Extraction d'information dans des documents manuscrits non contraints : application au traitement automatique des courriers entrants manuscrits

S. Thomas

Laboratoire d'Informatique,
de Traitement de l'Information et des Systèmes
Université de Rouen

Thèse en vue de l'obtention du titre de
Docteur en Informatique de l'Université de Rouen

Extraction d'information dans des documents manuscrits non contraints : application au traitement automatique des courriers entrants manuscrits

Simon Thomas

Version remise aux rapporteurs :

xxx xxx	-	xxx	-	Président
Christian Viard-Gaudin	-	Université de Nantes	-	Rapporteur
Thierry Artières	-	Université de Paris 6	-	Rapporteur
Laurent Heutte	-	Université de Rouen	-	Directeur de thèse
Thierry Paquet	-	Université de Rouen	-	Encadrant de thèse
Clément Chatelain	-	INSA de Rouen	-	Encadrant de thèse

Table des matières

Introduction Générale

La plus fondamentale avancée intellectuelle de notre espèce est sans aucun doute le développement et la structuration du langage qui a permis de faciliter la communication et l'échange de connaissances par le biais de la parole. La parole est cependant assujettie à de nombreuses limitations liées par exemple à de potentielles incompréhensions ou à nos capacités limitées de mémorisation. Un tournant et véritable moteur dans notre évolution réside dans la conservation des connaissances sur des supports physiques durables : les documents. Ils ont connu une évolution constante permise par des découvertes et inventions fondamentales (l'encre, le papier, l'imprimerie, etc.) et la structuration de nos écrits à base d'alphabet de symboles. Aujourd'hui, de grandes masses de documents sont accessibles en version électronique via Internet.

L'entrée récente dans l'ère du tout numérique n'a toutefois pas marqué la fin d'échanges de documents manuscrits. Au contraire, de nombreux échanges administratifs, professionnels et personnels sont toujours effectués par ce biais, impliquant le traitement de grandes masses de documents, traitement particulièrement chronophage pour un opérateur humain. Des systèmes de lecture sont maintenant opérationnels et traitent quotidiennement des millions de documents. C'est le cas par exemple du courrier, des chèques bancaires ou des formulaires. Une stratégie efficace pour traiter ces documents est l'extraction des informations pertinentes qu'ils contiennent, c'est à dire la mise en relation des données essentielles à leur compréhension afin de répondre à la question *Qui a fait quoi à qui, quand et où ?* Pour les adresses, chèques et formulaires, leur structure et leur vocabulaire, connus *a priori*, permettent de définir des modèles de lecture robustes guidant l'extraction d'information. Dans le cas de documents non contraints par une structure forte, comme par exemple des manuscrits d'archives ou des textes mansucrits libres, l'accès à un contenu précis n'est pas immédiat. La localisation et la reconnaissance de l'écriture sont des étapes incontournables pour détecter la présence de l'information pertinente qu'ils contiennent.

Notre étude se focalise sur le traitement de courriers entrants manuscrits reçus quotidiennement et massivement par des administrations et entreprises. Leur traitement nécessite l'extraction d'informations variées telles que : des mots clés (pour déterminer l'objet du courrier par exemple), des séquences numériques (pour récupérer des données relatives à l'expéditeur et permettre de l'identifier) ou encore des séquences alphanumériques (comme des dates

ou des références de dossier par exemple, pour apporter un complément d'information à la compréhension du courrier). Notre travail consiste donc à mettre en œuvre des outils permettant d'accéder automatiquement et rapidement à ces données. Ceci constitue une première étape vers l'extraction d'information et doit permettre de faciliter la recherche d'un document ou d'un ensemble de documents au sein d'une masse ou d'en guider le traitement.

Nous proposons d'apporter une solution au traitement de courriers manuscrits par le développement d'un système de détection de mots, de séquences numériques et de séquences alphanumériques. Dans ce but, nous avons conçu un modèle générique de ligne de texte capable de discriminer l'information pertinente, définie en amont de la reconnaissance sous la forme d'une ou d'un ensemble d'informations à détecter, de l'information non pertinente à rejeter. Nous nous attarderons ensuite sur l'amélioration de la discrimination locales des formes des lignes de texte. Nous montrerons qu'une architecture profonde constitue un classifieur d'observation robuste lorsqu'elle est associée aux modèles de Markov cachés et qu'elle permet d'améliorer les performances du système. Il en résulte un système complet et générique permettant de détecter des informations de tout type (alphabétique, numérique ou alphanumérique) dans des documents manuscrits non contraints.

Dans le premier chapitre, nous évoquons les principales stratégies de traitement de documents électroniques dont le contenu est directement accessible et compréhensible par un système. Ceci nous permet de positionner les problèmes relatifs au traitement automatique de documents manuscrits, contraints par une structure forte dans un premier temps (comme les chèques bancaires, les adresses postales et les formulaires par exemple), puis d'évoquer les difficultés supplémentaires lorsque l'on s'attaque au traitement automatique de documents non contraints dans un second temps. Nous terminons ce chapitre en présentant en détail notre sujet d'étude et ses points clés que sont la localisation de l'écriture et la reconnaissance de l'information recherchée.

Dans le deuxième chapitre, nous dressons un état de l'art des systèmes de reconnaissance de l'écriture manuscrite. À cet effet, nous présentons le compromis à trouver entre les trois points clés de ces systèmes qui sont : *(i)* la localisation de l'information pertinente caractérisée par une étape de segmentation, *(ii)* la reconnaissance des entités isolées lors de la première étape, *(iii)* le rejet de l'information non recherchée. L'analyse critique de cette étude bibliographique nous permettra de proposer notre première contribution sous la forme d'un modèle générique de ligne pour la détection d'information dans des documents manuscrits non contraints. Il met en concurrence l'information à rechercher (sous la forme d'une requête) et un modèle de rejet dans le but

d'aborder conjointement les trois points identifiés que sont la segmentation, la reconnaissance et le rejet.

Nous présentons dans le troisième chapitre l'implémentation, autour du modèle générique de ligne, du système complet de détection d'information dans des documents manuscrits. L'ensemble des traitements consiste à préparer les images de document manuscrit afin de fournir des données exploitables par le moteur de reconnaissance contraint par le modèle de ligne introduit au chapitre précédent. Les modèles de Markov cachés, permettant d'intégrer aisément des connaissances *a priori* et donc notre modèle de ligne, supportent le cœur du système formé par le module de segmentation/reconnaissance/rejet. Les expérimentations sont conduites sur une base d'images de courriers entrants manuscrits que nous avons annotées pour estimer des performances en détection d'informations alphabétiques, numériques et alphanumériques.

Dans le dernier chapitre, nous étudions des pistes d'amélioration du système de détection et introduisons notre seconde contribution sous la forme d'un classifieur d'observations basé sur un réseau de neurones profond. Il remplace les mélanges de Gaussiennes classiquement embarqués dans les états des modèles de Markov cachés. Après avoir présenté les avancées récentes en matière d'apprentissage d'architectures profondes, nous montrons que l'association du réseau de neurones profond avec les modèles de Markov cachés permet de mieux discriminer localement les observations et contribue à améliorer les performances du système en détection d'information. En outre, ce réseau de neurones permet d'inférer automatiquement des représentations de haut niveau lors de son apprentissage. Nous montrerons qu'il constitue une alternative au module d'extraction de caractéristiques.

Chapitre 1

Traitement automatique de documents

Sommaire

1.1 Introduction

Le sujet de notre étude est la détection d'information dans des documents manuscrits non contraints et son application aux courriers entrants manuscrits. Ces courriers sont reçus quotidiennement et en grand nombre par les administrations et entreprises. Cette étude entre donc dans le cadre de la gestion électronique de documents et plus précisément du traitement automatique d'images de documents, vaste et très actif domaine de recherche de par l'hétérogénéité des documents traités et leur contenu extrêmement variable.

Aujourd'hui, la quantité de documents produits, stockés ou échangés augmente de manière exponentielle, nous confrontant à des masses de documents (notamment électroniques) toujours plus grandes. Face à ces masses, différents besoins en traitement sont apparus afin d'éviter la lecture complète de tous les documents d'une base pour accéder à une information précise diluée dans celles-ci. Ces besoins ont été abordés sous la forme de trois disciplines distinctes du domaine de la gestion électronique de documents. Si l'on veut sélectionner un sous ensemble pertinent de documents par rapport à une requête utilisateur, on parle de **recherche d'information**. On peut également vouloir classer les documents pour en faciliter le stockage et l'accès ce qui correspond à une problématique de **catégorisation de documents**. Enfin, pour ordonner automatiquement le contenu pertinent des documents pour en guider la compréhension, on fait appelle à une stratégie d'**extraction d'information**.

Dans ce chapitre, nous présentons plus en détail les besoins en traitement de document puis les principales stratégies et méthodes de gestion des documents en se focalisant sur les documents électroniques. Dans une troisième partie, nous montrerons comment ces méthodes peuvent être adaptées aux problèmes liés au traitement d'images de documents et en particulier ceux contenant de l'écriture manuscrite. Nous finirons ce chapitre en exposant le cadre de notre étude autour du traitement automatique de courriers entrants manuscrits.

1.2 Les documents

Cette première section a pour but de mettre en avant l'importance des documents dans notre monde et les enjeux autour de leur gestion automatique. Nous commençons donc par définir ce qu'est un document.

Étymologiquement, le mot **document** vient du latin *documentum* composé de la racine *doceo* signifiant montrer, instruire et du suffixe *-umentum* indiquant de manière générale le moyen de faire quelque chose. Ce mot dénote donc de par ces origines un vecteur d'enseignement. Attardons nous sur la définition courante de ce mot : nous présentons ici celles proposées dans l'édition 2006 du *Petit Robert* illustrées par la figure 1.1

DOCUMENT [dɔkymã] n. m. — XII^e^ « enseignement » ; lat. *documentum* « ce qui sert à instruire *(docere)* » ; sens actuel issu de l'emploi jurid. « *Titres et documents* » → docteur* (encadré) **1.** Écrit, servant de preuve ou de renseignement. ⇒ **annales, archives, documentation, dossier, matériaux, papier, pièce.** *Document en un ou plusieurs exemplaires. Original, copie, photocopie d'un document. Conserver le double d'un document. Documents de première main.* ⇒ **information, source.** *Les époques plus récentes «nous ont laissé des milliers de documents contradictoires»* (Paulhan). — *Documents scientifiques. Document graphique, iconographique. Document sur papier, sur microfilm. Documents de travail. Mots-clés d'un document. Accès aux documents. Classer, archiver des documents. Gestion des documents* (⇒ **documentaliste, documentation**). — PAR EXT. *Document sonore.* ◊ CIN., TÉLÉV. *Document d'archives :* images filmées puisées dans des archives. **2.** Ce qui sert de preuve, de témoignage. *Objets saisis comme documents* (cf. Pièce à conviction*). *Enregistrements, films utilisés comme documents. « Portraits, statues, allégories, autographes, médailles, frontispices, tous ces documents parlent aux yeux »* (Henriot). **3.** DR. COMM. Pièce qui permet d'identifier une marchandise en cours de transport (connaissement, police d'assurance, factures). — *Document administratif unique (D. A. U.) :* en douanes, formulaire utilisé dans les échanges des pays de la Communauté européenne. **4.** TECHN. Projet entièrement élaboré d'une page illustrée, d'une affiche de publicité. ⇒ **maquette.**

FIG. 1.1: Ensemble des définitions du mot « document » extraites de [Rey-Debove 2006].

La première définition renseigne sur la nature la plus intuitive d'un document : l'écrit. Un raccourci vers l'écriture manuscrite peut d'ores et déjà être effectué et le support de l'écriture sous sa forme papier également. L'attention est ensuite portée à la dualité de la fonction que peut ou doit remplir un document : son caractère officiel et solennel (*preuve*) et son caractère informatif (*renseignement*). La seconde définition rejoint le caractère bivalent de la première. Cette description en fait une définition plus générale. La dernière

définition apporte une subtilité nouvelle : un document est créé suite à une réflexion quant à sa structuration afin d'en faciliter la compréhension. Dans cette perspective, un document est donc structuré de façon à mettre en avant l'information que son auteur cherche à véhiculer.

Trois composantes fondamentales doivent donc être prises en compte lorsque l'on fait référence à un document :

- son **support**
- l'**information** qu'il contient
- sa **structuration** i.e. comment l'information est organisée sur son support afin de faire passer au mieux l'information qu'il contient

Notons que le sens du mot document est très large et assez vague. Toute photographie, peinture, bande audio etc. est à juste titre un document. Cependant, nous nous limiterons dans ce chapitre à l'étude des documents contenant principalement de l'information sous forme textuelle.

Ces documents tiennent donc deux fonctions principales. D'un point de vue informatif, ils véhiculent des connaissances relatives à un sujet précis, l'accès à l'information qu'ils contiennent est donc essentiel. D'un point de vue administratif et professionnel, les documents sont des vecteurs de preuve et possèdent une certaine valeur juridique. Leur durée de vie est relativement courte et leur traitement doit se faire dans des délais de plus en plus brefs dans un souci évident de productivité et de rentabilité. Dans les deux cas, nous avons à faire à des quantités de documents très importantes impliquant potentiellement divers acteurs pour qui l'automatisation du traitement et de la gestion est nécessaire. Ces masses de documents peuvent être classées en deux grandes catégories :

1. les documents électroniques, produits par ordinateur dans un format électronique connu et dont le contenu est directement interprétable par nos machines. Leur gestion nécessite l'étude des mots et phrases qu'ils contiennent et fait donc intervenir des problématiques de traitement automatique du langage naturel.
2. les images de documents, produits par un humain à la main (document *manuscrit*) en respect de conventions d'écriture d'une langue donnée ou bien à l'aide d'un processus d'imprimerie (document *imprimé*). Ils sont uniquement visionnables à l'aide d'une machine sous la forme d'une image. Leur traitement nécessite une très lourde étape de reconnaissance afin d'en récupérer le contenu.

Les principales masses de documents électroniques sont disponibles sur internet et accessibles au grand public ou bien réservées à un cercle prédéfini d'utilisateurs d'un réseau restreint à une entreprise ou une administration.

La quantité de documents à disposition est alors colossale. À titre d'exemple, *Google* référençait en 2009 plus de 1000 milliards de pages web et autres documents électroniques. Des stratégies d'indexation et de recherche doivent être mises en place pour accéder à ces contenus et les rendre disponibles rapidement aux utilisateurs ayant exprimé un besoin précis en information.

Trois tâches sont identifiables pour la gestion d'une masse de documents :

- rechercher l'information pertinente dans un ensemble de documents à partir d'une demande spécifique d'un utilisateur (sa requête) dans le but d'assouvir ses besoins en information. En sortie du module de traitement, un sous ensemble de la masse de documents correspondant à sa demande lui est proposé : il s'agit de la problématique de **Recherche d'Information**.
- trier les documents par sujet (ou catégorie) par rapport aux informations qu'ils contiennent pour effectuer un classement de ceux-ci : il s'agit d'une problématique de **Catégorisation de document**
- récupérer un ensemble d'informations cibles prédéfinies dans des documents spécifiques afin de les traiter automatiquement : il s'agit de la problématique d'**Extraction d'Information**.

Nous introduisons plus en détail ces trois points dans la section suivante en nous focalisant dans un premier temps sur les documents électroniques.

1.3 Traitement automatique de documents électroniques

Trois tâches liées à la gestion et au traitement automatique de documents ont été identifiées dans la section précédente :

- **la recherche d'information** [Baeza-Yates 1999]
- **la catégorisation de document** [Joachims 1998]
- **l'extraction d'information** [Grishman 1997]

Ces trois tâches nécessitent l'accès au contenu textuel des documents. Dans le cas de documents électroniques, ce contenu est directement accessible et compréhensible par l'ordinateur. Ce n'est plus le cas lorsque l'on veut traiter des images de documents physiques numérisés contenant de l'écriture imprimée ou manuscrite. Dans cette section, nous présentons ces 3 tâches en nous focalisant sur les documents électroniques. Dans la suivante, nous élargirons cette étude au traitement d'images de documents.

1.3.1 Recherche d'information

Aujourd'hui, la capacité à retrouver l'information désirée de manière simple et rapide dans de grandes collections de documents est essentielle. La **recherche d'information** s'attaque aux problématiques liées à l'organisation et à la représentation de l'information contenue dans les documents afin d'y accéder de manière précise et rapide. L'exemple de système de recherche d'information le plus connu et répandu est le moteur de recherche internet : une requête est soumise au moteur qui retourne un ensemble de pages et de documents électroniques ordonnés par pertinence au regard des besoins caractérisés par cette requête.

Historiquement, les documents étaient physiquement stockés dans des librairies et bibliothèques. Un système de classification efficace permettant de retrouver les ouvrages intéressants par rapport à ses besoins personnels a été mis en place à la fin du 19ième siècle : le système décimal de Dewey. Dans cette classification, les œuvres sont réparties en 10 classes, elles-mêmes subdividées en dix divisions et chaque division en dix subdivisions. Par exemple, le code 093 fait référence à *l'informatique, l'information et les ouvrages généraux* pour le 0, aux *manuscrits et livres rares* pour le 9 et aux *incunables*[1] pour le 3. Une fois un ou plusieurs ouvrages présélectionnés grâce à ce système de classement, leur sommaire et leur index permettent de visualiser rapidement la pertinence de la présélection et ainsi faire son choix définitif. L'émergence récente du document électronique a fait augmenter considérablement le nombre et la taille des bases de données dans lesquelles ils sont stockés. La mise en place d'un système de classement similaire à celui de Dewey pour une telle masse de documents est aujourd'hui utopique. De nouvelles méthodes de recherche d'information ont alors vu le jour pour pallier ces limitations de classification : celles-ci se basent principalement sur la notion d'indexation des documents par leur contenu.

Les systèmes de recherche d'information doivent donc mettre en œuvre une méthode permettant d'associer un besoin utilisateur, exprimé sous la forme de mots combinés ou non par des opérateurs logiques, avec un sous ensemble de documents appartenant à une base de données potentiellement gigantesque. Cette approche repose généralement sur l'implémentation des 4 fonctions suivantes selon [Dupont 2011] :

- la représentation des documents
- la requête représentant le besoin informationnel de l'utilisateur
- la comparaison entre le document et la requête

[1] Livres imprimés entre 1450 et 1500 environ

– la présentation des réponses (souvent mis en relation avec la problématique de recherche liée au web)

Ce dernier point consiste à lister les résultats de recherche correspondant aux documents jugés pertinents par rapport à la requête. Diverses informations peuvent être ajoutées en marge des résultats comme le format du document, son auteur, un résumé ou encore les contextes dans lesquels apparaissent les différents termes de la requête. Nous évoquons maintenant les trois premiers points qui sont les plus importants.

Un système de recherche d'information efficace doit se baser sur une bonne stratégie d'indexation des documents afin que ceux étant réellement pertinents par rapport à une requête donnée soient rapidement retournés par le système. Cette requête est la plupart du temps formulée à l'aide d'une liste de mots ou en language naturel. La représentation choisie pour caractériser un document doit être de nature identique tout en étant suffisamment compacte pour respecter des contraintes d'espace de stockage mais aussi de temps de traitement. Dans [Kuropka 2004], l'auteur dresse une taxonomie des modèles de représentation utilisés en recherche d'information. Ceux-ci sont classés suivant deux critères : la base mathématique utilisée et les propriétés des modèles. Les trois représentations les plus largement utilisées sont les suivantes :

– la représentation booléenne standard parce qu'elle est à la base de la plupart des travaux en recherche d'information
– la représentation vectorielle parce qu'elle est aujourd'hui la plus robuste et la plus utilisée
– la représentation probabiliste du langage parce qu'elle est la plus complète et la plus souple pour la modélisation du contenu des documents

Le principe d'une modélisation booléenne est de répertorier les termes les plus importants d'un document dans un index [Baeza-Yates 1999]. Le but est de construire un annuaire inversé qui renvoie une liste des textes contenant les termes de la requête. La théorie des ensembles forme la base mathématique d'un tel modèle : l'index peut être formé par l'ensemble du vocabulaire d'un corpus et un document est entièrement décrit par un « sac de mots » explicitant de manière désordonnée les termes le composant. Un sac de mots est une structure de données utilisée pour la description du langage, introduit dans [Harris 1954]. De même, une requête utilisateur est vue comme une liste non ordonnée de termes. L'évaluation de la pertinence d'un document par rapport à celle-ci est réalisée simplement en prenant l'intersection des ensembles représentant ce document et la requête. Le nombre de termes d'indexation peut être réduit afin de diminuer la dimensionnalité de l'index. Une première méthode efficace consiste à supprimer les mots vides de sens (*stop-words* en

anglais) que sont les articles, prépositions et adverbes. Le *stemming* qui définit l'action de ne conserver que la racine pour un ensemble de mots ayant une base commune peut également être utilisé (*sports*, *sportif*, *sportivement* sont regroupés sous la racine générique *sport*). Enfin, la complexification des requêtes par l'ajout d'opérateurs booléens entre ses termes permet de rendre la recherche d'information plus discriminante.

Bien que simple, ce modèle ne repose que sur la présence ou l'absence des termes d'une requête dans les documents du corpus. Les modèles vectoriels permettent de pondérer les termes de l'index [Dupont 2011]. Ils sont définis par des modèles mathématiques pour représenter de manière compacte un ensemble de documents caractérisés par leur contenu. Cette représentation revient à considérer un espace dont la dimension est égale au nombre de mots pris en compte et à projeter les documents dans cet espace par rapport à leur contenu. Il en est de même pour les requêtes utilisateur. Une mesure de similarité entre un document et une requête peut donc être vue comme une distance entre les deux vecteurs qui les caractérisent. Les mots référencés par ces vecteurs caractérisant l'espace de projection sont pondérés par une contribution locale des mots dans les documents sous la forme de leur fréquence d'apparition *TF* (*Term Frequency*) et par une contribution globale des mots sur le corpus *IDF* (*Inverse Document Frequency*). Cette dernière est associée à une information de diffusion des mots sur l'ensemble des documents du corpus.

Afin de prendre en compte plus précisément le contenu des documents et ne pas se limiter aux mots les composant, des modèles probabilistes du langage forment une troisième alternative très utilisée en recherche d'information [Korfhage 2008]. Ces modèles reposent sur l'idée que la recherche d'information peut être assimilée au calcul de la probabilité de pertinence des documents sachant une requête donnée. La fonction d'ordonnancement des résultats d'une requête, nommée RSV (pour *Retrieval Status Value*), s'appuie sur une évaluation de cette probabilité. L'implémentation la plus classique repose sur la mise en œuvre de modèles stochastiques de langage. D'une manière générale, les modèles probabilistes de la langue sont utilisés pour le calcul d'occurence d'une séquence de mots étant données des mesures statistiques sur une langue en particulier. L'évaluation de la pertinence d'une requête sur un document revient à calculer la probabilité que la requête ait pu être générée par le modèle de langage calculé sur un document donné.

Actuellement, la recherche d'information est une tâche maîtrisée et les recherches se focalisent sur la personnalisation de la recherche par rapport à l'utilisateur dans des bases de données de tailles gigantesques [Dupont 2011]. Pour plus d'informations concernant la recherche d'information, nous proposons les lectures de [Baeza-Yates 1999, Korfhage 2008, Dupont 2011].

1.3.2 Catégorisation de documents

Alors que le domaine de la recherche d'information a pour but de proposer des solutions efficaces de récupération de documents dans un corpus de grande dimension par rapport aux besoins d'un utilisateur matérialisés par sa requête, la catégorisation de document vise à trier les documents d'une collection par thèmes [Joachims 1998]. Ces thèmes ou catégories sont caractérisés par des étiquettes déterminées en amont des traitements de catégorisation par des experts ou de manière automatique. La catégorisation peut être binaire (classement effectué entre une catégorie ou son opposée) ou bien multiple pour un tri plus fin. Les nombreux travaux [Joachims 1998, Sebastiani 2002, Ferilli 2011] et brevets récents autour de la catégorisation de documents illustrent l'intérêt actuel à l'égard de cette problématique.

La catégorisation désigne la tâche d'assigner des catégories à un ensemble de documents d'un corpus à partir de leur contenu. Les mots d'un document sont soumis à un classifieur dont ils forment les entrées. Les sorties correspondent aux différentes catégories souhaitées. Le nombre d'entrée du classifieur étant fixe, des prétraitements visant à sélectionner les termes les plus pertinents d'un document sont utilisés. À l'instar de la recherche d'information, il va s'agir du *stemming*, de la suppression des mots vides de sens ou encore de l'utilisation des pondérations *TF* et *IDF* par exemple [Baeza-Yates 1999]. Notons que ces prétraitements permettent de réduire la dimensionnalité du problème ce qui est indispensable lorsque l'on ne dispose pas d'une base d'apprentissage de très grande taille. Les classifieurs tels que les k-plus proches voisins ou les SVM [Joachims 1998] peuvent être utilisés pour attribuer une catégorie à un document.

L'application la plus commune en catégorisation de documents consiste à détecter les spams reçus quotidiennement dans les boîtes mails. On cherche alors à déterminer si un mail reçu dans sa boîte est désirable ou non. Il s'agit typiquement d'une catégorisation binaire. Plusieurs catégories peuvent être considérées pour effectuer un tri plus fin d'une base de données. Le tri de brèves journalistiques par domaine entre dans ce cadre au même titre que la désambiguïsation du sens de mots dans un document, l'organisation et le tri de documents par catégorie dans une base de données d'entreprise ou encore la gestion de l'information issue d'une masse de documents anciens de type archives [Rath 2002]. Il est à noter que la catégorisation de manière générale ne s'arrête pas aux frontières des documents textuels. La recherche de photographies dans une collection et leur classement par catégorie sont deux exemples illustrant l'étendue de ce domaine.

Notons également que recherche d'information et catégorisation de documents peuvent être utilisées de manière séquentielle pour indexer automatiquement un corpus de documents : les outils de recherche d'information tels que TF ou IDF permettent de sélectionner des termes discriminants servant de catégories pour l'étape d'indexation effectuée par une catégorisation des documents de ce corpus.

Catégorisation de documents et recherche d'information sont des outils puissants de gestion de masses de documents car permettent d'organiser un corpus et d'en faciliter l'accès. L'interprétation des résultats de catégorisation et de recherche est cependant entièrement laissée à l'utilisateur. L'extraction d'information permet de mettre en relation les informations essentielles à la compréhension des documents et constitue donc une stratégie complémentaire bien adaptée à leur gestion automatique. Nous l'introduisons dans la section suivante.

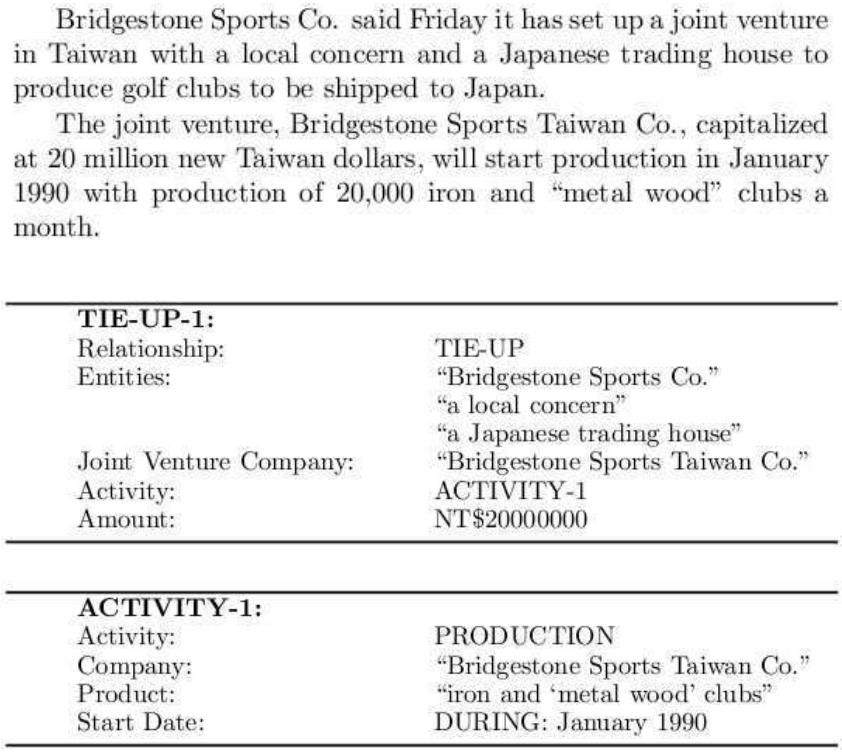
1.3.3 Extraction d'information

Initialement développée au cours des années 70 pour fournir des données financières en temps réel aux traders, l'extraction d'information a pour but de détecter les occurrences de certains types de données cibles dans le contenu textuel d'un document et d'identifier leur rôle [Ferilli 2011, Hobbs 2010, Grishman 1997]. L'objectif est de faciliter la compréhension de documents pour un utilisateur. La plupart des technologies relatives à l'extraction d'information ont répondu à un besoin d'automatisation du traitement de documents caractérisé par la création de conférences exclusivement dédiées à ce domaine : les MUC pour *Message Understanding Conference* [Grishman 1995, Chinchor 1998]. Pour chacune d'entre elles, des bases de données ont été proposées pour étalonner les performances des systèmes sur des tâches spécifiques telles que l'extraction d'information dans des messages relatifs à :

- des opérations navales avec *MUC-1* et *MUC-2*
- des événements terroristes en amérique latine avec *MUC-3* et *MUC-4*
- des fusions d'entreprises dans le domaine de la microélectronique *MUC-5*
- des changements de dirigeants dans les grandes entreprises *MUC-6*
- des rapports de lancement de satellites *MUC-7*

Les champs à extraire des documents peuvent être de nature différente (noms et prénoms de personnes et d'entreprises, adresses de lieux, montants de transactions, dates, etc.) et sont structurés sous la forme d'un modèle mis au point par un expert ou appris automatiquement. Par exemple, il peut s'agir de

parcourir un texte et d'en rechercher l'information essentielle pour répondre à la question : *Qui a fait quoi à qui, quand et où ?* [Hobbs 2010]. La figure 1.2 illustre un document et le modèle contenant les informations extraites. En termes de compréhension, une approche d'extraction d'information se situe entre la recherche d'occurrences de mots dans un texte et l'interprétation complète de celui-ci (de sa syntaxe à sa sémantique en incluant l'étude des intentions de l'auteur).



Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and "metal wood" clubs a month.

TIE-UP-1:	
Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co."
	"a local concern"
	"a Japanese trading house"
Joint Venture Company:	"Bridgestone Sports Taiwan Co."
Activity:	ACTIVITY-1
Amount:	NT$20000000

ACTIVITY-1:	
Activity:	PRODUCTION
Company:	"Bridgestone Sports Taiwan Co."
Product:	"iron and 'metal wood' clubs"
Start Date:	DURING: January 1990

FIG. 1.2: Extraction d'information : la partie supérieure représente un document non structuré, la partie inférieure représente le modèle d'extraction avec les informations extraites. Ces illustrations sont tirées de [Hobbs 2010].

Les stratégies en extraction d'information reposent principalement sur l'élaboration de règles d'extraction. Ces règles intègrent les données cibles à extraire (*targets*) et modélisent la façon dont ces dernières apparaissent dans les documents en utilisant un maximum de connaissances sur le(s) domaine(s) précis considéré(s) sous la forme d'un vocabulaire spécifique et de connaissances linguistiques par exemple. Dans un premier temps, ces règles ont été élaborées *manuellement* par des experts de ces domaines. Les systèmes d'extraction d'information se basent alors sur une cascade d'automates à état

fini imbriqués de manière séquentielle. Les connaissances linguistiques sont prises en compte dans les premiers étages, les règles d'extraction précises dans les derniers. Les mots du document étudié sont d'abord parcourus un à un. La lecture d'un nouveau mot fait transiter l'automate dans un nouvel état. Lorsqu'un état final est atteint, l'automate a détecté une entité qui va constituer l'entrée de l'étage suivant. Finalement, la mise en cascade d'automates permet de détecter des mots puis des phrases et enfin permet l'extraction des données cibles à partir des règles qu'ils caractérisent.

La mise au point manuelle des règles devenant fastidieuse avec l'augmentation de la taille des bases de données, des systèmes à base d'apprentissage ont été imaginés afin d'automatiser la création des règles d'extraction [Hobbs 2010]. Leur apprentissage automatique a rendu les systèmes plus robustes aux variations inhérentes aux langues. Trois types d'apprentissage sont distingués :

- l'apprentissage supervisé de règles syntaxiques pour l'extraction. Par exemple, les systèmes *Autoslog* [Riloff 1993], *PALKA* [Kim 1993] et *LIEF* [Huffman 1996] implémentent cette stratégie mais nécessitent une grande quantité de textes annotés. L'intervention d'un opérateur est souvent indispensable en début de phase d'apprentissage pour créer un premier jeu de règles d'extraction et lors de la phase de décision pour valider les règles d'extraction [Kim 1993, Huffman 1996] et les données à extraire [Riloff 1993].
- l'apprentissage supervisé de classifieurs séquentiels. À cet effet, des Modèles de Markov cachés [Gu 2006], Champs de Markov Aléatoires [Choi 2005] ou machines à vecteurs de support [Finn 2004] sont utilisés. Procédant généralement en deux passes, les modèles appris permettent de localiser les portions de documents potentiellement pertinentes dans un premier temps puis d'extraire réellement les données cibles dans un second temps [Gu 2006, Finn 2004]. L'automatisation et la probabilisation des règles d'extraction permettent de réduire considérablement l'intervention d'un opérateur humain.
- l'apprentissage non ou faiblement supervisé comme par exemple *AutoSlog-TS* [Riloff 1996, Patwardhan 2007] l'extension d'*Autoslog* [Riloff 1993]. L'apprentissage se fait également en deux passes : la première travaille sur un ensemble pré-annoté de documents et génère un nombre important de règles d'extraction candidates. Dans une seconde passe, la pertinence des règles candidates est évaluée en se basant sur la force de leur association avec les documents pertinents associés. Seules les meilleures règles sont conservées.

La conception de systèmes d'extraction d'information dépend grandement de la nature des documents à traiter mais également de l'environnement dans lequel se trouve l'information à extraire. Dans le cas de documents électroniques, une distinction est faite entre documents structurés, semi-structurés et non structurés [Ferilli 2011].

Les documents structurés sont entièrement définis par leur format et les champs à extraire peuvent l'être par simple analyse de la structure de ces documents. Par exemple, un document au format *xml* est interprété de manière très simple en associant les champs cibles à extraire aux balises présentes dans ce document. Dans l'exemple de la figure 1.3 ci-dessous, récupérer les nom et prénom de l'auteur se fait par détection de la balise *AUTHOR* dans un premier temps puis des balises *FIRSTNAME* et *LASTNAME*. La structure est connue préalablement aux traitements ce qui rend le document facilement interprétable automatiquement.

```
<EXAMPLE>
    <AUTHOR>
        <FIRSTNAME>Simon</FIRSTNAME>
        <LASTNAME>Thomas</LASTNAME>
    </AUTHOR>
    <TITLE>Exemple de fichier XML</TITLE>
    <DATEPUB>2012</DATEPUB>
</EXAMPLE>
```

FIG. 1.3: Illustration d'un document structuré de type xml.

Les documents semi-structurés (cf. figure 1.4) sont eux écrits en langage naturel mais possèdent une structure permettant leur interprétation et leur compréhension. Deux niveaux d'analyse sont alors considérés : des caractéristiques de position sont utilisées pour récupérer la structure d'un document puis une analyse syntaxique est effectuée pour récupérer les informations cibles à extraire.

La compréhension de documents non structurés est quant à elle entièrement dépendante de l'analyse linguistique du texte qu'ils contiennent et repose sur le traitement du langage naturel. Un exemple de document non structuré est illustré par la figure 1.5.

Actuellement, en termes de performances, [Hobbs 2010] présente les résultats d'extraction de quelques-uns des systèmes ayant pris part aux compétitions autour des conférences *MUC* (par exemple [Chinchor 1998]). Ceux-ci at-

Name: Dr. Jeffrey D. Hermes
Affiliation: Department of AutoImmune Diseases
Research & Biophysical Chemistry Merck Research Laboratories
Title: "MHC Class II: A Target for Specific Immunomodulation of the Immune Response"
Host/e-mail: Robert Murphy, murph@a.crf.cmu.edu
Date: Wednesday, May 3, 1995
Time: 3:30 p.m.
Place: Mellon Institute Conference Room
Sponsor: MERCK RESEARCH LABORATORIES

FIG. 1.4: Illustration d'un document semi-structuré correspondant à une annonce de séminaire extraite de [Chinchor 1998].

Professor John Skvoretz, U. of South Carolina, Columbia, will present a seminar entitled "Embedded Commitment," on Thursday, May 4th from 4-5:30 in PH 223D.

FIG. 1.5: Illustration d'un document non structuré correspondant à une annonce de séminaire extraite de [Chinchor 1998].

teignent près de 90% de bonnes extractions au niveau champ. Bien que ces résultats soient très corrects, les auteurs constatent que lorsque ces performances sont calculées sur des documents complets (il est commun de considérer un document comme correctement traité lorsqu'au moins quatre champs ont été bien extraits [Hobbs 2010]), celles-ci chutent à un palier correspondant à un *break-even point*[2] de 60%. Cette barrière est aujourd'hui un cap important potentiellement franchissable en incorporant des informations et connaissances plus globales dépassant celles des domaines traités [Hobbs 2010].

1.3.4 Synthèse sur le traitement automatique de documents électroniques

Les approches de recherche d'information, de catégorisation de documents et d'extraction d'information que nous venons de présenter permettent de traiter des documents électroniques à partir de leur contenu. Elles contribuent notamment au fonctionnement efficace des outils numériques de traitement de documents et de communication lié au web. Avec l'amélioration constante des machines de calculs, il a été envisagé, avec succès pour des applications

[2]Point de fonctionnement d'un système tel que rappel = précision

précises et très contraintes, de gérer de manière similaire des masses d'images de documents. Dans la section suivante, nous les présentons avant d'introduire le sujet de notre étude : la détection d'information dans des documents manuscrits non contraints.

1.4 Traitement automatique d'images de documents

Comme nous l'avons vu avec les documents électroniques, le contenu textuel d'un document est indispensable à son interprétation et donc à son traitement. Contrairement aux documents électroniques, l'accès au contenu de documents n'est pas immédiat lorsqu'ils sont obtenus à la suite d'une étape de numérisation. Cette étape engendre des dégradations et implique la mise en œuvre d'un moteur de reconnaissance. L'étape de reconnaissance est d'autant plus difficile que la numérisation est mauvaise, la taille du lexique est grande et la structure du document peu contrainte. Comme l'illustrent les images de la figure 1.6, le contenu d'images de documents est potentiellement très variable et peut contenir à la fois de l'information textuelle et de l'information graphique (images, enluminures, tampons, etc.). Dans le cas de documents imprimés, les polices utilisées pour un même document peuvent être nombreuses et contribuent à la difficulté de la tâche de lecture automatique. Dans le cas de documents manuscrits, les styles d'écriture sont également extrêmement variables.

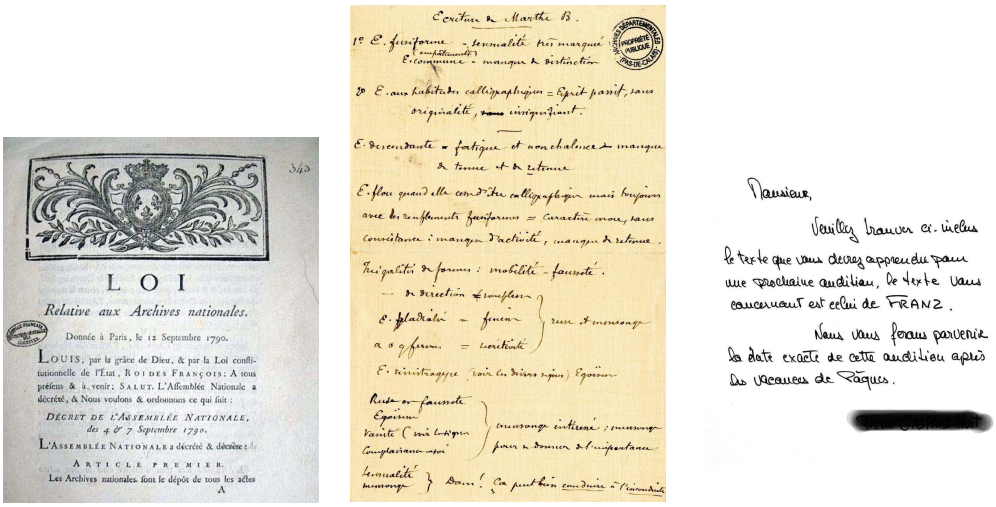


FIG. 1.6: Exemples d'images de documents numérisés. De gauche à droite : un document imprimé, un manuscrit ancien, un courrier manuscrit plus récent.

Pour les documents manuscrits, les seules applications industrialisées concernent pour l'heure uniquement des documents dont le contenu est attendu (vocabulaire connu ou restreint) et dont la structure est fortement contrainte. Ces applications répondent à un besoin réel d'automatisation afin d'éviter un travail fastidieux de lecture visuelle et de saisie manuelle par un opérateur. C'est le cas des applications historiques de la reconnaissance de l'écriture manuscrite que sont :

- la **lecture automatique de formulaires**
- la **lecture d'adresses postales**
- la **lecture de chèques bancaires**

La localisation des données est alors guidée par des connaissances *a priori* sur leur structure et leur vocabulaire. Nous les présentons dans un premier temps avant d'introduire les difficultés relatives au traitement d'images de documents manuscrits non contraints puis notre sujet d'étude.

1.4.1 Extraction d'information dans des formulaires

Un formulaire est typiquement un document dont la structure est très stable. Si un opérateur traite un tel document, il doit lire des champs prédéfinis dont les positions sont connues et toujours identiques. Les documents étant très contraints dans leur structure, l'extraction automatique des informations qu'ils contiennent est possible et guidée par un modèle prédéfini. Si l'on dispose d'un modèle qui renferme les connaissances *a priori* sur les champs (nom, type et position) alors leur extraction est facilitée [Niyogi 1996, Kavallieratos 1997]. La figure 1.7 illustre deux exemples de formulaires.

