Ramakrishnan, A. (2007). Line removal and restoration of handwritten strokes. *In Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, volume 3, pages 208–214.

[Bahl *et al.*, 1983] Bahl, L., Jelinek, F. et Mercer, R. (1983). A statistical approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2) :179–190.

[Belaïd et Saon, 1997] Belaïd, A. et Saon, G. (1997). Utilisation des processus markoviens en reconnaissance de l'écriture. *Traitement du Signal*, 14(2) :161–177.

[Bengio *et al.*, 2001] Bengio, Y., Ducharme, R. et Vincent, P. (2001). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(2) :1137–1155.

[Bengio *et al.*, 1995] Bengio, Y., LeCun, Y., Nohl, C. et Burges, C. (1995). LeRec : a NN/HMM hybrid for on-line handwriting recognition. *Neural Computation*, 7(6) :1289–1303.

[Bertolami et Bunke, 2008] Bertolami, R. et Bunke, H. (2008). Integration of n-gram language models in multiple classifier systems for offline handwritten text line recognition. *IJPRAI*, 22(7) :1301–1321.

[Bertolami *et al.*, 2006] Bertolami, R., Halter, B. et Bunke, H. (2006). Combination of multiple handwritten text line recognition systems with a recursive approach. *In Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition*, IWFHR '06.

[Bertolami *et al.*, 2007] Bertolami, R., Uchida, S., Zimmermann, M. et Bunke, H. (2007). Non-uniform slant correction for handwritten text line recognition. *In Proceedings of the 9th International Conference on Document Analysis and Recognition*, volume 1 de *ICDAR '07*, pages 18 –22.

[Bianne-Bernard, 2011] Bianne-Bernard, A.-L. (2011). *Reconnaissance de mots manuscrits cursifs par modèles de Markov cachés en contexte : application au français, à l'anglais et à l'arabe*. Thèse de doctorat, Télécom ParisTech.

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

[Boubaker *et al.*, 2010] Boubaker, H., El Baati, A., Kherallah, M., Alimi, A. et El-Abed, H. (2010). Online arabic handwriting modeling system based on the graphemes segmentation. *In Proceedings of the 15th International Conference on Pattern Recognition*, ICPR '00, pages 2061–2064.

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

[Broomhead et Lowe, 1988] Broomhead, D. S. et Lowe, D. (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems 2*, pages 321–355.

[Brown *et al.*, 1992] Brown, P., DeSouza, P., Mercer, R., Della-Pietra, V. et Lai, J. (1992). Class-based n-gram models of natural language. *Computational Linguistic*, 18(4) :467–479.

[Bunke *et al.*, 1995] Bunke, H., Roth, M. et Schukat-Talamazzini, E. G. (1995). Off-line cursive handwriting recognition using Hidden Markov Models. *Pattern Recognition*, 28(9) :1399–1413.

[Buse *et al.*, 1997] Buse, R., Liu, Z.-Q. et Caelli, T. (1997). A structural and relational approach to handwritten word recognition. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 27(5) :847–861.

[Cappé *et al.*, 2005] Cappé, O., Moulines, E. et Ryden, T. (2005). *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2 :303–314.

[Davis et Mermelstein, 1980] Davis, S. et Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4) :357–366.

[De Oliveira *et al.*, 2002] De Oliveira, J.J., J., de Carvalho, J., de A Freitas, C. et Sabourin, R. (2002). Feature sets evaluation for handwritten word recognition. *In Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, IWFHR '02, pages 446–450.

[Doetsch *et al.*, 2012] Doetsch, P., Hamdani, M., Ney, H., Gimenez, A., Andres-Ferrer, J. et Juan, A. (2012). Comparison of Bernoulli and Gaussian HMMs using a vertical repositioning technique for off-line handwriting recognition. *In Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition*, ICFHR '12, pages 3–7.

[Drira *et al.*, 2006] Drira, F., LeBourgeois, F. et Emptoz, H. (2006). Restoring ink bleed-through degraded document images using a recursive unsupervised classification technique. *In* Bunke, H. et Spitz, A., éditeurs : *Document Analysis Systems VII*, volume 3872 de *Lecture Notes in Computer Science*, pages 38–49. Springer.

[El Abed et Margner, 2009] El Abed, H. et Margner, V. (2009). How to improve a handwriting recognition system. *In Proceedings of the 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 1181–1185.

[El-Yacoubi *et al.*, 1999] El-Yacoubi, A., Sabourin, R., Suen, C. Y. et Gilloux, M. (1999). An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8) :752–760.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2) :179–211.

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

[Ficus, 1997] Ficus, J. G. (1997). A post-processing system to yield reduced error word rates : Recognizer output voting error reduction (ROVER). *In IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–354.

[Fink et Plotz, 2007] Fink, G. et Plotz, T. (2007). On the use of context-dependent modeling units for HMM-based offline handwriting recognition. *In Proceedings of the 9th International Conference on Document Analysis and Recognition*, ICDAR '07, pages 729–733.

[Fischer et Bunke, 2009] Fischer, A. et Bunke, H. (2009). Kernel PCA for HMM-based cursive handwriting recognition. *In Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, CAIP '09, pages 181–188. Springer.

[Fischer *et al.*, 2012] Fischer, A., Keller, A., Frinken, V. et Bunke, H. (2012). Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7) :934 – 942. Special Issue on Awards from ICPR 2010.

[Frinken et Bunke, 2010] Frinken, V. et Bunke, H. (2010). Self-training for handwritten text line recognition. *In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 6419 de *Lecture Notes in Computer Science*, pages 104–112. Springer.

[Fujisawa, 2008] Fujisawa, H. (2008). Forty years of research in character and document recognition-an industrial perspective. *Pattern Recogn.*, 41(8) :2435–2446.

[Fukushima, 1980] Fukushima, K. (1980). Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36 :193–202.

[Gatos *et al.*, 2006] Gatos, B., Pratikakis, I. et Perantonis, S. J. (2006). Adaptive degraded document image binarization. *Pattern Recognition*, 39(3) :317–327.

[Gatos *et al.*, 2010] Gatos, B., Stamatopoulos, N. et Louloudis, G. (2010). ICFHR 2010 handwriting segmentation contest. *In Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition*, ICFHR '10, pages 737–742.

[Gers *et al.*, 2000] Gers, F. A., Schmidhuber, J. et Cummins, F. A. (2000). Learning to forget : Continual prediction with LSTM. *Neural Computation*, 12(10) :2451–2471.

[Gers *et al.*, 2002] Gers, F. A., Schraudolph, N. N. et Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3 :115–143.

[Giebel, 1971] Giebel, H. (1971). Feature extraction and recognition of handwritten characters by homogeneous layers. *In* Grüsser, O.-J. et Klinke, R., éditeurs : *Pattern Recognition in Biological and Technical Systems*, pages 162–169. Springer.

[Gilloux, 1994] Gilloux, M. (1994). Reconnaissance de chiffres manuscrits par modèle de Markov pseudo-2d. *In Actes du 3ème Colloque National sur l'Écrit et le Document*, pages 11–17.

[Giménez et Juan, 2009] Giménez, A. et Juan, A. (2009). Embedded Bernoulli mixture HMMs for handwritten word recognition. *In Proceedings of the 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 896–900.

[Good, 1953] Good, I. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40 :237–264.

[Graves, 2012] Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer.

[Graves, 2013] Graves, A. (2013). Rnnlib : A recurrent neural network library for sequence learning problems. `http://sourceforge.net/projects/rnnl/`.

[Graves *et al.*, 2006] Graves, A., Fernández, S., Gomez, F. J. et Schmidhuber, J. (2006). Connectionist temporal classification : labelling unsegmented sequence data with recurrent neural networks. *In International Conference on Machine Learning*, pages 369–376.

[Graves *et al.*, 2007] Graves, A., Fernandez, S. et Schmidhuber, J. (2007). Multi-dimensional recurrent neural networks. Rapport technique, IDISIA.

[Graves *et al.*, 2009] Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H. et Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5).

[Graves et Schmidhuber, 2005] Graves, A. et Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6) :602–610.