Si la position des champs à extraire est connue à l'intérieur du document, une étape de recalage de l'image de document par rapport à son modèle est nécessaire. Pour cela, des ancres de recalage sont définies pendant l'élaboration du modèle pour se repositionner précisément sur les différentes zones de lecture des champs. Ces ancres peuvent être représentées par des lignes spécifiques du document, des logos voire des symboles imprimés spécialement pour tenir ce rôle. Généralement, des rateaux permettant de remplir proprement le formulaire constituent les ancres les plus robustes. Une fois localisés, les champs sont reconnus. Des techniques d'OCR et d'ICR (*Optical* et *Intelligent Character Recognition*) sont utilisées à cet effet. Lorsque les scores de reconnaissance sont inférieurs à un seuil de confiance, l'intervention d'un opérateur est nécessaire dans un souci de contrôle et de vérification.

Peu de travaux récents sont publiés dans ce domaine précis. Au contraire, nous assistons à un nombre grandissant de dépôts de brevets et par conséquent la création et le développement de systèmes industrialisés. Citons par exemple

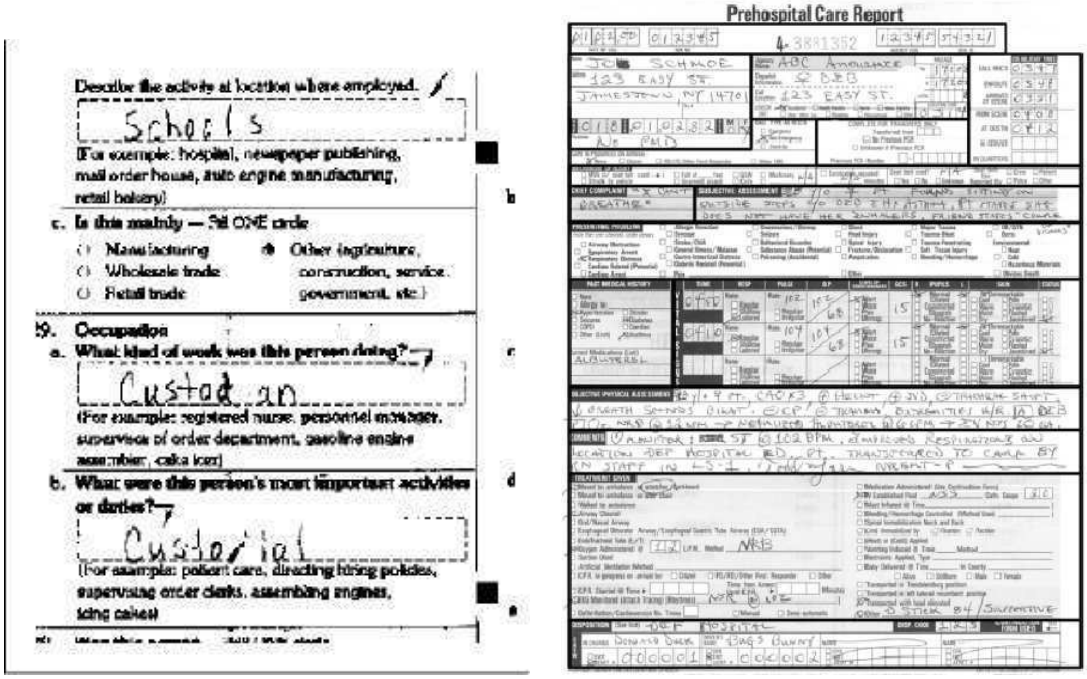


FIG. 1.7: Exemples de formulaires contenant de l'information manuscrite.

les solutions logicielles d'*EMC Captiva* telles que *Dispatcher* et *InputAccel*[3].

1.4.2 Extraction d'information dans des adresses postales

La reconnaissance d'adresses postales est un des domaines historiques de développement de systèmes de reconnaissance de l'écriture manuscrite. Les systèmes de traitement et de gestion de courrier trient aujourd'hui automatiquement des millions de courriers échangés par le biais des services postaux nationaux tels que *La Poste* en France ou l'*US Postal Services* aux États-Unis et permettent d'en assurer le bon acheminement. Par rapport aux formulaires, la lecture automatique d'adresses présente un niveau de difficulté supérieur : la position et la structure des adresses sont assujeties à de plus grandes variations que celles dues à la numérisation. Dans le cas idéal, il va s'agir de reconnaitre les noms et prénoms du destinataire, son adresse complète composée d'un numéro, d'un type et d'un nom de voirie ainsi que d'un code postal précédent le nom de la ville lui correspondant. Les exemples de blocs adresses isolés de la figure 1.8 illustrent ces informations à reconnaitre.

[3] http://www.emc.com

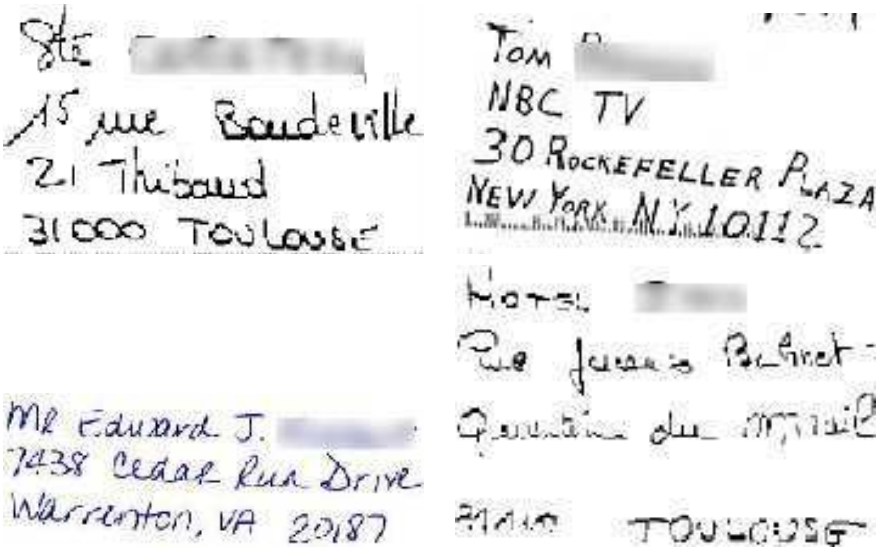


FIG. 1.8: Exemples d'adresses postales extraites sur des enveloppes.

Avant la lecture d'un bloc adresse, sa localisation sur une enveloppe ou un courrier est indispensable. Des méthodes à base de clustering sont le plus souvent choisies [Lii 1995, Pfister 2000] : on cherche à isoler les pixels d'écriture du reste de l'enveloppe. En ce qui concerne la reconnaissance, le bloc adresse est segmenté en lignes afin de décomposer le problème [Srihari 1997]. Le code postal est lu dans un premier temps. Ceci permet de réduire les possibilités quant au nom de ville qui est lu dans un second temps. Enfin, la ligne correspondant à la voirie ainsi que celle contenant les nom et prénom du destinataire sont éventuellement reconnues [El-Yacoubi 2002].

Bien que complexes, les solutions permettant de résoudre la problématique de lecture de bloc adresse existent et sont grandement facilitées par les connaissances *a priori* sur leur structure ainsi que les correspondances existant entre les codes postaux et les noms de villes ou entre les noms de villes et ses noms de rue. Des heuristiques permettent également de faciliter la reconnaissance des différents champs : en France par exemple, un code postal se situe sur la dernière ligne sur la grande majorité des enveloppes et précède le nom de la ville qui lui est associé. Enfin, la ligne de texte se situant juste au dessus de celle-ci renferme les informations concernant la voirie dans un ordre prédéfini : numéro, type puis nom. Ces informations sont intégrées aux modèles d'extraction d'information et permettent de contraindre la reconnaissance [El-Yacoubi 2002].

1.4.3 Extraction d'information dans des chèques bancaires

Autre domaine historique de la reconnaissance de l'écriture manuscrite, la reconnaissance de chèques bancaires est supportée par des systèmes de plus en plus fiables depuis les années 90. Ce type de document est un des plus échangés dans le monde et en particulier en France. Typiquement, un chèque contient un mélange d'informations manuscrites et d'informations imprimées (informations relatives à la banque et à son client) comme illustré par les exemples de la figure 1.9. La lecture automatique d'un chèque nécessite donc l'extraction des informations manuscrites permettant de le traiter. On cherche à extraire les deux montants : le montant légal (montant en toutes lettres) [Knerr 1997, Suen 1999] et le montant de courtoisie [Lee 1997, Dimauro 1997, Heutte 1997], la date d'émission du chèque [Morita 2003, Xu 2003], la signature du payeur [Suen 1999, Madasu 2003] et éventuellement le destinataire et le lieu d'émission.

FIG. 1.9: Exemples de chèques bancaires.

À l'instar de la lecture automatique d'adresses, les différents champs à reconnaitre doivent être localisés dans un premier temps. Ceci peut être effectué

par des opérations de morphologie mathématique [Ye 1999], par l'utilisation de modèles de chèques [Dimauro 1997] ou encore par la recherche d'ancres de recalage [Gorski 2001]. La structure d'un chèque étant *a priori* connue par le biais d'un modèle, celle-ci est utilisée : un encadré entoure le montant de courtoisie qui est également précédé d'un symbole € alors que de longs traits horizontaux et parallèles soutiennent le montant légal. La zone inférieure droite est principalement réservée à la date d'émission et à la signature du payeur dans le cas de chèques français.

Une fois localisés, les différents champs sont extraits lors d'une phase de reconnaissance. Un maximum de contraintes est utilisé : vocabulaire limité à une vingtaine de mots pour la reconnaissance du montant légal, vérification du montant littéral par validation croisée avec le montant numérique [Gorski 2001, Kim 1997] pour améliorer et fiabiliser la lecture, formatage des dates prédéfinies par des modèles [Morita 2003], etc.

1.4.4 Stratégies d'accès au contenu de documents manuscrits non contraints

Les trois applications que nous venons de présenter ont fait l'objet de nombreux brevets, ont été industrialisées et sont utilisées avec succès par entreprises et administrations. Elles tirent profit de l'ensemble des connaissances *a priori* sur les documents à traiter : structure et modèle contraints et/ou vocabulaire bien défini et restreint. Lorsque l'on se focalise sur des documents manuscrits dits libres comme des courriers, des documents d'archives, des brouillons d'auteurs, etc. (cf. exemples de la figure 1.6), les faibles connaissances *a priori* complexifient grandement la lecture automatique :

- structure inconnue
- vocabulaire inconnu et potentiellement infini
- contexte omniscripteur

L'accès à leur contenu reste toutefois le dénominateur commun pour traiter ces documents.

Dans la littérature, ce problème a été abordé sous des différents angles : en prenant en entrée d'un système des requêtes sous la forme d'images de mots isolés [Adamek 2007, Rath 2003b, Rodríguez-Serrano 2009b] ou de séquences *ASCII* [Choisy 2007, Rodríguez-Serrano 2008, van der Zant 2008] ou bien en se basant sur des vocabulaires de grande taille [Graves 2009, Espana-Boquera 2011].

En ce qui concerne la première catégorie d'approche dite de *Word-Spotting*, les occurrences des requêtes images sont recherchées dans un ensemble de documents à l'aide d'algorithmes de *pattern matching*

[Rath 2002, Rath 2004, Manmatha 1996] et ne se base donc pas sur une étape de reconnaissance. Ces approches sont rapides à mettre en place mais ne sont pas robustes aux variations de l'écriture. Elles sont donc préférentiellement utilisées dans un contexte monoscripteur pour l'indexation et la recherche de documents par exemple [Rath 2002].

Les systèmes dont les entrées sont définies sous la forme d'une ou d'un ensemble de requêtes *ASCII* se base majoritairement sur une étape de reconnaissance. Ils autorisent une plus grande souplesse et robustesse face aux variations de l'écriture. En revanche, ils sont plus lourds à mettre en œuvre car nécessitent une phase d'apprentissage. Deux approches considérant des entrées *ASCII* se distinguent pour accéder au contenu des documents :

La lecture complète vise à reconnaitre les documents entièrement [Marti 2000, Vinciarelli 2004, Favata 1998, Senior 1998, Graves 2009, Espana-Boquera 2011]. Cela nécessite la prise en compte d'un lexique de très grande taille pour couvrir l'ensemble des mots de la langue du document et des connaissances linguistiques telles que des modèles de langage pour contraindre la reconnaissance [Lorette 1999, Aubert 2002]. Ceci implique des temps de lecture longs pour des performances qui décroissent rapidement lorsque la taille du lexique augmente [Koerich 2002b].

Détection de mots clés ou Keyword-Spotting vise à détecter la présence d'un contenu spécifique dans des documents sans chercher à reconnaitre l'information non pertinente [Choisy 2007, Rodriguez 2008, van der Zant 2008]. Cette approche permet des traitements plus rapides mais nécessite de modéliser à la fois l'information pertinente et l'information non pertinente [El-Yacoubi 2002, Koch 2006, Fischer 2010, Frinken 2010]. La modélisation de l'information non pertinente constitue le point clé de ces approches et sera abordé dans le chapitre suivant.

Notre proposition dans cette thèse consiste à développer une approche de détection d'informations de tout type (séquences alphabétiques, numériques et alphanumériques) pour accéder à un contenu précis dans des documents manuscrits non contraints. Nous présentons dans la section suivante le contexte précis de notre étude ainsi que les difficultés auxquelles nous serons confrontés.

1.5 Sujet de notre étude

Le travail de recherche de cette thèse s'inscrit dans le cadre d'un programme de développement pour la réalisation d'un système de lecture auto-

matique de documents hétérogènes sur support papier. L'un des points critiques identifiés dans ce contexte concerne en particulier le traitement des documents manuscrits.

Le but de cette thèse est de concevoir un système d'extraction de séquences alphanumériques dans des documents manuscrits non contraints. Ces séquences alphanumériques se caractérisent par des requêtes *ASCII* dont on souhaite retrouver les séquences dans une image ou un ensemble d'images de documents. Les objectifs d'un tel système intéressent donc l'ensemble des organismes, professionnels comme administratifs, dont le fonctionnement prévoit le traitement journalier du courrier entrant en gros volumes (cf. figure 1.10).

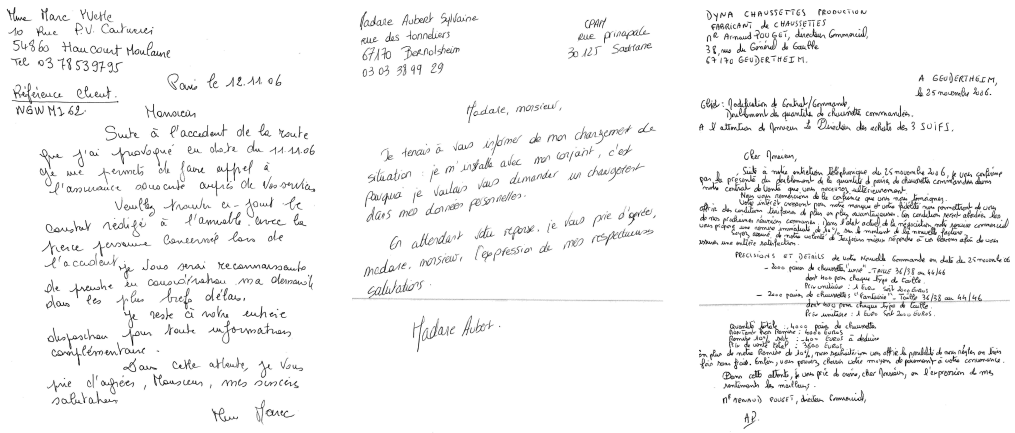


FIG. 1.10: Exemples de courriers entrants manuscrits extraits de la base de données *RIMES* [Grosicki 2009].

À l'heure actuelle, la gestion du courrier entrant dans les entreprises constitue une tâche partiellement automatisée : le courrier est réceptionné et ouvert par un automate prévu à cet effet, puis numérisé à l'aide d'un scanner et archivé pour d'éventuelles consultations ultérieures. Le type de document est ensuite reconnu. Enfin, l'acheminement de versions numérisées des documents vers les départements concernés est effectué par voie électronique. Tout ceci représente bien évidemment un coût, tant du point de vue financier que du point de vue temps de traitement. À titre d'exemple, aux Etats-Unis, devant l'importance de cette problématique, se sont développés depuis plusieurs années des départements d'entreprises et des sociétés, de type service bureau, dédiés à la gestion de ce courrier entrant. Pour certains groupes, le nombre de documents traités dépasse le million par jour. Comme nous l'avons mentionné dans la section précédente, certains

types de courrier (essentiellement les formulaires imprimés) peuvent d'ores et déjà être traités automatiquement si le contenu des champs à traiter est connu. Il reste néanmoins une grande partie du courrier entrant, les courriers manuscrits libres, qui sont traités manuellement. En revanche, il est existe des applications de reconnaissance concernant des sous-problèmes plus réalistes et correspondant à des besoins industriels identifiés. C'est le cas de notre étude qui vise à extraire, à partir de requêtes, l'information pertinente (mots-clés, expressions, entités nommées, numéro de dossier, référence client, numéro de téléphone, ...) dans des documents manuscrits dont le contenu est attendu (courrier entrant). Il s'agit là d'une première étape avant la mise en relation automatique de ces informations pour, par exemple, récupérer l'identité de l'expéditeur, le service destinataire concerné, l'objet du courrier, etc.

La détection de la présence ou non de certains mots clés dans un courrier peut permettre de catégoriser ces documents, d'en déterminer l'objet ou de les indexer, c'est à dire de les classer afin de les retrouver de manière facile et rapide. Sur la figure 1.11, l'extraction des mots *client*, *noter*, *nouvelle* et *adresse* indique au lecteur que le courrier concerne un changement d'adresse. Avec la reconnaissance de l'écriture en elle-même, la principale difficulté consiste à localiser cette information ce qui est loin d'être trivial et immédiat.

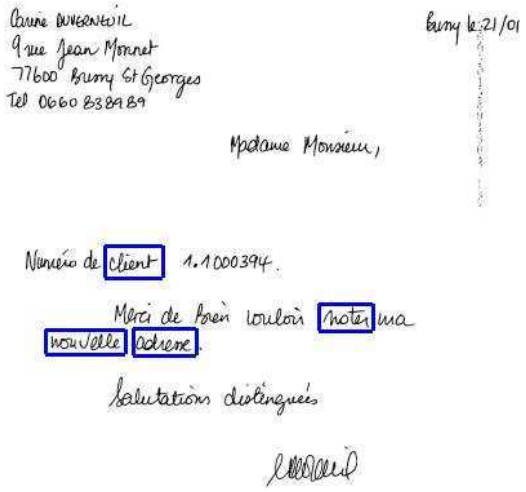


FIG. 1.11: Illustration d'un courrier manuscrit (extrait de la base *RIMES* [Grosicki 2009]) et des mots clés extraits permettant de déterminer son objet.

Les séquences numériques sont également très riches en information. Il peut s'agir comme l'illustre la figure 1.12 :

- du code postal permettant la localisation géographique du client
- du code client permettant son identification
- du numéro de téléphone permettant à la fois de l'identifier et de le contacter

Au même titre que les mots clés discriminants d'un document, les séquences numériques constituent des informations permettant par exemple de déterminer le rédacteur du courrier via ses données personnelles (numéro de téléphone, code client, adresse etc.). D'autres informations contenant des données chiffrées peuvent être extraites. Dates, montants et adresses en sont des exemples. Associées à une base de données contenant les champs décrits ci-dessus, ces séquences numériques permettent d'identifier de manière certaine l'individu ayant écrit le courrier. Contrairement aux problématiques bien définies de lecture automatique de documents manuscrits non contraints, nous sommes confrontés à la difficulté de localiser ces informations sans connaissance *a priori* sur la structure du document.

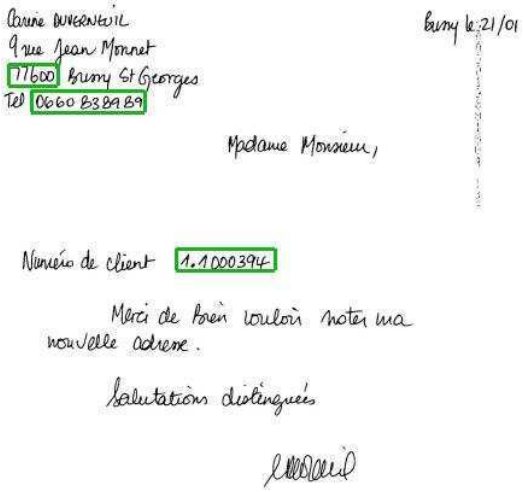


FIG. 1.12: Exemples de séquences numériques à extraire d'un courrier manuscrit (extrait de la base *RIMES* [Grosicki 2009]).

Notre étude entre donc dans le cadre du traitement automatique d'images de documents manuscrits non contraints. Elle se situe dans le prolongement des travaux de Clément Chatelain [Chatelain 2006b] et de Guillaume Koch [Koch 2006]. Dans [Chatelain 2006b], l'auteur a développé une approche visant à extraire des séquences numériques en utilisant leur syntaxe. Il se sert par exemple du fait qu'un numéro de téléphone se compose de 10 chiffres

généralement regroupés par 2, ou encore que les codes postaux possèdent 5 chiffres. Cette approche ne convient pas malgré tout à la détection de mots clés. Dans [Koch 2006], l'auteur a développé un système de détection de mots clés pour la catégorisation de documents sans toutefois s'intéresser aux séquences numériques et alphanumériques.

Notre but est de développer un seul et même système capable de détecter n'importe quelle séquence dans des documents manuscrits non contraints de type courrier entrant : mots clés, séquences numériques ou séquences alphanumériques définies automatiquement ou par un utilisateur sous la forme d'une requête ou d'un lexique. Des exemples de requêtes sont données par le tableau de la figure 1.13.

Requêtes alphabétiques	Requêtes numériques	Requêtes alphanumériques
contrat	26600	DPCES73
résilier	76800	ETVQY17
abonnement	16/12/10	16 décembre 2010
Jérôme Feugier	01-12-83	5 jan 09
Mme Durand	0631557013	76000 Rouen
Saint-Etienne	06-84-12-16-72	8 rue Saint-Gervais
...	...	...

FIG. 1.13: Exemples de requêtes avec lesquelles on souhaite interroger le système sur une base de documents.

Sur la figure 1.14, nous illustrons un exemple d'utilisation du système à concevoir : l'interrogation d'une base de documents par requêtage. Nous souhaitons pouvoir isoler un ensemble de documents pertinents par rapport à une requête ou un ensemble de requêtes. La première requête, un mot clé correspondant à un nom propre, est recherchée dans une base de documents limitée ici à trois courriers entrants manuscrits. L'occurrence du mot correspondant à cette requête est retrouvée dans le second document, celui-ci est donc selectionné pour sa pertinence. Deux autres types de requêtes sont illustrés, à savoir une séquence numérique sous la forme d'un numéro de téléphone et une séquence alphanumérique (une date) et dont des occurrences sont respectivement détectées dans les premier et troisième documents.

Les points clés à aborder dans cette étude concernent donc la localisation et la reconnaissance de l'information pertinente (les requêtes) ainsi que la gestion de l'information non pertinente à rejeter. Lorsque l'on travaille sur

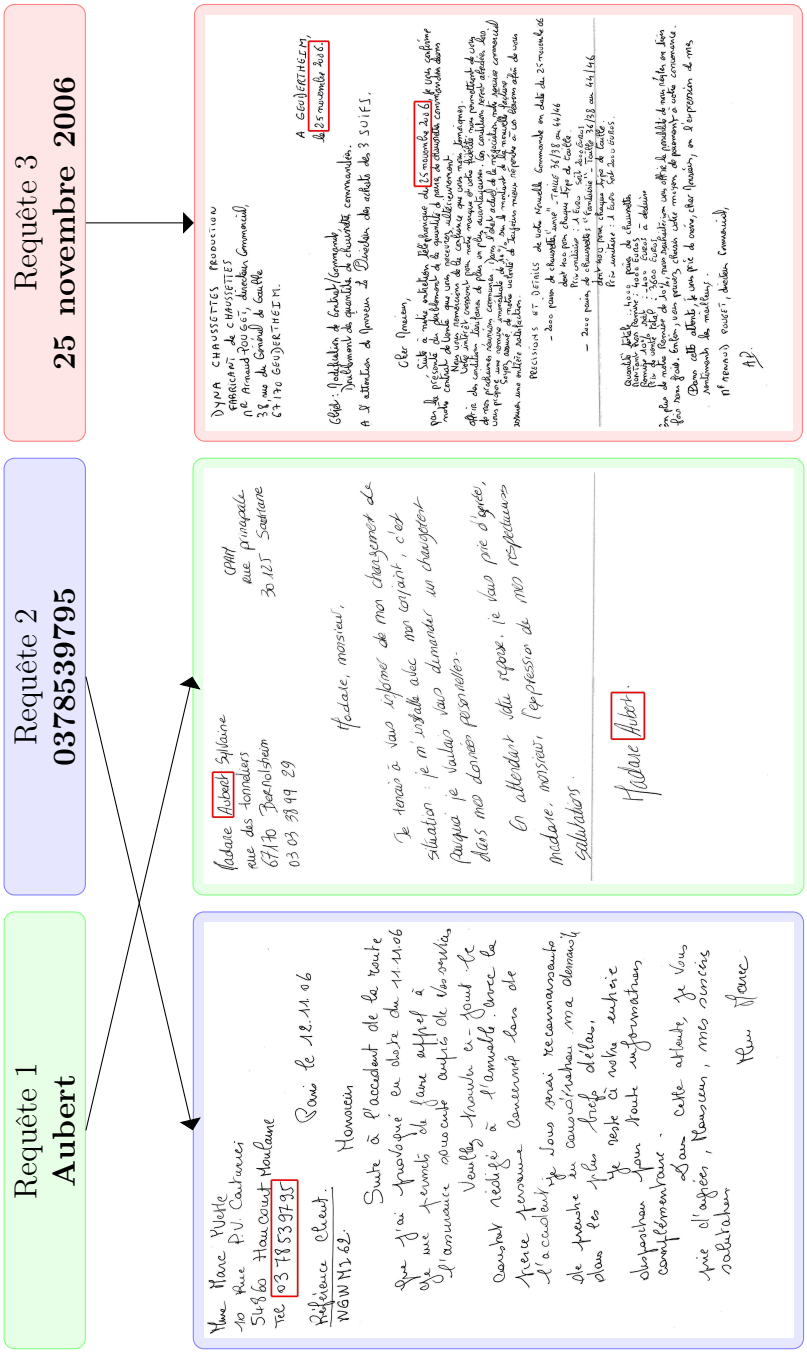


FIG. 1.14: Trois requêtes (alphabétique, numérique et alphanumérique) sont lancées sur une base de 3 documents (extraits de la base de données *RIMES* [Grosicki 2009]). Les documents dans lesquelles l'information pertinente est détectée sont isolés.

des images de documents non structurés, comme c'est le cas par exemple pour les courriers entrants, peu de contraintes et de connaissances *a priori* guident la lecture automatique. Les seules hypothèses que nous pouvons faire *a priori* sur ces documents sont que :

- ils sont écrits dans une langue donnée et connue (en français dans notre cas)
- ils sont organisés en lignes de texte, unique contrainte connue sur la structure du document avant son traitement
- l'orientation de ces lignes de texte est globalement horizontale

Dans ce manuscrit, nous allons donc présenter un système complet de détection d'informations alphabétiques, numériques et alphanumériques dans des documents manuscrits non contraints basé sur une modélisation globale des lignes de texte.

Nous avons donc besoin d'aborder, dans un second chapitre, la problématique de la **modélisation** de l'écriture manuscrite. Nous débuterons par la présentation des trois points clés incontournables à la mise en place d'une stratégie d'accès au contenu de documents manuscrits. Nous introduirons ensuite notre première contribution qui concerne la modélisation générique de ligne de texte pour la détection d'informations quelconques (alphabétique, numérique et alphanumérique) dans des documents manuscrits non contraints, au regard de ces trois points.

Le troisième chapitre présente l'**implémentation** complète d'un système de détection autour de ce modèle de ligne. Nous y présenterons les principaux modules de traitement, en nous référant à l'état de l'art, et présenterons plus en détail l'implémentation du modèle de ligne pour la détection. Nous conduirons une première série d'expérimentations afin d'éprouver notre système et effectuerons une critique objective de ses atouts et de ses points faibles.

Le quatrième chapitre introduira notre deuxième contribution sous la forme d'une piste d'**optimisation** du système et de son moteur de reconnaissance. Nous montrerons que celui-ci peut être pertinemment amélioré par l'ajout d'un pouvoir localement discriminant à la modélisation globale des lignes de texte grâce à des réseaux de neurones profonds. Également, nous montrerons que les réseaux de neurones profonds permettent de s'affranchir de la délicate étape de conception d'un extracteur de caractéristiques, rendant le système plus générique encore.

1.6 Conclusion

La numérisation de documents produits des quantités toujours plus grandes de documents électroniques et d'images de documents. Les besoins

en traitement automatique de ces documents sont aujourd'hui proportionnels aux tailles des masses de documents stockées et échangées. S'il peut être considéré que la gestion des documents électroniques est aujourd'hui bien maitrisée, ce n'est pas le cas des images de documents, notamment ceux qui contiennent de l'écriture manuscrite et qui sont faiblement voire non contraints. Au cours de ce chapitre, nous avons montré l'intérêt d'extraire de l'information (sous la forme de mots clés, de séquences numériques et alphanumériques) pour traiter les documents manuscrits non contraints et en particulier les courriers entrants manuscrits. Notre étude se situe dans ce contexte et nous proposons, pour guider le traitement des images de courriers entrants manuscrits non contraints, d'accéder à une partie spécifique de leur contenu en le détectant parmi de l'information non pertinente à rejeter.

Dans le chapitre suivant, nous allons aborder les points clés qui doivent être pris en compte dans la conception d'un tel système d'extraction d'information dans des documents manuscrits à savoir la modélisation et la reconnaissance de l'écriture manuscrite ainsi que le rejet de l'information non pertinente présente dans ces documents. Nous insisterons sur les choix de modélisation retenus pour contraindre la reconnaissance des informations à extraire et à rejeter et donc se rapprocher de ce qui se fait dans les domaines maitrisés de la lecture de formulaires, de chèques bancaires et d'adresses postales.

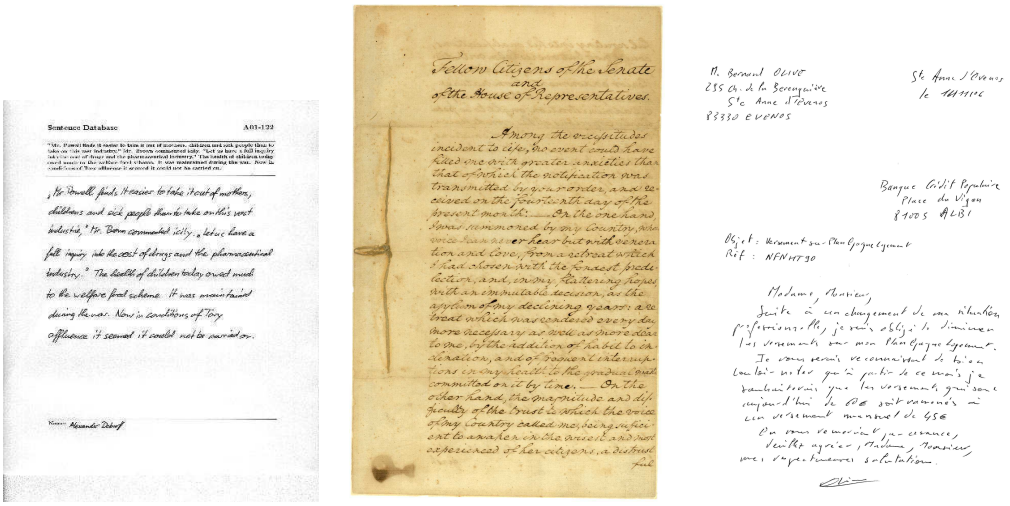
Chapitre 2

Modélisation de l'écriture et stratégies d'accès au contenu de documents manuscrits

Sommaire

2.1 Introduction

Au cours du chapitre précédent, nous avons mis en avant la nécessite de traiter automatiquement des masses de documents et l'importance de la modélisation pour accéder à l'information pertinente contenue dans leur document de manière générale. En ce qui concerne les méthodes de gestion des documents électroniques, c'est à dire la recherche d'information, la catégorisation de documents et l'extraction d'information, des modèles conçus par des experts ou appris automatiquement permettent de mettre en relation des données textuelles de ces documents directement accessibles et interprétables par nos machines. Dans le cas d'images de documents, des connaissances *a priori* sur leur structure, lorsqu'elles sont suffisamment contraintes, permettent de définir des modèles de lecture dans le but de localiser les régions d'intérêt puis d'en extraire les champs clés.



(a) (b) (c)

FIG. 2.1: Exemples de documents manuscrits : (a) un texte de la base *IAM sentence* [Marti 2002], (b) un manuscrit ancien de la base *Washington* [Rath 2002], (c) un courrier de la base *RIMES* [Grosicki 2009].

Dans notre sujet d'étude, les documents que nous avons à traiter sont manuscrits et non contraints par une structure donnée si ce n'est le fait qu'ils sont organisés en lignes de texte (comme l'illustre la figure 2.1). Le traitement de ces documents nécessite la mise en place d'une stratégie d'accès au contenu de ces lignes. Nous avons identifié trois points clés afin de mettre en place une telle stratégie :

– la segmentation des lignes du document en mots (et séquences numé-

riques lorsqu'il y en a). Elle est indispensable pour pouvoir intégrer des connaissances linguistiques de type lexique et modèles de langage lorsque cela est possible [Lorette 1999, Plamondon 2000, Aubert 2002].
- la reconnaissance de ces entités isolées, étape incontournable pour faire face à la variabilité de l'écriture manuscrite, notamment dans notre contexte omniscripteur.
- le rejet de l'information non pertinente. Cette étape permet de fiabiliser la reconnaissance ou de rejeter l'information non souhaitée.

Ce chapitre a pour but de présenter ces trois notions essentielles et d'introduire les solutions retenues dans notre cadre d'extraction d'information (cf. section 1.5). Nous commencons par présenter les stratégies de segmentation de lignes de texte en sous entités.

2.2 Stratégies de segmentation des lignes en mots

La segmentation d'une ligne est l'action d'isoler les entités qu'elle contient afin de se ramener à une problématique de reconnaissance de séquences de caractères isolées (mots ou séquences numériques). Il s'agit d'un problème de classification à deux classes : il faut déterminer si les blancs entre composantes connexes successives d'une ligne suivant l'axe horizontal sont des espaces inter-mot ou des espaces intra-mot (cf. figure 2.2). Cette opération de segmentation peut être réalisée de deux manières distinctes et antagonistes : explicitement [Louloudis 2009, Nikolaou 2010], on cherche à découper définitivement la ligne en mots, ou implicitement, en intégrant le processus de segmentation à la phase de reconnaissance qui ne retient alors que les hypothèses de segmentation les plus probables.

2.2.1 Segmentation explicite des lignes en mots

Les méthodes permettant de réaliser cette tâche sont nombreuses : [Louloudis 2009, Nikolaou 2010, Khurshid 2008, Lüthy 2007, Marti 2001a]. Les principales stratégies de segmentation explicite de lignes en mots se basent notamment sur :
- un seuillage sur les valeurs des histogrammes de projection verticale des pixels d'écriture [Louloudis 2009] ou sur les distances entre masses connexes [Marti 2001a]
- l'utilisation d'un classifieur pour discriminer les espaces inter-mot des espaces intra-mot [Nikolaou 2010, Lüthy 2007]

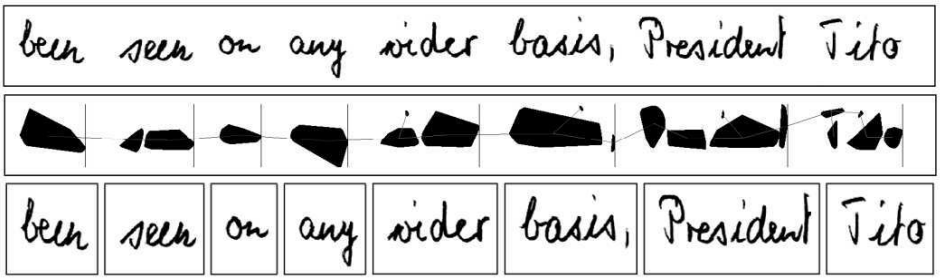


FIG. 2.2: Illustration de la problématique de segmentation d'une ligne manuscrite en mots extraite de [Marti 2001a]. Les distances entre masses connexes sont utilisées à cet effet.

Dans [Louloudis 2009], les auteurs filtrent dans un premier temps les distances euclidiennes entre les composantes connexes de manière à ne considérer que celles dont les projections verticales ne se superposent pas. Un seuil global pour une image est ensuite déterminé à l'aide d'heuristiques et permet de prendre une décision quant à la nature des espaces considérés aboutissant à la séparation des mots entre eux. Dans [Nikolaou 2010, Khurshid 2008], les auteurs utilisent une méthode de discrimination d'espaces basée sur un Run-Length. Celui-ci permet à la fois d'extraire les lignes d'une image de document mais aussi les mots d'une ligne. Une méthode basée sur les modèles de Markov cachés ou MMC a été présentée dans [Lüthy 2007]. Les auteurs se ramènent à un problème de discrimination d'espaces inter ou intra mots en utilisant une fenêtre glissante le long d'une ligne de texte depuis laquelle des densités de pixels sont extraites. Les MMC permettent de labeliser les trames de la ligne comme appartenant à un mot (caractère ou espace au sein du mot) ou à un espace entre deux mots. De la même manière, des CRF peuvent également être utilisés [Shetty 2007].

2.2.2 Segmentation implicite des lignes en mots

Contrairement à une segmentation explicite où l'on cherche à déterminer les frontières entre les mots de manière définitive, la segmentation implicite d'une ligne est une approche globale. En effet, le processus de détection d'espace inter-mots est couplé à la reconnaissance des mots contenus dans cette ligne [Marti 2000, Marukatat 2001, Vinciarelli 2004, Graves 2009]. Pour guider la reconnaissance et donc conjointement la segmentation de la ligne

en mots, des modèles de ligne sont utilisés [Marti 2001b, Vinciarelli 2004]. Ces modèles de lignes intègrent des connaissances linguistiques dans leurs transitions sous la forme de modèles de langage définissant les enchainements de mots les plus probables. Ceux-ci se caractérisent par des N-grammes ou des N-classes de mots [Quiniou 2007]. Le schéma de la figure 2.3 illustre le modèle de ligne utilisé dans [Marti 2001b] et dont les transitions sont définies par un modèle de langage de type bigramme de mots. Nous décrirons en détail cette approche dans la section suivante puisqu'il s'agit d'une extension de la reconnaissance de séquences isolées basée sur une segmentation implicite de ces séquences en caractères.

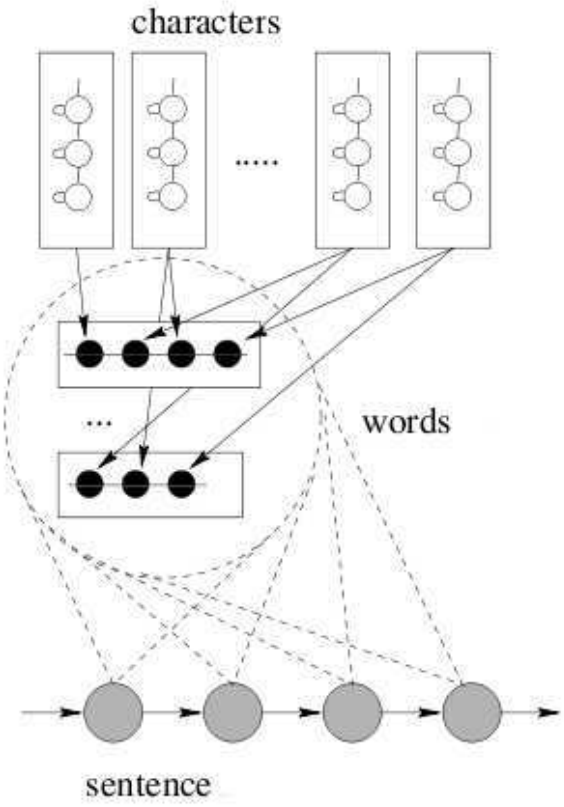


FIG. 2.3: Modélisation classique d'une ligne de texte par des modèles de mots constitués de la concaténation de leurs modèles de caractère. Illustration extraite de [Marti 2001b].

Il faut tout de même noter que la complexité combinatoire de ce type d'approche est importante. Celle-ci est notamment à l'utilisation d'un modèle de ligne très complexe pour être représentatif d'une langue. Si l'on considère

un lexique de 60000 mots[1] est utilisé en reconnaissance, un modèle de langage définissant $3,6.10^9$ transitions doit être construit.

Malgré cela, cette approche présente l'avantage de ne pas chercher à prendre de décisions définitives quant aux frontières entre les mots d'une ligne. On s'affranchit ainsi du problème de propagation des erreurs de mauvaise segmentation en mots le long de la chaine de reconnaissance et la détection des espaces est fiabilisée à l'aide de la reconnaissance. Pour ces dernières raisons, nous avons choisi de nous orienter vers une segmentation implicite des lignes en mots et donc de définir un modèle de ligne intégrant des connaissances linguistiques pour contraindre la reconnaissance et par là même guider la segmentation en mots.

Évoquées en fin de premier chapitre, les stratégies d'accès au contenu de documents manuscrits sont étroitements liées à la méthode de segmentation choisie. C'est également le cas des stratégies de reconnaissance de séquences manuscrites que nous présentons dans la section suivante.

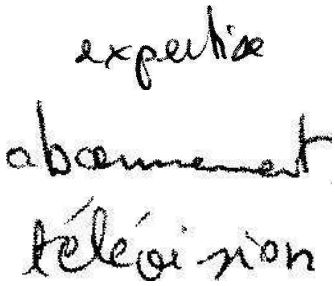
2.3 Stratégies pour la reconnaissance de séquences

La reconnaissance de séquences de caractères manuscrits est un des domaines historiques de la reconnaissance de l'écriture manuscrite hors ligne. En effet, il s'agit des problématiques abordées lorsqu'il a été envisagé d'automatiser la lecture de chèques [Knerr 1997, Heutte 1997], d'adresses [El-Yacoubi 2002, Pfister 2000] ou de formulaires manuscrits [Niyogi 1996, Kavallieratos 1997]. Une fois isolées, une étape de reconnaissance des séquences est à considérer. Cette étape se confronte au paradigme de segmentation - reconnaissance énoncé dans [Sayre 1973]. Celui-ci stipule que les caractères composant une séquence ne peuvent être reconnus sans les avoir au préalable localisés mais pour les localiser, il faut dans un premier temps les avoir reconnus. La modélisation de ces séquences manuscrites qui doivent être reconnues est donc fondamentale.