[Graves et Schmidhuber, 2008] Graves, A. et Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. *In Neural Information Processing Systems*, pages 545–552.

[Grosicki *et al.*, 2006] Grosicki, E., Carré, M., Augustin, E. et Prêteux, F. (2006). La campagne d'évaluation RIMES pour la reconnaissance de courriers manuscrits. *In Colloque International Francophone sur l'Ecrit et le Document.*

[Grosicki et El-Abed, 2009] Grosicki, E. et El-Abed, H. (2009). ICDAR 2009 handwriting recognition competition. *In Proceedings of the 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 1398–1402.

[Grosicki et El-Abed, 2011] Grosicki, E. et El-Abed, H. (2011). ICDAR 2011 : French handwriting recognition competition. *In Proceedings of the 11th International Conference on Document Analysis and Recognition*, ICDAR '11.

[Guyon *et al.*, 1991] Guyon, I., Albrecht, P., LeCun, Y., Denker, J. et W., H. (1991). Design of a neural network character recognizer for a touch terminal. *Pattern Recognition*, 24(2) :105–119.

[Hamdani *et al.*, 2009] Hamdani, M., Abed, H. E., Kherallah, M. et Alimi, A. M. (2009). Combining multiple hmms using on-line and off-line features for off-line arabic handwriting recognition. *In Proceedings of the 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 201–205.

[Hochreiter *et al.*, 2001] Hochreiter, S., Bengio, Y., Frasconi, P. et Schmidhuber, J. (2001). Gradient flow in recurrent nets : the difficulty of learning long-term dependencies. *In A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

[Hochreiter et Schmidhuber, 1997] Hochreiter, S. et Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8) :1735–1780.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8) :2554–2558.

[Hornik, 1991] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2) :251–257.

[Hubel et Wiesel, 1968] Hubel, D. H. et Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, 195 :215–243.

[Katz, 1987] Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3) :400–401.

[Kavallieratou et Stamatatos, 2006] Kavallieratou, E. et Stamatatos, E. (2006). Improving the quality of degraded document images. *In Proceedings of the Second International Conference on Document Image Analysis for Libraries*, DIAL '06, pages 340–349. IEEE Computer Society.

[Knerr et Augustin, 1998] Knerr, S. et Augustin, E. (1998). A neural network-hidden markov model hybrid for cursive word recognition. *In Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1518–1520 vol.2.

[Kohonen, 1989] Kohonen, T. (1989). *Self-organization and associative memory : 3rd edition*. Springer.

[Kozielski *et al.*, 2012] Kozielski, M., Forster, J. et Ney, H. (2012). Moment-based image normalization for handwritten text recognition. *In Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition*, ICFHR '12, pages 256–261.

[Kumar et Doermann, 2011] Kumar, J. et Doermann, D. (2011). Fast rule-line removal using integral images and support vector machines. *In Proceedings of the 11th International Conference on Document Analysis and Recognition*, ICDAR '11, pages 584–588.

[Lavrenko *et al.*, 2004] Lavrenko, V., Rath, T. M. et Manmatha, R. (2004). Holistic word recognition for handwritten historical documents. *In Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*, pages 278–287.

[LeCun *et al.*, 1998] LeCun, Y., Bottou, L., Bengio, Y. et Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324.

[Lee et Kim, 1995] Lee, S.-W. et Kim, Y.-J. (1995). A new type of recurrent neural network for handwritten character recognition. *In Proceedings of the 3th International Conference on Document Analysis and Recognition*, ICDAR '95, pages 38–41.

[Lelore et Bouchara, 2011] Lelore, T. et Bouchara, F. (2011). Super-resolved binarization of text based on the fair algorithm. *In Proceedings of the 11th International Conference on Document Analysis and Recognition*, ICDAR '11, pages 839–843.

[Lemaitre *et al.*, 2009] Lemaitre, A., Camillerapp, J. et Coüasnon, B. (2009). Multi-script baseline detection using perceptive vision. *In Proc. 14th Biennial Conference of the International Graphonomics Society.*

[Lemaitre *et al.*, 2011] Lemaitre, A., Camillerapp, J. et Coüasnon, B. (2011). A perceptive method for handwritten text segmentation. *In Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7874.

[Lemarié, 1993] Lemarié, B. (1993). Practical realization of a radial basis function network for handwritten digit recognition. *In* Mira, J., Cabestany, J. et Prieto, A., éditeurs : *New Trends in Neural Computation*, volume 686 de *Lecture Notes in Computer Science*, pages 131–136. Springer.

[Likforman-Sulem, 1998] Likforman-Sulem, L. (1998). Extraction d'éléments graphiques dans les images de manuscrits. *In Colloque International Francophone sur l'Écrit et le Document.*

[Likforman-Sulem, 2003] Likforman-Sulem, L. (2003). Apport du traitement des images à la numérisation des documents manuscrits anciens. *Documents Numérique*, 7(3-4) :13–26.

[Likforman-Sulem et Barney-Smith, 2012] Likforman-Sulem, L. et Barney-Smith, E. (2012). *Imagerie, reconnaissance des formes : théorie et pratique sous Matlab*. Technosup. Ellipses.

[Likforman-Sulem *et al.*, 2011] Likforman-Sulem, L., Darbon, J. et Barney Smith, E. H. (2011). Enhancement of historical printed document images by combining total variation regularization and non-local means filtering. *Image and Vision Computing*, 29(5) :351 – 363.

[Likforman-Sulem *et al.*, 2007] Likforman-Sulem, L., Zahour, A. et Taconet, B. (2007). Text line segmentation of historical documents : a survey. *Int. J. Doc. Anal. Recognit.*, 9(2) :123–138.

[Liwicki *et al.*, 2007] Liwicki, M., Graves, A., Bunke, H. et Schmidhuber, J. (2007). A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. *In Proceedings of the 9th International Conference on Document Analysis and Recognition.*

[Lladós *et al.*, 2008] Lladós, J., Karatzas, D., Mas, J. et Sánchez, G. (2008). A generic architecture for the conversion of document collections into semantically annotated digital archives. *J. UCS*, 14(18) :2912–2935.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. *In Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–1157.

[Lu et Shridhar, 1996] Lu, Y. et Shridhar, M. (1996). Character segmentation in handwritten words — an overview. *Pattern Recognition*, 29(1) :77 – 96.

[M. Blumenstein *et al.*, 2002] M. Blumenstein, M. M., Cheng, C. K. et Liu, X. Y. (2002). New preprocessing techniques for handwritten word recognition. *In Second IASTED International Conference on Visualization*, pages 480–484.

[Maddouri *et al.*, 2008] Maddouri, S. S., Samoud, F. B., Bouriel, K., Ellouze, N. et El-Abed, H. (2008). Baseline extraction : Comparison of six methods on IFN/ENIT database. *In Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition*, ICFHR '08.

[Markram, 2006] Markram, H. (2006). The Blue Brain Project. *Nature Reviews Neuroscience.*

[Marti et Bunke, 1999] Marti, U.-V. et Bunke, H. (1999). A full english sentence database for off-line handwriting recognition. *In Proceedings of the 5th International Conference on Document Analysis and Recognition*, ICDAR '99, pages 705–708.

[Marti et Bunke, 2000] Marti, U.-V. et Bunke, H. (2000). Handwritten sentence recognition. *In Proceedings of the 15th International Conference on Pattern Recognition*, ICPR '00, pages 467–470.

[Marti et Bunke, 2001] Marti, U.-V. et Bunke, H. (2001). Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *IJPRAI*, 15(1) :65–90.

[McCulloch et Pitts, 1943] McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4) :115–133.

[Menasri *et al.*, 2012] Menasri, F., Louradour, J., Bianne-Bernard, A.-L. et Kermorvant, C. (2012). The A2iA french handwriting recognition system at the Rimes-ICDAR 2011 competition. *In Proceedings of Electronic Imaging - Document Recognition and Retrieval (DRR) XIX.*

[Morillot *et al.*, 2012a] Morillot, O., Grosicki, E. et Likforman-Sulem, L. (2012a). Reconnaissance de courriers manuscrits par HMMs contextuels et modèle de langage. *In Actes de CIFED 2012.*

[Morillot *et al.*, 2013a] Morillot, O., Grosicki, E. et Likforman-Sulem, L. (2013a). Reconnaissance de courriers manuscrits par HMM contextuels et modèle de langage. *Document Numérique*, 16(2/2013) :69–90.