Pour illustrer toute la difficulté de la tâche de reconnaissance de séquences, les figures 2.4 et 2.5 présentent respectivement des images de mots et des séquences numériques manuscrites déjà isolées. Les formes des caractères appartenant aux séquences dépendent fortement de leur contexte et des possibles

[1] 60000 comme le nombre de mots dans le petit Robert 2006 [Rey-Debove 2006]

contacts avec leurs voisins directs.

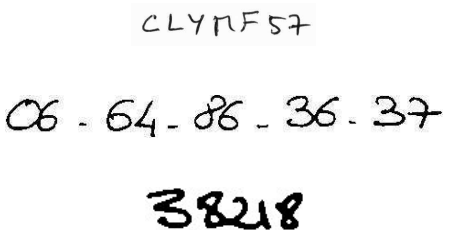


expertise

abonnement

télévision

FIG. 2.4: Exemples d'images de mots manuscrits isolés et leur étiquette extraites de la base de données RIMES [Grosicki 2009].



référence client

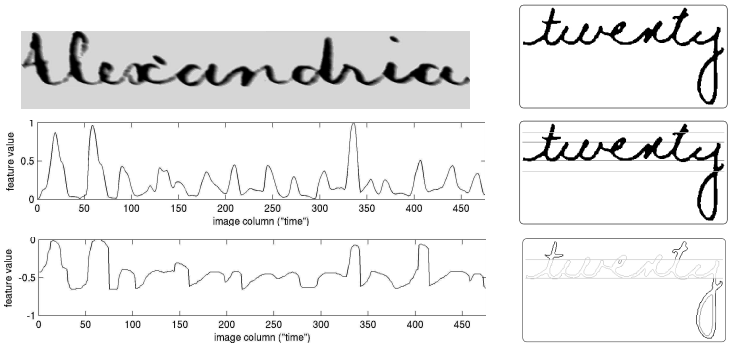
numéro de téléphone

code postal

FIG. 2.5: Exemples d'images de séquences numériques et alphanumériques isolées et leur type extraites de la base de données RIMES [Grosicki 2009].

La problématique de la modélisation de séquences de caractères peut être abordée de manière globale si la taille du lexique associé à la reconnaissance est petite. On parle alors d'approche holistique. Cette modélisation a par exemple été mise en œuvre avec succès dans les problématiques de reconnaissance de montant légal sur les chèques bancaires [Knerr 1997, Suen 1999]. On cherche alors à représenter les images de mots à reconnaitre par des caractéristiques globales extraites depuis l'empreinte de ces mots par exemple (cf. figure 2.6). Cette approche est envisageable lorsque le lexique est connu et de taille limitée [Gorski 2001, Lee 1997, Govindaraju 1996, Knerr 1997, Weliwitage 2003, Rath 2004] mais ne convient plus lorsque l'on considère des séquences numériques dont le nombre de chiffres est *a priori* inconnu (l'utilisation d'un lexique pour contraindre la reconnaissance est impossible) ou lorsque le problème de reconnaissance est contraint par un lexique de grande taille. Afin de traiter des problèmes où les lexiques sont plus importants, une segmentation des séquences en sous entités (lettres, chiffres, graphèmes) est nécessaire.

Dans le cas de mots isolés (cf. la figure 2.4), deux méthodes de segmentation peuvent être considérées : une approche explicite au cours de laquelle des



(a) (b)

FIG. 2.6: Exemples de caractéristiques globales sur des images de mots. De haut en bas : (a) image d'origine, projection horizontale des pixels d'écriture et profil supérieur [Lavrenko 2004], (b) image d'origine, détection des lignes de base, position et taille des hampes et jambages [Guillevic 1994].

hypothèses de frontières entre les sous entités sont soumises à un classifieur [Sadri 2007, Koch 2004], ou une approche implicite au cours de laquelle tous les points de la forme de la séquence de caractères sont considérés comme des points de segmentation potentiels [Vinciarelli 2002, Koerich 2003]. Dans les deux approches, la segmentation est guidée par la reconnaissance des entités segmentées, la reconnaissance étant contrainte par l'utilisation d'un lexique prédéfini couvrant tous les mots observables.

Dans le cas de séquences numériques, aucun lexique n'est disponible pour contraindre la reconnaissance : des classifieurs discriminants sont généralement utilisés pour reconnaitre les chiffres qui doivent dans un premier temps être isolés de la séquence [Oliveira 2002]. La segmentation peut être dirigée ou non par la reconnaissance, on parle d'approches de segmentation / reconnaissance ou de segmentation puis reconnaissance.

Dans cette section, nous présentons les approches de reconnaissance de séquences de caractères manuscrites isolées basées sur une étape de segmentation. Nous évoquerons les modélisations :

- à base de segmentation explicite
- à base de segmentation implicite

Les méthodes holistiques (sans segmentation en caractères) ne seront pas détaillées. Nous finirons cette section en présentant les méthodes de modélisation de séquences de mots permettant d'apporter un complément d'informations pour la reconnaissance de phrases sous la forme de connaissances linguistiques.

2.3.1 Reconnaissance à base de segmentation explicite

La reconnaissance d'images de séquences isolées à base de segmentation explicite vise à retrouver les frontières entre les caractères d'un mot ou les chiffres d'une séquence numérique afin de se ramener à une problématique de reconnaissance de sous entités dont le nombre est restreint par un alphabet de graphèmes, de lettres ou de chiffres. Bien que les principes de segmentation concernant ces deux types de séquences soient les mêmes, les méthodes peuvent différer : alors que dans une image de séquence numérique manuscrite, les chiffres peuvent être joints (cf. figure 2.7) mais sont le plus souvent dissociés, une image de mot, en particulier lorsque l'écriture est cursive, ne possède pas nécessairement de frontières franches entre ses caractères. Ceci est illustré par la figure 2.8.

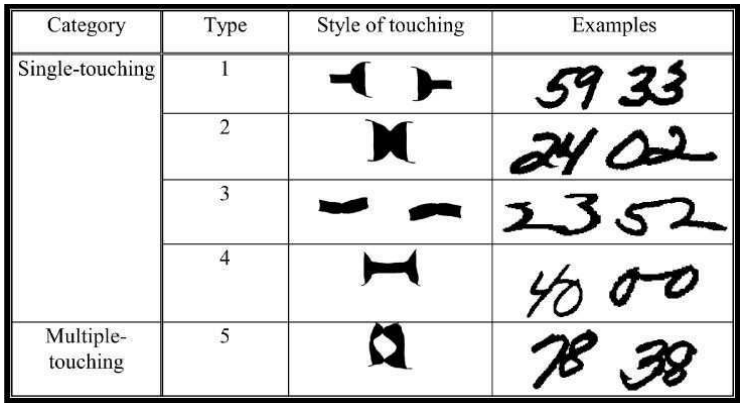


FIG. 2.7: Différents types de contacts entre deux chiffres liés selon [Chen 2000].

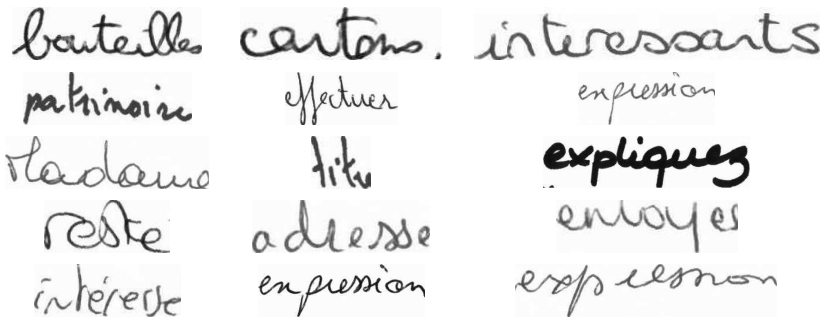


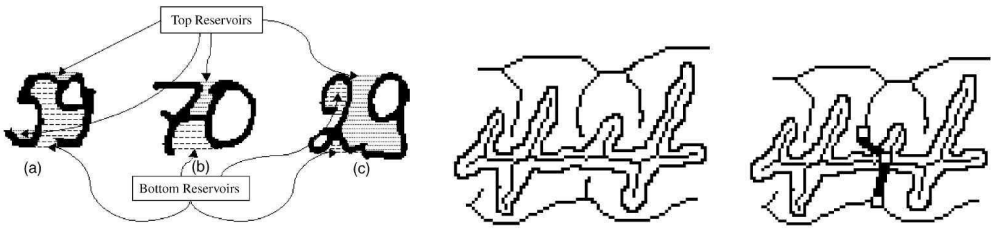
FIG. 2.8: Illustrations de ligatures entre caractères dans des mots manuscrits extraits de la base RIMES mots [Grosicki 2009].

Deux types de reconnaissance à base de segmentation explicite peuvent

donc être considérés :

- une approche séquentielle au cours de laquelle l'étape de segmentation cherche à isoler les caractères de manière définitive. Cette étape précède la phase de reconnaissance des caractères ainsi isolés.
- une approche au cours de laquelle des hypothèses de segmentation sont produites dans un premier temps puis validées à l'aide d'une étape de reconnaissance.

Bien qu'employées pour la reconnaissance de séquences numériques (cf. figure 2.9), les prises de décision sans vérification par une étape de reconnaissance font de la première approche une stratégie caduque.



(a) (b)

FIG. 2.9: Exemples de stratégies de segmentation de chiffres : (a) à l'aide du concept de *water reservoir* introduit par Pal dans [Pal 2003], (b) à l'aide de l'étude du squelette de l'écriture et de l'arrière plan [Chen 2000].

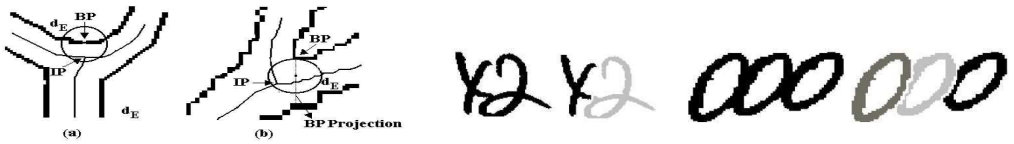
A contrario, une approche de segmentation/reconnaissance évalue la vraisemblance des hypothèses de segmentation par le biais d'un module de reconnaissance. Les scores de reconnaissance produits par ce module sont utilisés pour valider les meilleures hypothèses de segmentation tout en rejetant les moins probables. Cette propriété fait de la segmentation/reconnaissance une stratégie globalement plus robuste par rapport à la variabilité des formes que celle de segmentation puis reconnaissance. Nous la présentons dans ce qui suit.

2.3.1.1 Segmentation des séquences

Le but de l'étape de segmentation est de produire des hypothèses de frontières entre les caractères composant une séquence. La segmentation de deux caractères disjoints est basée sur la détection des zones de pixels de fond entre eux. Mais la détection des caractères joints et surtout leur segmentation présentent plus de difficultés.

En ce qui concerne la détection de chiffres collés, l'étude des boîtes englobantes peut être effectuée : les proportions des chiffres composant une séquence numérique sont souvent très semblables [Chatelain 2006b]. Pour ce

qui est de la segmentation, les hypothèses sont déterminées à l'aide des caractéristiques extraites depuis l'image de la séquence. Ces caractéristiques peuvent être extraites depuis les pixels d'écriture [Oliveira 2002], les pixels du fond [Casey 1996] ou bien les deux [Sadri 2007]. Le squelette des composantes connexes d'écriture et l'étude des jonctions multiples sont également utilisés [Elnagar 2003, You 2003, Chen 2000]. Des méthodes plus originales telles que [Congedo 1995] et [Pal 2003] reposent sur deux concepts relatifs à l'écoulement d'un fluide pour déterminer les points ou chemins de segmentation entre deux chiffres collés.



(a) (b)

FIG. 2.10: Exemple de segmentation de chiffres joints dans [Oliveira 2002] : (a) des caractéristiques sur les points de jonctions du squelette de l'écriture sont recherchés, (b) résultats de segmentation/reconnaissance.

En ce qui concerne plus spécifiquement les images de mots manuscrits, les styles d'écriture peuvent conduire à des lettres disjointes (écriture scripte), liées (écriture cursive) ou dans le cas le plus complexe, un mélange des deux styles (cf. exemples de la figure 2.8). Dans le cas d'une écriture purement cursive, retrouver les frontières entre les lettres est loin d'être trivial puisqu'il n'existe pas de segmentation parfaite : pour une même image d'un mot manuscrit, deux individus pourraient donner des frontières entre caractères différentes. Ceci impose la conception d'un module de segmentation robuste face à ces variations. Il doit être capable de produire un ensemble de points de segmentation potentiels suffisamment grand pour contenir tous les points de segmentation corrects mais également suffisamment réduit pour limiter les temps de calculs et les possibilités de combinaison (cf. figure 2.11). À cet effet, des points caractéristiques des contours inférieur et supérieur, des profils de ces contours voire directement un histogramme de projection verticale des pixels d'écriture sont utilisés [Koerich 2002a, El-Yacoubi 1999] pour déterminer cet ensemble de points de segmentation. Ces points caractéristiques correspondent généralement aux minima et maxima locaux [Koch 2004], ou peuvent reposer sur la détection de ligatures entre caractères [Shetty 2007]. L'étude des pixels de fond au voisinage des pixels d'écriture peut également permettre de déterminer un ensemble de points de segmentation potentiels [Casey 1996]. Afin de limiter la combinatoire, des heuristiques peuvent être utilisées pour limiter le

nombre de points de segmentation potentiels [El-Yacoubi 1999]. L'estimation de la largeur moyenne d'un caractère en est un exemple [Koch 2006].

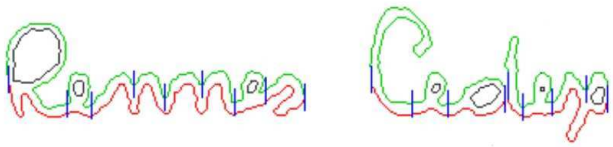


FIG. 2.11: Illustration d'une segmentation explicite à l'aide des contours d'après [El-Yacoubi 1999].

2.3.1.2 Reconnaissance des séquences

Une fois un ensemble de points de segmentation potentiels estimé, tous les graphèmes[2] constituants la séquence sont regroupés en niveaux dans un treillis de segmentation. Les différents niveaux de ce treillis correspondent aux recombinaisons de ces graphèmes :

- le premier niveau est constitué de tous les graphèmes pris individuellement.
- le second niveau est constitué de toutes les combinaisons de deux graphèmes consécutifs.
- et ainsi de suite jusqu'à ce que le nombre de niveaux désiré soit atteint.

Le nombre de niveaux de recombinaison de graphèmes doit être suffisamment grand pour s'assurer que tous les caractères de la séquence sont présents de manière complète au moins une fois sur un des niveaux. Il varie donc en fonction de la tâche de reconnaissance (mots ou séquences numériques) et de la méthode de segmentation considérée. Le treillis permet de prendre en compte le fait que les caractères (lettres comme chiffres) ne sont pas toujours constitués d'un même nombre de graphèmes. Un m est logiquement plus découpé qu'un i. Les figures 2.12 et 2.13 *(b)* représentent respectivement des treillis de segmentation d'une image de mot et d'une image de séquence numérique.

La phase de reconnaissance vise à évaluer toutes les hypothèses de recombinaison des graphèmes. Un algorithme de programmation dynamique [Bellman 1962] permet ensuite de rechercher le meilleur chemin de segmentation parmi les hypothèses produites dans un premier temps. L'algorithme le plus largement utilisé à cet effet est le *Dynamic Time Warping*[3] ou *DTW* présenté dans [Sakoe 1978]. [Koerich 2002a, Morita 2003] pour la reconnaissance de mots et [Oliveira 2004, Sadri 2007, Liu 2004] pour la reconnaissance

[2] plus petite unité distinctive de l'écriture et est l'analogue d'un phonème pour la parole
[3] Déformation Temporelle Dynamique

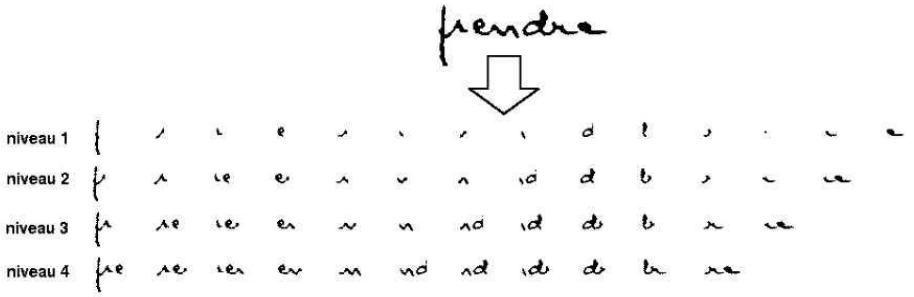
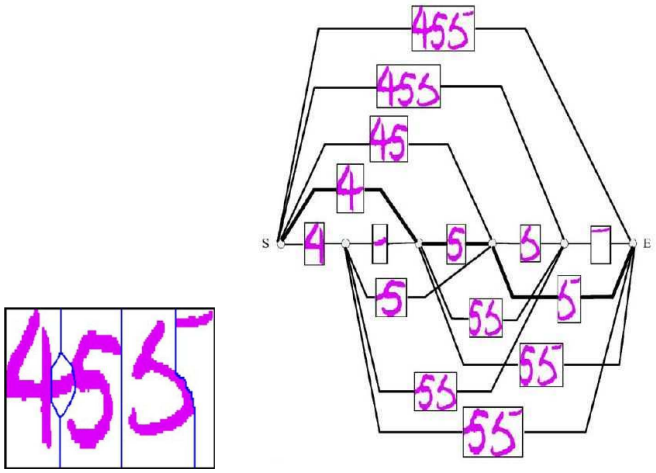


FIG. 2.12: Illustration d'un treillis de segmentation d'un mot à 4 niveaux extrait de [Koch 2006].



(a) (b)

FIG. 2.13: Illustration d'un treillis de segmentation en modélisation de séquences numériques d'après [Sadri 2007] : (a) chemins de segmentation proposé au moteur de reconnaissance chiffre, (b) parcours du treillis pour isoler le meilleur chemin de segmentation (en gras) et reconnaitre la séquence.

de séquences numériques utilisent le DTW. Le principe des algorithmes de programmation dynamique est illustré par la figure 2.14. Lorsque des images de mots manuscrits sont considérées, la recherche du meilleur chemin est contrainte par un lexique permettant de définir *a priori* les enchainements possibles de lettres dans la séquence. On parle de reconnaissance dirigée par le lexique. Ce n'est plus possible pour les séquences numériques dont la longueur en nombre de chiffres est *a priori* inconnue.

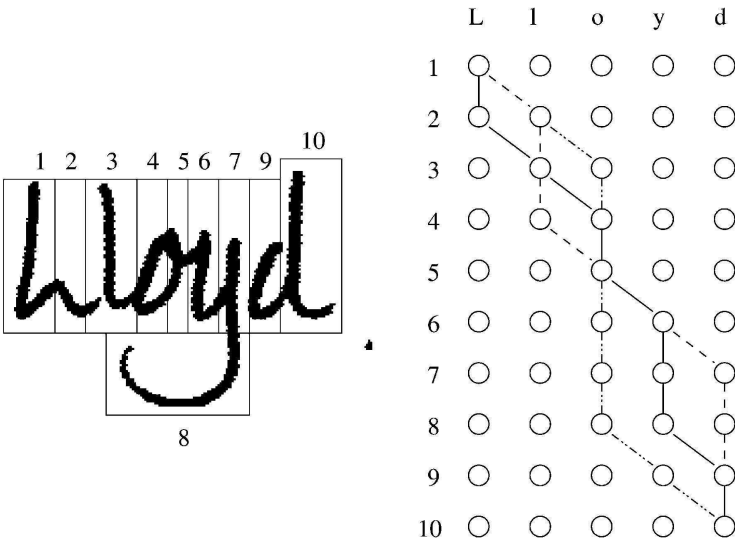


FIG. 2.14: Illustration du principe de la programmation dynamique par rapport à la segmentation explicite d'un mot en graphèmes. Figure extraite de [Vinciarelli 2002].

2.3.1.3 Reconnaissance des caractères

L'évaluation des hypothèses de segmentation dans les approches explicites se fait à l'aide d'un moteur de reconnaissance de caractères. Des vecteurs de caractéristiques sont extraits depuis toutes les formes de tous les niveaux du treillis puis soumis à un classifieur dont le but est de discriminer les formes représentant des caractères des formes représentant des fragments de caractères voire plusieurs caractères.

Un vecteur de caractéristiques doit modéliser efficacement les formes des différentes classes du problème tout en réduisant la dimension de leur représentation [Trier 1996]. Dans l'espace des caractéristiques, cela doit se traduire par le regroupement des formes appartenant à une même classe dans une même zone tout en les éloignant au maximum des régions englobant potentiellement les autres classes. Des exemples de caractéristiques sont donnés par les figures 2.15 et 2.16.

La phase de classification vise à attribuer une étiquette de caractère à une forme d'entrée. Cette opération est typiquement réalisée par un classifieur discriminant, le plus souvent statistique paramétrique [Jain 2000]. L'apprentissage des modèles consiste donc à déterminer les frontières de décision séparant les formes appartenant aux différentes classes dans l'espace de représentation. Les classifieurs les plus utilisés actuellement sont les SVM[4]

[4]de l'anglais *Support Vector Machine* ou Machine à vecteur de support

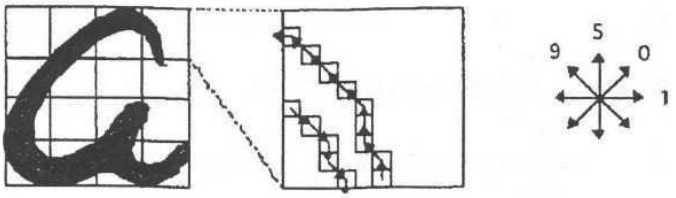


FIG. 2.15: Illustration du processus d'extraction du chaincode d'un caractère après son découpage en $4 * 4$ zones [Kimura 1994].

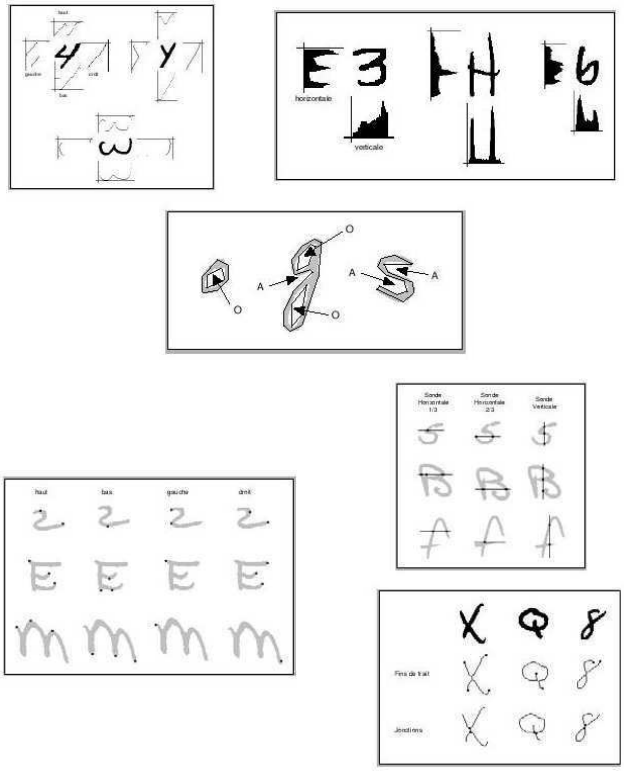


FIG. 2.16: Exemples de caractéristiques structurelles et statistiques extraites de [Heutte 1998]. De haut en bas et de gauche à droite : projection des profils, histogrammes de projection horizontale et verticale, détection d'occlusions et de concavités, extremum locaux du tracé suivant les 4 directions, nombre d'intersection avec une sonde verticale médiane et deux sondes horizontales aux tiers, nombre de fins de trait et de jonctions.

[Vapnik 1998] et les réseaux de neurones [LeCun 1990].

Malgré l'utilisation d'un treillis, l'approche de modélisation des séquences par segmentation explicite présente l'inconvénient majeur de chercher à produire des hypothèses de segmentation indépendamment de la reconnaissance. En effet, les méthodes de segmentation explicite ont pour but de détecter les caractères afin d'utiliser des classifieurs discriminants performants. À l'inverse, les méthodes à base de segmentation implicite cherchent à reporter la détection directement au niveau de la séquence. Ce type d'approche permet ainsi d'étudier des hypothèses de séquences où certaines décisions locales sont incertaines.

2.3.2 Reconnaissance à base de segmentation implicite

Lorsque l'on fait référence à une approche à base de segmentation implicite, chaque point du contour de la séquence est considéré comme un point de segmentation potentiel. Dans ce but, une fenêtre glissante est utilisée pour générer une séquence d'observations (ou trames) qui est décodée au regard d'un modèle (cf. figure 2.17). Ce modèle est caractérisé par une entrée du lexique dans le cas d'une reconnaissance dirigée ou par un modèle ergodique en l'absence d'un lexique [Ha 1997, Ha 1996].

Au lieu de chercher à segmenter explicitement une séquence en caractères ou en graphèmes, une sur-segmentation est volontairement visée. Celle-ci est matérialisée par une fenêtre glissant de gauche à droite (dans le sens de l'écriture) sur l'image de la séquence à reconnaitre. Pour fiabiliser la détection des points de segmentation et introduire un contexte dans chaque trame, un chevauchement entre deux fenêtres consécutives est souvent effectué. Deux hyperparamètres sont alors à fixer : la largeur de la fenêtre glissante et le recouvrement entre deux fenêtres successives.

Des caractéristiques sont extraites depuis chacune des trames ainsi obtenues. Elles permettent de représenter de manière compacte les formes contenues dans ces trames (également appelées observations). La recherche de la meilleure séquence de caractères au regard des observations, contrainte ou non par un lexique, peut être supportée par différents modèles : les modèles de Markov cachés [Marti 2001a, Vinciarelli 2003], les champs aléatoires conditionnels (CRF pour *Conditional Random Fields*) [Lafferty 2001], les réseaux de neurones récurrents [Graves 2009] ou les combinaisons hybrides neuro-markoviennes [Poisson 2004, Espana-Boquera 2011, Soullard 2011, Do 2011].

Les modèles de markov cachés (MMC) [Rabiner 1990] sont des modèles

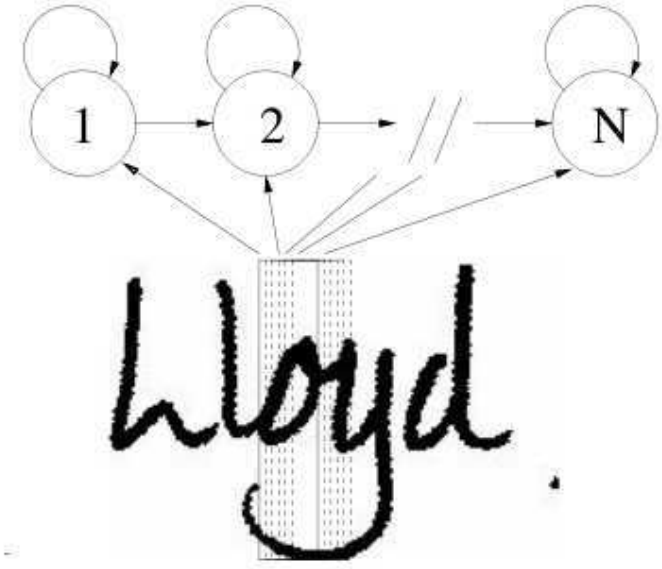


FIG. 2.17: Illustration du principe de la fenêtre glissante et de son utilisation avec des modèles de markov cachés de caractères dans [Vinciarelli 2003].

génératifs et probabilistes bien adaptés à la modélisation de données séquentielles. Ils ont été largement utilisés, notamment en traitement du langage (reconnaissance de la parole [Rabiner 1993] et de l'écriture [Plamondon 2000] par exemple) car ils permettent d'incorporer directement dans leur modèle des connaissances *a priori*. Les MMC cherchent à décoder, classiquement à l'aide de l'algorithme de Viterbi [Fornay 1973], une séquence d'observations par rapport à des modèles de mots appartenant à un lexique [Marti 2001a, Vinciarelli 2003]. Ces modèles de mots sont construits à partir de la concaténation des MMC de caractères les composant et modélisent l'enchainement des observations à l'aide d'états cachés. Une des raisons du succès et de l'utilisation massive des MMC en reconnaissance de la parole et de l'écriture réside dans la mise au point de l'algorithme de Baum-Welch [Baum 1972] permettant d'apprendre conjointement et de manière embarquée tous les MMC de caractères. Ceci permet de s'affranchir de la segmentation manuelle des données d'apprentissage en caractères pour l'apprentissage des modèles. L'apprentissage au regard du maximum de vraisemblance contribue cependant à rendre les MMC peu discriminants. Une autre limitation majeure des MMC réside dans l'hypothèse forte et généralement non vérifiée d'indépendance des observations.

À partir de ces MMC, deux pistes d'évolution ont été proposées pour ten-

ter de les rendre plus discriminants tout en gardant leurs propriétés modélisantes. D'un côté, il a été envisagé l'amélioration de la capacité discriminante des MMC en modifiant leur critère d'apprentissage. Les critères du Minimum d'Erreur de Classification [Juang 1992] (MCE pour *Minimum Classification Error*), du Maximum d'Information Mutuelle [Woodland 2002] (MMI pour *Maximum Mutual Information*) ou encore de maximisation de la marge [Do 2009, Sha 2007] ont été implémentés avec succès sur des tâches de reconnaissance de séquences diverses. En parallèle à ces travaux concernant spécifiquement l'apprentissage, il a été envisagé d'améliorer la modélisation locale des observations en combinant les MMC avec des classifieurs discriminants de type réseaux de neurones [Bengio 1995, Marukatat 2001, Bourlard 1998, Trentin 2001, Knerr 1998, Poisson 2004] ou SVM [Altun 2003] pour coupler les avantages des deux types de classifieurs. Toutefois, la même limitation d'indépendance des observations que pour les MMC est faite. L'entrainement des structures hybrides ainsi obtenues est complexifiée :

- par le réglage des hyperparamètres des deux classifieurs
- par leur apprentissage qui nécessite un étiquetage des données à différents niveaux : au niveau mots pour les MMC et au niveau états de modèles de caractères pour les classifieurs discriminants.

Ce problème est en partie résolu par l'utilisation d'algorithmes d'apprentissage conjoints et itératifs de type Expectation-Maximization [Dempster 1977]. Nous reviendrons plus en détail sur ces modélisations hybrides dans le dernier chapitre.

Pour contourner le problème du faible pouvoir discriminant des MMC, les CRF ont été utilisés en lieu et place des MMC pour modéliser des séquences [Shetty 2007, Lafferty 2001, McCallum 2005, Sicard 2006, Do 2006]. Plutôt que de chercher à modéliser la probabilité jointe d'observer la séquence de trames et son étiquette associée, les CRF modélisent directement la probabilité *a posteriori* (probabilité conditionnelle) d'observer une étiquette au regard de la séquence d'observation. Ceci confère aux CRF des pouvoirs discriminants que les MMC n'ont pas et permet de lever l'hypothèse sur l'indépendance des observations. Dans la pratique, cela permet aux CRF de mieux modéliser le contexte des observations. Comme dans le cas des travaux d'hybridation des MMC, les CRF ont également été combinés avec des classifieurs discrimants comme des réseaux de neurones profonds [Do 2011], des MMC [Altun 2003, Quattoni 2007, Andrew 2006, Soullard 2011] ou bien les deux [Vinel 2011]. La figure 2.18 illustre graphiquement les différences entre les modèles classiquement utilisés en reconnaissance de l'écriture manuscrite : les MMC génératifs intégrant des modèles de mélanges de gaussiennes, les combinaisons hybrides neuro-markoviennes et les CRF.

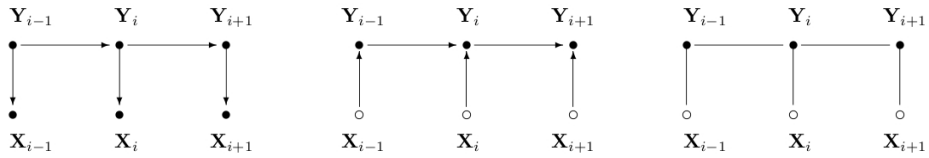


FIG. 2.18: Représentation graphique de 3 types de modélisation d'une séquence de trames. Illustration extraite de [Lafferty 2001]. X représente la séquence de trames à reconnaitre et Y la séquence de modèles. De gauche à droite : un MMC gauche-droite, une combinaison hybride MMC-réseau de neurones et un CRF. Les cercles blancs indiquent que les variables ne sont pas générées par le modèle.

Développés récemment et présentant de très bons résultats dans des domaines variés concernant la reconnaissance de séquences [Schuster 1997, Fernández 2007, Graves 2006], les réseaux de neurones récurrents, à l'image du *BLSTMN* [5] ont été adapté avec succès en reconnaissance de l'écriture manuscrite [Frinken 2010, Graves 2009]. Le BLSTMN a la capacité de prendre un compte un contexte de grande taille et bidimensionnel ce qui lui permet d'être très discriminant localement. Une couche de sortie caractérisée par un *CTC* [Graves 2006] [6] prenant directement en compte les sorties des deux couches cachées (une couche cachée pour le sens gauche-droite et une autre pour le sens droite-gauche) formées par des *LSTMN* permet d'atteindre de très bonnes performances de reconnaissance, comme en attestent les résultats de la compétition de reconnaissance de mots manuscrits *RIMES* [Grosicki 2009] remportée par un système intégrant cette combinaison de classifieurs. La couche de sortie CTC peut être associée à n'importe quel type de réseau de neurones, il a cependant été montré que le réseau permettant de prendre en compte le maximum de contexte présente les meilleurs résultats de reconnaissance [Graves 2005]. Le processus de décodage d'une séquence d'observations par un réseau BLSTM-CTC est illustré sur la figure 2.19. Comme pour les MMC, l'apprentissage du réseau global peut se faire de manière embarquée et contourne ainsi un des principaux inconvénients des réseaux récurrents, jusqu'alors cantonnés aux tâches de reconnaissance d'entités isolées et couplés aux MMC [Poisson 2004, Senior 1998, Bengio 1995]. Par contre, de telles structures présentent les inconvénients d'être très difficiles à dimensionner et paramétrer [Graves 2009]. Il est à noter que les couches cachées à base de BLSTMN ont également été combinées

[5] BLSTMN pour *Bidirectional Long-Short Term Memory Networks*
[6] CTC pour *Connectionist Temporal Classification*

avec succès à des réseaux bayésiens dynamiques (DBN pour *Dynamic Bayesian Networks*) [Wollmer 2010] dont la structure est proche de celle des CRF.

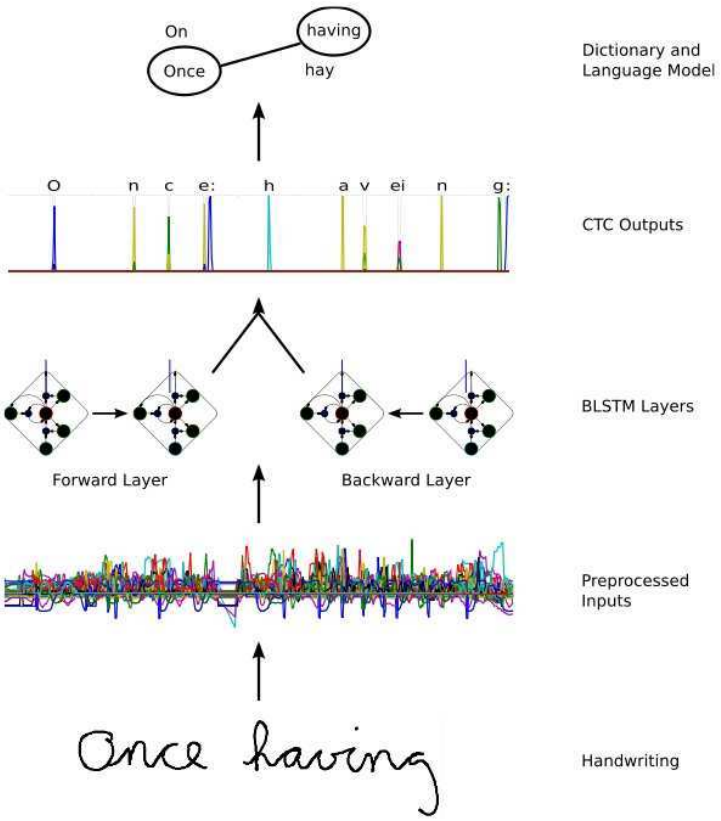


FIG. 2.19: Décodage d'une séquence d'observations par un réseau *BLSTM-CTC* [Graves 2009]. Illustration extraite de [Graves 2009].

Une stratégie de reconnaissance à base de segmentation implicite permet la conception de systèmes plus robustes aux variations de l'écriture que dans le cas d'une segmentation explicite car elles se basent notamment sur des descriptions locales des mots à travers ses trames et également plus de souplesse en associant le processus de segmentation à celui de la reconnaissance. Elles permettent ainsi d'atteindre des performances correctes sur de grands lexiques allant jusqu'à plusieurs dizaines de milliers de mots [Vinciarelli 2002, Koerich 2003].

Les modélisations de séquences isolées, qu'elles soient à base de segmentation explicite ou implicite, doivent être étendues pour permettre le passage à l'échelle des phrases puis des documents. Déjà évoqué, le problème du manque

de structure forte pour reconnaitre le contenu de documents non contraints est abordé par le biais de modèles de langage. Ces derniers forment, avec un lexique de très grande taille, les seules informations à notre disposition en l'absence de connaissances sur les documents à traiter. Certains modèles comme les MMC par exemple permettent d'intégrer des connaissances *a priori*.

2.3.3 Intégration de connaissances linguistiques

La modélisation de phrases à l'aide de modèles de langage permet de caractériser des connaissances linguistiques et donc de contraindre la reconnaissance de séquences [Lorette 1999, Plamondon 2000, Aubert 2002]. Les modèles de langage permettent de fournir au système une information *a priori* sur l'enchainement probable de séquences de mots. Développés dans un premier temps pour le décodage de la parole continue [Rabiner 1993, Aubert 2002], ils sont également utilisés en reconnaissance de l'écriture manuscrite que ce soit en-ligne [Perraud 2003, Quiniou 2007] ou hors-ligne [Vinciarelli 2004, Marti 2001a]. Les méthodes de décodage de séquences décrites dans la section précédente peuvent aisément être transposées au décodage de séquences de mots en prenant en compte un modèle supplémentaire caractérisant l'espace. À ce titre, les MMC sont très largement utilisés car ils permettent d'intégrer facilement cette contrainte linguistique lors du décodage [Marukatat 2001, Marti 2000, Vinciarelli 2004] sous la forme de transitions entre les MMC de mots.

Dans ses travaux de thèse, Quiniou dresse une taxonomie des modèles du langage [Quiniou 2007]. La modélisation est dite structurelle quand le langage est défini par des règles sur les mots. Ceux-ci sont répartis en catégories lexicales et sont associés suivant des règles prédéfinies ou inférées à partir d'un corpus d'apprentissage pour former des phrases. On va parler de modélisation statistique lorsque le langage est représenté par la fréquence d'apparition de suites de mots. Les *N-grammes* de mots permettent d'estimer ces fréquences. Leur principe est le suivant : ils approximent la probabilité de voir apparaitre un mot w_m en m^{ime} position d'une séquence de m mots en considérant que celui-ci ne dépend que des $N-1$ mots précédents, soit :

$$P(w_m|w_1, w_2, ..., w_{m-1}) \quad = \quad P(w_m|w_{m-N+1}, w_{m-N+2}, ..., w_{m-1}) \qquad (2.1)$$

L'estimation de l'ensemble de ces probabilités augmente exponentiellement avec N et nécessite un grand nombre d'exemples d'apprentissage pour assurer fiabilité et fidélité dans la modélisation d'un langage. Lorsque la taille du lexique est grande, un problème de dimensionnalité apparait. Les *N'-classes*

permettent de réduire la complexité des N-grammes en modélisant les relations entre des classes de mots. Ces classes peuvent être prédéfinies (en catégories morpho-syntaxiques par exemple : nom, verbe, adjectif...) ou bien être inférées automatiquement sur une base d'apprentissage dédiée. Moins complexes que les modèles de N-grammes, les modèles de N'-classes sont également moins précis.

En ce qui concerne des travaux sur la reconnaissance de l'écriture manuscrite hors ligne, nous pouvons citer [Vinciarelli 2004, Marti 2001a, Marti 2001b, Zimmermann 2004b]. Les auteurs utilisent des bigrammes de mots pour contraindre le décodage des MMC de lignes. Ces bigrammes sont classiquement appris sur les exemples de phrases de leur base d'apprentissage [Marti 2002] sous forme de fréquences d'apparition de paires de mots, ce qui correspond formellement à :

$$P(w_m|w_1, w_2, ..., w_{m-1}) \quad = \quad P(w_m|w_{m-1}) \tag{2.2}$$

L'équation 2.2 caractérise le fait qu'un mot w_m ne dépend que du mot qui le précède avec cette modélisation. Lorsqu'elle est restreinte à des bigrammes de mots, l'utilisation d'un modèle de langage pour contraindre la reconnaissance de phrases engendre un coût computationnel important pour un gain de performances minime voire nul s'ils sont estimés sur une base d'apprentissage trop petite.

Des modèles de langage peuvent également être défini au niveau caractère sous la forme de N-grammes de lettres et remplacer un lexique. La reconnaissance n'est plus dirigée par un lexique mais contrainte par ce modèle comme c'est le cas pour les *OCR*[7] et les *ICR*[8] [Ait-Mohand 2011]. Les N-grammes de caractères permettent également de caractériser les transitions d'un modèle de remplissage [Rodríguez-Serrano 2009a, Frinken 2012] dont nous reparlerons en détail dans la section suivante.

L'intégration de connaissances sous la forme de modèles de langage ajoute donc à la complexité combinatoire de la reconnaissance mais permet d'atteindre des performances plus élevées. Nous pouvons difficilement nous en passer pour garantir une plus grande robustesse de nos systèmes et nous verrons dans la fin de ce chapitre comme nous les utilisons pour apporter des connaissances pour contraindre la reconnaissance.

[7] OCR pour *Optical Character Recognition*
[8] ICR pour *Intelligent Character Recognition*

2.3.4 Synthèse sur la reconnaissance de séquences manuscrites

Les problématiques présentées dans cette section ont mis en avant une des difficultés majeures du domaine de la reconnaissance de l'écriture manuscrite hors ligne : la segmentation en sous-entités. Les différentes stratégies de reconnaissance évoquées caractérisent la façon dont l'écriture peut être modélisée. Les approches de modélisation sans segmentation conviennent bien à des problématiques très contraintes dont le vocabulaire est connu, fixé et limité à une cinquantaine de mots [Lavrenko 2004, Weliwitage 2003]. Pour pallier les limitations de ces approches, des méthodes de modélisation basées sur la segmentation ont été développées. Les méthodes implicites présentent l'avantage de bien modéliser de manière probabiliste les séquences [Vinciarelli 2003, Marti 2001a] contrairement aux méthodes à base de segmentation explicite qui se basent davantage sur la reconnaissance de caractères isolés. Ces dernières utilisent l'idée qu'un caractère bien segmenté sera reconnu avec un score élevé pour sa classe et qu'un fragment de caractère sera caractérisé par des scores de reconnaissance faibles pour toutes les classes, ce qui n'est pas systématiquement le cas.

Lorsque l'on connait l'information à rechercher (l'information pertinente) dans un document ou un ensemble de documents, il nous faut considérer le reste de l'information qui est par conséquent à rejeter. La modélisation de cet ensemble est toutefois peu évidente à définir et caractériser de manière précise car les connaissances *a priori* sur celui-ci sont limitées voire inexistantes. La modélisation du rejet constitue donc, après les étapes de segmentation et de reconnaissance, le troisième point clé à considérer lors de la conception d'un système d'accès au contenu de documents. Nous abordons ce point dans la section suivante.

2.4 Stratégies de modélisation du rejet

Nous voulons accéder au contenu manuscrit de documents non contraints et en extraire de l'information pertinente consistant en des mots clés, des séquences numériques ou des séquences alphanumériques. On veut donc être capable de rejeter l'information qui n'appartient pas au lexique : l'information non pertinente représenté par les séquences hors lexique que nous appellerons OOVS, anagramme de *Out of Vocabulary Sequences*.

Ce problème est abordé sous différents angles dans la littérature. Une

première solution consiste à reconnaitre les documents manuscrits dans leur intégralité pour ensuite rejeter les OOVS. Il faut alors considérer un lexique de très grande taille pour tenter de couvrir le vocabulaire complet de la langue. La pertinence syntaxique ou grammaticale des phrases reconnues est vérifiée en post-traitement à l'aide de modèles de langages. Un seuillage permet de valider ou rejeter les phrases reconnues [Bertolami 2006, Zimmermann 2004a]. Dans un second temps, la méthode de détection des OOVS est basée sur l'estimation de scores de confiance. Une mesure de distance est utilisée pour rejeter les séquences trop différentes des entrées du lexique [Wessel 2001, Choisy 2007, Rodríguez-Serrano 2009a]. Dans [Bazzi 2000, Manos 1997, Frinken 2010], la détection des OOVS est associée à la reconnaissance en utilisant un modèle de remplissage.

Le rejet de l'information non désiré (OOVS) peut donc être effectué de 3 différentes façons :

- par un rejet de distance entre une image de séquence isolée et une requête image. Cette stratégie est implémentée dans les systèmes de *Word Spotting*
- par une méthode de rejet par seuillage sur les scores de reconnaissance de séquences. On parle alors de *Keyword-Spotting*
- par une méthode de rejet à l'aide d'un modèle de remplissage en mettant en concurrence ce modèle avec des modèles classiques de séquences pendant la reconnaissance. Il s'agit d'extraire l'information recherchée noyée au milieu de l'information non pertinente[9].

Nous présentons ces approches dans les sections suivantes.

2.4.1 Rejet de distance

Cette stratégie de rejet est implémenté dans des approches de détection de mots clés par requêtes images [Rath 2002, Saykol 2004, Adamek 2007, Rodríguez-Serrano 2009b]. Elle se base sur une mesure de distance entre une image de référence, généralement extraite manuellement d'un document et servant de requête, et un ensemble d'images à tester segmentées depuis un document pour lequel on veut accéder rapidement au contenu.

Pour rejeter l'information jugée non pertinente, i.e. ne correspondant pas à la requête, une mesure de distance entre l'image de référence et les hypothèses de localisation des mots clés, produites par la segmentation du document en mots, doit être mise en place. La méthode de calcul de distance

[9]Une distinction est à faire entre le sens donné à *extraction d'information* ici et celui utilisé dans le chapitre 1 qui faisait référence au fait de mettre en relation les données essentielles à la compréhension d'un document

la plus simple consiste à utiliser directement les pixels de la requête image et ceux de l'image candidate pour la comparaison. Des distances classiques peuvent être utilisées comme un *XOR* (cf. figure 2.20), une mesure *EDM*[10] qui étend la mesure de distance basée sur le XOR pour diminuer l'influence du bruit (dû à la numérisation par exemple), une mesure *SSM*[11] qui équivaut à calculer la corrélation entre l'image cible et l'image candidate [Rath 2002] ou bien plus complexes mais aussi plus robustes aux déformations comme la distance *SLH*[12] [Manmatha 1996] ou la distance d'Hausdorff [Leydier 2009].

(a) Image 1

(b) Image 2

(c) XOR-Image of Image 1 and 2

FIG. 2.20: Distance entre deux images à l'aide d'un *XOR* d'après [Rath 2002].

Une méthode moins lourde en temps de traitement consiste à utiliser un vecteur de caractéristiques extrait depuis l'image du mot à détecter ou rejeter. Afin de modéliser au mieux le mot, des caractéristiques issues des contours (le *chaincode* ou les profils par exemple [Kimura 1994, Rath 2003b, Terasawa 2005]), structurelles, de concavités, issues de gradient ou encore des densités de pixels peuvent être considérées [Rothfeder 2003, Manmatha 1996, Saykol 2004] voire combinées comme dans les travaux de [Rath 2003a] avec le vecteur *GSC*[13] ou dans les travaux de [Adamek 2007] avec le vecteur *MCC-DCT*[14].

[10]EDM pour *Euclidean Distance Mapping*
[11]SSM pour *Sum of Squared Differences*
[12]SLH pour *Scott and Longuet-Higgins*
[13]GSC pour *Gradient, Structural and Concavity*
[14]MCC-DCT pour *Multi-scale Convexity Concavity - Discrete Cosine Transform*

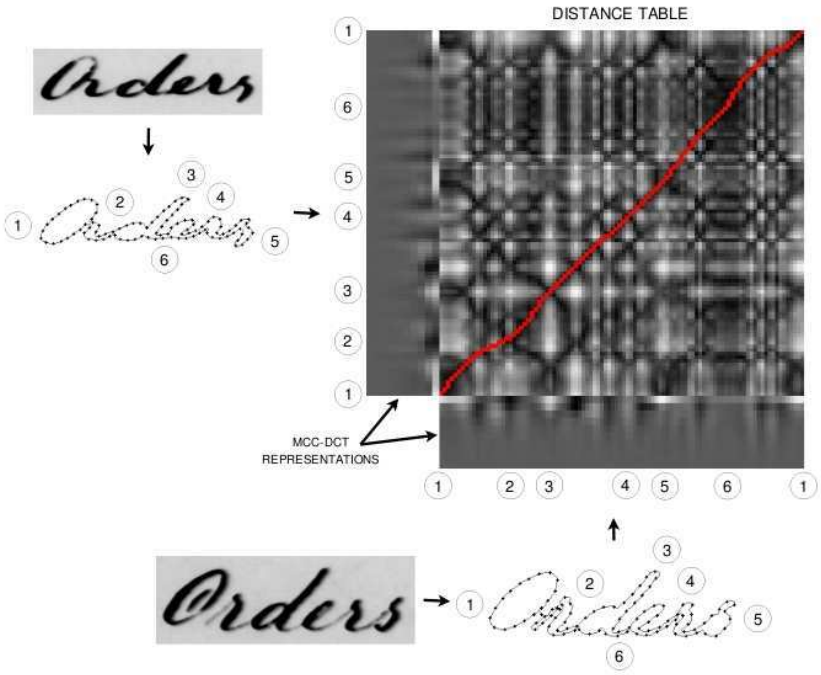


FIG. 2.21: Distance mesurée par DTW entre deux images caractérisées par le vecteur *MCC-DCT*. Illustration extraite de [Adamek 2007].

Un fenêtrage sur les images de mot peut aussi être effectué afin d'obtenir une séquence de vecteurs de caractéristiques. Pour comparer deux séquences de requête et de test, des algorithmes de programmation dynamique sont massivement utilisés [Bellman 1962]. Le plus commun d'entre eux, également le plus utilisé dans ce cadre est l'algorithme de déformation temporelle dynamique [Vintsyuk 1968, Sakoe 1978] ou *DTW*[15]. Il est utilisé pour calculer la mesure de distance entre une image requête et une image cible dans [Adamek 2007, Rath 2003b, Rath 2004, Rodríguez-Serrano 2009b] (cf. figure 2.21).

Cette stratégie de rejet, de par la nature des requêtes prises en compte, n'autorise que peu de flexibilité en ce qui concerne les types d'écriture à reconnaitre. Des applications très ciblées d'indexation ou de recherche d'information dans les manuscrits anciens se basent sur une telle approche [Rath 2004, Sankar 2007, Terasawa 2005]. Pour des problématiques nécessitant plus de souplesse quant aux requêtes et plus de robustesse face à la variabilité de l'écriture, une stratégie de rejet intervenant après une phase de

[15]DTW pour *Dynamic Time Warping*

reconnaissance est nécessaire. Elle permet notamment plus de souplesse en ce qui concerne les requêtes de mots clés. Nous la présentons dans la section suivante.

2.4.2 Rejet par reconnaissance - seuillage

Contrairement à l'approche de rejet par mesure de distance décrite dans la section précédente, une stratégie de rejet par reconnaissance - seuillage se base sur une étape de reconnaissance servant de base pour ensuite rejeter les OOVS ou les séquences mal reconnues. L'étape de reconnaissance vise à produire un score par rapport à un modèle construit au regard d'une requête ASCII ou de séquences appartenant à un lexique et se rapproche donc d'une problématique de reconnaissance de séquences isolées décrite en section 2.3. Le point clé et également le principal inconvénient de cette stratégie de rejet réside dans le fait qu'il faille normaliser ces scores de reconnaissance afin de déterminer si l'image candidate est à accepter ou à rejeter par rapport à la requête. Le terme de normalisation est généralement calculé par rapport à la séquence candidate X [Kolcz 2000, Choisy 2007, Rodríguez-Serrano 2008] en cherchant à estimer $P(X)$ et le rejet se fait par rapport à un seuil sur le score normalisé.

La normalisation des scores de reconnaissance peut être effectuée à deux échelles :

- au niveau mot [Kolcz 2000, van der Zant 2008, Choisy 2007, Rodríguez-Serrano 2009a]
- au niveau phrase [Zimmermann 2006, Bertolami 2006, Zimmermann 2003]

En ce qui concerne la normalisation au niveau mot, cette stratégie de rejet est notamment implémentée dans les systèmes de détection de mots clés et nécessite donc une phase de segmentation explicite en mots. La normalisation se caractérise par l'estimation d'un score de confiance permettant d'accepter ou de rejeter la séquence isolée traitée.

Dans [Choisy 2007], le score d'une séquence isolée et celui du terme de normalisation sont estimés lors de deux reconnaissances distinctes par le même classifieur (à base de NSHP-HMM[16] illustré par la figure 2.22). Le score de reconnaissance est estimé à partir d'un modèle de séquence construit par la concaténation des modèles de caractères la composant alors que le terme de normalisation est estimée par un modèle générique de mot construit à partir

[16]NSHP-HMM pour *Non-Symmetric Half Plane - Hidden Markov Model* introduit dans [Saon 1997]

d'un unique modèle générique de caractère appris sur l'ensemble des caractères de la base d'apprentissage. L'idée sous jacente est que si un mot correspond effectivement au modèle du mot clé recherché, la valeur du score de confiance normalisée est élevée. Dans le cas contraire, cette valeur est faible. Un seuillage sur ce score permet ensuite d'accepter une image par rapport à une requête ou de le rejeter.

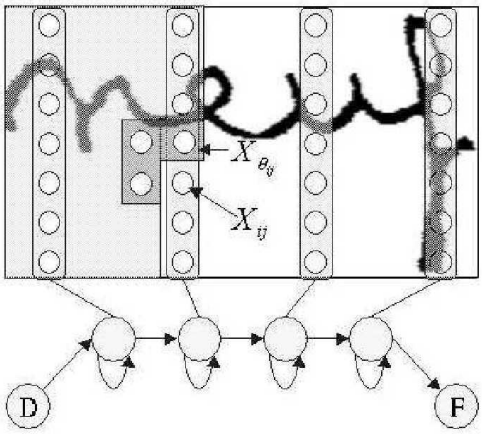


FIG. 2.22: Illustration du NSHP-HMM extraite de [Choisy 2007] : les MMC modèlisent l'enchainement de colonnes alors que les champs de Markov permettent de prendre en considération le voisinage des pixels.

Dans [Rodríguez-Serrano 2008], les auteurs se basent sur la même idée et l'estimation d'un score de confiance similaire. Toutefois, aucune étape de reconnaissance supplémentaire n'est effectuée et la normalisation ne nécessite pas l'apprentissage d'un nouveau modèle. Dans ce cas, la reconnaissance des mots étant supportée par des MMC, les GMM embarquées dans ces MMC sont directement utilisées pour estimer $P(X)$. Un seuillage équivalent à [Choisy 2007] permet d'accepter ou de rejeter une image de mot candidate. Le système complet de détection de mots clés est représenté par l'illustration de la figure 2.23.

Le rejet de phrases et donc de séquences de mots se base également sur l'estimation d'un score de confiance, la stratégie d'accès au contenu sous jacente étant la reconnaissance complète de document. Un lexique de grande taille est pris en compte en entrée et la reconnaissance des lignes de texte est guidée par l'utilisation de modèles de langage. Le but est de créer une retranscription du contenu du document [Marti 2000, Vinciarelli 2004, Favata 1998, Senior 1998] et de vérifier celui-ci par des post-traitements validant ou non les hypothèses de reconnaissance en utilisant les connaissances linguistiques véhiculées par les modèles de langage.

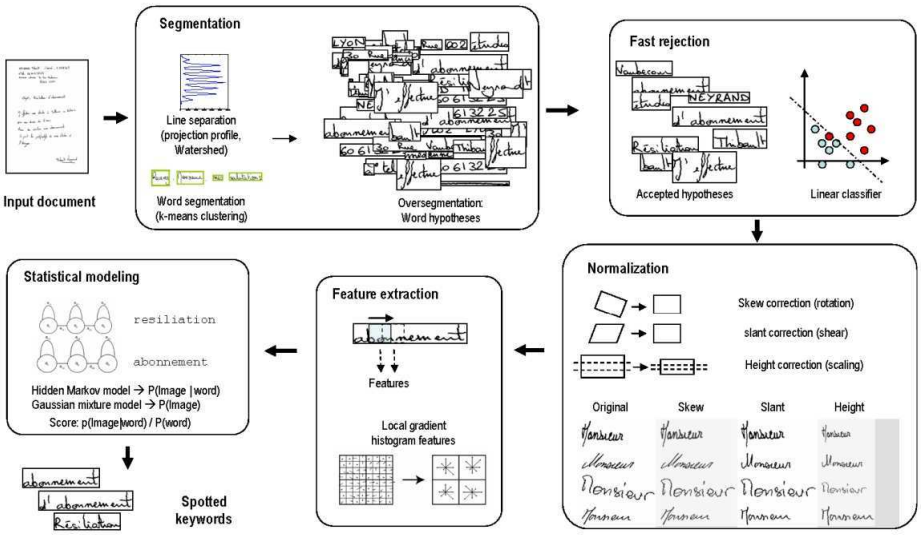


FIG. 2.23: Illustration de la chaine de traitements complète du système présenté dans [Rodríguez-Serrano 2009a] : les MMC modèlisent l'enchainement de colonnes alors que les champs de Markov permettent de prendre en considération le voisinage des pixels.

Le terme de normalisation sur les phrases est donc estimé à partir des modèles de langage [Bertolami 2006, Zimmermann 2003]. Les auteurs s'inspirent de travaux similaires dans le cadre de la reconnaissance de l'écriture en-ligne à l'image de [Marukatat 2002, Quiniou 2007, Perraud 2003] mais aussi dans les nombreuses investigations menées dans le domaine de la reconnaissance de la parole continue [Sanchis 2000, Zeppenfeld 1997].

Dans [Zimmermann 2006], les auteurs utilisent un modèle grammatical de phrases (le *Stochastic Context-Free Grammar*) à l'image de N-classes de mots [Perraud 2003, Quiniou 2007]. Celui-ci est extrait depuis un corpus de plusieurs millions de mots. Ces N-classes sont utilisées pour analyser la qualité syntaxique des phrases. Une pénalité est introduite après la reconnaissance pour une phrase n'ayant pas une grammaire correcte. Ce module permet de faire remonter les hypothèses syntaxiquement correctes lors de la reconnaissance en diminuant les non-sens grammaticaux.

Dans [Brakensiek 2004], les auteurs cherchent à valider ou rejeter des adresses postales. Les scores de reconnaissance sont normalisés par rapport au nombre de trames « consommées » par l'adresse à reconnaitre. Dans [Gorski 1997], ce sont les montants littéraux des chèques qui sont vérifiés. Une vingtaine de caractéristiques extraites sur les meilleurs hypothèses de vraisemblance produits par le moteur de reconnaissance nourrit un réseau de

neurones. Celui-ci produit une mesure de confiance qui, comparé à un seuil, permet de valider ou non l'hypothèse de départ.

Cette stratégie de rejet convient bien lorsque l'on souhaite fiabiliser la détection de mots clés ou la reconnaissance complète d'un document manuscrit. Toutefois, les performances diminuent lorque la taille du lexique augmente et la séquentialisation des traitements accroit les temps de calculs. La normalisation peut également être dépendante des données tout comme le paramétrage du seuil de rejet. Dans la section suivante, nous présentons une troisième stratégie de rejet visant à réunir reconnaissance et rejet en une étape unique.

2.4.3 Rejet de mots par modèle de remplissage

Le principe de rejet par modèle de remplissage est de mettre en concurrence, pendant la reconnaissance, les modèles de séquences à rechercher (appartenant à un lexique donné ou constitués d'une requête ou d'un ensemble de mots clés) avec un modèle spécifique pour rejeter les OOVS. Ce modèle de rejet, caractérisé par un modèle de remplissage[17] a pour but d'englober les OOVS non représentés explicitement par des requêtes ou des entrées du lexique. Il est constitué de l'ensemble des caractères de l'alphabet interconnectés entre eux. Les transitions au sein de ce modèle peuvent être équiprobables, on parle alors de modèle ergodique [El-Yacoubi 2002], ou bien fixées par un modèle de bigrammes de caractères [Koch 2006, Fischer 2010].

Une approche utilisant un tel modèle de rejet se base donc directement sur la reconnaissance et non plus sur les scores qu'elle produit. Les méthodes de rejet par modèle de remplissage ont été développées dans un premier temps pour détecter la présence d'OOVS en reconnaissance de la parole [Bazzi 2000, Fetter 1998, Parada 2010, Manos 1997] puis implémentées en reconnaissance de l'écriture manuscrite [El-Yacoubi 2002, Koch 2006, Fischer 2010, Frinken 2010]. Un des inconvénients peut être d'absorber des fragments de signal assimilés comme des OOVS alors qu'il s'agit de séquences appartenant au lexique. Ce problème peut être contrôlé via l'utilisation d'une probabilité *a priori* sur ce modèle ou bien d'une pénalité contrôlant le taux d'apparition d'OOVS [Bazzi 2000, Manos 1997, Fischer 2010]. Un modèle de remplissage joue donc le rôle de modèle générique de séquences ou de « modèle poubelle » (comme souvent désigné en anglais par *garbage model*) et entre en compétition avec les séquences du lexique au cours de la reconnaissance.

[17] de l'anglais *filler model*

Dans [Koch 2006], les mots hors lexique sont représentés par un modèle de remplissage entrant en concurrence avec les mots clés appartenant à un lexique de taille réduite (moins de 500 mots) lors de la reconnaissance. Le modèle de ligne utilisé est présenté sur le schéma de la figure 2.24. Deux types de modèles de remplissage sont comparés : le premier dont les transitions entre les modèles de caractères sont apprises sur la base d'apprentissage, le second ergodique. Les meilleurs résultats d'extraction sont obtenus pour la modélisation du modèle de remplissage la plus fine.

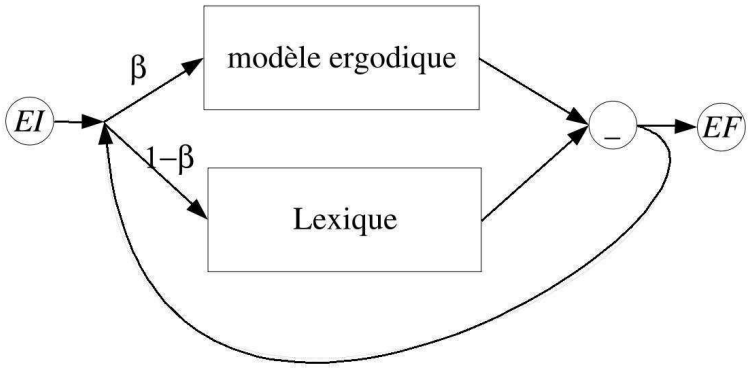


FIG. 2.24: Illustration du modèle de ligne pour la détection de mots clés de [Koch 2006]. Celui-ci intègre un modèle de remplissage à 26 états représentant les 26 caractères de l'alphabet

Dans [El-Yacoubi 2002], les auteurs utilisent un modèle de remplissage ergodique afin d'absorber l'information non pertinente et donc isoler l'information recherchée, dans ce cas précis les noms de rue dans des adresses postales. Un modèle générique de ligne à base de MMC est utilisé à cet effet. Globalement, une ligne est constituée de deux modèles ergodiques en début et fin de ligne avec potentiellement des noms de rue modélisés par la concaténation de MMC de caractères. La structure de ces derniers permet de couvrir la variablité de l'écriture mais également de prendre en compte des cas de sur et sous segmentation des caractères. Ce modèle contraint la reconnaissance et permet d'extraire les noms de rues à partir de séquences de graphèmes segmentés explicitement. Il est illustré par la figure 2.25.

Également à base de MMC et d'un modèle générique de ligne, les travaux présentés dans [Fischer 2010] intègrent une gestion des OOVS à l'aide d'un modèle de remplissage (cf. figure 2.26). Celui-ci est ergodique et constitué par l'ensemble des MMC de caractères appris. Des séquences de vecteurs de 9 caractéristiques [Marti 2000] sont extraites depuis les images de lignes d'entrée

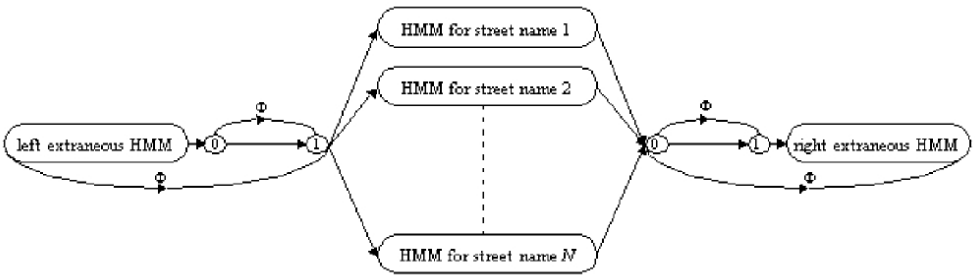


FIG. 2.25: Illustration d'un modèle de ligne pour la reconnaissance de nom de rue dans [El-Yacoubi 2002]. Deux modèles de remplissage (*left* et *right extraneous HMM*) sont appris pour absorber les portions d'écriture ne correspondant pas à des noms de rue.

à l'aide d'une fenêtre glissante (approche de segmentation implicite). Alors que dans [Koch 2006], la taille du lexique atteint 500 mots, dans [Fischer 2010, Frinken 2010] un seul mot est recherché et donc aucun lexique n'est pris en compte. De plus, le modèle de ligne présenté est construit pour que le mot clé recherché soit présent dans la ligne cible. Le score de cette ligne est normalisé par un score obtenu sur cette même ligne en contraignant la reconnaissance par le modèle de remplissage seul. Un seuillage est ensuite effectué pour valider ou rejeter l'hypothèse. Si l'hypothèse de reconnaissance est validée, et que le mot à rechercher est à la fois trouvé dans la ligne et présent dans son étiquette, une bonne détection est comptabilisée, sans que sa position ne soit toutefois vérifiée précisément.

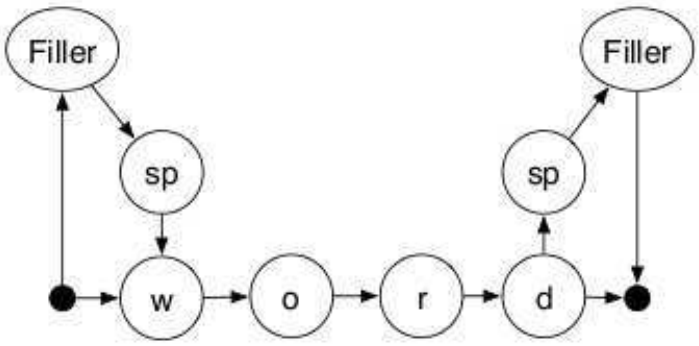


FIG. 2.26: Illustration d'un modèle de ligne pour l'extraction d'information dans [Fischer 2010].

Dans [Frinken 2010, Frinken 2012], ces mêmes auteurs ont travaillé

à rendre plus discriminant localement le modèle de ligne décrit dans [Fischer 2010]. Un classifieur de type réseau de neurones récurrents est utilisé à cet effet. Plus précisément, les mélanges de gaussiennes sont remplacés par les réseaux de neurones bidimensionnels présentés dans [Graves 2009] et abordé dans la section 2.3.2. Comparativement aux MMC, ces réseaux neuronaux semblent offrir une meilleure précision pour des valeurs de rappel convenables (un mot clé sur deux bien détecté pour 95 % de précision). Bien que la mise en place d'un protocole expérimental pertinent soit délicate, les mêmes limitations expérimentales que dans [Fischer 2010] doivent être notées.

2.4.4 Synthèse sur la modélisation de l'information à rejeter

Dans cette section, nous avons présenté trois stratégies de rejet. La première stratégie cherche à mesurer la distance entre une requête image et une image candidate. Elle est rapide mais impose une étape de segmentation du document en mots. Elle est peu souple en termes de requêtes puisqu'il faut disposer d'exemples sous la forme d'images ce qui la rend également peu robuste aux variabilités de l'écriture et donc inadaptée à un contexte omniscripteur.

La seconde stratégie vise à utiliser les scores de reconnaissance pour accepter ou rejeter des données. S'il s'agit de mots, l'approche présente plus de souplesse que la précédente en termes de requêtes mais possède l'inconvénient d'être séquentielle. Le rejet dépend alors de la segmentation en mots et de la normalisation choisie pour des scores de reconnaissance. S'il s'agit de phrases, le rejet, tout comme la reconnaissance, est lourde en termes de temps de traitements car le lexique et les modèles de langage à prendre en compte peuvent être très grands.

La troisième et dernière stratégie de rejet basée sur la modélisation de l'information non pertinente par un modèle de remplissage présente l'avantage de paralléliser les prises de décision d'acceptation et de rejet lors de la reconnaissance. Elle n'alourdit donc pas le système avec un module de post-traitements.

Dans la section suivante, nous justifions nos choix stratégiques au regard de cet exposé bibliographique et présentons le modèle de ligne mis en œuvre pour l'extraction d'information.

2.5 Un modèle de ligne pour l'extraction d'information

Nous avons présenté notre sujet d'étude à la fin du chapitre précédent : nous voulons développer un système complet de détection d'informations alphabétiques (mots clés), numériques (code postaux, numéros de téléphones, etc.) et alphanumériques (code client, dates, etc.) dans des documents manuscrits non contraints à partir de requêtes définies automatiquement ou par un utilisateur.

Dans ce chapitre, nous avons jusqu'à présent décrit les trois points clés à considérer afin de concevoir un tel système, à savoir : la segmentation, la reconnaissance et le rejet. Nous avons mis en avant la complexité de ces trois tâches, leur interdépendance et la nécessité de fournir au système des connaissances *a priori* pour fiabiliser leurs décisions.

Afin de minimiser l'impact d'éventuelles erreurs de segmentation, de reconnaissance ou de rejet sur les performances globales du système en détection d'information, nous avons décidé de paralléliser au maximum ces trois traitements. Cela se traduit par :

- une **segmentation implicite des lignes** en trames par une méthode à base de fenêtre glissante
- un moteur de reconnaissance choisissant les **meilleurs points de segmentation conjointement aux meilleures hypothèses de reconnaissance**
- une modélisation générique de la ligne intégrant la ou les requêtes à rechercher dans les documents et l'information à rejeter caractérisée par un **modèle de remplissage**

Nos choix se sont orientés vers une modélisation globale des lignes de texte. Utilisé pour guider et contraindre la reconnaissance, un tel modèle de ligne permet d'effectuer dans un même temps la segmentation des lignes en sous entités (caractères, mots, etc.), la reconnaissance des informations pertinentes qu'elle peut contenir au regard des requêtes et le rejet de l'information non pertinente. Notre première contribution concerne la conception de ce modèle de ligne.

Une ligne peut être vue comme une succession de séquences alphabétiques, numériques et/ou alphanumériques séparées par des espaces. Deux types de séquences sont donc à discriminer au sein d'une ligne :

- **les requêtes** désignant les séquences que l'on souhaite **rechercher** dans un document, pouvant être constituées de n'importe quel enchainement

de caractères (lettres, chiffres, minuscules, majuscules, ponctuation)
– **toutes les autres séquences** que l'on souhaite **rejeter** constituées par un modèle de remplissage pour représenter les OOVS entourant l'information pertinente

Nous proposons donc un modèle de ligne qui intègre en parallèle les modèles de requêtes et le modèle de remplissage et qui fait apparaitre un modèle d'espace entre ces deux types d'information. Le modèle global de ligne est illustré par le schéma de la figure 2.27.

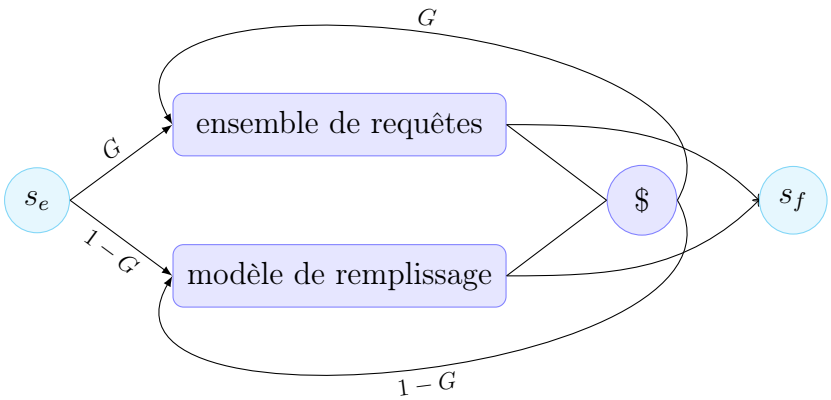


FIG. 2.27: Illustration du modèle générique de ligne. s_e et s_f sont des modèles structurels initial et final et $ représente le modèle d'espace.

Ce modèle met en concurrence les deux types d'information modélisés : les requêtes caractérisées par la concaténation de leurs modèles de caractères (cf. figure 2.28) et le modèle de remplissage (lettres minuscules et majuscules, chiffres, signes de ponctuation ... cf. figure 2.29). Il est à noter que ce choix de modélisation nous permet de considérer des requêtes de différente nature : alphabétique, numérique et alphanumérique. La variable G est un hyperparamètre du système à optimiser et probabilise la transition d'un type d'information à l'autre lors de la reconnaissance d'une ligne de texte. Il peut également être vu comme la proportion d'information pertinente recherchée dans une ligne et présente *a priori* dans celle-ci.

Les détails d'implémentation du modèle générique de ligne et des modèles de requêtes et de remplissage le composant seront donné dans le chapitre suivant qui traitera plus spécifiquement de la mise en œuvre pratique du système complet.

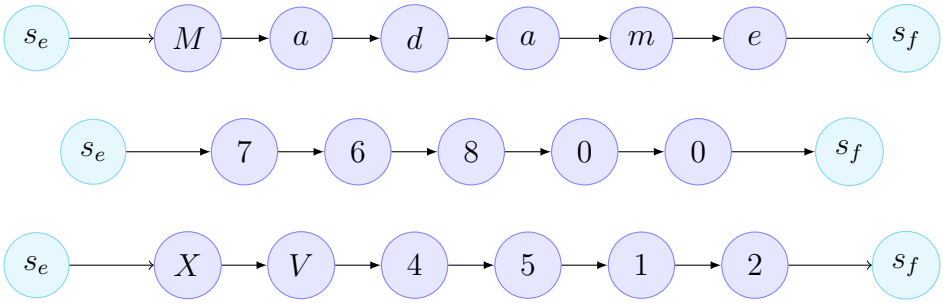


FIG. 2.28: Illustration de modèles de requêtes alphabétiques (*Madame* sur l'exemple), numériques (*76800*) et alphanumériques (*XV4512*). s_e et s_f sont des modèles structurels de début et de fin de modèle.

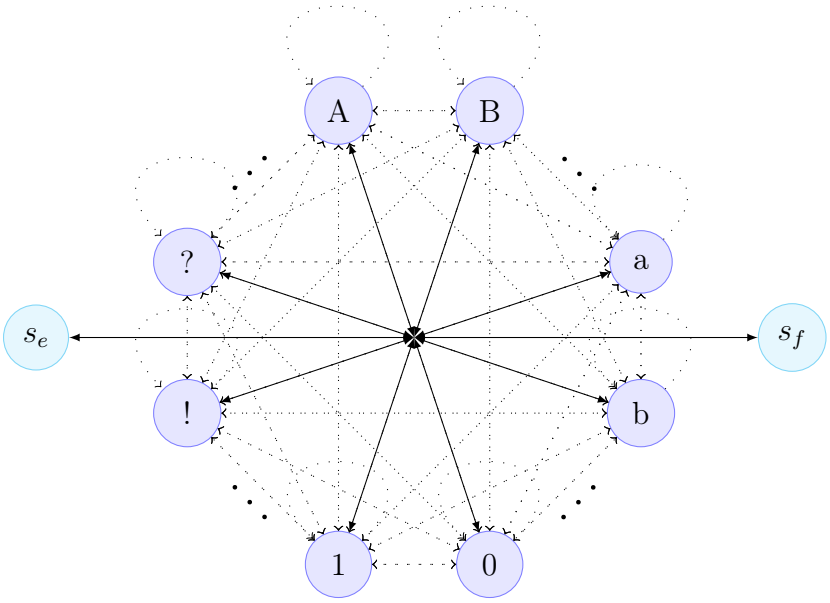


FIG. 2.29: Illustration du modèle de remplissage. s_e et s_f sont des modèles structurels initial et final.

2.6 Conclusion

Au cours de ce chapitre, nous avons identifié et présenté trois points essentiels liés à l'accès au contenu d'images de documents manuscrits : la segmentation des lignes d'un document en mots, la reconnaissance de l'écriture manuscrite de manière générale et les stratégies de rejet d'hypothèses de reconnaissance. Nous avons notamment mis en avant le point critique qu'est la segmentation de séquences en sous-entités afin de se ramener à la problématique maitrisée de la reconnaissance de caractères et le dilemme que

forment ces trois étapes essentielles.

Pour accéder à un contenu précis de documents manuscrits, nous avons choisi une approche entièrement parallélisée en ce qui concerne les prises de décision de segmentation, de reconnaissance et de rejet au niveau des lignes. Pour ce faire, une stratégie de segmentation implicite à l'aide d'une fenêtre glissante a été retenue pour sa souplesse, la reconnaissance étant guidée par un modèle de ligne générique probabiliste intégrant les deux types d'information à discriminer : l'information pertinente constituée par les requêtes et l'information non pertinente représenté par un modèle de remplissage.

Ces choix correspondent à une approche d'extraction d'information. Cette approche tend à se démarquer des stratégies classiques de lecture complète de documents et de détection de mots clés dans leur façon d'aborder le point sensible qu'est la modélisation de l'information non pertinente à savoir les OOVS. La principale différence en termes de traitements réside dans la parallélisation des prises de décision rendant un tel système moins sensible à la propagation d'erreurs au cours des traitements. Le chapitre suivant présente la mise en œuvre du système complet d'extraction d'information autour de cette modélisation des lignes de texte dont nous présenterons également les détails d'implémentation.

Chapitre 3

Un système d'extraction de mots clés dans des documents manuscrits

Sommaire

3.1 Introduction

Le chapitre précédent nous a permis de présenter les stratégies d'accès au contenu de documents manuscrits et la modélisation de l'écriture. En particulier, nous avons introduit un modèle générique de ligne manuscrite permettant d'extraire de l'information. Il met en concurrence un modèle de l'information pertinente à extraire et un modèle de remplissage dont le but est d'absorber l'information non pertinente. Ce chapitre a pour objet de présenter l'implémentation du système complet de détection d'information autour de ce modèle.

Tout système de reconnaissance de documents manuscrits intègre une série de traitements incontournables visant à récupérer l'information qu'ils contiennent. Dans notre cadre de détection d'information, ces traitements sont supportés par un moteur de reconnaissance contraint par notre modèle de ligne. Les lignes d'écriture d'un document doivent donc être isolées et traitées afin de produire des séquences d'observations qui vont être décodées au regard du modèle de ligne. La figure 3.1 présente la chaine de traitements proposée pour l'extraction d'information dans un document manuscrit. Dans la partie *compilation du modèle*, le modèle de ligne est construit à partir d'une ou plusieurs requêtes d'un utilisateur constituant un ensemble de séquences à détecter et du modèle de remplissage appris sur la base d'apprentissage. Dans la partie *préparation des données*, il s'agit de segmenter le document en lignes de textes, de prétraiter ces lignes pour diminuer la variabilité de l'écriture qu'elles contiennent et de les caractériser. Nous obtenons des séquences d'observations que l'on va décoder par rapport au modèle de ligne construit.

Dans ce chapitre, nous présentons de manière détaillée les implémentations des modules de préparation des données de la figure 3.1 en les confrontant aux solutions proposées dans la littérature. Le réglage et l'apprentissage des paramètres du système sont également exposés. Les expérimentations conduites pour éprouver notre système sont ensuite présentées. Un protocole de détection d'information a été conçu pour cela. Nous discuterons enfin des résultats obtenus avec ce protocle sur une base d'images de courriers manuscrits.

3.2 Implémentation du système d'extraction d'information

Dans cette section, nous présentons les principaux modules du système de détection d'information implémenté. Pour cela, nous décrivons les traitements

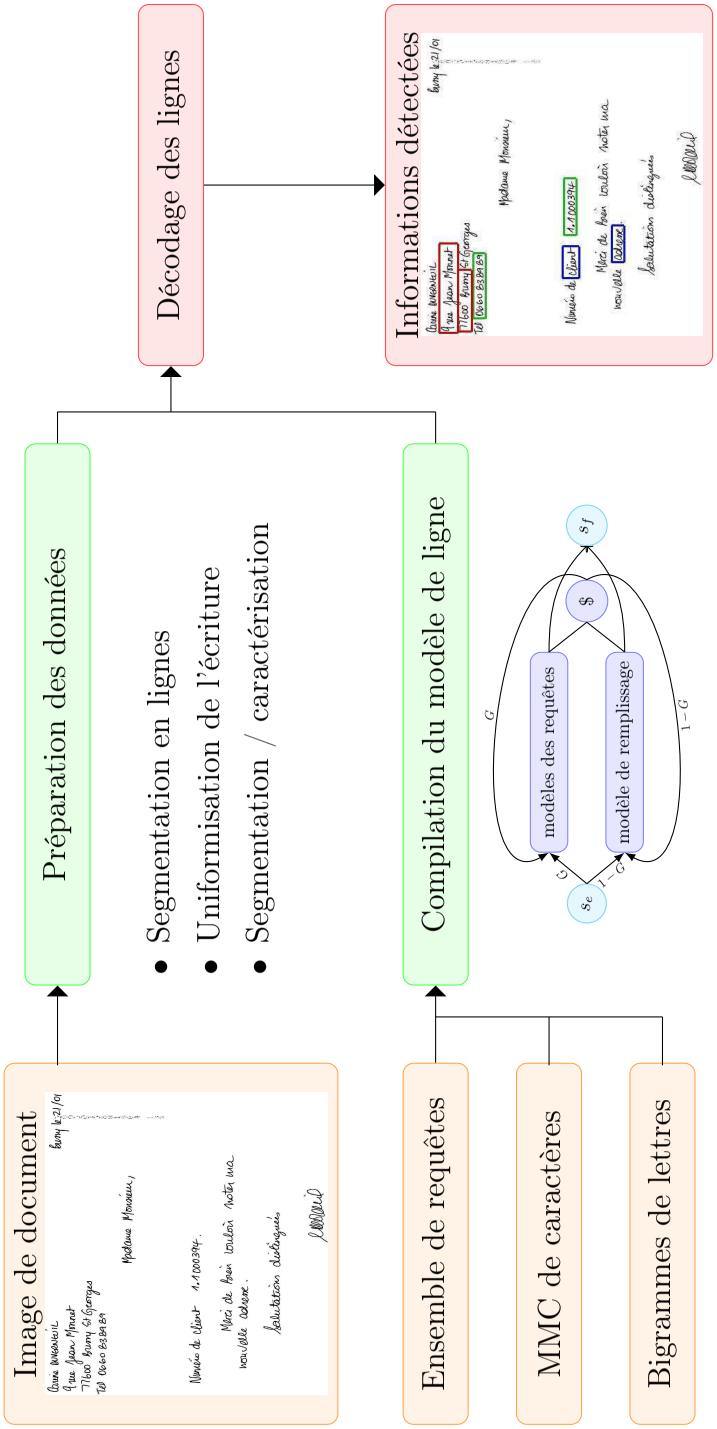


FIG. 3.1: Vue générale du système d'extraction d'information dans des documents manuscrits.

représentés sur le schéma de la figure 3.1 à savoir la segmentation du document

en lignes, les prétraitements sur ces lignes, leur segmentation et caractérisation visant à produire des séquences d'observations à décoder par le dernier module. Nous commençons par présenter le module de segmentation.