[Morillot *et al.*, 2012b] Morillot, O., Likforman-Sulem, L. et Grosicki, E. (2012b). Construction of language models for an handwritten mail reading system. *In Proceedings of Electronic Imaging - Document Recognition and Retrieval (DRR) XIX*.

[Morillot *et al.*, 2013b] Morillot, O., Likforman-Sulem, L. et Grosicki, E. (2013b). Comparative study of HMM and BLSTM segmentation-free approaches for the recognition of handwritten text-lines. *In Proceedings of the 12th International Conference on Document Analysis and Recognition*, ICDAR '13.

[Morillot *et al.*, 2013c] Morillot, O., Likforman-Sulem, L. et Grosicki, E. (2013c). New baseline correction algorithm for text-line recognition with bidirectional recurrent neural networks. *SPIE Journal of Electronic Imaging*, 22(2).

[Morillot *et al.*, 2013d] Morillot, O., Oprean, C., Likforman-Sulem, L., Mokbel, C., Chammas, E. et Grosicki, E. (2013d). The UOB-Telecom ParisTech Arabic handwriting recognition and translation systems for the OpenHaRT 2013 competition. *In NIST-OpenHaRT Workshop*.

[Natarajan *et al.*, 2008] Natarajan, P., Saleem, S., Prasad, R., MacRostie, E. et Subramanian, K. (2008). Multi-lingual offline handwriting recognition using hidden Markov models : a script-independent approach. *In Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition*, SACH'06, pages 231–250. Springer.

[Ney *et al.*, 1994] Ney, H., Essen, U. et Kneser, R. (1994). On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8 :1–38.

[NIST, 2010] NIST (2010). NIST 2010 open handwriting recognition and translation evaluation plan. `http://www.nist.gov/itl/iad/mig/upload/OpenHaRT2010_EvalPlan_v2-8.pdf`.

[NIST, 2013a] NIST (2013a). NIST 2013 open handwriting recognition and translation evaluation plan. `http://www.nist.gov/itl/iad/mig/upload/OpenHaRT2013_EvalPlan_v1-7.pdf`.

[NIST, 2013b] NIST (2013b). Open handwriting recognition and translation evaluation. `http://www.nist.gov/itl/iad/mig/hart.cfm`.

[Oprean *et al.*, 2013a] Oprean, C., Likforman-Sulem, L. et Mokbel, C. (2013a). Handwritten word preprocessing for database adaptation. *In Proceedings of Electronic Imaging - Document Recognition and Retrieval (DRR) XX*.

[Oprean *et al.*, 2013b] Oprean, C., Likforman-Sulem, L., Popescu, A. et Mokbel, C. (2013b). Using the web to create dynamic dictionaries in handwritten out-of-vocabulary word recognition. *In Proceedings of the 12th International Conference on Document Analysis and Recognition*, ICDAR '13.

[Otsu, 1979] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1) :62–66.

[Perraud *et al.*, 2006] Perraud, F., Viard-Gaudin, C. et Morin, E. (2006). Language Independent Statistical Models for on-Line Handwriting Recognition. *In Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition*, IWFHR '06.

[Plamondon et Srihari, 2000] Plamondon, R. et Srihari, S. (2000). Online and offline handwriting recognition : A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :63–84.

[Poisson *et al.*, 2004] Poisson, E., Viard-Gaudin, C. et Lallican, P.-M. (2004). Système TDNN/HMM de reconnaissance de mots cursifs en ligne à apprentissage simplifié. *In Actes de CIFED 2004*.

[Pratikakis *et al.*, 2011] Pratikakis, I., Gatos, B. et Ntirogiannis, K. (2011). ICDAR 2011 document image binarization contest (DIBCO 2011). *In Proceedings of the 11th International Conference on Document Analysis and Recognition*, ICDAR '11, pages 1506–1510.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *In Proceedings of the IEEE*, pages 257–286.