3.2.1 Segmentation du document en lignes de texte

Le but d'un tel module est d'isoler les lignes de texte manuscrites présentes dans un document. Cette phase d'extraction des lignes de texte se base sur les règles de nomenclature de l'écriture manuscrite occidentale : nous écrivons de gauche à droite par rapport à un support imaginaire horizontal, lorsque l'on atteint une extrémité du support d'écriture, nous recommencons l'écriture sur un support imaginaire inférieur en évitant le chevauchement de deux lignes successives. La figure 3.2 illustre la segmentation d'une image de document manuscrit en lignes. Elle met en avant la complexité de la tâche et présente différents types d'erreurs de segmentation.

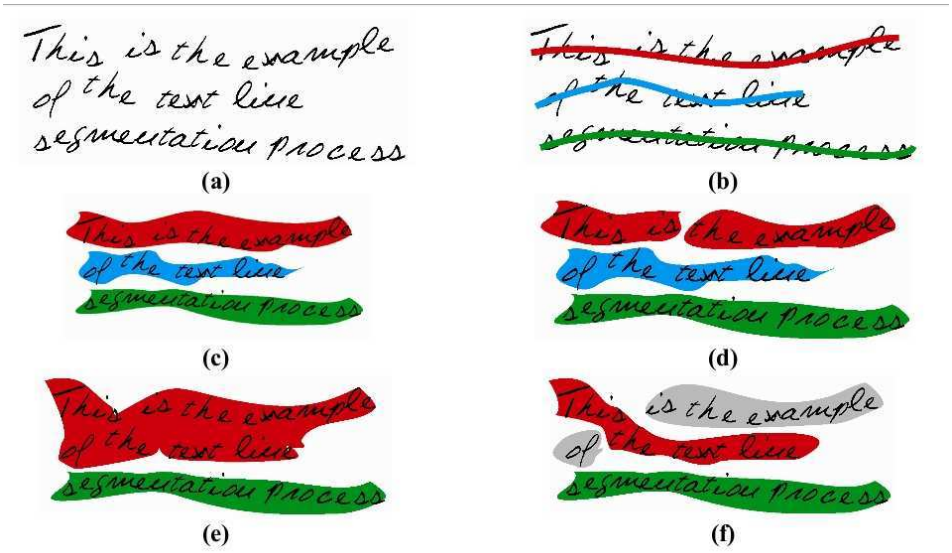


FIG. 3.2: Segmentation d'un document manuscrit en lignes d'après [Brodic 2011]. (a) image d'origine. (b) alignements de référence. (c) Segmentation en ligne correcte. (d) Sur segmentation du document en lignes. (e) Sous segmentation du document en lignes. (f) Segmentation en lignes avec des mots assignés à des lignes différentes.

Deux stratégies sont rencontrées dans la littérature pour extraire les images de ligne de texte dans un document [Likforman-Sulem 2007] :

– en considérant le document dans son intégralité, la position des lignes de texte peut être estimée par découpage de ce document en sous-portions,

idéalement en lignes. Une telle approche est qualifiée de descendante [Marti 2001a, Zheng 2005].

– en considérant des sous-entités localisées dans un document (pixels d'écriture, composantes connexes etc.), leur regroupement en lignes constitue une seconde approche : celle-ci est qualifiée d'ascendante [Wong 1982, Nikolaou 2010, Gatos 2010].

Etant donnée la nature peu contrainte de l'écriture manuscrite, il est probable de rencontrer des chevauchements entre deux lignes successives engendrant un recouvrement de leur tracé dans l'image du document. Afin de tirer avantage de ces deux approches de segmentation, celles-ci sont souvent combinées et donnent lieu à des systèmes mixtes plus complexes mais permettant de contourner les inconvénients de chaque approche [Likforman-Sulem 1994, Likforman-Sulem 1995].

3.2.1.1 Méthodes de segmentation en lignes descendantes

Les méthodes descendantes englobent différents types d'approche, notamment celles basées sur l'étude des histogrammes de projections horizontales des pixels d'écriture [Marti 2001a, Zheng 2005]. Le fait que l'écriture présente une plus forte densité de tracé dans le corps de ligne est ainsi utilisé pour déterminer les points de segmentation optimaux d'un document en lignes [Marti 2001a]. Le principal inconvénient de cette méthode est qu'elle peut ne pas convenir aux documents dont les lignes sont trop fortement inclinées. Pour contourner ce problème, [Arivazhagan 2007] rend la méthode plus robuste en se basant sur des histogrammes de projection locaux. La prise de décision concernant le chemin de segmentation interligne optimal est alors guidée localement rendant la stratégie mixte. Dans [Zheng 2005], les auteurs utilisent des modèles de Markov cachés sur les valeurs de ces histogrammes de projection locaux pour extraire les lignes d'un document. Le décodage est réalisé par un algorithme de Viterbi.

Une approche de segmentation à base de modèles markoviens est également présentée dans les travaux de [Nicolas 2006]. L'auteur utilise des champs conditionnels aléatoires 2D (ou CRF pour *Conditional Random Field*). Les CRF permettent de prendre en compte l'état des pixels voisins et donc l'agencement local des pixels d'écriture. Ainsi, chaque pixel de l'image est classifié comme appartenant à une certaine ligne du document.

Basu et al. ont également développé une méthode descendante robuste à l'inclinaison de l'écriture et au chevauchement reposant sur un concept de zones inondables (espace inter-ligne) et de zones insubmersibles (pixels d'écriture) dans un document manuscrit [Basu 2007]. Quelques paramètres fixés empiriquement permettent de règler l'écoulement de l'eau. Les zones « inon-

dées » sont supprimées en fin de traitement et permettent d'isoler les lignes de texte entre ces zones.

3.2.1.2 Méthodes de segmentation en lignes ascendantes

En ce qui concerne les approches de segmentation dite ascendantes, les relations entre les pixels ou les composantes connexes d'un document sont étudiées. Les méthodes basées sur une segmentation à l'aide du RLSA (pour *Run-Length Smoothing Algorithm*) entrent dans ce cadre [Wong 1982, Nikolaou 2010, Gatos 2010]. Elles cherchent à déterminer l'appartenance de chaque pixel d'écriture à une ligne. Partant du document complet, les *run-lengths* sont calculés pour chacune des lignes de pixels de l'image. Un seuil déterminé empiriquement ou à l'aide d'heuristiques (à partir d'une estimation de la longueur des mots [Wong 1982] par exemple) permet de permuter la valeur de certains pixels appartenant potentiellement à une ligne de texte ou un espace inter-ligne. Ce processus permet d'isoler autant de masses noires qu'il y a de lignes de texte. Une telle méthode reste relativement fiable lorsque le document possède une écriture uniforme et peu inclinée mais ses performances chutent pour des documents dont la taille de l'écriture ou l'inclinaison des lignes est trop variable. Dans [Nikolaou 2010], les auteurs présentent une méthode basée sur un RLSA adaptatif (*ARLSA*) présentant des solutions robustes à ces problèmes de variabilité.

Les méthodes ascendantes peuvent également être basées sur la transformée de Hough et le regroupement de composantes connexes. La transformée de Hough, uniformisée dans [Duda 1972] permet, à partir d'un ensemble de points dans un plan, de déterminer l'inclinaison d'une droite. Dans [Likforman-Sulem 1994, Likforman-Sulem 1995], les auteurs l'utilisent sur les centres de gravité des composantes connexes pour déterminer les meilleurs combinaisons de celles-ci afin d'obtenir un ensemble de lignes suffisamment parallèles entre elles. Les conflits d'alignement générés par ce regroupement des composantes connexes sont gérés de manière globale. Cette méthode peut donc être qualifiée de mixte. [Louloudis 2009] a développé une méthode similaire à la différence du choix des points considérés lors de la transformée de Hough : chaque composante connexe est découpée en trames de largeur uniforme égale à la largeur estimée d'un caractère du document et les centres de gravité de ces trames sont calculés. Ceci permet de fiabiliser les hypothèses d'alignements produits par la transformée de Hough sur ces points.

3.2.1.3 Approche retenue

Le choix d'une méthode de segmentation dépend principalement de la nature plus ou moins complexe des documents à traiter. Dans notre cas, les

documents manuscrits étant non contraints dans un contexte omni-scripteur, l'écriture présente un tracé très variable en taille et en inclinaison. Il nous faut donc une méthode à la fois souple (peu d'hyperparamètres à régler) et robuste aux variations : une approche ascendante visant à regrouper les composantes connexes en alignement a été choisie dans [Koch 2006]. Celle-ci est inspirée des travaux de [Likforman-Sulem 1995].

La segmentation d'une image de document en lignes est réalisée par les étapes suivantes (cf. figure 3.3) :

- les composantes connexes dont la taille est supérieure à un seuil sont d'abord regroupées en lignes potentielles dans une direction horizontale prise en compte par un hyperparamètre α. Le critère de distance suivant est utilisé pour le regroupement :

$$d(a,b) = (x_a - x_b)^2 \times \alpha(y_a - y_b)^2$$

  où a et b sont deux composantes connexes et x et y leur centre de gravité. α est déterminé de manière empirique.
- les alignements produits lors de la première étape sont analysés : deux alignements suffisamment proches sont fusionnés. En effet, plusieurs regroupements peuvent appartenir à une même ligne. Un critère de distance est une nouvelle fois utilisé : il dépend d'une estimation de la taille moyenne des interlignes calculée sur l'ensemble du document.
- enfin, les composantes connexes filtrées lors de la première étape, principalement les signes diacritiques à cause de leur petite taille, sont assignées au regroupement le plus proche.

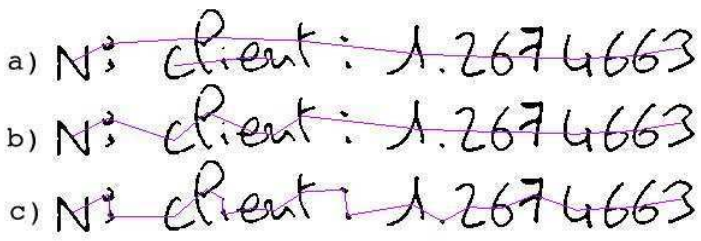


FIG. 3.3: Les trois étapes de la segmentation d'un document en lignes : a) regroupement des composantes connexes suffisamment grandes, b) fusion des alignements trop proches, c) intégration des petites composantes connexes comme les signes diacritiques. Illustration tirée de [Koch 2006].

En fin de segmentation du document, les regroupements restants sont considérés comme des lignes de texte segmentées. Ce processus est illustré par la figure 3.3. Les lignes de textes sont ensuite uniformisées par redresse-

ment de l'écriture et normalisées.

3.2.2 Uniformisation de l'écriture

Afin de réduire la variabilité de l'écriture manuscrite produite par différents scripteurs, il est pertinent de chercher à en uniformiser le tracé. Ceci est notamment réalisé par la correction de l'inclinaison de l'écriture, la normalisation de l'épaisseur du tracé de l'écriture ou encore de la taille des images de lignes.

3.2.2.1 Correction de l'inclinaison de l'écriture

L'écriture manuscrite possède une inclinaison horizontale naturelle[1] correspondant à une inclinaison générale de la ligne par rapport à un support virtuel parfaitement horizontal. Une inclinaison verticale[2] du tracé correspondant à l'inclinaison des caractères par rapport à la direction orthogonale à leur support peut également être observée. La correction de ces inclinaisons entre dans le cadre de la normalisation de l'écriture manuscrite et permet de diminuer sa variabilité. Pour ce faire, deux catégories de méthodes existent. Elles sont illustrées par le schéma de la figure 3.4 où seules les boîtes englobantes des formes manuscrites (mots ou lignes) sont représentées.

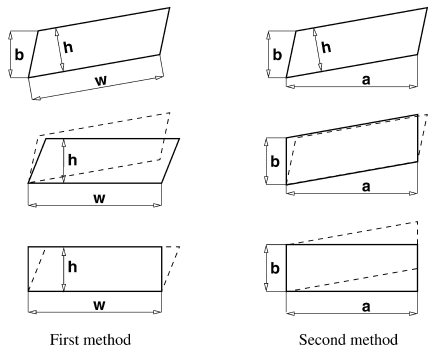


FIG. 3.4: Illustration des deux façons de corriger l'inclinaison de l'écriture manuscrite tirée de [Slavík 2001].

La première catégorie de méthodes consiste à corriger le *slope* dans un premier temps par rotation puis de corriger le *slant* par une méthode de cisaillement horizontal. La détection du paramètre de rotation (son angle) dans la première étape est primordiale. Elle peut se faire par exemple

[1] *slope* en anglais
[2] *slant* en anglais

à l'aide de la détection des lignes de base de l'écriture par l'analyse des histogrammes des directions de *Freeman* [Koch 2006] ou encore à l'aide des transformées de Hough ou de Radon comme dans [Kapoor 2004]. Dans [Espana-Boquera 2011], les auteurs utilisent 4 réseaux de neurones distincts pour corriger l'inclinaison de l'écriture au niveau de chaque pixel par rapport à son voisinage proche. Toutefois, l'apprentissage des différents paramètres du système semble particulièrement lourd et gourmand en temps de calcul. La seconde catégorie de méthodes consiste à corriger en premier lieu le *slant* par une méthode de cisaillement horizontal puis corriger le *slope* par une méthode de cisaillement vertical, les angles d'inclinaison ayant était calculés au préalable. [Slavík 2001] compare les 2 méthodes et montre qu'elles sont équivalentes en termes de performance de reconnaissance. Les deux méthodes introduisant un crénelage dû aux transformées par translation, un lissage du tracé de l'écriture par morphologie mathématique (ouverture, dilatation etc.) est nécessaire.

Le contexte omni-scripteur et la nature hétérogène des documents sur lesquels nous travaillons impliquent des traitements de normalisation visant à uniformiser l'écriture. Il faut donc être prudent quant aux transformations envisagées afin de ne pas introduire de nouveaux facteurs de variation. Les prétraitements les plus classiques de correction des inclinaisons horizontales et verticales sont considérés dans notre système et effectués dans cet ordre [Koch 2006].

Dans un premier temps, la ligne est redressée de façon à ce que celle-ci soit la plus horizontale possible. Une estimation de l'angle d'inclinaison de la ligne par rapport à l'horizontale est calculée et une rotation de l'image de cet angle est effectuée. Le *slope* est ainsi corrigé. Dans un second temps, nous cherchons à évaluer l'inclinaison de l'écriture par rapport à un axe vertical. Les traitements permettant d'effectuer ces corrections sont les suivants :

1. Le chaincode [Kimura 1997] est extrait depuis le contour de l'ensemble des composantes connexes de l'image de la ligne.
2. À partir de l'histogramme des directions de *Freeman* (extrait depuis le chaincode), l'angle d'inclinaison verticale est globalement estimé.
3. Cette inclinaison est corrigée par une méthode de cisaillement correspondant à une translation des lignes de pixels de l'image pour que celle ci soit décalée d'un angle égal à la valeur calculée à l'étape précédente.
4. Des *escaliers* sur les contours des pixels d'écriture sont introduits par le cisaillement. Ils sont atténués et lissés par une ouverture morphologique.

Le *slant* est ainsi corrigé.

3.2.2.2 Normalisation de l'écriture et autres prétraitements

Une fois les lignes d'écriture redressées, il peut être intéressant de normaliser celles-ci en taille. Cette normalisation n'est pas indispensable mais permet de produire des images plus homogènes en proportion. La hauteur des lignes peut donc être normalisée à hauteur fixe [Vinciarelli 2004]. Cette transformation des lignes ne prend pas en compte son contenu et en particulier la présence ou non de hampes et de jambages. Cela peut avoir des conséquences sur la modélisation de l'écriture et donc les performances globales du système.

D'autres techniques de normalisation se basent donc sur la détection des hampes et des jambages avant la reconnaissance des lignes et nécessite d'estimer la position des lignes de base de l'écriture. Ceci peut se faire de différentes manières :

- à l'aide de l'histogramme de projection horizontale des pixels d'écriture, en utilisant le fait que la majorité de l'information manuscrite soit contenue entre les deux lignes de base. Deux seuils sur les données de l'histogramme permettent de déterminer les positions de ces lignes de base [Vinciarelli 2002].
- à l'aide de l'étude des minimas et des maximas locaux. Un perceptron multicouches [Espana-Boquera 2011] ou une transformée de Hough [Kim 1996] peuvent être utilisés dans le but de sélectionner les points les plus à même d'appartenir à ces lignes de base.

Les lignes de base détectées sont alors utilisées pour redimensionner la ligne de texte et pour obtenir des images dont les zones correspondant au corps de texte, aux hampes et aux jambages aient des proportions constantes [Espana-Boquera 2011]. Outre la normalisation de la taille des images de lignes, ces lignes de base peuvent être utiles pour le calcul de caractéristiques discriminantes sur les formes contenues dans la ligne de texte.

L'épaisseur du tracé est également un facteur de variation important dépendant du style d'écriture mais aussi de son vecteur (plume, bille, crayon de bois, etc.). L'uniformisation de cette épaisseur n'est pas une tâche aisée car elle requiert des connaissances relatives au parcours du stylo, parcours complexe voire impossible à récupérer. Cependant, la réduction de cette épaisseur à une largeur d'un pixel a fait l'objet de nombreuses études [Blum 1967, Senior 1998, Elnagar 2003, Chen 2000, Oliveira 2002, Keaton 1997]. On parle alors de squelettisation. Ce prétraitement présente l'avantage de réduire le nombre de pixels d'écriture à son minimum et la rendre plus compacte. Des méthodes à base d'amincissement topologique sont fréquemment utilisées. À l'image de la détection des lignes de base, le squelette d'une forme peut être utile à sa segmentation [Elnagar 2003, Chen 2000] et à l'extraction

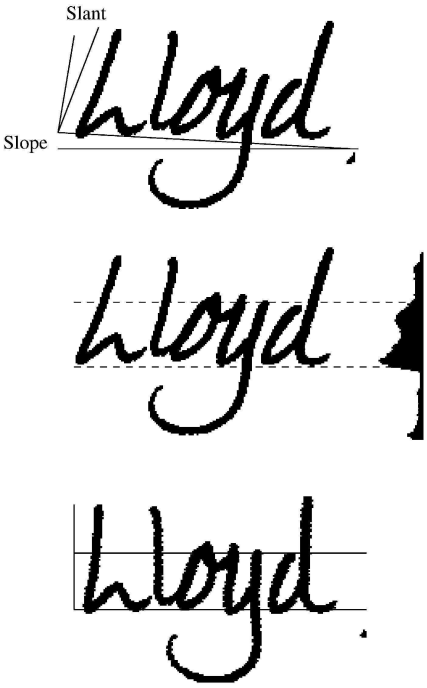


FIG. 3.5: Illustration d'une détection de lignes de base de l'écriture à l'aide de l'histogramme de projection horizontal des pixels d'écriture tirée de [Vinciarelli 2002].

de caractéristiques comme la position et le nombre de points de jonction [Blum 1967, Oliveira 2002].

Dans notre système, la détection des lignes de base de l'écriture constitue le dernier prétraitement. Ces lignes de bases sont caractérisées par deux ensembles de segments de droite et vont être utiles pour le calcul de certaines des caractéristiques. La stratégie de détection de ces lignes repose sur la localisation des minima et maxima locaux des composantes connexes appartenant à la même ligne de texte. Les points correspondant à des minima locaux sont d'abord analysés pour déterminer la position de la ligne de base basse, support virtuel de l'écriture. À partir de cette ligne et en considérant l'ensemble des points correspondant à des maxima locaux, la limite supérieure du corps de ligne est approximée : la ligne de base haute de l'écriture est extraite. Pour plus de détails quant à l'implémentation de cette méthode de détection des lignes de base, nous renvoyons à la lecture de [Koch 2006]

L'étape qui suit l'extraction et la normalisation des lignes de texte est leur reconnaissance. Le module de reconnaissance forme le cœur du système. Nous le présentons dans la section suivante.

3.2.3 Reconnaissance des lignes de texte

La reconnaissance d'une ligne de texte, contrairement à la reconnaissance de mot, ne peut se faire de manière holistique. Elle nécessite une étape de segmentation de la ligne en observations. Cette étape est suivie d'une étape de caractérisation des observations et une dernière étape de décodage de la séquence de vecteur de caractéristiques produite.

3.2.3.1 Segmentation implicite des lignes de texte

La segmentation des lignes en sous-entités est indispensable à leur reconnaissance et peut se faire de manière explicite ou implicite. Nos réflexions, présentées dans le chapitre précedent, ont conduit à modéliser les lignes de texte de manière globale en prenant en compte les modèles des requêtes correspondant à l'information pertinente recherchée dans les documents et un modèle de remplissage permettant d'absorber les portions d'écriture non pertinente. Ces deux modèles sont mis en concurrence dans le modèle de ligne lors du décodage (cf. section 2.5). Une approche reposant sur une segmentation implicite s'est imposée, notamment par la souplesse du choix des points de segmentation des lignes en mots et des mots en caractères. Dans une telle approche, la caractérisation d'une ligne consiste à produire une séquence de vecteurs de caractéristiques extraits depuis les trames de cette ligne. Ces trames sont isolées par une fenêtre glissante. La séquence de vecteurs produite est décodée au regard du modèle de ligne par un classifieur adéquat.

3.2.3.2 Classifieurs de séquences d'observations

Dans la littérature, les réseaux de neurones récurrents [Graves 2009, Graves 2006], les modèles de Markov cachés [Rabiner 1990, Marti 2001a, Vinciarelli 2003], les champs aléatoires conditionnels [Lafferty 2001, Feng 2006] et des combinaisons hybrides [Poisson 2004, Espana-Boquera 2011, Soullard 2011, Do 2011] peuvent être utilisés pour décoder des séquences d'observations produites par une segmentation implicite. Contrairement aux MMC, les réseaux de neurones récurrents sont complexes à dimensionner et entrainer. Leur utilisation nécessite d'être combinés avec un classifieur complexe à paramétrer tel que le CTC de [Graves 2006] par exemple. Les MMC possédent un algorithme d'apprentissage efficace et embarqué des modèles de caractères : l'algorithme de *Baum-Welch* [Rabiner 1993]. Bien qu'ils font l'hypothèse forte d'indépendance des observations, contrairement aux CRF qui ne permettent pas d'intégrer facilement des connaissances lexicales, les MMC ont déjà été utilisés avec succès en reconnaissance de l'écriture manuscrite [Plamondon 2000, Lorette 1999]

mais aussi dans des domaines variés où la modélisation est un point clé tels que le traitement de la parole continue [Young 1989, Rabiner 1990] et le séquencement de l'ADN en biologie [Krogh 1994] par exemple. Notre choix s'est orienté dans un premier temps vers les MMC [Rabiner 1990], modèles séquentiels probabilistes. Ils permettent en effet d'intégrer facilement des connaissances linguistiques, notamment des lexiques [Koerich 2003, Vinciarelli 2002] et des modèles de langage de grandes tailles [Zimmermann 2006, Vinciarelli 2004] mais aussi des modèles de remplissage pour l'extraction d'information [Hobbs 2010, Gu 2006, El-Yacoubi 2002, Fischer 2010].

Le processus de décodage d'une ligne au regard d'un modèle de ligne construit par la concaténation de MMC de caractères est illustré par la figure 3.26.

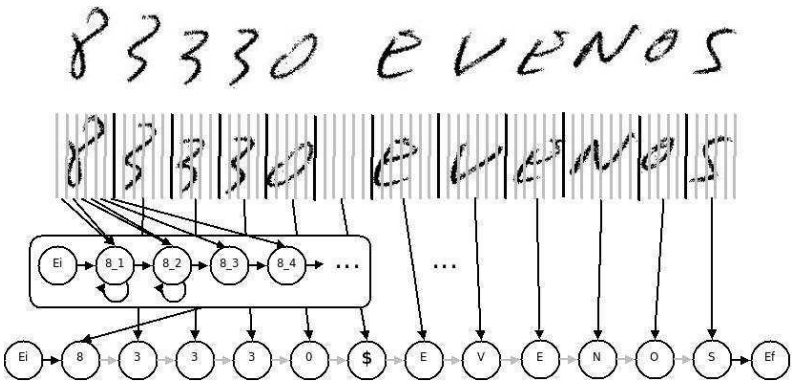


FIG. 3.6: Illustration d'un alignement d'une image de ligne avec le modèle MMC correspondant à sa transcription. $ représente le modèle d'espace.

3.2.3.3 Caractérisation des trames

Dans la littérature, de nombreux systèmes intègrent une approche à base de segmentation implicite [Marti 2001b, Zimmermann 2004b, Bertolami 2006, Vinciarelli 2003, Fischer 2010, Ait-Mohand 2011, Rodriguez 2008] et traitent donc de la caractérisation des trames. Les choix de modélisation de ces trames sont nombreux et de nature variée. En voici quelques exemples, parmi les plus communément utilisés :

– le *zoning* : il consiste à calculer différents types de données (densités de pixels, configurations, orientations, etc.) dans $n \times m$ régions de l'image prédéfinies par une fenêtre de taille fixe [Vinciarelli 2004]. Ceci permet de cartographier les images suivant leurs régions d'intérêts contenant de l'information (de l'écriture).

- les moments invariants : ils possèdent la propriété d'être invariants par rapport à la taille, aux proportions, à la position et à l'orientation des zones d'intérêt contenues dans une image. Ce sont des mesures statistiques de la distribution des pixels autour du centre de gravité des formes présentes dans l'image. Initialement présentés dans [Hu 1962], ils ont été étendus dans de nombreux travaux et ce dans des domaines variés [Flusser 2000].
- les contours : ils renferment la totalité de l'information présente dans l'image puisqu'il s'agit des frontières entre les pixels blancs du fond de l'image et les pixels noirs de la forme. Une des façons efficaces d'utiliser les contours est de calculer leur histogramme des directions de Freeman comme par exemple dans [Kimura 1994] avec son *chaincode* extrait depuis des zones de la trame. Les différentes valeurs normalisées des histogrammes forment l'ensemble des caractéristiques considérées.
- les transformées et développements en séries (Fourier, Hough, ondelettes par exemple) : les caractéristiques considérées sont directement les valeurs des coefficients de la transformée de Fourier des pixels d'écriture de l'image. Cela permet d'obtenir une représentation fréquentielle de l'image. Le premier coefficient définit l'allure générale de la forme, le second est considéré comme une mesure d'ellipticité de la forme, etc. Ces caractéristiques sont par exemple utilisées dans [Espana-Boquera 2011] ou [Rodriguez 2008]

À titre d'exemple, [Rodriguez 2008] propose un vecteur de caractéristiques extraites depuis une fenêtre découpée en 4×4 cellules. Ces caractéristiques correspondent à des histogrammes de gradients locaux. Pour chaque cellule, 16 valeurs sont utilisées pour discrétiser les histogrammes et donc 256 caractéristiques sont finalement extraites. Ce vecteur permet de modéliser finement l'orientation générale du tracé représenté par les pixels d'écriture dans chacune des cellules de la fenêtre. Le processus complet d'extraction est illustré par la figure 3.7.

La figure 3.8 présente deux jeux de caractéristiques extraits depuis une fenêtre glissante. Dans [Marti 2001b], les auteurs combinent des caractéristiques structurelles et statistiques avec succès : un vecteur à 9 caractéristiques est extrait pour chaque trame de 1 pixel de large. Ce vecteur permet de modéliser de manière compacte la répartition des pixels d'écriture dans une colonne de la ligne. [Vinciarelli 2003] propose un vecteur à 16 caractéristiques basé sur la densité des pixels d'écriture dans chacune des cellules de la fenêtre découpé

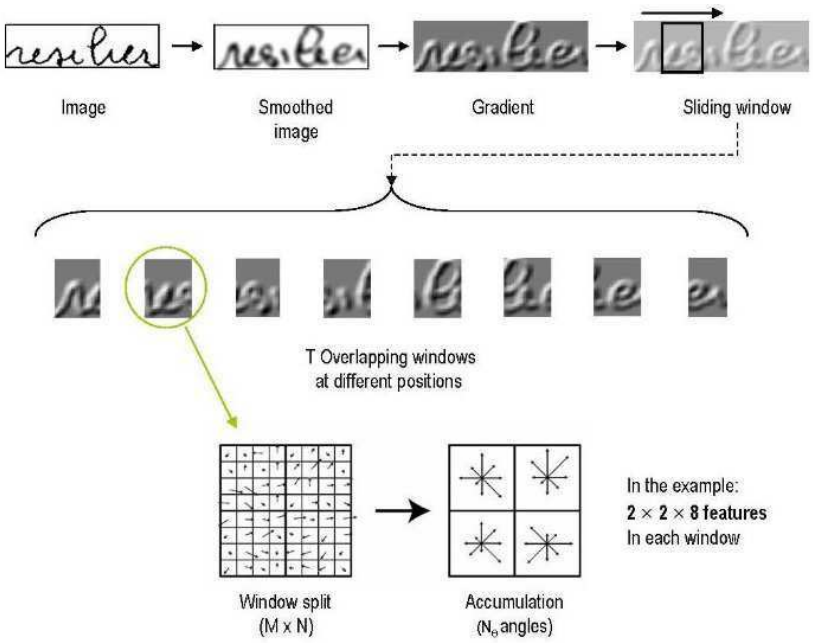


FIG. 3.7: Illustration du processus d'extraction des histogrammes de gradient [Rodriguez 2008].

en 4×4 cellules. Ceci permet de cartographier la répartition des pixels dans la fenêtre.

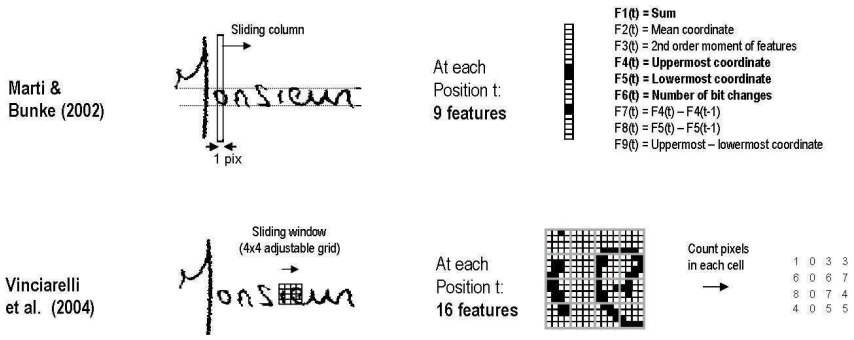


FIG. 3.8: Illustration de deux jeux de caractéristiques fréquemment combinés à une approche de segmentation implicite à base de fenêtre glissante [Marti 2001b, Vinciarelli 2003].

Il est à noter que ces vecteurs peuvent être extraits à partir d'une fenêtre *cropée* [Vinciarelli 2003] ou non [Marti 2001a, Rodriguez 2008] autour des pixels d'écriture de la fenêtre. Un *crop* consiste à supprimer les plages de pixels de fond en dessous et au dessus de la forme contenue dans la fenêtre. Dans [Rodriguez 2008], les auteurs montrent que les différences en reconnaissance sont relativement faibles sous la contrainte d'avoir suffisamment d'échantillons d'apprentissage. Dans les deux cas, des caractéristiques extraites à partir des lignes de base permettent de prendre en compte la position de l'écriture dans la trame.

Nous travaillons avec des MMC de caractères intégrant des mélanges de gaussiennes qui sont difficiles à estimer quand la dimension augmente. Les formes à modéliser doivent toutefois être suffisamment bien représentées par des caractéristiques choisies dans le but de discriminer ces formes. Une combinaison de caractéristiques structurelles et statistiques forme généralement un bon compromis [Heutte 1998, Chatelain 2006b, Koch 2006]. Elle a par exemple été utilisée avec succès dans des systèmes à base de MMC en reconnaissance de l'écriture [Marti 2001a, Vinciarelli 2004] mais également en extraction d'information [Fischer 2010]. De dimension très raisonnable (9 primitives), le vecteur de caractéristiques de [Marti 2001a, Vinciarelli 2004, Fischer 2010] est extrait à partir d'une fenêtre de 1 pixel de largeur uniquement. Ceci implique des MMC avec de nombreux états pour modéliser au mieux les caractères. A contrario, [Rodriguez 2008] utilise un vecteur de 256 caractéristiques extrait depuis une fenêtre plus large. Il travaille donc avec des MMC de caractères plus compacts.

Vecteur de caractéristiques retenu

Notre choix de vecteur de caractéristiques s'est donc orienté vers le vecteur présenté dans [Al-Hajj 2007]. Il s'agit d'un vecteur intégrant des caractéristiques statistiques et structurelles dont une partie dépend des positions des lignes de base. Ce vecteur est intégré dans un système de reconnaissance de mots manuscrits isolés [Kessentini 2010] ayant notamment très bien figuré à la compétition de reconnaissance de mots manuscrits d'ICDAR 2009 [Grosicki 2009].

Le vecteur de caractéristiques retenu est de dimension $20 + d$, d étant la largeur de la fenêtre glissante. 17 de ces caractéristiques sont relatives à la position des lignes de base permettant de détecter facilement la présence de hampes et de jambages, $3 + d$ caractéristiques sont indépendantes des lignes de base. En voici le détail :

f1 caractérise la densité des pixels d'écriture dans la trame.

f2 caractérise l'organisation générale de l'écriture dans la trame. Chaque trame est divisée verticalement en $\frac{h}{4}$ cellules où h est la hauteur de la trame en pixels. $f2$ représente le nombre de transitions Noir/Blanc entre ces cellules.

f3 caractérise la différence de position entre les centres de gravité des pixels d'écriture dans deux trames consécutives.

f4 caractérise la position verticale normalisée du centre de gravité des pixels d'écriture par rapport à la ligne de base basse.

f5-f6 caractérisent la densité des pixels d'écriture au dessus et en dessous de la ligne de base basse.

f7 caractérise le nombre de transitions Noir/Blanc entre les cellules situées au dessus de la ligne de base basse.

f8 représente la zone à laquelle appartient le centre de gravité de l'écriture dans la fenêtre (zone supérieure $f8 = 1$, zone médiane $f8 = 2$, zone inférieure $f8 = 3$).

f9 - f13 caractérisent la manière dont les pixels d'écriture sont agencés dans la trame. Les configurations locales des pixels d'écriture sont utilisées à cet effet. Ces configurations sont illustrées par la figure 3.9 et ces caractéristiques représentent les proportions de chacune des 5 configurations répertoriées.

f14 - f18 sont caractérisées par les configurations locales des pixels d'écriture situés dans la zone médiane séparée par les lignes de base haute et basse.

f19 - f19+d caractérisent enfin les densités de pixels noirs dans chaque colonne de la trame. Elles permettent de renseigner sur la répartition des pixels de l'écriture sur la largeur de la trame.

L'hyperparamètre d ainsi que le recouvrement o entre deux fenêtres successives sont fixés avant l'apprentissage des MMC sur une base de validation.

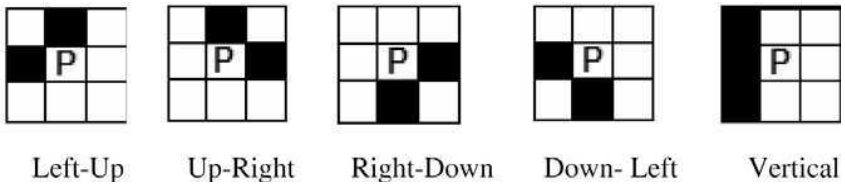


FIG. 3.9: Illustration des différentes configurations locales des pixels pour le calcul des caractéristiques $f9$ à $f18$. P représente le pixel d'écriture courant du contour de la forme contenue dans sa trame.

À l'issue de la phase de caractérisation des trames, une séquence de vecteurs de caractéristiques est obtenue. La phase de reconnaissance consiste à décoder ces séquences sous la contrainte de notre modèle de ligne.

3.2.3.4 Décodage des lignes de texte

Classiquement, le décodage des lignes de texte est contraint par un lexique de mots pour contraindre les séquences de caractères entre les espaces et d'un modèle de langage de type bigramme de mots pour contraindre l'enchainement des mots dans la ligne de texte. Le problème de la reconnaissance de séquences de mots s'écrit alors suivant l'équation 3.1 où $\hat{L}$ représente la meilleure séquence de mots par rapport à la séquence d'observations (de trames) O :

$$\hat{L} = \arg\max_{L}\{P(L|O)\} \tag{3.1}$$

$$= \arg\max_{L}\left\{\frac{P(O|L)P(L)}{P(O)}\right\} \tag{3.2}$$

$$= \arg\max_{L}\{P(O|L)P(L)\} \tag{3.3}$$

où les différents termes sont :

- $P(O|L)$, la probabilité d'observer la séquence O sachant la séquence de mots L
- $P(L)$, la probabilité a priori de la séquence de mots L
- $P(O)$, la probabilité a priori d'observer la séquence O

Le terme $P(O)$ est difficile à estimer. Puisqu'il est constant, il n'affecte pas la recherche de la meilleure séquence de mots par rapport aux observations et peut donc être supprimé des calculs. Dans les équations 3.2 et 3.3, l'information $P(L)$ représente la probabilité *a priori* d'une séquence de mots. Celle-ci est donc définie par un modèle de langage et donc par notre modèle de ligne déjà présenté. Il contraint la reconnaissance dans une optique d'extraction d'information.

Le terme $P(O|L)$ est quant à lui estimé à l'aide des MMC. En reconnaissance de l'écriture manuscrite, l'algorithme de Viterbi [Rabiner 1990] est utilisé pour déterminer la meilleure séquence de modèles de caractères sous la contrainte d'un lexique n'autorisant que certaines séquences correspondant à des mots. Des algorithmes basés sur Viterbi ont été étendus aux problématiques plus spécifiques de la reconnaissance de séquences de mots. Ceux-ci utilisent notamment la structure des langues : la concaténation de caractères appartenant à un alphabet permet de former des mots qui eux-mêmes concaténés

deux à deux permettent de former des phrases. Ces méthodes de recherche de meilleures séquences de modèles sont optimales. Elles permettent de diminuer la complexité de l'espace de recherche tout en garantissant le même niveau de performances et un coût computationnel réduit [Moore 2002]. Ces algorithmes de décodage peuvent être répartis en trois catégories [Rabiner 1993] :

- *Two level Dynamic Programming Algorithm* : il procède en deux passes. Dans un premier temps, tous les mots du lexique sont comparés avec toutes les portions de la séquence de trames à reconnaitre. Dans un second temps, le treillis de mots précédemment formé est parcouru et la phrase ou séquence la plus probable au regard des MMC appris est isolée [Rabiner 1990]. L'utilisation d'un modèle de langage de type bigrammes de mots permet d'affiner la recherche de la solution optimale lors du parcours du treillis.
- *Level Building Algorithm* : il procède en niveaux. Dans un premier temps, les meilleurs mots candidats sont recherchés pour le début de la séquence, puis pour chaque hypothèse, les meilleurs candidats pour la suite de la séquence sont recherchés. Ceci est réalisé jusqu'à ce que toutes les trames soient parcourues [Myers 1981]. L'utilisation d'un modèle de langage permet d'apporter un supplément d'information quant aux transitions entre chaque niveau lors du décodage.
- *One Pass Algorithm* : il cherche à cartographier trame par trame la séquence à reconnaitre avec les formes du lexique. Il est aussi appelé *frame synchronous level building algorithm* car la partie calculatoire sur la séquence à tester se fait de manière synchrone avec les trames. De nombreux algorithmes de décodage sont dérivés de celui-ci comme par exemple le *token passing algorithm* [Young 1989] implémenté dans la *toolbox HTK* [Young 1999] ou encore le plus connu *Beam Search decoding* [Moore 2002] implémenté dans la *toolbox Torch* [Collobert 2002].

Étant donné la quasi généralisation de l'utilisation du troisième type d'algorithme et la disponibilité d'implémentations dans des *toolboxes* accessibles à tous, nous nous sommes orientés vers un algorithme de type *One Pass*. Dans des conditions expérimentales équivalentes, l'algorithme introduit dans [Moore 2002] présente des résultats équivalents en termes de performances mais plus rapide que son homologue implémenté dans la *toolbox HTK* [Young 1989]. Nous avons donc choisi d'utiliser cet algorithme, décrit ci-après par l'algorithme de la figure 3.10.

Soit une séquence d'observations à décoder $X = \{x_1, x_2, ..., x_n, ...x_N\}$ extraite depuis une image de ligne à reconnaitre. Soit $p(w_{v'}|w_v)$ un modèle de langage définissant les probabilités de transition entre deux mots d'indices v et v' d'un lexique de V mots. Un mot w_v est constitué par un ensemble d'états

$s_{i,v}$ dont l'agencement est défini par la topologie des MMC de caractères et donc du mot. On note $w_v = \{s_{e,v}, ..., s_{i,v}, ..., s_{f,v}\}$ où $s_{e,v}$ et $s_{f,v}$ sont des états structurels, non-émetteurs marquant le début et la fin du modèle du mot w_v.

Si l'hypothèse d'une transition vers l'état $s_{i',v}$ est faite à l'instant $n-1$ alors l'état $s_{i',v}$ va émettre l'observation x_n à l'instant n avec une certaine probabilité (calculée à l'aide du modèle de mélanges de Gaussiennes associé à cet état) et transiter vers un nouvel état $s_{i,v}$ selon la topologie du MMC du mot w_v.

Étant données ces notations, l'équation de récurrence pour des transitions à l'intérieur d'un modèle de mot est donnée par :

$$Q(x_n, s_{i,v}) = \max_{i'}\{p(x_n, s_{i,v}|s_{i',v}).Q(x_{n-1}, s_{i',v})\} \tag{3.4}$$

où $Q(x_{n-1}, s_{i',v})$ est le score accumulé jusqu'à l'état $s_{i',v}$ et $p(x_n, s_{i,v}|s_{i',v})$ est la probabilité jointe d'émettre l'observation x_n quand on est dans l'état $s_{i',v}$ puis de transiter vers l'état $s_{i,v}$.

L'équation de transition entre deux modèles de mot est ensuite donnée par :

$$Q(x_n, s_{e,v'}) = \max_{v}\{p(w_{v'}|w_v).Q(x_n, s_{f,v})\} \tag{3.5}$$

où $p(w_{v'}|w_v)$ définit la probabilité de transiter vers le mot $w_{v'}$ à partir du mot w_v. $s_{f,v}$ et $s_{e,v'}$ étant non-émetteurs, il n'y a pas d'émission d'observation lors de cette transition.

Enfin, l'équation de transition entre l'état initial du mot suivant et ses successeurs potentiels selon la topologie de ce modèle de mot se produit de manière similaire :

$$Q(x_n, s_{i,v'}) = \max_{i}\{p(s_{i,v'}|s_{e,v'}).Q(x_n, s_{e,v'})\} \tag{3.6}$$

Une fois encore, étant donnée la nature structurelle de l'état s_e, v, aucune observation n'est émise.

3.2.4 Apprentissage du système complet

Les modules de traitement permettant de détecter de l'information quelconque dans des images de document manuscrit ont été présentés dans les

Entrée : $X = \{x_1, x_2, ..., x_N\}$ la séquence d'observations à décoder, un lexique de séquences $W = \{w_1, w_2, ..., w_V\}$ avec $w_v = \{s_{e,v}, ..., s_{i,v}, ..., s_{f,v}\}$ et un modèle de langage caractérisant les probabilités de transition $p(w_{v'}|w_v)$ du mot w_v vers le mot $w_{v'}$
Sortie : séquence de mots optimale
DÉBUT
pour chaque observation x_n, $1 \leq n \leq N$ **faire**
pour chaque mot w_v du lexique, $1 \leq v \leq V$ **faire**
pour chaque état $s_{i,v}$ (sauf le premier) du mot w_v, $s_{e,v} < s_{i,v} \leq s_{f,v}$, i' est un indice sur les états suivants de i dans le modèle du mot w_v **faire**
Calculer 3.4, le score accumulé jusqu'à la trame n alors que l'on se trouve à l'intérieur du modèle du mot w_v :

$$Q(x_n, s_{i,v}) = \max_{i'}\{p(x_n, s_{i,v}|s_{i',v}).Q(x_{n-1}, s_{i',v})\}$$

fin pour
pour chaque état initial $s_{e,v'}$ du mot $w_{v'}$, v' est un indice sur les mots du lexique potentiellement successeur du mot w_v **faire**
Calculer 3.5, le score accumulé après avoir transité dans un nouveau modèle de mot $w_{v'}$:

$$Q(x_n, s_{e,v'}) = \max_{v}\{p(w_{v'}|w_v).Q(x_n, s_{f,v})\}$$

pour tous les états $s_{i,v}$ suivants de $s_{e,v}$, i est un indice sur les états suivants de l'état initial du mot w_v **faire**
Calculer 3.6, le score accumulé après avoir émis le premier symbole du modèle du mot $w_{v'}$:

$$Q(x_n, s_{i,v'}) = \max_{i}\{p(s_{i,v'}|s_{e,v'}).Q(x_n, s_{e,v'})\}$$

fin pour
fin pour
fin pour
fin pour
FIN

FIG. 3.10: *Time-Synchronous Beam Search* d'après [Moore 2002].

sections précédentes. Dans cette section, nous nous attardons sur l'apprentissage du système complet et de ses paramètres. Cela se traduit par :

- le réglage des hyperparamètres de la fenêtre glissante pour la segmentation des images de ligne en séquences d'observations
- l'apprentissage des paramètres des modèles de Markov cachés et des modèles de mélanges de Gaussiennes qu'ils intègrent
- l'apprentissage des transitions du modèle de remplissage et du modèle de ligne

Nous présentons ces trois points dans cette section.

3.2.4.1 Réglages des hyperparamètres

Soient d et o les hyperparamètres correspondant respectivement à la largeur de la fenêtre glissante utilisée pour l'extraction des caractéristiques et au recouvrement entre deux fenêtres successives. Bien que la largeur des différents caractères soit potentiellement très variable, un i étant très souvent moins large qu'un m par exemple, nous avons fait le choix de considérer des modèles ayant toujours le même nombre d'états dans un souci de simplicité d'implémentation. Il existe des méthodes efficaces d'adaptation de la structure des MMC qui peuvent être utilisées afin d'affiner la modélisation des modèles de caractères. Nous renvoyons le lecteur aux travaux de [Ait-Mohand 2011]. Pour le couple (d, o) définissant la fenêtre glissante et donc le nombre d'états dans un modèle de caractère, un compromis entre rapidité et performance doit être trouvé. La largeur de la fenêtre doit être suffisamment fine pour bien décrire les caractères mais pas trop pour être rapide en décision. Un recouvrement entre deux fenêtres successives n'est pas nécessaire [Marti 2001a] mais très souvent considéré [Vinciarelli 2004, Koerich 2002b]. Il permet de prendre en compte de manière redondante une partie du tracé de l'écriture contenue dans la fenêtre et donc introduire un certain contexte entre deux fenêtres successives.

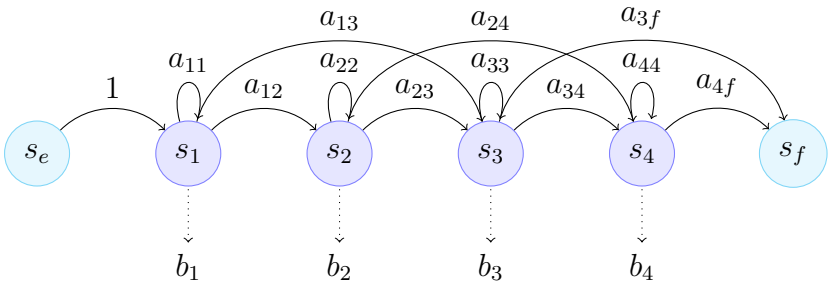


FIG. 3.11: MMC de caractère à 4 états émetteurs. s_e et s_f sont des états structurels de début et de fin, s_i les états émetteurs, a_{ij} les transitions et b_i les symboles émis.

Ces hyperparamètres ont été réglés sur la base de données Ironoff chèques

[Viard-Gaudin 1999] comprenant des images de mots isolés provenant de chèques bancaires (lexique de 23 mots). Le nombre d'états émetteurs pour les MMC de caractères a été fixé à 4 ce qui est un bon compromis entre représentation des formes et rapidité en décision. Le modèle retenu est illustré par le schéma de la figure 3.11. Dans cette configuration, le couple (d, o) ayant obtenu les meilleurs résultats sur une tâche de reconnaissance de mots isolés est le couple $(8, 5)$ avec 91% de taux de reconnaissance. Le nombre de gaussiennes par état est lui fixé à 5 dans un souci de rapidité. Le choix du couple d'hyperparamètres (d, o) a également été validé sur une seconde base de données de séquences isolées (mots et séquences numériques) afin de vérifier la robustesse du système aux changements de base. Une base de données interne de 6000 images de séquences isolées a été utilisée à cet effet. Avec un lexique de 100 séquences tirées aléatoirement parmi les étiquettes des exemples de la base, le système atteint 79% de taux de reconnaissance, 93% en *Top 5* et 97% en *Top 10*.

3.2.4.2 Apprentissage des modèles de caractères

L'apprentissage des modèles de caractères est effectué à l'aide de l'algorithme de Baum-Welch [Baum 1966]. Il consiste à déterminer les paramètres des MMC à partir d'un ensemble de séquences d'observations étiquetées. Son but est de maximiser la probabilité que les modèles (construits à partir des annotations) aient émis les séquences d'observations d'apprentissage. À cet effet, un critère de maximum de vraisemblance est fréquemment utilisé [Rabiner 1990]. Une des propriétés intéressantes d'un tel apprentissage réside dans le fait que tous les modèles de caractères sont appris conjointement et itérativement : pour une image de ligne donnée, un modèle de Markov caché est construit à partir de sa transcription et est aligné sur la séquence d'observations correspondante comme illustré sur la figure 3.12. Tous les modèles de caractères étant appris de manière embarquée, nous nous affranchissons de la délicate et fastidieuse tâche de la segmentation des images en sous entités (mots et caractères). Nous avons finalement considéré $N_c = 71$ modèles de caractères :

- 26 caractères minuscules : *a, b, ..., z*
- 26 caractères majuscules : *A, B, ..., Z*
- 10 chiffres : *0, 1, ..., 9*
- 1 modèle d'espace noté $
- 4 caractères accentués : *è, é, ê, à*
- 4 séparateurs ou signes de ponctuation : *.,-,',/*

Notons que certains caractères de la base d'apprentissage trop peu présents ont été fusionnés au sein d'autres modèles. C'est le cas par exemple de la

virgule fusionnée avec le point ou encore du *ë* fusionné avec le *ê*.

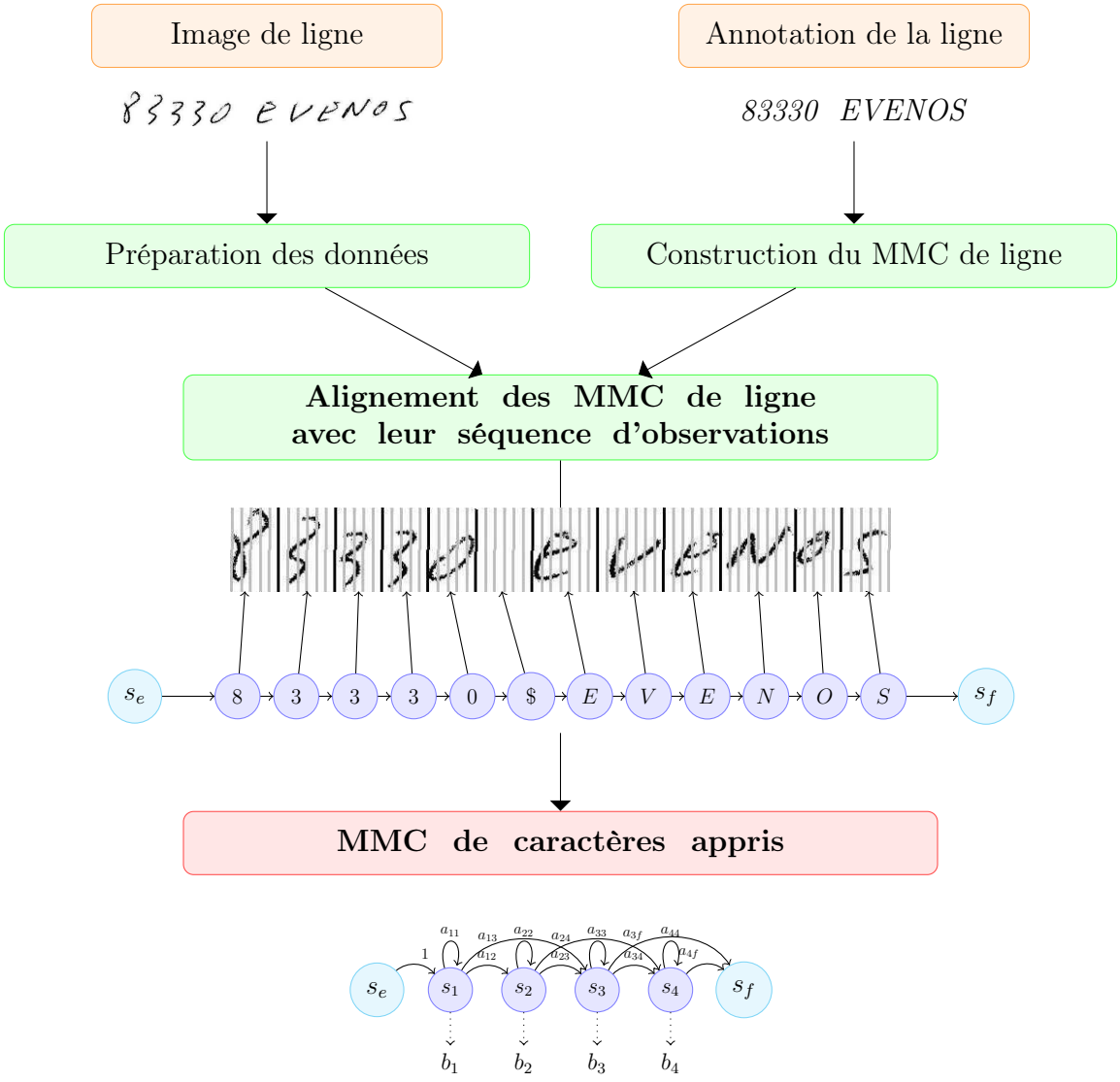


FIG. 3.12: Illustration de l'apprentissage des MMC de caractères.

3.2.4.3 Apprentissage du modèle de détection d'information

L'apprentissage du modèle de ligne pour la détection d'information a été réalisé sur la base d'apprentissage RIMES [Grosicki 2009] que nous présenterons en détail dans la section suivante. En supposant les MMC de caractères appris et étant donné le modèle présenté dans la section 2.5, seules les transitions au sein du modèle de ligne sont à régler. Il va donc s'agir de fixer les transitions entre les différents types d'information du modèle de ligne

c'est à dire entre le modèle d'espace, les caractères du modèle de remplissage et de requêtes.

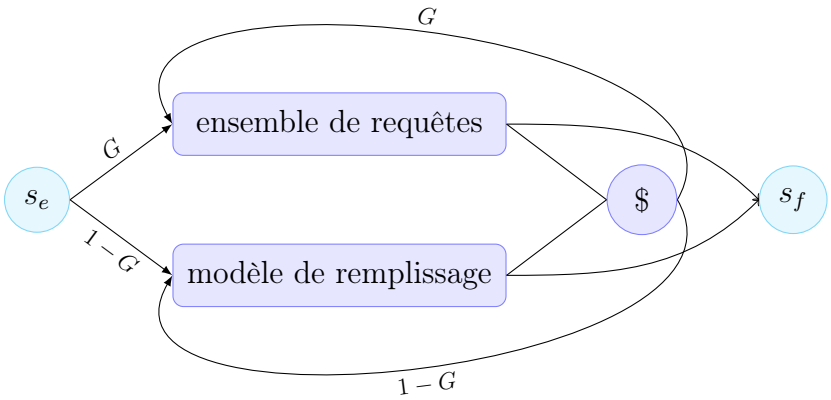


FIG. 3.13: Modèle de ligne pour la détection d'information dans des documents manuscrits non contraints présenté dans la section 2.5.

Le modèle de ligne est présenté de nouveau sur le schéma de la figure 3.13. Les transitions de ce modèle peuvent être représentées sous une forme matricielle, comme sur la figure 3.14.

Les entrées de cette matrice (cf. figure 3.14) correspondent aux transitions du modèle de ligne et le caractérisent de manière complète.

- $\boldsymbol{\alpha_m}$ représente la transition de l'état initial s_e ou du modèle d'espace \$ vers le mot W_m du lexique
- $\boldsymbol{\beta_r}$ représente la transition de l'état initial ou du modèle d'espace vers le caractère E_r du modèle de remplissage
- $\boldsymbol{\gamma_{rr'}}$ représente la transition au sein du modèle de remplissage entre les caractères E_r et $E_{r'}$
- $\boldsymbol{\delta_m}$ représente la transition du mot W_m vers le modèle d'espace \$
- $\boldsymbol{\epsilon_m}$ représente la transition du mot W_m vers l'état final s_f du modèle de ligne
- $\boldsymbol{\zeta_r}$ représente la transition du caractère E_r du modèle de remplissage vers le modèle d'espace \$
- $\boldsymbol{\eta_r}$ représente la transition du caractère E_r du modèle de remplissage vers l'état final s_f du modèle de ligne
- **les autres transitions**, n'apparaissant pas sur le modèle de ligne (figure 3.13), sont interdites et donc fixées à 0. Elles caractérisent un enchai-

$$\left(\begin{array}{c|ccccc|ccccc|cc}
0 & \alpha_1 & \cdots & \alpha_m & \cdots & \alpha_M & \beta_1 & \cdots & \beta_r & \cdots & \beta_R & 0 & 0 \\
\hline
0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \delta_1 & \epsilon_1 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \delta_m & \epsilon_m \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \delta_M & \epsilon_M \\
\hline
0 & 0 & \cdots & 0 & \cdots & 0 & \gamma_{11} & \cdots & \gamma_{r1} & \cdots & \gamma_{R1} & \zeta_1 & \eta_1 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & \cdots & 0 & \gamma_{1r} & \cdots & \gamma_{rr} & \cdots & \gamma_{Rr} & \zeta_r & \eta_r \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & \cdots & 0 & \gamma_{1R} & \cdots & \gamma_{rR} & \cdots & \gamma_{RR} & \zeta_R & \eta_R \\
\hline
0 & \alpha_1 & \cdots & \alpha_m & \cdots & \alpha_M & \beta_1 & \cdots & \beta_r & \cdots & \beta_R & 0 & 0 \\
\hline
0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0
\end{array}\right)$$

FIG. 3.14: Modèle de ligne pour l'extraction d'information représenté sous forme matricielle. Les lignes et les colonnes représentent, de gauche à droite ou de haut en bas, l'*état initial* de la ligne s_e, les mots du lexique $W_1, ..., W_m, ..., W_M$, les caractères du modèles de remplissage $E_1, ..., E_r, ..., E_R$, le modèle d'espace \$ et l'*état final* s_f.

nement impossible entre deux entités du modèle global de ligne. Il est impossible de transiter d'un mot vers le modèle de rejet (et inversement) sans passer par le modèle d'espace.

Étant données ces notations et le modèle graphique d'extraction représenté sur la figure 3.13, nous pouvons les expliciter. Si l'on souhaite avoir un modèle probabiliste, ce qui est notre cas, la somme des éléments d'une ligne doit être égale à 1. Ceci revient à normaliser les transitions de la matrice.

En ce qui concerne les α et β, nous avons :

$$\sum_m \alpha_m + \sum_r \beta_r = 1 \tag{3.7}$$

Or α_m est caractérisé par l'hyperparamètre G que l'on pondère par $p(W_m)$, la probabilité *a priori* d'observer le mot W_m par rapport aux M mots du lexique. Étant donnée la variabilité en contenu de la base d'apprentissage,

cette probabilité est assimilée à $\frac{1}{M}$, ce qui donne :

$$\alpha_m = \frac{G}{M} \tag{3.8}$$

En remplacant dans l'équation 3.7, on obtient :

$$\sum_r \beta_r = 1 - M \times \frac{G}{M} \tag{3.9}$$

$$= 1 - G \tag{3.10}$$

Pour que l'équation 3.7 somme à 1, on normalise les termes β_r par la probabilité *a priori* $p(E_r)$ d'observer le caractère correspondant E_r du modèle de remplissage. On obtient alors :

$$\beta_r = \frac{1-G}{p(E_r)} \tag{3.11}$$

En ce qui concerne les entités δ et ϵ de la matrice globale, on doit résoudre le système suivant :

$$\begin{cases} \delta_1 + \epsilon_1 = 1 \\ \cdots \\ \delta_m + \epsilon_m = 1 \\ \cdots \\ \delta_M + \epsilon_M = 1 \end{cases} \tag{3.12}$$

Or ϵ_m, $m \in \{1, M\}$ représente la transition d'un mot du lexique vers l'état final du modèle de ligne. Il peut donc être apparenté à la probabilité d'observer le dernier mot de la ligne et donc être estimé par l'inverse du nombre moyen de mots par ligne N_{mpl}. Soit :

$$\epsilon_m = \frac{1}{N_{mpl}} \tag{3.13}$$

Ce qui nous donne pour δ_m :

$$\delta_m = 1 - \epsilon_m = \frac{N_{mpl} - 1}{N_{mpl}} \tag{3.14}$$

En ce qui concerne les transitions depuis un caractère du modèle de rem-

plissage (entrées de la matrice notées γ, ζ et η), le système suivant est à considérer :

$$\begin{cases} \zeta_1 + \eta_1 + \sum_{r'-\{\$\}} \gamma_{1r'} &= 1 \\ \cdots \\ \zeta_r + \eta_r + \sum_{r'-\{\$\}} \gamma_{rr'} &= 1 \\ \cdots \\ \zeta_R + \eta_R + \sum_{r'-\{\$\}} \gamma_{Rr'} &= 1 \end{cases} \tag{3.15}$$

Les entrées $\gamma_{rr'}$ de la matrice 3.14 caractérisent les transitions à l'intérieur du modèle de remplissage. Nous rappelons qu'elles doivent être fixées de manière à modéliser l'information à rejeter (cf. section 2.5). Il peut s'agir d'un modèle :

- purement ergodique : les transitions entre les modèles de caractères sont toutes identiques et fixées à $1/R$ où R représente le nombre de modèles de caractères
- appris sous la forme d'un modèle de langage de type bigrammes de lettres :
  - sur un corpus de texte de grande taille (en libre accès sur internet par exemple) pour être très représentatif de la langue considérée
  - sur les documents ayant servi à l'apprentissage des modèles de caractère pour être représentatif des données utilisées

Dans ses travaux sur la détection de mots hors vocabulaire en reconnaissance de la parole [Manos 1997], Manos montre que la représentation la plus proche des données à traiter permet d'atteindre de meilleurs résultats. Pour mieux coller aux courriers manuscrits (présence de mots, de séquences numériques et alphanumériques) nous avons également choisi d'apprendre les transitions du modèle de remplissage sur nos données d'apprentissage. Soit $p(E_r|E_{r'})$ la probabilité d'observer le bigramme composé des lettres $E_{r'}$ et E_r. Pour résoudre le système d'équations 3.15, nous pouvons nous servir du fait que la somme des transitions d'un caractère donné vers tous les autres caractères de l'alphabet (espace compris) est égale à la probabilité *a priori* d'observer ce caractère, ce qui revient à écrire :

$$\sum_r p(E_r|E_{r'}) = p(E_{r'}) \tag{3.16}$$

L'équation 3.16 est équivalente à :

$$\sum_r \frac{p(E_r|E_{r'})}{p(E_{r'})} = 1 \tag{3.17}$$

Dans le système 3.15, les termes $\zeta_r + \eta_r$ représentent la probabilité de finir un

mot n'appartenant pas au lexique par le caractère d'indice E_r du modèle de remplissage, le premier terme désignant un mot quelconque de la ligne et le second désignant le dernier mot. En décomposant la somme de l'équation 3.17 pour en sortir l'espace que l'on décompose ensuite en \$ et s_f pour respecter notre modèle de ligne, il vient :

$$\sum_r \frac{p(E_r|E_{r'})}{p(E_{r'})} = \sum_{r-\{\$\}} \frac{p(E_r|E_{r'})}{p(E_{r'})} + \frac{p(\$|E_{r'})}{p(E_{r'})} \quad (3.18)$$

$$= \sum_{r-\{\$\}} \frac{p(E_r|E_{r'})}{p(E_{r'})} + \delta_m.\frac{p(\$|E_{r'})}{p(E_{r'})} + \epsilon_m.\frac{p(s_f|E_{r'})}{p(E_{r'})} \quad (3.19)$$

En identifiant avec $\gamma_{rr'}$, ζ_r, η_r, on obtient :

$$\begin{cases} \gamma_{rr'} = \frac{p(E_r|E_{r'})}{p(E_{r'})} \\ \zeta_{r'} = \frac{N_{mpl}-1}{N_{mpl}}.\frac{p(\$|E_{r'})}{p(E_{r'})} \\ \eta_{r'} = \frac{1}{N_{mpl}}.\frac{p(s_f|E_{r'})}{p(E_{r'})} \end{cases} \quad (3.20)$$

Soit pour entièrement caractériser la matrice, nous obtenons :

$$\begin{cases} \alpha_m = \frac{G}{M} \\ \beta_m = \frac{1-G}{p(E_r)} \\ \gamma_{rr'} = \frac{p(E_r|E_{r'})}{p(E_{r'})} \\ \delta_m = \frac{N_{mpl}-1}{N_{mpl}} \\ \epsilon_m = \frac{1}{N_{mpl}} \\ \zeta'_r = \frac{N_{mpl}-1}{N_{mpl}} \times \frac{p(\$|E_{r'})}{p(E_{r'})} \\ \eta'_r = \frac{1}{N_{mpl}} \times \frac{p(s_f|E_{r'})}{p(E_{r'})} \end{cases} \quad (3.21)$$

ce qui caractèrise entièrement les transitions de notre modèle de ligne pour la détection d'information.

Afin de tester les performances de notre système, des expérimentations ont été conduites sous le contrôle d'un protocole expérimental que nous présentons dans la section suivante.

3.3 Expérimentations

Nous venons de présenter notre système complet de reconnaissance de l'écriture manuscrite et nous voulons maintenant le tester en détection de séquences dans des documents manuscrits de type courrier entrant. À notre connaissance, la seule base de données publique et donc disponible de courriers entrants est francophone : il s'agit de la base communément appelée *RIMES*. Celle-ci a notamment servi de support à des compétitions de reconnaissance de l'écriture manuscrite [Grosicki 2009].

Dans un premier temps, nous commencons par donner des exemples d'utilisation de notre système pour appréhender le principe d'interrogation du système par requêtes *ASCII* sur quelques courriers entrants manuscrits de la base de test. Dans un second temps, nous présenterons les travaux permettant de construire les bases de données et les rendre utilisables pour le réglage, l'apprentissage et le test du système. Nous décrivons ensuite le protocole expérimental imaginé afin d'éprouver le système en détection d'information pour différents types de données (alphabétique, numérique et alphanumérique) et avec un nombre de requêtes plus ou moins important. Enfin, nous disserterons sur les résultats obtenus en détection d'information par rapport au protocole imaginé.

3.3.1 Exemples d'utilisation du système en détection d'information

La première série d'expérimentations a pour but de mettre en avant le fonctionnement du système de détection d'information par l'exemple : étant donné une image de courrier et une requête, nous interrogeons le système de détection sur la présence ou non de cette information spécifique, recherchée dans ce document.

Les deux premiers exemples montrent les résultats de détection obtenus sur un même courrier pour deux requêtes différentes : *formulaire* sur l'exemple de la figure 3.15 et *d'abonnement* sur la figure 3.16. La partie gauche de ces figures illustre graphiquement les résultats en ne considérant que les coordonnées des mots effectivement extraits. La partie droite présente la sortie brute du moteur de reconnaissance obtenue après décodage des images de lignes de texte prétraitées. Dans les résultats de décodage de ces deux figures, un espace est volontairement inséré entre toutes les entités reconnues ce qui permet de facilement différencier les caractères du modèle de remplissage des caractères appartenant à un modèle de séquence à extraire appartenant au lexique. Le

modèle d'espace est quant à lui matérialisé par le symbole **$**.

Les figures 3.17 et 3.18 représentent respectivement des exemples d'extraction de mots (de gauche à droite et de haut en bas *changement*, *délai*, *investir* et *reconcidérer*) et de séquences de mots (*changement de situation*, *demande de résiliation*, *dégats des eaux* et *traitement médical*) dans des documents produits par différents scripteurs et ayant donc des écritures très variables.

Enfin, les figures 3.19, 3.20 et 3.21 mettent en avant la capacité du système à extraire des données de type varié tel que des mots clés (*demande*, *licenciement* et *précisions*), des séquences numériques telles que des codes postaux (*54490*, *67110* et *60149*) ou des numéros de téléphone (*0315725979*) et enfin des séquences alphanumériques comme des dates (*20 Novembre 2006* dans l'exemple de la figure 3.20) ou des noms de rues (*34 rue Principale* dans un des exemples de la figure 3.21).

Ces quelques exemples de détection permettent d'appréhender le potentiel du système et en particulier la souplesse de notre modèle de ligne qui peut supporter des requêtes de tout type. Dans un contexte industriel, il est évident que nous devons travailler sur des courriers entiers et non sur des lignes de texte isolées. La présence de types de données variés tels que des noms communs, des noms propres, des chiffres, des séquences numériques et alphanumériques nous permet d'envisager le test de notre système en détection d'informations alphabétiques, numériques et alphanumériques.

La base de données *RIMES Courriers* comprend un total de 1150 images de courriers manuscrits, quelques-uns ont été utilisés dans cette section pour illustrer la détection d'information à partir de requêtes. Depuis ces courriers, ont été isolées et mises à notre disposition des images de mots et de séquences numériques (pour des tâches de reconnaissance d'entités isolées) ainsi que la retranscription du corps de texte des courriers (accessible sous la forme de fichiers *xml*). Deux problèmes se présentent :

- pour calculer des performances en détection de séquences, il nous faut connaitre les positions exactes des séquences dans les images de courrier ce qui n'est pas disponible immédiatement.
- pour apprendre les modèles de Markov cachés de caractères et le modèle de remplissage, il nous faut des images de lignes de texte annotées. Celles-ci ne sont pas disponibles non plus.

Ces problèmes sont illustrés par le courrier de la figure 3.22 : sur cet exemple, seules les boîtes englobantes des mots ayant été annotés et isolés dans une base distincte *RIMES Mots* sont représentées. Cela représente pour cet exemple 14 images sur les 80 mots et séquences numériques du courrier. Sur la totalité de la base seulement 33% des mots et séquences numériques sont annotés. Un

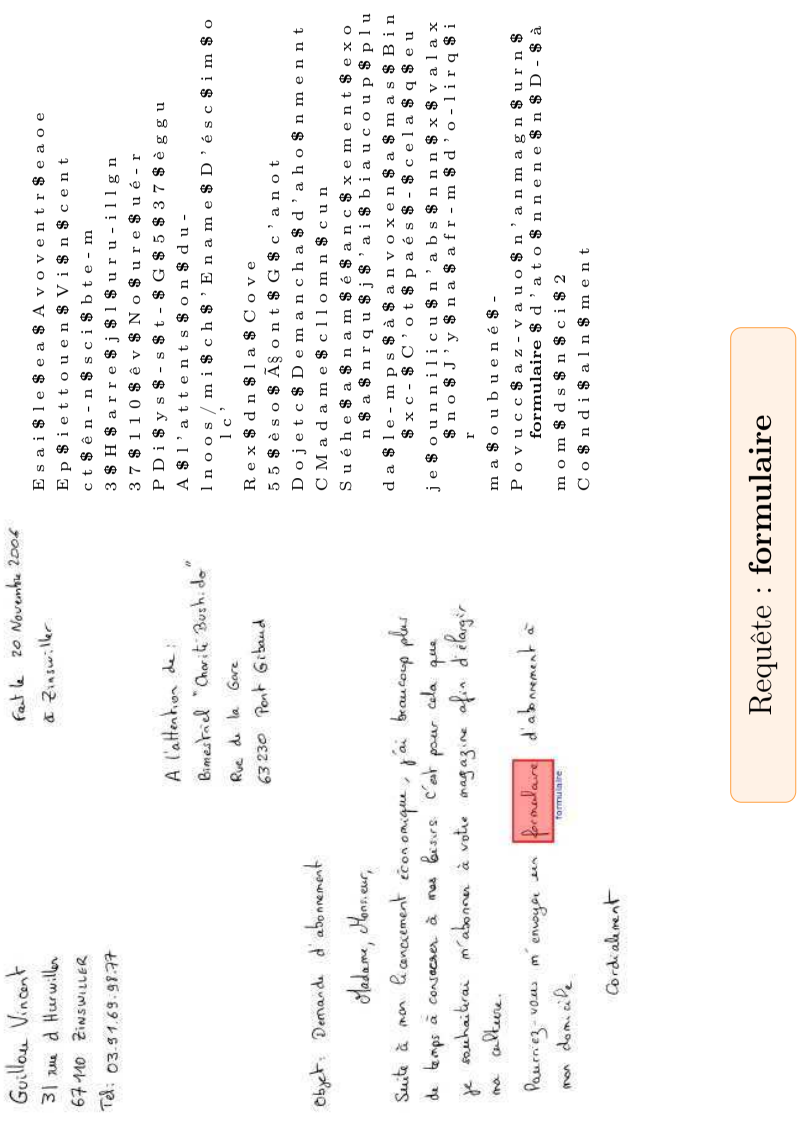


FIG. 3.15: Détection de la requête *formulaire* dans un courrier entrant et le résultat de décodage contraint par le modèle de ligne construit à partir de cette requête.

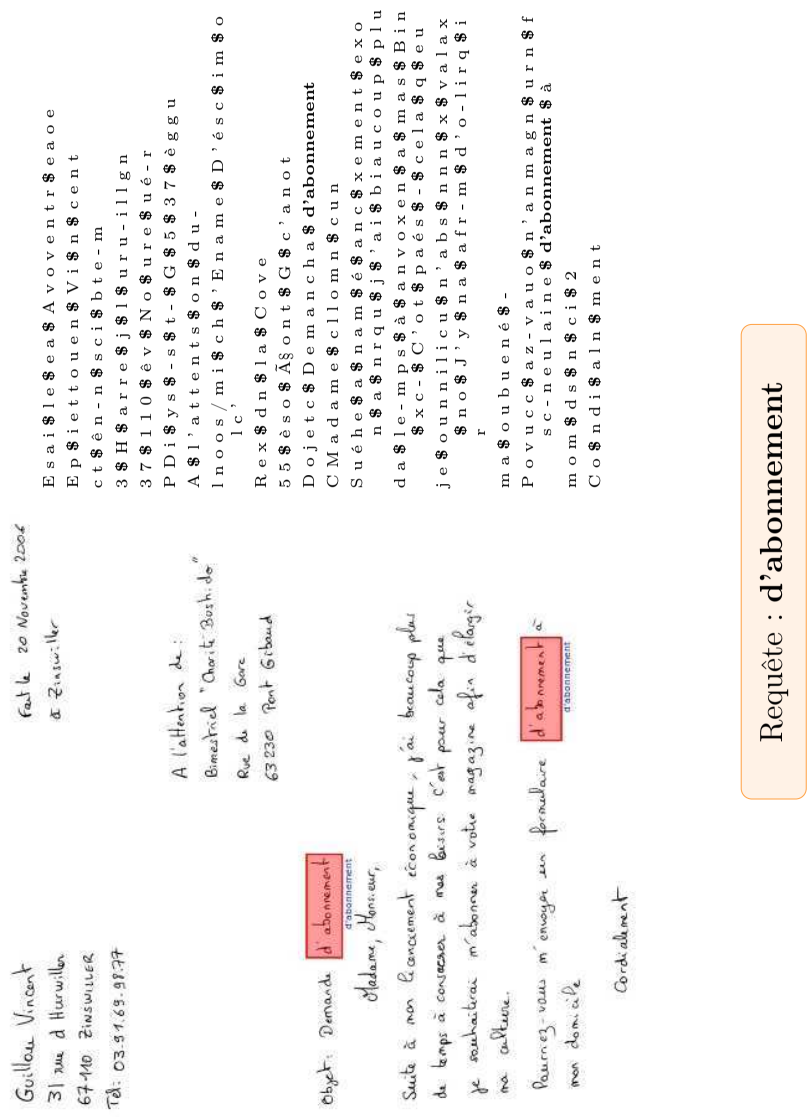
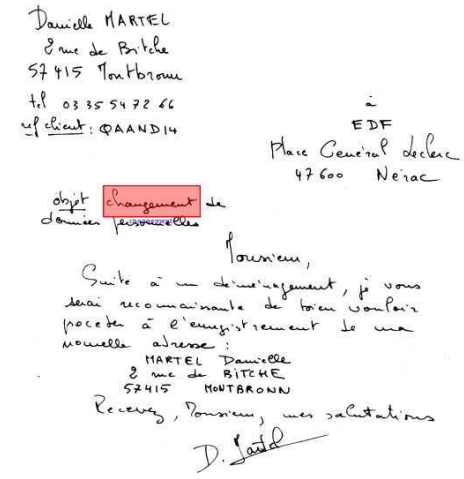
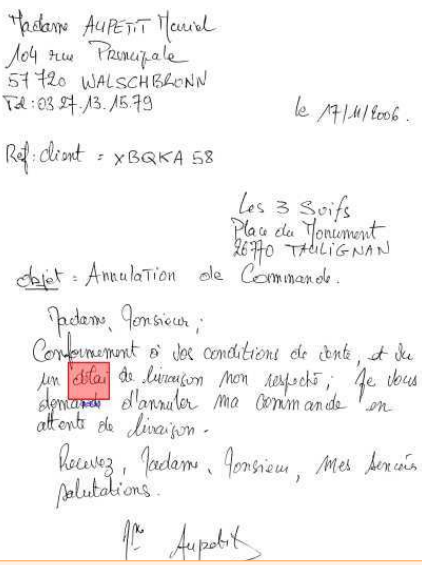


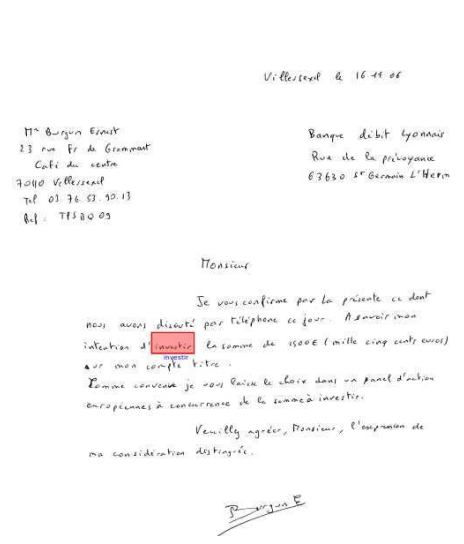
FIG. 3.16: Détection de la requête *d'abonnement* dans un courrier entrant et le résultat de décodage contraint par le modèle de ligne construit à partir de cette requête.



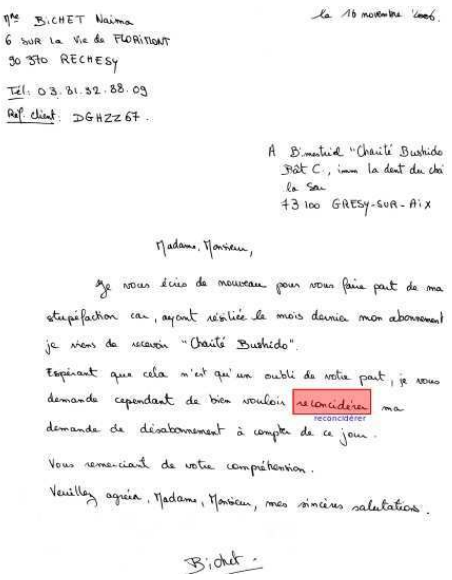
Requête : **changement**



Requête : **délai**

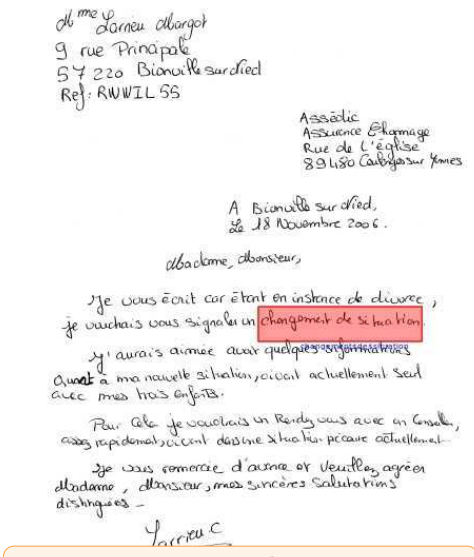


Requête : **investir**

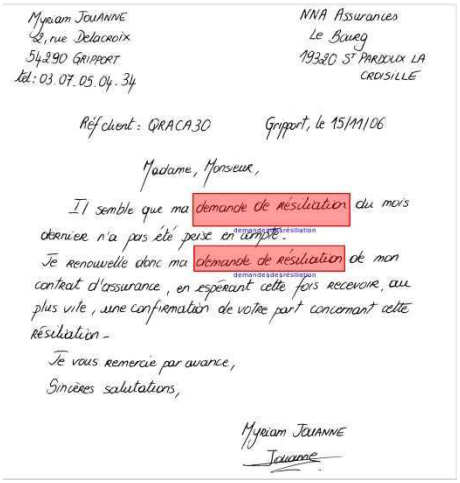


Requête : **reconcidérer**

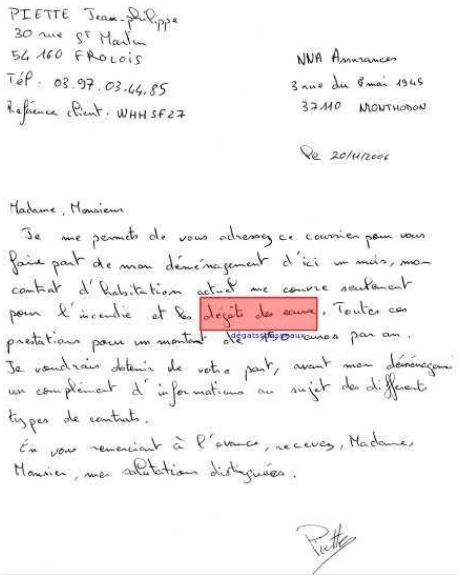
FIG. 3.17: Exemples de détection de mots clés dans des courriers produits par différents scripteurs. De gauche à droite et de haut en bas, les mots *changement*, *délai*, *investir* et *reconcidérer* sont respectivement détectés correctement dans les courriers.



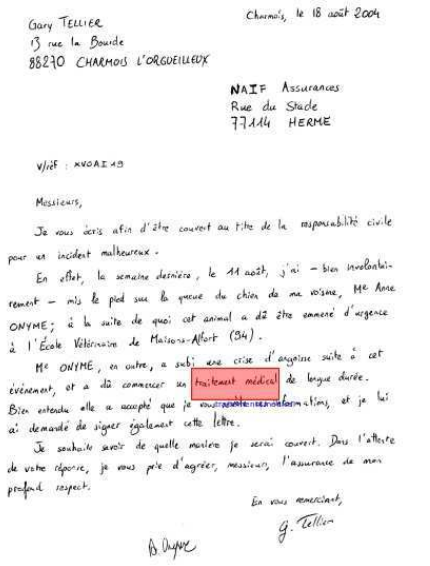
Requête : **changement de situation**



Requête : **demande de résiliation**



Requête : **dégats des eaux**



Requête : **traitement médical**

FIG. 3.18: Exemples de détection de séquences de mots dans des courriers produits par différents scripteurs. De gauche à droite et de haut en bas, les séquences de mots *changement de situation*, *demande de résiliation*, *dégats des eaux* et *traitement médical* sont ainsi détectées avec succès.