[Rodríguez-Serrano et Perronnin, 2008] Rodríguez-Serrano, J. A. et Perronnin, F. (2008). Local gradient histogram features for word spotting in unconstrained handwritten documents. *In Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition*, ICFHR '08.

[Rodríguez-Serrano et Perronnin, 2009] Rodríguez-Serrano, J. A. et Perronnin, F. (2009). Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recognition*, 42(9) :2106–2116.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The Perceptron : probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 :386–408.

[Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1986). Learning internal representations by error propagation. *In* Rumelhart, D. E., McClelland, J. L. et PDP Research Group, C., éditeurs : *Parallel distributed*

processing : explorations in the microstructure of cognition, vol. 1, pages 318–362. MIT Press.

[Rybach *et al.*, 2011] Rybach, D., Hahn, S., Lehnen, P., Nolden, D., Sundermeyer, M., Tüske, Z., Wiesler, S., Schlüter, R. et Ney, H. (2011). RASR - the RWTH aachen university open source speech recognition toolkit. *In IEEE Automatic Speech Recognition and Understanding Workshop*.

[Saul et Jordan, 1986] Saul, L. K. et Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. *In Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546.

[Schambach, 2003] Schambach, M.-P. (2003). Model length adaptation of an HMM based cursive word recognition system. *In Proceedings of the 7th International Conference on Document Analysis and Recognition*, ICDAR '03, pages 109–113.

[Schlapbach *et al.*, 2005] Schlapbach, A., Kilchherr, V. et Bunke, H. (2005). Improving writer identification by means of feature selection and extraction. *In Proceedings of the 8th International Conference on Document Analysis and Recognition*, ICDAR '05, pages 131–135.

[Schuster et Paliwal, 1997] Schuster, M. et Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11) :2673–2681.

[Senior et Robinson, 1998] Senior, A. W. et Robinson, A. J. (1998). An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 :309–321.

[Steinherz *et al.*, 1999] Steinherz, T., Rivlin, E. et Intrator, N. (1999). Offline cursive script word recognition - a survey. *International Journal of Document Analysis and Recognition*, 2(2) :90–110.

[Stolcke, 2002] Stolcke, A. (2002). SRILM : An extensible language modeling toolkit. *In International Conference on Spoken Language Processing*, pages 901–904.

[Tay *et al.*, 2001] Tay, Y. H., Lallican, P.-M., Khalid, M., Viard-Gaudin, C. et Knerr, S. (2001). Offline handwritten word recognition using a hybrid Neural Network and Hidden Markov Models. *In 6th International Symposium on Signal Processing and its Applications*.

[Thomas *et al.*, 2013] Thomas, S., Chatelain, C., Heutte, L. et Paquet, T. (2013). Un modèle neuro markovien profond pour l'extraction de séquences dans des documents manuscrits. *Documents Numérique*, 16(2) :49–69.

[Tong *et al.*, 2013] Tong, A., Przybocki, M., Märgner, V. et El-Abed, H. (2013). NIST 2013 open handwriting recognition and translation OpenHaRT'13) evaluation. *In NIST-OpenHaRT Workshop*.

[Uchida *et al.*, 2001] Uchida, S., Taira, E. et Sakoe, H. (2001). Nonuniform slant correction using dynamic programming. *In Proceedings of the 6th International Conference on Document Analysis and Recognition*, ICDAR '01, pages 434–438.

[Viard-Gaudin *et al.*, 1999] Viard-Gaudin, C., Lallican, P.-M., Knerr, S. et Binter, P. (1999). The ireste on/off (ironoff) dual handwriting database. *In Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 455–458.

[Vinciarelli *et al.*, 2004] Vinciarelli, A., Bengio, S. et Bunke, H. (2004). Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 :709–720.

[Vinciarelli et Luettin, 2001] Vinciarelli, A. et Luettin, J. (2001). A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9) :1043–1050.

[Vinciarelli et Perrone, 2003] Vinciarelli, A. et Perrone, M. (2003). Combining online and offline handwriting recognition. *In Proceedings of the 7th International Conference on Document Analysis and Recognition*, ICDAR '03, pages 844–.