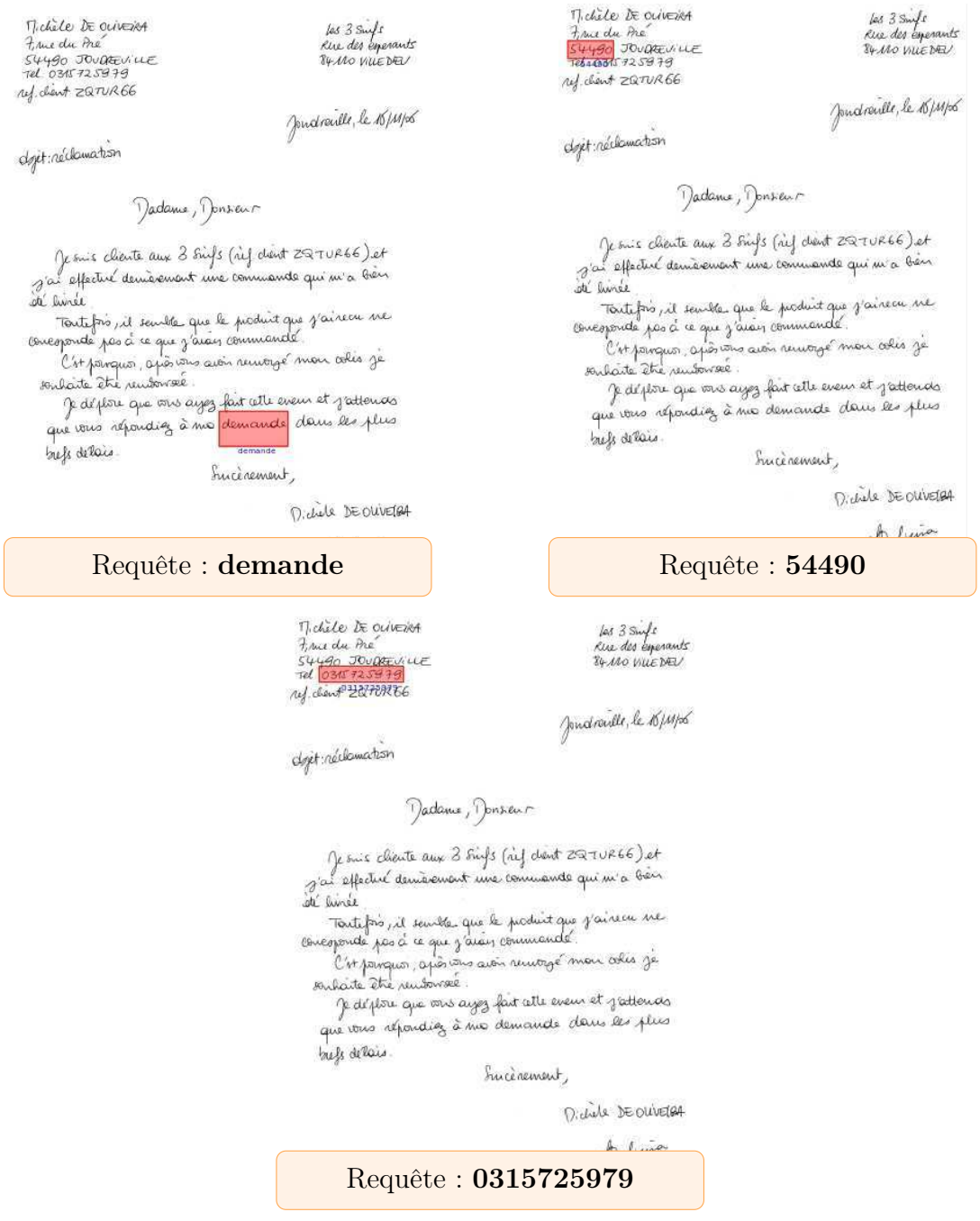
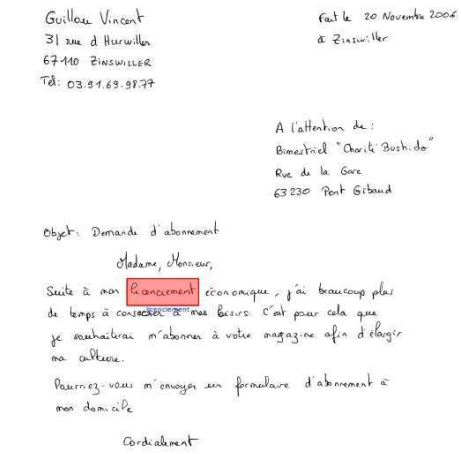


FIG. 3.19: Exemples de détection de requêtes de différents types dans un même courrier : le mot clé *demande*, le code postal *54490* et le numéro de téléphone *0315725979* sont détectés du courrier avec succès.

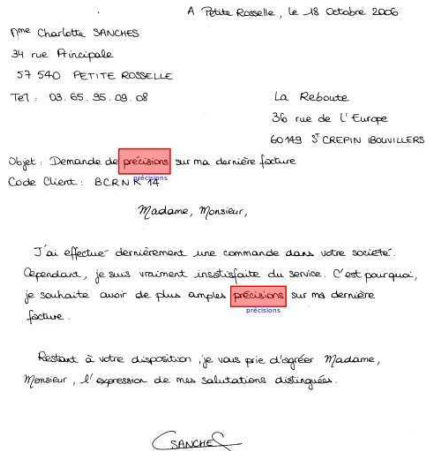


Requête : **licenciement**

Requête : **67110**

Requête : **20 Novembre 2006**

FIG. 3.20: Exemples de détection de requêtes de différents types dans un second courrier : le mot clé *licenciement*, le code postal *67110* et la date *20 Novembre 2006* sont détectés de ce courrier avec succès.


A Petite Rosselle, le 18 Octobre 2006

Mme Charlotte SANCHES
34 rue Principale
57 540 PETITE ROSSELLE
Tél : 03.65.35.09.08

La Reboute
36 rue de L'Europe
60149 St CREPIN IBOUVILLERS

Objet : Demande de précisions sur ma dernière facture
Code Client : BCRNK14

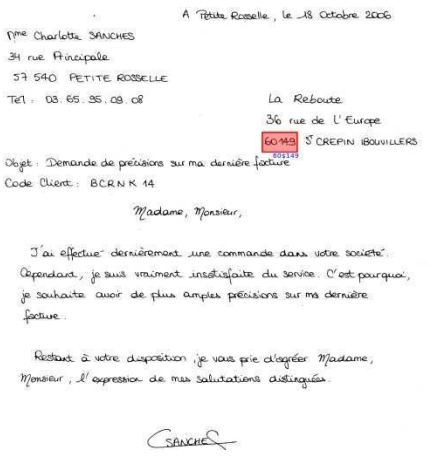
Madame, Monsieur,

J'ai effectué dernièrement une commande dans votre société. Cependant, je suis vraiment insatisfaite du service. C'est pourquoi, je souhaite avoir de plus amples précisions sur ma dernière facture.

Restant à votre disposition, je vous prie d'agréer Madame, Monsieur, l'expression de mes salutations distinguées.

SANCHES

Requête : **précisions**


A Petite Rosselle, le 18 Octobre 2006

Mme Charlotte SANCHES
34 rue Principale
57 540 PETITE ROSSELLE
Tél : 03.65.35.09.08

La Reboute
36 rue de L'Europe
60149 St CREPIN IBOUVILLERS

Objet : Demande de précisions sur ma dernière facture
Code Client : BCRNK14

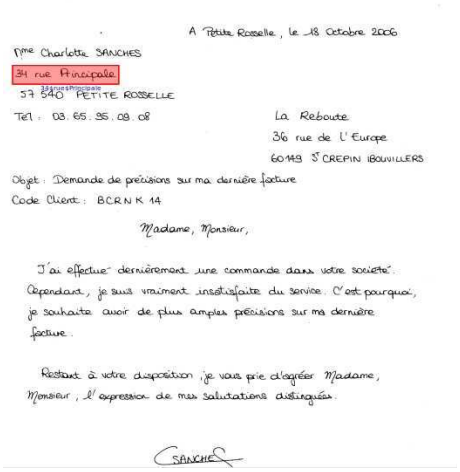
Madame, Monsieur,

J'ai effectué dernièrement une commande dans votre société. Cependant, je suis vraiment insatisfaite du service. C'est pourquoi, je souhaite avoir de plus amples précisions sur ma dernière facture.

Restant à votre disposition, je vous prie d'agréer Madame, Monsieur, l'expression de mes salutations distinguées.

SANCHES

Requête : **60149**


A Petite Rosselle, le 18 Octobre 2006

Mme Charlotte SANCHES
34 rue Principale
57 540 PETITE ROSSELLE
Tél : 03.65.35.09.08

La Reboute
36 rue de L'Europe
60149 St CREPIN IBOUVILLERS

Objet : Demande de précisions sur ma dernière facture
Code Client : BCRNK14

Madame, Monsieur,

J'ai effectué dernièrement une commande dans votre société. Cependant, je suis vraiment insatisfaite du service. C'est pourquoi, je souhaite avoir de plus amples précisions sur ma dernière facture.

Restant à votre disposition, je vous prie d'agréer Madame, Monsieur, l'expression de mes salutations distinguées.

SANCHES

Requête : **34 rue Principale**

FIG. 3.21: Exemples de détection de requêtes de différents types dans un troisième courrier : le mot clé *précisions*, le code postal *60149* et le nom de rue *34 rue Principale* sont correctement détectés.

important travail d'annotation a donc été réalisé avant d'envisager l'utilisation de cette base pour calculer des performances en détection d'information. Nous le présentons dans la section suivante.

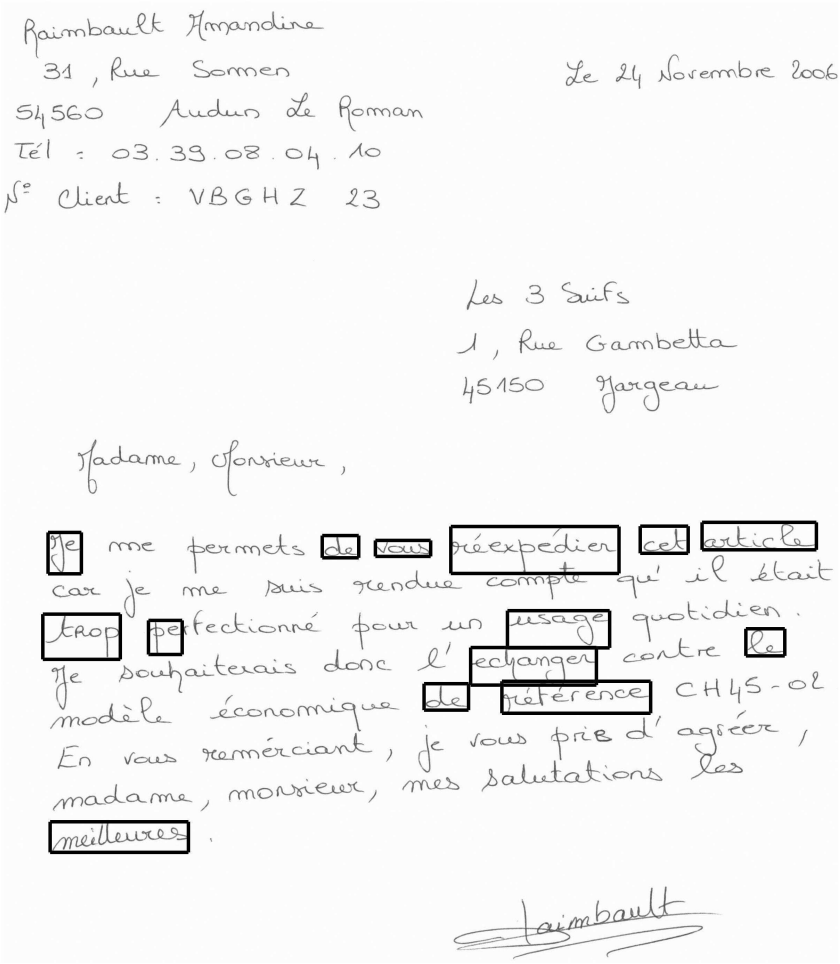


FIG. 3.22: Exemple de courrier annoté dans de la base de données *RIMES*

3.3.2 Étiquetage et annotation de la base de données

Pour mesurer les performances du système mis en œuvre, il nous faut avoir accès à un nombre suffisant de documents étiquetés au niveau mot pour les tests et au niveau lignes pour les apprentissages. Ces tâches étant très fastidieuses pour un opérateur, il a été envisagé de les automatiser le plus possible. Nous présentons maintenant les solutions retenues pour créer nos bases d'apprentissage et de test.

3.3.2.1 Création d'une base d'apprentissage

L'apprentissage des modèles de caractères et de lignes nécessite des images de lignes de texte annotées. Une annotation au niveau mot n'est pas forcément nécessaire puisque les MMC de caractères sont appris de manière embarquée, un modèle de ligne pouvant être vu comme un modèle de mot intégrant un ou plusieurs modèles d'espace suivant le nombre de mots de la ligne. La figure 3.23 illustre les données à notre disposition : des images de courriers et des images de mots étiquetés et extraits depuis ces courriers sans que leurs positions dans ceux-ci ne soient connus. A partir de ces données, nous voulons donc reconstruire des images de lignes étiquetées.

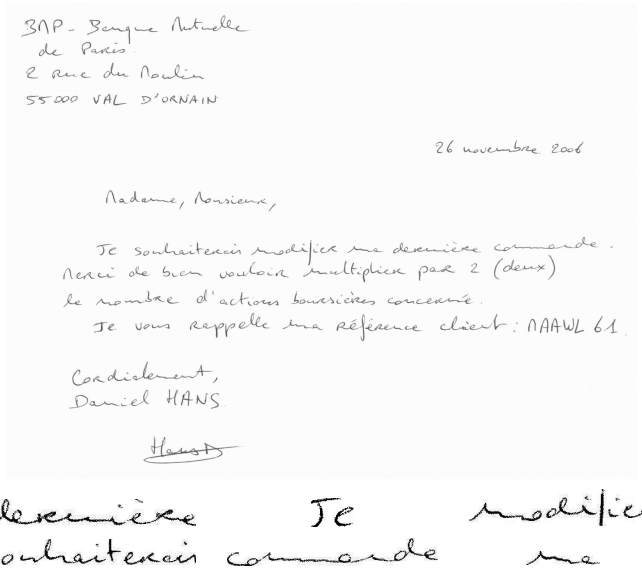
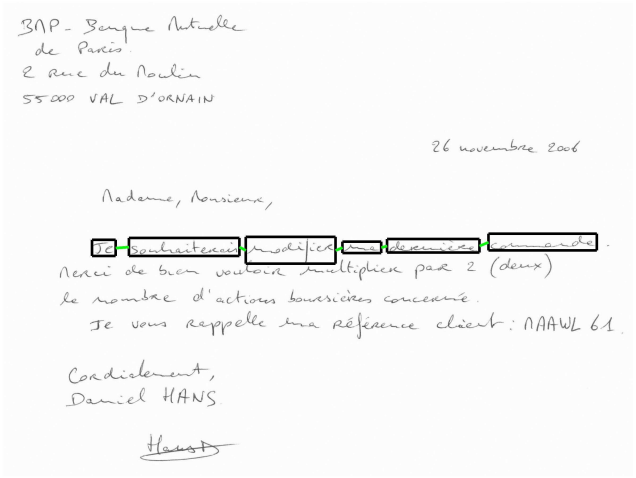


FIG. 3.23: Exemple de jeux de données disponibles dans la base de données *RIMES* : courrier manuscrit non étiqueté (en haut) et 6 exemples d'images de mots isolés depuis ce courrier mais dont la position est inconnue (en bas).

Dans un premier temps, un algorithme d'appariement d'images (*template matching*) a été utilisé pour retrouver la position précise des images de mots dans les images courriers. Ces positions ont ensuite été comparées afin de regrouper les mots suffisamment proches en termes de distances verticale et horizontale dans le but de reconstruire des lignes de texte. L'annotation finale d'une ligne de texte est obtenue en concaténant les annotations des mots séparés par des espaces. Ceci est illustré par la figure 3.24. Afin de ne pas produire des mauvaises données d'apprentissage, deux critères de distances verticale et horizontale ont été utilisés et permettent de ne pas

fusionner deux mots dont l'appartenance à une même ligne n'est pas certaine.



Je souhaiterais modifier ma dernière commande

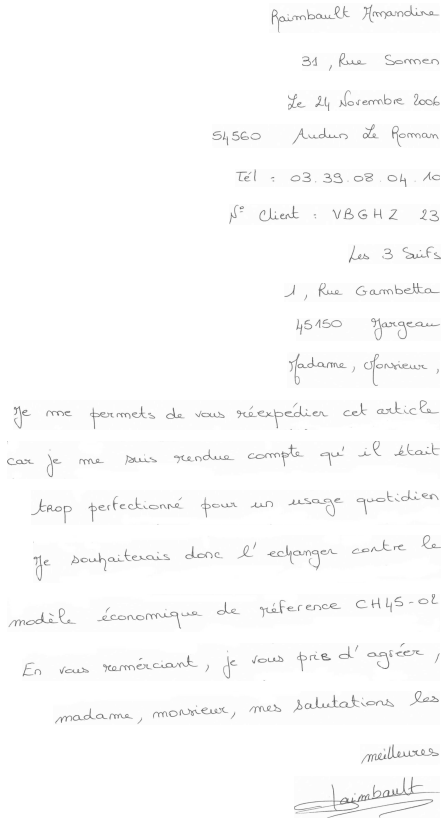
FIG. 3.24: Illustration du processus de *template matching* et d'alignement des mots localisés dans leur courrier (en haut), de l'extraction de l'image de ligne correspondante et de la construction de l'étiquette globale de la ligne (en bas).

Cet algorithme nous a permis d'annoter un ensemble de 15683 lignes de texte regroupant 37312 mots et séquences numériques soit une moyenne d'un peu moins de 2,4 mots par ligne. Dans la réalité de nos courriers, cette moyenne peut paraitre ridiculement petite mais tous les mots des courriers n'ont pas été isolés : certaines lignes ne possèdent qu'un seul mot ou une seule séquence numérique (cf. figure 3.22) et d'autres ont été tronquées pendant la procèdure pour éviter le regroupement d'images de mots n'appartenant pas au même alignement.

3.3.2.2 Création des bases de validation et de test

Pour calculer les performances en extraction d'information, il nous faut les positions exactes des images de mots dans les courriers pour pouvoir les comparer avec les positions des mots reconnus par notre système. Or, les mots du courrier n'ont pas tous été isolés (cf. figure 3.22) : la méthode d'étiquetage basée sur un template matching ne suffit plus.

Pour contourner ce problème, nous avons développé un algorithme d'annotation semi-automatique. Dans un premier temps, des images de courriers ont été annotées manuellement de manière complète en respectant la segmentation en lignes des documents produites par notre algorithme de segmentation en lignes (cf. figure 3.25).



Raimbault Amandine
31 , Rue Somen
Le 24 Novembre 2006
54560 Audun Le Roman
Tél : 03.39.08.04.10
N° Client : VBGHZ 23
Les 3 Suifs
1 , Rue Gambetta
45150 Jargeau
Madame, Monsieur,
Je me permets de vous réexpédier cet article
Car je me suis rendue compte qu'il était
trop perfectionné pour un usage quotidien
Je souhaiterais donc l' echanger contre le
modèle économique de référence CH45-02
En vous remerciant, je vous prie d'agréer,
madame, monsieur, mes salutations les
meilleures
Raimbault

FIG. 3.25: Illustration de l'extraction des lignes de texte depuis un courrier manuscrit pour automatiser leur annotation.

Un alignement du MMC correspondant à l'annotation de la ligne est effectué sur son image. Ceci nous permet de récupérer les bornes horizontales de chacun des mots de cette ligne. Combinées aux coordonnées verticales de leur ligne, nous récupérons ainsi les boîtes englobantes de tous les mots de tous les courriers. Un parcours visuel des résultats de segmentation en lignes a été effectué pour constater que :

- très peu de cas de sous-segmentation en lignes ont été rencontrés.
- les quelques cas de sur-segmentation observés n'ont pas entrainé de séparation de mots ou de séquences numériques en deux dans deux parties de lignes distinctes.

– l'alignement effectué avec les MMC en termes de segmentation en mots ne présente pas d'erreurs. Des décalages d'une trame sont quelquefois observés pour les bornes de segmentation des mots : celles-ci seront prises en compte dans le calcul des performances.

Le processus d'alignement est illustré par le schéma de la figure 3.26 et met en avant la manière dont les bornes extrêmes des mots sont récupérées.

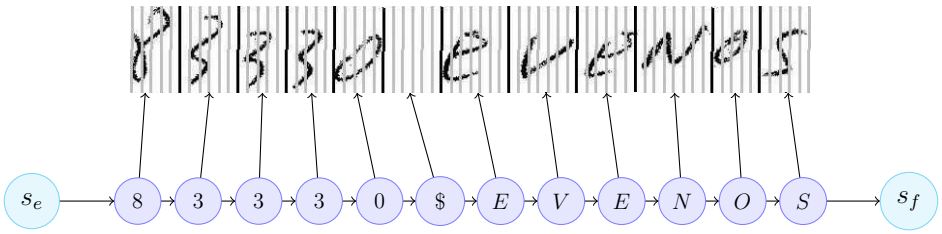


FIG. 3.26: Illustration d'un alignement d'une image de ligne avec le modèle MMC correspondant à sa transcription.s_e et s_f sont des états structurels non émetteurs de début et de fin de modèle, $ représente le modèle d'espace.

3.3.2.3 Répartition des données dans les bases

La base d'apprentissage est composée de 950 courriers depuis lesquels 15683 lignes ont été reconstruites lesquelles embarquant un total de 37312 séquences. La base de validation est composée d'un ensemble de 100 courriers englobant un total de 8095 séquences. Enfin, la base de test est composée d'un autre ensemble de 100 courriers renfermant un total de 2050 lignes de texte et 7932 mots, séquences numériques et alpha-numériques. La répartition des données dans les différentes bases est illustrée par le tableau de la figure 3.27.

Base	App	Valid	Test
Nb courriers	950	100	100
Nb lignes	15683	2130	2050
Nb séquences	37312	8095	7932

FIG. 3.27: Répartition des données dans les différentes bases.

3.3.3 Protocole expérimental

Afin de tester notre système, plusieurs expérimentations ont été mises en place, leur but étant de tester les capacités du système en reconnaissance, en requêtage (une seule séquence recherchée) et en détection d'information (un

ensemble de séquence est recherché).

Pour mesurer les performances du système en reconnaissance pure, une tâche de reconnaissance de mots isolés est dans un premier temps considérée. Un lexique de 100 mots est construit pour chaque image de mot à reconnaitre et est constitué de l'étiquette de ce mot complétée de 99 étiquettes tirées aléatoirement parmi les étiquettes des mots de la base de données de test. Un score est attribué par le moteur de reconnaissance pour chacune des entrées du lexique. Le taux de bonnes reconnaissances est enfin calculé en confrontant l'entrée correspondant au meilleur score et l'étiquette du mot à reconnaitre.

Pour mesurer les performances de notre système en requêtage (correspondant à une tâche de keyword-spotting), nous cherchons à retrouver une séquence dans un courrier manuscrit. La séquence représentant la requête est tirée aléatoirement dans le vocabulaire du courrier. Il peut donc s'agir aussi bien d'un mot que d'une séquence numérique ou alphanumérique. Pour une requête donnée, un modèle de ligne est construit autour de cette requête et la reconnaissance est lancée sur le courrier sous la contrainte de ce modèle. L'expérience est menée sur l'ensemble des courriers de la base de test et reconduite pour 10 requêtes pour chaque document.

Enfin, pour tester notre système en détection d'information, nous avons fait le choix de simuler des requêtes multiples en définissant automatiquement des lexiques de séquences à détecter. Une première approche consiste à considérer un ensemble de séquences appartenant aux 100 documents de la base de test. Étant donnée l'hétérogénéité de ceux-ci en termes de contenu, l'ensemble des informations communes à ces documents est très faible. S'il est possible de trouver un certain nombre de mots qui leur sont communs, les séquences numériques prises individuellement sont uniquement présentes dans un seul document. Ceci est dû à l'unicité des numéros de téléphone et la grande quantité de codes postaux par exemple. Afin de garantir une certaine quantité d'information à détecter dans chaque document, les expériences que nous avons menées se basent sur un ensemble dynamique de requêtes construit spécifiquement pour chacun des documents de la base de test sous la forme de lexique. Pour un document donné, nous avons donc choisi de prélever aléatoirement 10 séquences appartenant à son vocabulaire et de les compléter de $N - 10$ séquences n'appartenant pas à son vocabulaire. 100 lexiques de N séquences (1 lexique par document) sont donc constitués pour chacune des différentes expériences menées.

Il est à noter que les séquences sélectionnées pour constituer les requêtes ou

les lexiques sont choisies aléatoirement. Dans le cas de lexique, elles sont également discriminées par type. Ainsi, nous séparons les mots clés des séquences numériques et alphanumériques. Un critère sur la longueur des séquences en nombre de caractères est utilisé afin de s'assurer que les mots clés ne soient pas des mots vides tels que des articles ou pronoms comme par exemple : *le, la, les, il, elle, nous, etc.* ou des chiffres isolés tels que des numéros de rues par exemple. Ce critère a été fixé à 5 caractères. En ce qui concerne la base de test, dont les effectifs en nombre de documents, de lignes et de séquences sont récapitulés dans le tableau de la figure 3.27, les 7932 séquences remplissant les critères donnés se répartissent de la manière suivante :

- 3650 mots possèdent 5 caractères ou plus
- 375 séquences numériques possèdent 5 chiffres ou plus (codes postaux, numéros de téléphones et autres champs numériques)

Étant donné la faible quantité de séquences alphanumériques (comme *VBGHZ 23* et *CH45-02* dans l'exemple de la figure 3.25) et la difficulté d'annoter et d'extraire automatiquement des séquences alphanumériques complexes de type adresse (*31 , Rue Somen*, *1 , Rue Gambetta* ou *45150 Jargeau* sur la figure 3.25) ou date (*24 Novembre 2006* sur la figure 3.25), nous avons fait le choix de composer nos lexiques alphanumériques d'un *mix* des deux types de séquences alphabétiques et numériques en intégrant, lorqu'elles sont présentes, les séquences alphanumériques comme *VBGHZ 23* et *CH45-02*.

Afin d'étalonner notre approche de détection d'information, nous la comparons avec une approche de lecture complète de documents. Pour un document donné, nous nous plaçons dans les mêmes conditions expérimentales : le même lexique de séquences est à extraire. Dans une telle approche, un lexique distinct de plus grande taille doit cependant être considéré pour la reconnaissance. Trois tailles différentes ont été utilisées :

(i) un lexique de reconnaissance correspondant exactement au vocabulaire du document traité. Ce choix favorise l'approche de reconnaissance complète. Bien qu'inenvisageable en pratique, il permet d'étalonner notre approche d'extraction.

(ii) un lexique constitué de l'ensemble du vocabulaire des documents de la base de test soit 3188 mots.

(iii) un lexique ouvert composé de 25000 mots. Ce cas correspond donc davantage à une situation réelle.

3.3.4 Calcul des performances

En détection d'information, le rappel et la précision sont les deux indices permettant de mesurer les performances d'un système. En fonction de l'application, plusieurs points de fonctionnement peuvent être considérés. Nous voulons pouvoir choisir un point de fonctionnement suivant que l'on désire privilégier le rappel ou la précision. L'hyperparamètre G du modèle de ligne nous permet de définir de tels points de fonctionnement et donc de se déplacer dans le plan rappel/précision. Nous rappelons que G représente la proportion d'information pertinente présente dans une ligne et pondère donc la transition vers les modèles de séquences à extraire (cf. section 2.4) : augmenter G permet de forcer la reconnaissance de ces séquences et par conséquent améliorer le rappel du système. Inversement, diminuer G permet d'augmenter la précision du système au détriment du rappel. Ceci est illustré par le schéma de la figure 3.28.

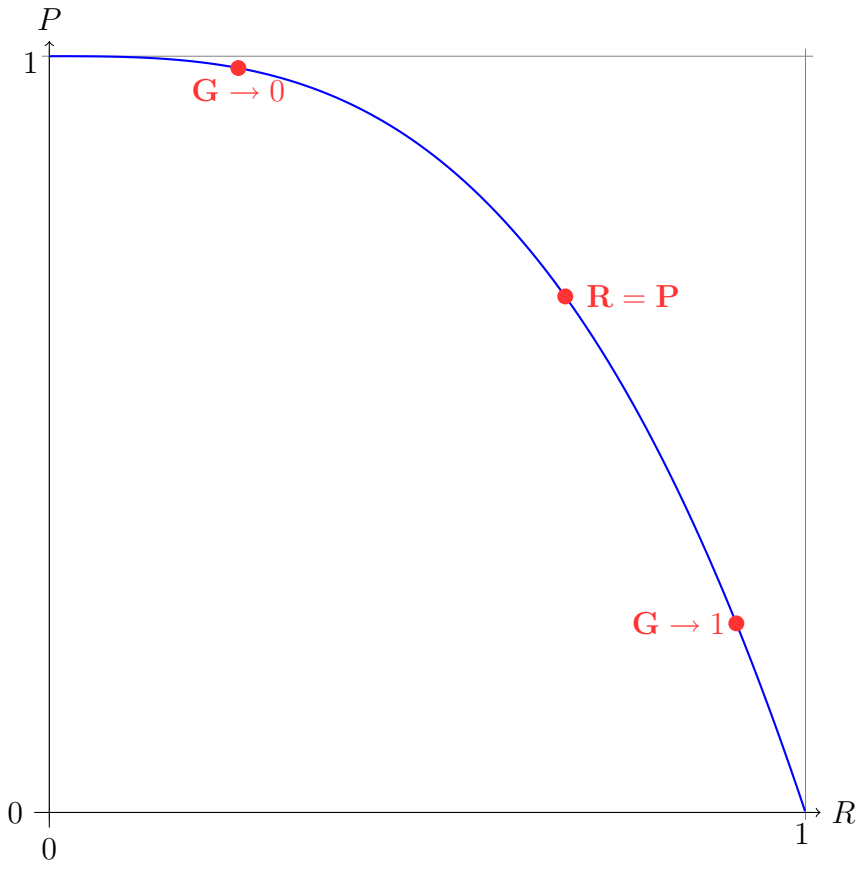


FIG. 3.28: Illustration de l'influence du paramètre G sur le rappel et la précision du système.

Plus concrètement, plaçons-nous dans l'exemple de la figure 3.29. Le modèle de ligne a été construit de manière à détecter la présence du mot *commande* comme l'illustre le modèle de ligne de la figure 3.30. Une valeur de G proche de 0 va favoriser le décodage des entités du modèle de remplissage au détriment du modèle du mot *commande* et donc retourner en sortie une séquence de caractères non contraintes par le lexique. A contrario, lorsque nous allons faire tendre la valeur de G vers 1, nous allons forcer la détection de séquences du lexique et donc le décodage du modèle de mot *commande* dans ce cas précis. Nous obtenons en sortie du moteur de reconnaissance une séquence constituée d'un certain nombre de mots *commande*. Un seul est correctement détecté alors que les autres sont incorrectement détectés engendrant des erreurs qu'il nous faut prendre en compte dans le calcul des performances.

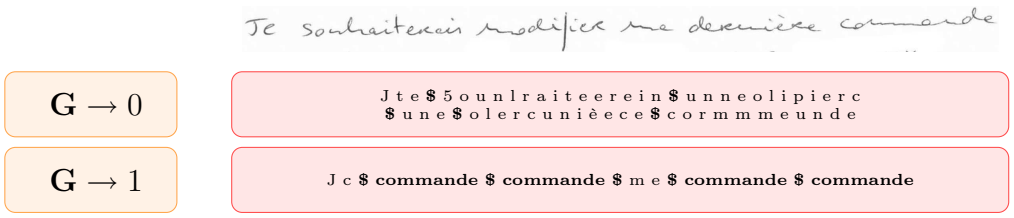


FIG. 3.29: Illustration de l'influence du paramètre G sur la recherche du mot *commande* dans une image de ligne de texte. La partie supérieure illustre le décodage de l'image lorsque la valeur de G tend vers 0 résultant en une non détection, la partie inférieure présente le résultat du décodage de cette même image lorsque la valeur de G tend vers 1 résultant en une surdétection du mot recherché.

Formellement, le calcul du rappel et de la précision est donné par les équations 3.22 et 3.23 suivantes :

$$Rappel = \frac{\text{Card}(BD)}{\text{Card}(BD) + \text{Card}(FR)} \tag{3.22}$$

$$Precision = \frac{\text{Card}(BD)}{\text{Card}(BD) + \text{Card}(FD)} \tag{3.23}$$

Ils reposent sur le recensement des événements suivants :

Card(BD) nombre de Bonnes Détections (BD) correspondant au nombre de séquences correctement détectées.

Card(FD) nombre de Fausses Détections (FD) correspondant au nombre de séquences détectées à tort.

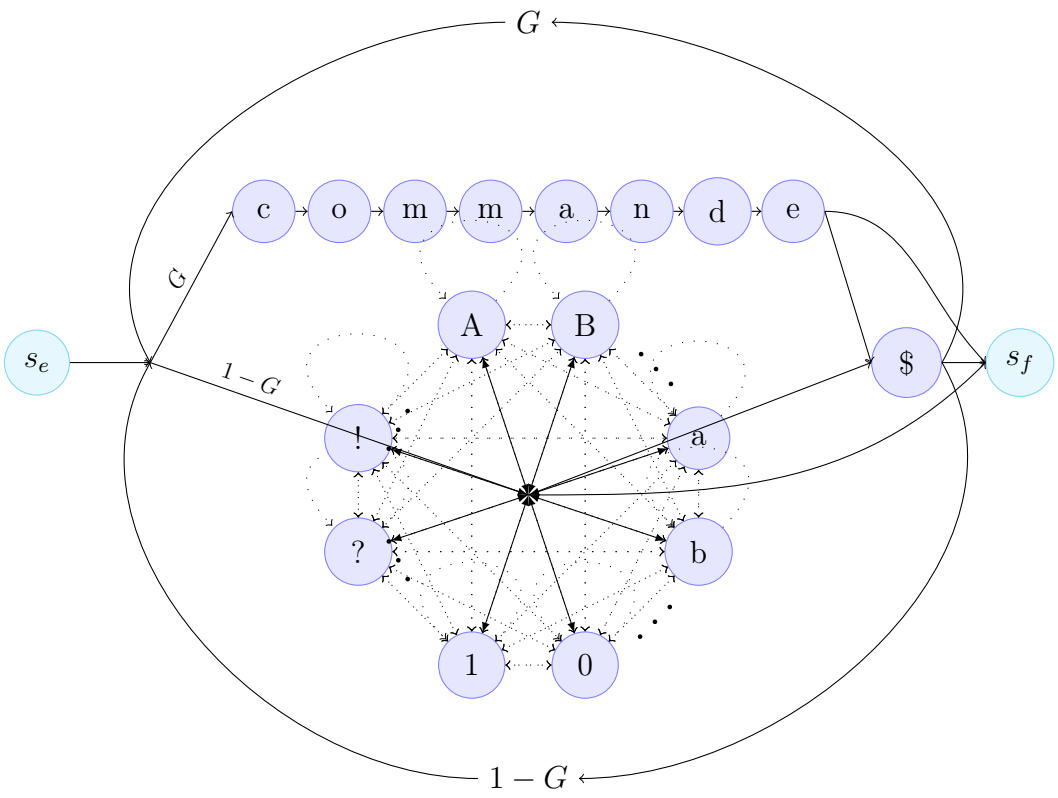


FIG. 3.30: Illustration du modèle de ligne créé pour la recherche du mot *commande*.

Card(BR) nombre de Bons Rejets (BR) correspondant au nombre de séquences correctement rejetées.

Card(FR) nombre de Faux Rejets (FR) correspondant au nombre de séquences non détectées.

Ces évènements sont illustrés par la figure 3.31. Dans cet exemple, nous souhaitons de nouveau rechercher le mot *commande*. Les différentes configurations correspondent alors :

a) au cas idéal, c'est à dire que le mot recherché est bien détecté et les 5 autres sont biens rejetés soit 5 BR et 1 BD

b) au rejet complet de la ligne soit 5 BR et 1 FR

c) à une fausse alarme et un mauvais rejet soit 4 BR, 1 FD et 1 FR

d) à une bonne détection et une fausse alarme soit 4 BR, 1 FD et 1 BD

e) à une détection ambigüe soit 5 BR, 1 FD et 1 FR

Notons que les performances sont mesurées par rapport à la vérité terrain (c'est à dire l'annotation des lignes de texte). En procédant de la sorte, des cas particuliers peuvent apparaitre comme c'est le cas dans l'exemple *e)* de la figure 3.31. Dans cette configuration, le dernier espace est mal reconnu et le mot détecté recouvre deux mots de la vérité terrain. Nous comptabiliserons donc un mauvais rejet pour la partie du mot à détecter non-détectée et une mauvaise détection pour la partie du mot à rejeter incorrectement détecter. Nous nous plaçons dans la situation la moins avantageuse en ne comptant pas de bonne détection et en comptant deux erreurs pour un mot partiellement détecté.

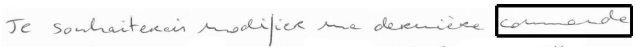
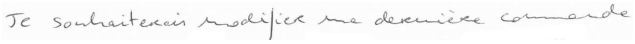
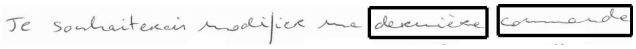
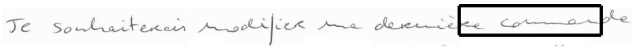
	Illustrations	Résultats
a)		5 BR, 1 BD
b)		5 BR, 1 FR
c)		4 BR, 1 FD, 1 FR
d)		4 BR, 1 FD, 1 BD
e)		5 BR, 1 FD, 1 FR

FIG. 3.31: Illustration du calcul des performances pour différentes configurations d'extraction du mot *commande*.

Le protocole expérimental ainsi que les bases ont été présentés, nous nous intéressons dans la section suivante aux performances atteintes par notre système lors des expérimentations que nous avons conduites.

3.4 Résultats

Dans cette section, nous présentons les performances du système sur différentes tâches de détection d'information. Avant cela, une première expérimentation a été menée sur une tâche de reconnaissance de mots isolés.

3.4.1 Résultats en reconnaissance de mots isolés

La base de test utilisée est la base de données *RIMES mots*. Elle a fait l'objet d'une compétition de reconnaissance de mots isolés [Grosicki 2009], nous pouvons donc comparer notre moteur de reconnaissance à ces systèmes en notant toutefois que les lexiques que nous avons utilisés ne sont pas les mêmes que ceux considérés lors de cette compétition. N'ayant pas participé à

cette compétition, nous n'avons pas eu accès ces lexiques. Nous les avons donc générés dynamiquement : l'étiquette de l'image du mot considéré est ajoutée à 99 étiquettes tirées aléatoirement dans le vocabulaire de la base pour former un lexique de 100 mots. Les résultats de reconnaissance sont illustrés par le tableau de la figure 3.32. Les performances des systèmes ayant pris part à la compétition ont été ajoutées afin d'avoir un point de comparaison pour notre moteur de reconnaissance.

Système	Top 1	Top2	Top5	Top 10
TUM	98,4	-	-	99,9
UPV	96,4	-	-	99,6
Siemens	95,3	-	-	99,7
LITIS	91,4	-	-	98,3
IRISA	91,2	-	-	96,3
Paris Tech (système 1)	86,4	-	-	88,8
Paris Tech (système 2)	82,0	-	-	87,2
Paris Tech (système 3)	79,9	-	-	88,0
Notre Approche	**78,7**	**87,6**	**93,8**	**97,2**
ITESOFT	74,6	-	-	85,3

FIG. 3.32: Comparaison des résultats obtenus avec les systèmes ayant pris part à la compétition de reconnaissance de mots isolées *RIMES* avec un lexique fixé à 100 mots [Grosicki 2009].

Nous remarquons que les performances de notre système en reconnaissance de mots isolés sont modérées. Signalons toutefois que lorsque l'on considère le $Top10$, notre système est capable de faire remonter les bonnes solutions. Gardons à l'esprit qu'il ne s'agit pas la de notre problématique qui est la détection d'information. Nous présentons maintenant les résultats de notre système dans ce cadre.

3.4.2 Résultats du système en requêtage (*keyword-spotting*)

Dans un premier temps, nous testons le système de détection d'information en lui soumettant une seule requête à la fois. Comme décrit dans le protocole expérimental, la séquence correspondant à la requête est présente dans le courrier et tient le rôle de l'information à rechercher. Toutes les autres séquences du courrier sont à rejeter. Trois points de fonctionnement du système sont définis en fixant G de la manière suivante :

- une valeur proche de 1 favorisant la détection et donc le rappel au détriment de la précision
- une valeur proche de 0 favorisant le rejet et donc la précision au détriment du rappel
- une valeur tel que $Rappel \approx Prcision$

Les rappels et précisions sont calculés par rapport aux 10 interrogations du système pour chaque courrier et moyennés sur l'ensemble des courriers de la base de test. Ceci nous permet de calculer les écarts-types des rappels et précisions. Les résultats correspondant sont donnés sur la figure 3.33.

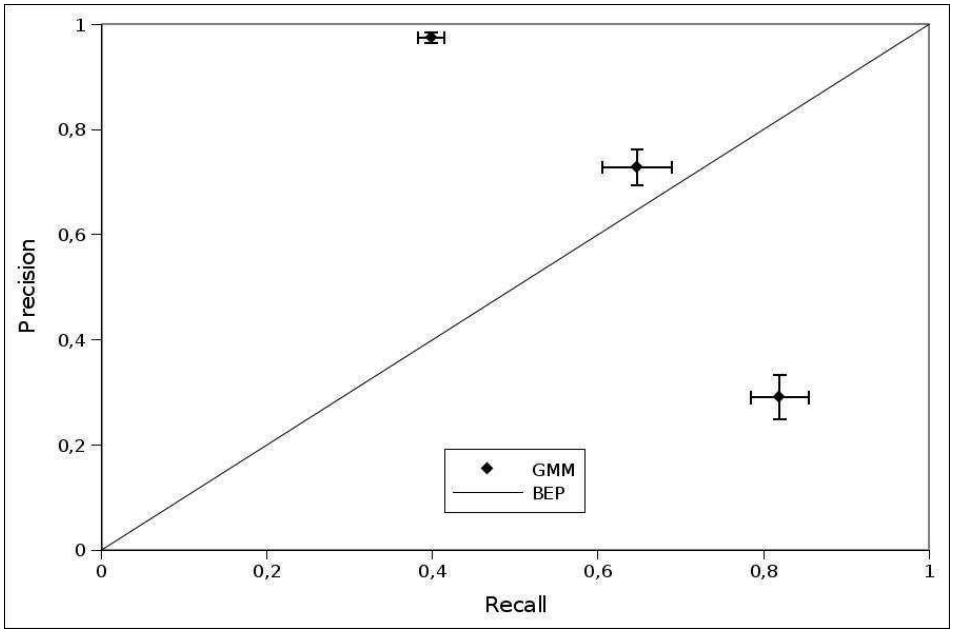


FIG. 3.33: Rappel / Précision pour 3 points de fonctionnement du système en requêtage à base de MMC.

Cette expérimentation, récapitulée dans le tableau de la figure 3.34, met en avant les performances du système lorsque nous l'interrogeons sur la présence

Point de fonctionnement	R	P	$\sigma(R)$	$\sigma(P)$
$G \rightarrow 0$	39,8%	97,5%	1,6%	0,9%
G tel que $R \approx P$	64,7%	72,8%	4,1%	3,4%
$G \rightarrow 1$	81,9%	29,0%	3,5%	4,2%

FIG. 3.34: Performances obtenues par notre système de détection pour 3 points de fonctionnement en ne considérant qu'une requête à la fois.

Système	BDD	nb requêtes	BEP
Fischer	IAM database (929 lignes)	3421	≈ 30%
	GW database (675 lignes)	1067	≈ 40%
Rodriguez	Privée (630 courriers)	1	≈ 75%
Frinken	IAM database (929 lignes)	2807	≈ 72%
	GW database (675 lignes)	1067	≈ 65%

FIG. 3.35: BEP obtenus par 3 systèmes de *keyword-spotting* de la littérature avec leurs différentes conditions expérimentales. Fischer fait référence à [Fischer 2010], Rodriguez à [Rodríguez-Serrano 2009a] et Frinken à [Frinken 2012].

d'une séquence dans un document.

Lorsque l'on se place dans la première configuration, le système est capable de détecter de manière quasiment certaine près de 40% des séquences, sans fausse alarme. Dans la seconde configuration, nous nous placons proche du *break-even point* (*BEP*). Celui-ci se situe autour de 70%, c'est à dire que pour 10 requêtes unitaires, 7 vont être bien détectés, 3 séquences ne vont pas être retrouvée et 3 fausses alarmes vont être commises. En ce qui concerne la dernière configuration, le rappel est proche de 82% pour une précision voisine de 30% soit, pour 10 séquences à retrouver (correspondant à 10 requêtes), 8 qui le sont correctement, 2 non retrouvée et 19 fausses alarmes. Notons que les écarts-types en rappel et en précision sont plus faibles lorsque G est petit et ils augmentent lorsque l'on cherche à avoir améliorer le rappel.

À titre indicatif, nous donnons les BEP obtenus par quelques systèmes de *keyword-spotting* de la littérature présentés dans la section 2.4 [Fischer 2010, Rodríguez-Serrano 2009a, Frinken 2012] (cf. tableau de la figure 3.35). Nous devons nuancer ces résultats car les bases utilisées sont différentes à la notre : une base privée pour [Rodríguez-Serrano 2009a] et les bases IAM et George Washington (GW) pour [Fischer 2010, Frinken 2012]. Le protocole expérimental reste toutefois comparable puisqu'une seule requête est recherchée à la fois. Notons également que [Rodríguez-Serrano 2009a] ne donne des résultats que pour une seule requête, correspondant au mot *résiliation*. Ce mot est malgré tout recherché dans l'ensemble des documents de la base ce qui ajoute à la complexité de la recherche puisqu'il ne représente qu'environ $0,4\%$ des mots de la base et nécessite donc le rejet de $99,6\%$ de l'information contenue dans cette base. En ce qui concerne les systèmes [Fischer 2010] et [Frinken 2012], notons que les auteurs travaillent sur des lignes déjà segmentées et n'ont donc pas d'erreurs de segmentation des documents en lignes. Leur protocole expérimental implique la présence du mot recherché dans la ligne,

nous rappelons que dans cette expérimentation, nous nous sommes imposés la présence du mot à rechercher dans un document donné. Les trois protocoles sont finalement relativement comparables.

Nous constatons que les BEP obtenus par dans [Frinken 2012] et [Rodríguez-Serrano 2009a] sont du même ordre de grandeur, en gardant à l'esprit les différences entre nos protocoles. Notre système possède cependant l'avantage de pouvoir rechercher plusieurs séquences en même temps. Dans la section suivante, nous présentons les résultats obtenus lorsque nous travaillons sur des lexiques de mots clés à détecter.

3.4.3 Résultats en détection d'information

Au cours d'une seconde série d'expérimentations, les performances du système de détection d'information mis en œuvre sont évaluées sur la base de test créée à cet effet, sur des lexique constitués uniquement de mots clés. Nous rappelons que nous avons choisi des lexiques constitués de 10 mots appartenant au vocabulaire du document considéré complétées de $N - 10$ mots tirés aléatoirement dans le vocabulaire du reste de la base. Dans cette expérimentation, N prend les valeurs 10, 100 et 500. Les résultats sont illustrés par le schéma de la figure 3.36.

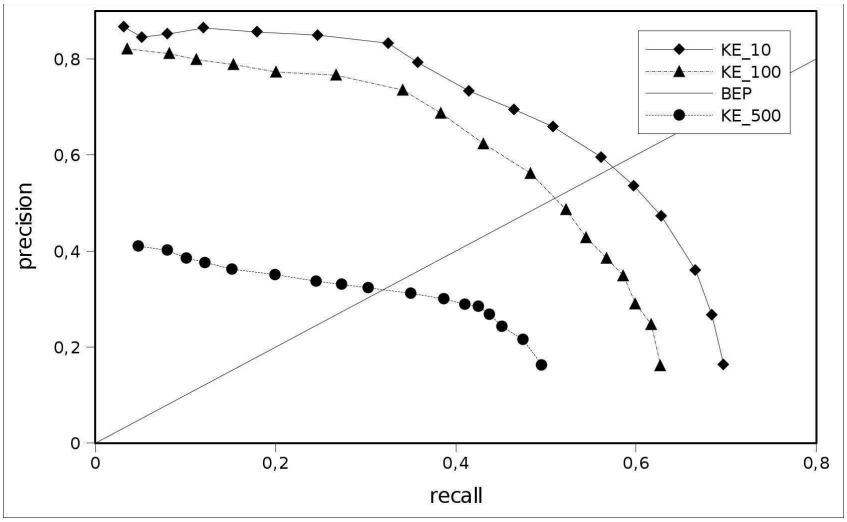


FIG. 3.36: Courbes rappel/précision de notre système d'extraction d'information pour différentes tailles de lexique de mots clés alphabétiques. Les tailles de lexique sont de 10 (résultats sur la courbe *KE_10*), 100 (*KE_100*) et 500 mots clés (*KE_500*).

L'augmentation de la taille des lexiques permet d'éprouver notre modèle

de détection lorsque la dimensionnalité du problème augmente. Cette comparaison met en avant la diminution générale des performances en rappel et en précision lorsque la taille du lexique de requêtes augmente. Des BEP [3] supérieurs à 50% sont toutefois à noter pour des tailles de lexique inférieures à 100 mots. Cette diminution est constatée de manière plus importante au niveau des précisions alors que des niveaux intéressants sont toujours atteints en rappel (50% atteint pour toutes les courbes). Ceci s'explique par la structure même de notre modèle d'extraction : en augmentant progressivement G, on force le système à extraire de plus en plus d'information. On fait augmenter le taux de bonnes détections et donc le rappel mais également grimper le taux de fausses détections.

Lexique	R_{max}	P_{max}	P_{moy}	BEP
KE_10	69,7%	86,7%	66,1%	57,5%
KE_100	62,2%	82,1%	57,1%	50,9%
KE_500	49,5%	41,1%	31,5%	32%

FIG. 3.37: Points caractéristiques de fonctionnement du système par rapport aux courbes de la figure 3.36.

Les points caractéristiques des courbes de la figure 3.36 sont répertoriés dans le tableau de la figure 3.37. On constate que la précision moyenne des points des courbes diminue fortement lorsque le nombre de requêtes devient plus important mais reste correcte (supérieur à 57%) pour des lexiques de 100 et 500. Une expérience antérieure sur un ensemble plus restreint de documents (seulement 20 documents) mais sur plus de taille de lexique mettait en avant la stabilité du système jusqu'à 200 requêtes et montrait une dégradation plus rapide des performances en détection à partir d'un lexique de 300 séquences.

3.4.4 Comparaison avec une tâche de lecture complète

Dans une seconde expérimentation, nous comparons notre approche de détection d'information avec une approche de lecture complète utilisant le modèle de ligne (cf. section 3.3.4). Trois tailles de lexique de reconnaissance sont utilisées pour la lecture complète et sont notés : *FR_ doc* pour un lexique restreint au vocabulaire du document courant, *FR_ 3200* pour le lexique composé de l'ensemble du vocabulaire de la base de test et *FR_ 25000* pour le lexique ouvert. Le lexique de requêtes à détecter est dans un premier temps fixé à 10 mots clés par documents, les résultats sont illustrés sur la figure 3.38.

[3] Le break-even point est le point de fonctionnement d'un système tel que le rappel est égal à la précision

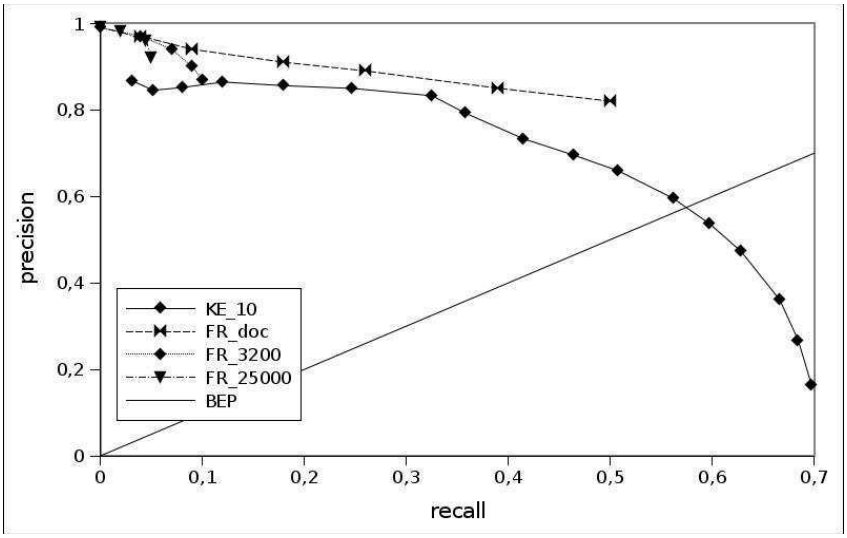


FIG. 3.38: Comparaison des systèmes de détection d'information et de lecture complète de documents dans le plan rappel/précision pour un lexique de requêtes à détecter fixé à 10 mots.

Nous constatons sur cette figure que :

- le système contraint par le plus petit lexique (représenté par la courbe *FR_ doc*) est plus performant que notre approche de détection pour des rappels inférieurs à 50%.
- le système contraint par un lexique intermédiaire (courbe *FR_ 3200*) est aussi plus performant pour des rappels inférieurs à 10%.
- le système travaillant avec un lexique ouvert (courbe *FR_ 25000*) est également plus performant pour des rappels inférieurs à 6%.

Pour tous les points dont le rappel est supérieur à ces valeurs, notre système de détection se comporte mieux que la reconnaissance complète. Notre approche permet donc d'atteindre des rappels bien supérieurs allant jusqu'à 80% pour une des précisions supérieures à 12%. Ces résultats sont à nuancer. En effet, se placer dans une configuration de reconnaissance où seul le vocabulaire du document courant est considéré n'est pas réaliste. Dans une configuration plus réaliste (lexique de reconnaissance à 3200 et 25000 mots), la précision des systèmes de lecture complète est supérieure à celle atteinte par notre système de détection mais le rappel reste plus faible. Par exemple, le rappel atteint 11% pour une précision de 86% sur la courbe *FR_ 3200* et 5% pour 92% en précision sur la courbe *FR_ 25000*. Avec notre approche de détection, même si la précision est inférieure (autour de 80%), nous arrivons à atteindre 80% de rappel pour une précision de 12%. Un *break-even point* proche de 60% est ainsi observé.

Un lexique de requêtes de 100 mots clés est ensuite considéré. Les résultats correspondant sont représentés sur la figure 3.39.

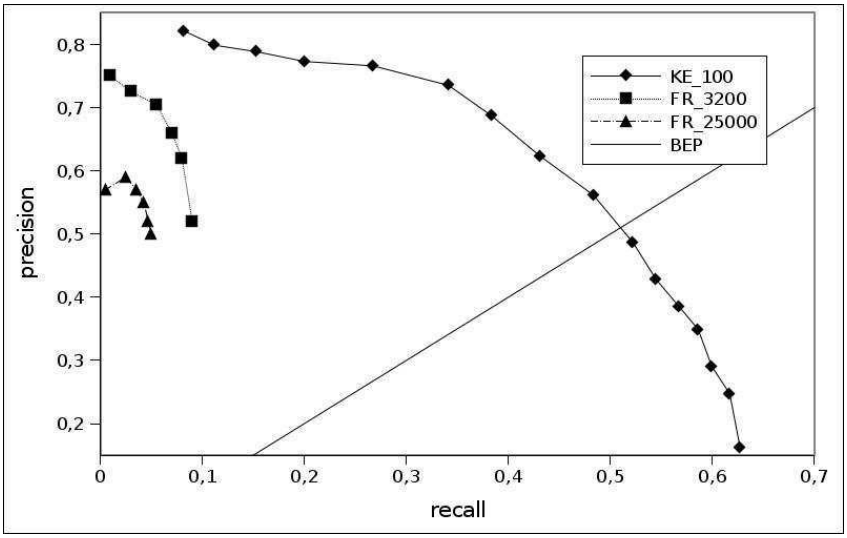


FIG. 3.39: Comparaison des systèmes de détection d'information et de lecture complète de documents dans le plan rappel/précision pour un lexique de requêtes à détecter fixé à 100 mots.

Les performances atteintes s'avèrent globalement plus faibles lorsqu'un lexique de taille plus importante est considéré. Les confusions de reconnaissance dans une approche de lecture complète de documents apparaissent clairement sur cette figure. Ceci se traduit par des baisses sensibles de la précision à respectivement 70% et 60% pour les courbes *FR_3200* et *FR_25000*. Nous pouvons également noter que le rappel est toujours en dessous de 10%. En ce qui concerne notre modèle de détection, les performances sont également en baisse mais ne s'effondrent pas : un break-even point à 45% est obtenu.

La taille du lexique de requêtes à détecter est enfin portée à 500 mots. Les résultats sont présentés dans la figure 3.40. Cette figure met en avant les avantages de notre stratégie avec une taille de lexique de requêtes supérieure. En effet, on constate une chute commune des rappels et précisions pour la stratégie de reconnaissance complète de documents. Ceci s'explique par le fait que les confusions locales au niveau mot sont nombreuses. Lorsque les scores de reconnaissance sur les mots détectés sont seuillés, les scores correspondants à des mauvaises détections sont supérieurs à ceux des mots

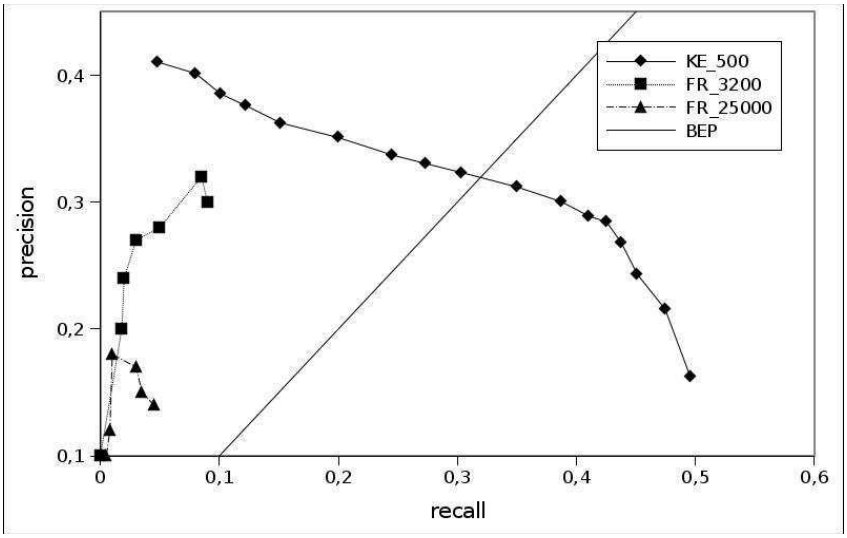


FIG. 3.40: Comparaison des systèmes de détection d'information et de lecture complète de documents dans le plan rappel/précision pour un lexique de requêtes à détecter fixé à 500 mots.

correctement détectés. On assiste donc à une baisse commune du rappel et de la précision vers des valeurs proches de zéro.

Le fait que l'approche de lecture complète ne soit pas contrainte par l'utilisation d'un modèle de langage de type bigramme de mots voire tri-gramme de mots explique en partie ce phénomène. Toutefois, l'estimation de tels modèles de langage est hasardeuse dans notre cas à cause de la présence d'un vocabulaire très varié (près 3200 séquences différentes pour moins de 8000 séquences au total) et ded l'unicité des séquences numériques et alphanumériques. À cela s'ajoute le fait que seule la première hypothèse de reconnaissance est considérée pour le calcul des résultats. Un module de post-traitements prenant en considération une liste des meilleurs résultats permettrait sans doute d'améliorer les performances [Bertolami 2006]. La lecture complète n'étant pas l'objet de notre étude et dans le but de se placer dans des conditions expérimentales proches, une telle liste d'hypothèses n'a pas été considérée.

3.4.5 Comparaison sur différents types d'informations à détecter

Dans une troisième série d'expérimentations, nous testons notre approche de détection sur des données de types différents et non plus uniquement sous la forme de séquences alphabétiques. La détection des séquences numériques

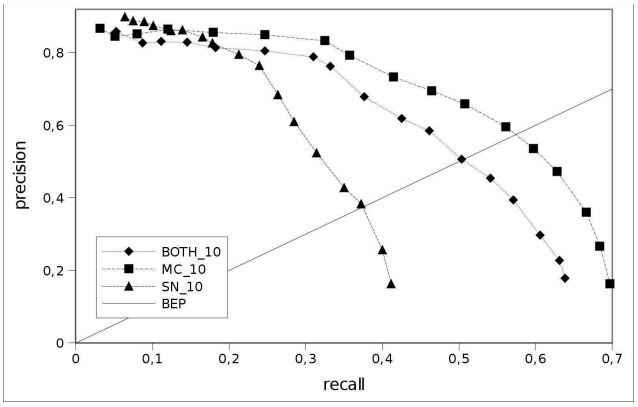
et alphanumériques présentes dans les courriers de test est ainsi évaluée. Trois types de lexique de requêtes sont donc considérés :

- un lexique alphabétique (déjà présenté) que nous nommons **MC**. Celui-ci est uniquement composé de mots clés sous les mêmes contraintes de longueur précédemment décrites, c'est-à-dire avec un minimun de 5 caractères par mot.
- un lexique comprenant uniquement des séquences numériques que nous nommons **SN**. Par séquence numérique, nous faisons référence à des champs constitués de chiffres ou d'un mélange de symboles (séparateurs tels que − ou / présents notamment dans les numéros de téléphone) et de chiffres et dont la longueur est supérieure ou égale à 5 caractères. Ceci exclut donc les numéros de rue seuls mais permet de considérer les codes postaux et les numéros de téléphone par exemple.
- un lexique comprenant un mélange de mots clés et de séquences ainsi que les séquences comprenant à la fois des lettres et des chiffres lorsque présentes dans un document. Nous appelerons ce lexique **BOTH**.

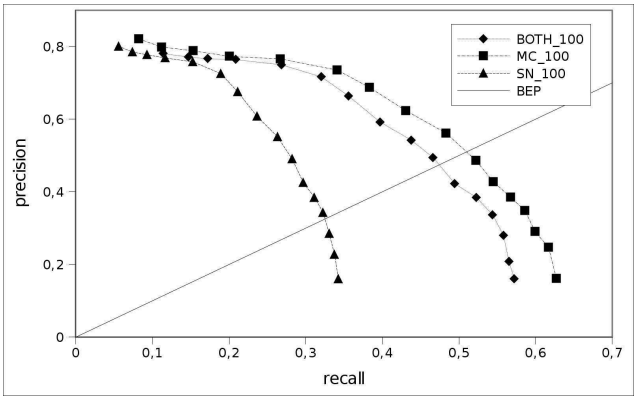
Le protocole de construction des lexiques de séquences à détecter est légèrement modifié pour prendre en compte les nouveaux types de données (séquences numériques et alphanumériques). Étant donné que les séquences numériques apparaissent rarement dans un document, le lexique de requêtes correspondant au cas SN est composé de toutes les séquences numériques du courrier considéré complétées par des séquences apparaissant dans les autres courriers de la base en évitant les possibles doublons. Il est à noter que les documents de la base contiennent en moyenne 3 séquences numériques. En ce qui concerne le lexique correspondant à l'expérimentation $BOTH$, les séquences alphanumériques (lorsqu'il y en a) sont isolées depuis l'annotation du courrier considéré et insérées dans le lexique de requêtes. Nous ajoutons ses séquences numériques et complétons de mots appartenant à son vocabulaire pour arriver à un total de 10 séquences à détecter. Éventuellement, ce lexique est complété de $N - 10$ mots et séquences diverses tirés aléatoirement dans le vocabulaire de la base pour former un lexique de N requêtes comme lors des expériences précédentes.

Les résultats en détection d'information pour les 3 différents types de requêtes sont illustrés par les graphes de la figure 3.41. Les points caractéristiques des courbes sont récapitulés dans le tableau de la figure 3.42. Nous présentons également des exemples de détection d'information dans des images de courriers manuscrits avec les figures 3.43, 3.44 et 3.45.

a)



b)



c)

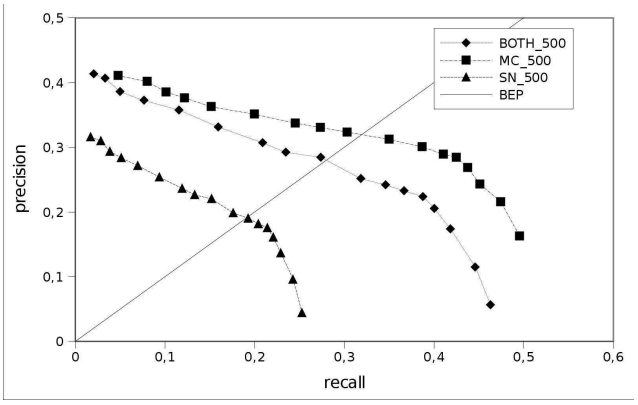


FIG. 3.41: Courbes rappel/précision avec différents tailles et types de lexique.

À l'instar des expérimentations menées avec des requêtes uniquement alphabétiques, de bonnes précisions sont atteintes lorsque l'information recherchée est constituée de 10 séquences : de l'ordre de 90% pour 10 séquences numériques (noté **SN_10** sur le graphe 3.41 et dans le tableau 3.42) et supérieures à 85% pour les expérimentations **MC_10** et **BOTH_10**. Les rappels maximum avoisinent les 70% pour les courbes **MC_10** et **BOTH_10** mais sont inférieurs à 40% pour tous les points de la courbe

Lexique	R_{max}	P_{max}	P_{moy}	$_{BEP}$
MC_10	69,7%	86,7%	61,1%	57,5%
SN_10	41,1%	89,8%	62,6%	37,5%
BOTH_10	63,9%	85,8%	56,4%	51%
MC_100	62,2%	82,1%	57,1%	50,9%
SN_100	34,3%	80%	54,7%	32,6%
BOTH_100	52,7%	78,1%	53,9%	47,3%
MC_500	49,5%	41,1%	31,5%	32%
SN_500	25,3%	31,6%	20,7%	19,1%
BOTH_500	46,3%	41,1%	27,3%	28,2%

FIG. 3.42: Points de fonctionnement du système pour différents types et tailles de lexique.

SN_10. Les précisions moyennes tournent autour de 60% pour les trois courbes alors que les BEP sont respectivement de 57.5%, 51% et 37.5% pour les expérimentations **MC_10**, **BOTH_10** et **SN_10**.

Lorsque le nombre de requêtes est porté à 100 séquences pour les 3 types considérés, une légère baisse des performances est observée. Pour les expérimentations **MC_100**, **BOTH_100** et **SN_100** :

- les rappels maximum diminuent respectivement de 7.5, 11.2 et 6.8 points soient des baisses de performances autour de 15% par rapport aux expérimentations conduites avec des ensembles de 10 séquences à rechercher.
- les précisions maximales diminuent respectivement de 4.6, 7.7 et 9.8 points équivalentes à des baisses inférieures à 10%
- les précisions moyennes sont relativement stables autour de 55% (diminutions de 5 points en moyenne)
- les BEP sont autour de 50% pour les expérimentations **MC_100** et **BOTH_100** et baisse à 30% pour la courbe **SN_100**

Enfin, lorsque la quantité d'information à rechercher en une passe est portée à 500 séquences, les performances du système de détection connaissent une plus forte baisse :

- les rappels maximum enregistrent une baisse respectivement de 12.7, 7.4 et 6.8 points pour les expérimentations **MC_500**, **BOTH_500** et **SN_500** par rapport aux expérimentations **MC_100**, **BOTH_100** et **SN_100** soient des diminutions respectives de 20%, 14% et 36%.
- les précisions maximales diminuent quant à elles de respectivement de

4.6, 7.7 et 9.8 points soit des baisses proches de 50%
- les précisions moyennes chutent à des valeurs autour de 30% pour les courbes **MC_500** et **BOTH_500** et 20% pour **SN_500**
- enfin, en ce qui concerne les BEP, leurs valeurs sont relativement proches des précisions moyennes, soient respectivement 32%, 28.2% et 19.1% pour les expérimentations **MC_500**, **BOTH_500** et **SN_500** correspondant à des baisses de plus de 40% par rapport aux expérimentations menées sur des ensembles de 100 séquences

De manière générale, nous constatons que la précision du système diminue plus rapidement que le rappel lorsque le nombre de requêtes augmente. C'est particulièrement le cas lorsque le nombre de séquences à rechercher est porté à 500 séquences et que celles-ci sont de nature numérique. Lorsque l'information à détecter consiste en un mélange de séquences numériques, alphabétiques et alphanumériques, les performances diminuent moins brutalement par rapport à l'alphabétique mais restent largement plus hautes que pour les séquences numériques.

Nous avons vu précédemment que les approches de modélisation de séquences numériques isolées les plus efficaces sont basées sur une segmentation explicite pilotée par la reconnaissance des chiffres isolés à l'aide d'un classifieur discriminant [Ha 1997, Chatelain 2006a]. Ce n'est pas le cas dans notre approche où la segmentation est implicite et ou les observations sont classifiées par des modèles de mélanges de Gaussiennes peu discriminantes. Plusieurs explications peuvent être données et liées, notamment, à la remarque précédente à savoir que pour bien reconnaitre et extraire une séquence numérique, il faut être capable de la localiser dans un premier temps dans le document comme par exemple dans les travaux de [Chatelain 2006a] et d'en localiser les chiffres dans un second temps.

Également, la détection des lignes de base n'est pas forcément nécessaire à la reconnaissance de chiffres manuscrits et donc de séquences numériques. Par contre, elles sont quasi indispensables à la bonne reconnaissance de mots manuscrits. Celles-ci sont utilisées pour caractériser les formes contenues dans les lignes. Or, la méthode de détection implémentée entraine une variabilité dans la position des lignes de base sur des images contenant des séquences numériques ce qui a pu entrainer une modélisation moins fine des chiffres comparativement aux lettres. Ceci explique en partie la baisse des performances générales en extraction observée sur les courbes rappel/précision correspondantes de la figure 3.41. La modélisation plus fine et plus spécifique des séquences numériques en fonction de leur syntaxe par exemple [Chatelain 2006a] permettrait de faire grimper les performances en

extraction de séquences numériques.

Enfin, nous comparons notre approche à celles de Koch [Koch 2006] et de Chatelain [Chatelain 2006b], nos travaux s'inscrivant dans la suite de ceux-ci.

Les travaux de [Chatelain 2006b] visait à extraire des séquences numériques en s'appuyant sur leur syntaxe. La difficulté réside dans la localisation des champs numériques ainsi que dans leur reconnaissance, non dirigée par un lexique de séquences. Notre approche de détection, consistant à rechercher la présence ou non de séquences numériques sous la forme de requêtes dans des courriers manuscrits, est difficilement comparable à ces travaux étant donné les différences dans les objectifs de nos systèmes et les protocoles expérimentaux mis en place. Toutefois, notons que la reconnaissance des chiffres à l'aide d'un classifieur discriminant de type réseau de neurones permet d'atteindre des performances intéressantes en extraction. Le couplage d'un tel classifieur avec les MMC modélisant que nous avons utilisé forme une piste d'optimisation intéressante pour notre système.

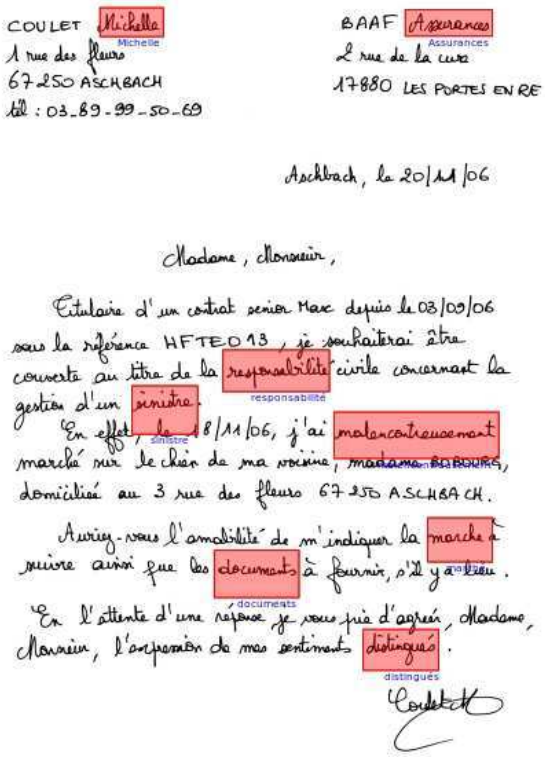
Cette piste peut également être justifier par rapport aux travaux de [Koch 2006]. Ceux-ci visaient à détecter des mots clés pour catégoriser des courriers manuscrits et donc en extraire le sujet. À cet effet, l'auteur a montré que les performances en catégorisation augmentaient lorsqu'il ne cherchait pas à détecter les mots clés de manière certaine mais uniquement leur racine en effectuant un prétraitement de *stemming* sur ces mots clés. Malgré le fait que nous cherchons à détecter des mots clés *entiers*, de meilleurs précisions sont obtenus (40% maximum dans les expérimentations de Koch) mais une baisse plus rapide des précisions est observée pour notre système. L'utilisation d'un classifieur discriminant pour reconnaitre les lettres segmentées peut expliquer les meilleures précisions. Nous allons donc chercher à améliorer la reconnaissance locale des observations pour tenter d'améliorer les performances globales de notre système à la fois en précision et en rappel mais aussi en détection de séquences numériques et alphanumériques qui sont en deça de celles des séquences alphabétiques classiques (mots clés).

3.5 Conclusions

Dans ce chapitre, un système complet de détection d'information dans les documents manuscrits a été présenté. Il se base sur l'utilisation du modèle générique de ligne. Dans un premier temps, les lignes de texte sont isolées les unes des autres dans l'image du courrier considérée. Ces lignes sont ensuite prétraitées dans le but d'uniformiser l'écriture manuscrite qu'elles contiennent. Elles sont enfin segmentées implicitement et caractérisées

par un ensemble de primitives statistiques et structurelles. Le moteur de reconnaissance décodant les séquences de trames produites pour chacune des lignes se base sur le formalisme des modèles de Markov cachés et, contraint par le modèle de ligne, permet de détecter de l'information définie par rapport aux besoins d'un utilisateur. Au cours des expérimentations menées, nous avons simulé ces besoins par des tirages aléatoires dans le vocabulaire des documents de notre base de test et avons montré par l'exemple l'intérêt et le potentiel de notre approche en détection d'informations alphabétiques, numériques et alphanumériques. À notre connaissance, aucun système n'est à l'heure actuelle capable de travailler sur des types de données variées et très peu sont capables d'extraire de l'information en se basant sur des lignes de textes entière [Frinken 2012, Koch 2006, Chatelain 2006a]. Il apparait difficile de comparer les performances de ces systèmes car les objectifs et donc les protocoles expérimentaux permettant de mesurer leurs performances diffèrent : extraction d'un seul mot clé [Fischer 2010, Frinken 2012], extraction d'un ensemble de mots clés pour la catégorisation de documents [Koch 2006], extraction de séquences numériques contraintes par leur syntaxe [Chatelain 2006a].

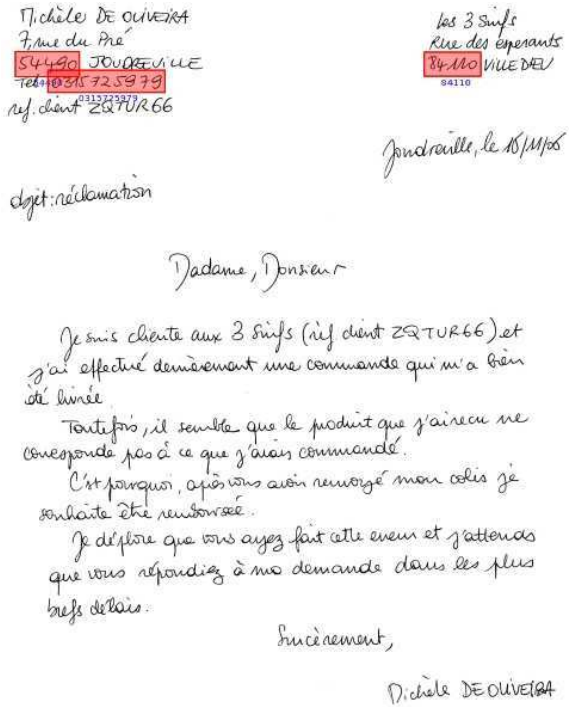
Le protocole expérimental décrit à la suite de la présentation du système a permis de mesurer les performances du système dans notre cadre de détection d'information hétérogène. Le modèle de ligne permet d'atteindre des rappels intéressants ce qui met en valeur les capacités de détection du système. Par contre, lorsque la quantité d'information que l'on souhaite extraire augmente, précision et rappel baissent sensiblement. Également, les performances concernant la détection de séquences numériques s'avèrent plus faibles que celles obtenues en détection de mots clés, en rappel comme en précision. Or, les MMC sont connus pour être de très bons modélisateurs de séquences mais moins efficaces en ce qui concerne la discrimination locale des formes. La faiblesse des résultats en détection de séquences numériques et les baisses rapides de performances en précision lorsque la quantité d'information à rechercher augmente s'expliquent notamment par l'approche de segmentation implicite retenue et les carences des modèles de mélanges de Gaussiennes en discrimination des observations. Au cours du chapitre suivant, nous allons donc nous focaliser sur la discrimination de ces formes locales afin d'améliorer les performances globales du système en détection d'information.



Requêtes alphabétiques

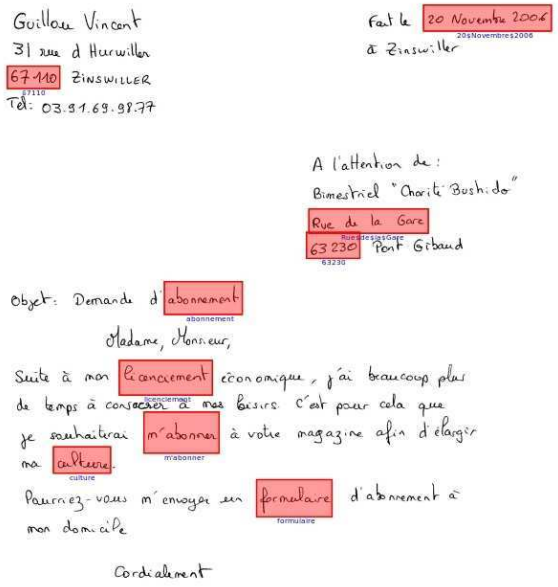
BD Michelle
BD Assurances
BD responsabilité
BD sinistre
BD malencontreusement
BD documents
BD marche
BD distinguées
FR réponse
FR amabilité

FIG. 3.43: Exemple de détection d'information avec 10 requêtes de type mot clé. Les 8 premiers mots sont correctement détectés (**BD**). Les deux derniers sont incorrectement rejetés (**FR**).



Requêtes numériques	
BD	54490
BD	84110
BD	0315725979
BR	26600
BR	76800
BR	18800
BR	0375426430
BR	0475747269
BR	0684121672
BR	0631557013

FIG. 3.44: Exemple de détection d'information avec 10 requêtes de type séquence numérique. Les 3 premières séquences, à détecter, le sont correctement (**BD**). Les 7 autres sont correctement rejetées (**BR**).



Requêtes quelconques	
FR	Vincent
BD	abonnement
BD	licenciement
BD	culture
BD	formulaire
BD	m'abonner
BD	20 Novembre 2006
BD	Rue de la Gare
BD	67110
BD	63230
FR	03-91-69-98-77

FIG. 3.45: Exemple de détection d'information avec 10 requêtes quelconques. Le premier mot (*Vincent*) et la dernière séquence (*le numéro de téléphone*) sont incorrectement rejetés (**FR**). Les autres séquences sont correctement détectées (**BD**).

Chapitre 4

Modélisation profonde et modèles de Markov cachés pour l'extraction d'information

Sommaire

4.1 Introduction

Dans le chapitre précédent, nous avons présenté un système complet de détection d'informations alphanumériques dans des documents manuscrits non contraints à base de modèles de Markov cachés. Nous avons présenté des résultats prometteurs en contraignant la reconnaissance par un modèle de ligne intégrant une ou un ensemble de séquences (modélisant des requêtes) à détecter, mis en concurrence avec un modèle de remplissage. Précédemment utilisé en reconnaissance de la parole pour détecter des mots hors vocabulaire [Bazzi 2000] ou en reconnaissance de l'écriture manuscrite pour la catégorisation de documents [Koch 2006], nous avons montré qu'il pouvait être intéressant en détection d'information. Contrairement aux autres systèmes de détection d'information [Frinken 2012, Rodríguez-Serrano 2009a], notre système est capable de rechercher un lexique pouvant contenir plusieurs centaines de séquences variées (alphabétiques ou numériques) en un seul traitement. Bien que performants en modélisation, les MMC sont aujourd'hui dépassés en classification de séquences [Graves 2009]. Nous pensons que l'amélioration des performances de notre système de détection d'information passe par une meilleure discrimination locale des trames.

Dans ce chapitre, nous allons tenter de rendre plus robuste la discrimination locale de l'écriture et proposons de substituer les mélanges de Gaussiennes classiquement embarqués dans les états des MMC par des modèles plus discriminants tels que les réseaux de neurones [Hinton 2006a], les SVM [Vapnik 1998] ou encore les forêts aléatoires [Breiman 2001]. Notre choix s'est orienté vers les réseaux de neurones profonds [Bengio 2007c] pour les raisons suivantes. Premièrement, ils ont permis d'obtenir les meilleures performances sur de nombreuses bases de référence (MNIST[1], CIFAR[2], Reuters[3], etc.), en détronant notamment les SVM sur de nombreux problèmes [Xue 2008, Rifai 2011b]. Deuxièmement, ils permettent d'obtenir par l'apprentissage non supervisé d'une partie de leur structure, une représentation de haut niveau des données permettant une discrimination efficace. Par ailleurs, de nombreux travaux ont montré l'efficacité des combinaisons neuro-markoviennes sur des problèmes de reconnaissance de séquences (reconnaissance de l'écriture [Gorski 2001, Poisson 2005, Espana-Boquera 2011] et de la parole [Bengio 1995, Bourlard 1998, Trentin 2001]).

Au cours de ce chapitre, nous allons montrer comment combiner ces ré-

[1] http://yann.lecun.com/exdb/mnist/
[2] http://www.cs.toronto.edu/~kriz/cifar.html
[3] http://about.reuters.com/researchandstandards/corpus/

seaux profonds avec les MMC pour améliorer les performances de reconnaissance de notre système de détection d'information. Nous présentons dans un premier temps les réseaux de neurones « classiques », puis nous présenterons les méthodes récentes permettant l'apprentissage des structures dites « profondes », c'est à dire des réseaux de neurones comportant un grand nombre de couches cachées. Nous présenterons ensuite les méthodes de combinaison de réseaux de neurones avec les modèles de Markov cachés et introduirons l'implémentation de notre modélisation neuro-markovienne. Nous terminerons par une comparaison du système obtenu avec le système présenté dans le chapitre précédent embarquant les modèles classiques de mélanges de Gaussiennes.

4.2 Réseaux de neurones artificiels et réseaux profonds

Un réseau de neurones artificiel est un classifieur statistique discriminant [Jain 2000]. Ces classifieurs ont permis d'atteindre de bonnes performances de classification depuis le fin des années 80 sur des tâches de reconnaissance de l'écriture manuscrite grâce à des algorithmes d'apprentissage efficaces [LeCun 1987, LeCun 1990]. Ces algorithmes se basent sur des méthodes itératives à gradient qui convergent vers un minimum en se basant sur la direction donnée par le gradient de l'erreur. Elles sont efficaces mais ne garantissent pas d'atteindre le minimum global.

Les réseaux de neurones sont constitués d'un ensemble de fonctions élémentaires interconnectées entre elles : les neurones formels [McCulloch 1943] illustré par la figure 4.1. Un neurone isolé effectue une somme pondérée des entrées soumise à une fonction d'activation non linéaire comme la fonction de *Heaviside* dans le cas du perceptron [Rosenblatt 1958] ou ses variantes qui utilisent les fonctions tangente hyperbolique ou sigmoïde par exemple.

L'agencement d'un ensemble de neurones formels sous la forme d'un réseau permet de considérer des problèmes non linéaires complexes en classification ou en regression tels que ceux liées à la reconnaissance de l'écriture manuscrite [LeCun 1995, Collobert 2008, Fernández 2007] ou à la reconnaissance de la parole [Trentin 2001, Bourlard 1998]. L'agencement de ces neurones permet également de décrire différentes topologies de réseaux de neurones, les connaissances relatives à un problème donné étant contenues dans les connexions entre neurones. On distingue les réseaux de neurones à passe avant (ou *feedforward* en anglais, cf. figure 4.2), les réseaux de neurones récurrents et les réseaux de neurones totalement interconnectés. Nous présentons ces grandes familles de réseaux dans cette section.