[Waibel *et al.*, 1989] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. et Lang, K. (1989). Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3) :328–339.

[Werbos, 1974] Werbos, P. (1974). *Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences*. Thèse de doctorat, Harvard University.

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time : what it does and how to do it. *Proceedings of the IEEE*, 78(10) :1550–1560.

[Wöllmer *et al.*, 2010] Wöllmer, M., Eyben, F., Graves, A., Schuller, B. et Rigoll, G. (2010). Improving keyword spotting with a tandem BLSTM-DBN architecture. *In* Solé-Casals, J. et Zaiats, V., éditeurs : *Advances in Nonlinear Speech Processing*, volume 5933 de *Lecture Notes in Computer Science*, pages 68–75. Springer.

Reconnaissance de textes manuscrits par Modèles de Markov Cachés et Réseaux de Neurones Récurrents : application à l'écriture latine et arabe

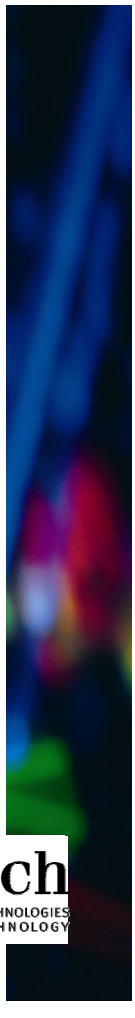
Olivier MORILLOT

RÉSUMÉ : La reconnaissance d'écriture manuscrite est une composante essentielle de l'analyse de document. Une tendance actuelle de ce domaine est de passer de la reconnaissance de mots isolés à celle d'une séquence de mots. Notre travail consiste donc à proposer un système de reconnaissance de lignes de texte sans segmentation explicite de la ligne en mots. Afin de construire un modèle performant, nous intervenons à plusieurs niveaux du système de reconnaissance. Tout d'abord, nous introduisons deux méthodes de prétraitement originales : un nettoyage des images de lignes de texte et une correction locale de la ligne de base. Ensuite, nous construisons un modèle de langage optimisé pour la reconnaissance de courriers manuscrits. Puis nous proposons deux systèmes de reconnaissance à l'état de l'art fondés sur les HMM (*Hidden Markov Models*) contextuels et les réseaux de neurones récurrents BLSTM (*Bi-directional Long Short-Term Memory*). Nous optimisons nos systèmes afin de proposer une comparaison de ces deux approches. Nos systèmes sont évalués sur l'écriture cursive latine et arabe et ont été soumis à deux compétitions internationales de reconnaissance d'écriture. Enfin, en perspective de notre travail, nous présentons une stratégie de reconnaissance pour certaines chaînes de caractères hors-vocabulaire.

MOTS-CLEFS : reconnaissance d'écriture manuscrite, hors-ligne, lignes de texte, français, arabe, preprocessing, correction de la ligne de base, modèles de Markov cachés, HMM, HMM en contexte, réseaux de neurones récurrents, BLSTM, modèle de langage

ABSTRACT : Handwriting recognition is an essential component of document analysis. One of the popular trends is to go from isolated word to word sequence recognition. Our work aims to propose a text-line recognition system without explicit word segmentation. In order to build an efficient model, we intervene at different levels of the recognition system. First of all, we introduce two new preprocessing techniques : a cleaning and a local baseline correction for text-lines. Then, a language model is built and optimized for handwritten mails. Afterwards, we propose two state-of-the-art recognition systems based on contextual HMMs (*Hidden Markov Models*) and recurrent neural networks BLSTM (*Bi-directional Long Short-Term Memory*). We optimize our systems in order to give a comparison of those two approaches. Our systems are evaluated on arabic and latin cursive handwritings and have been submitted to two international handwriting recognition competitions. At last, we introduce a strategy for some out-of-vocabulary character strings recognition, as a prospect of future work.

KEY-WORDS : handwriting recognition, offline, text lines, cursive, french, arabic, prétraitement, baseline correction, hidden Markov models, HMM, contextual HMMs, recurrent neural networks, BLSTM, language model



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY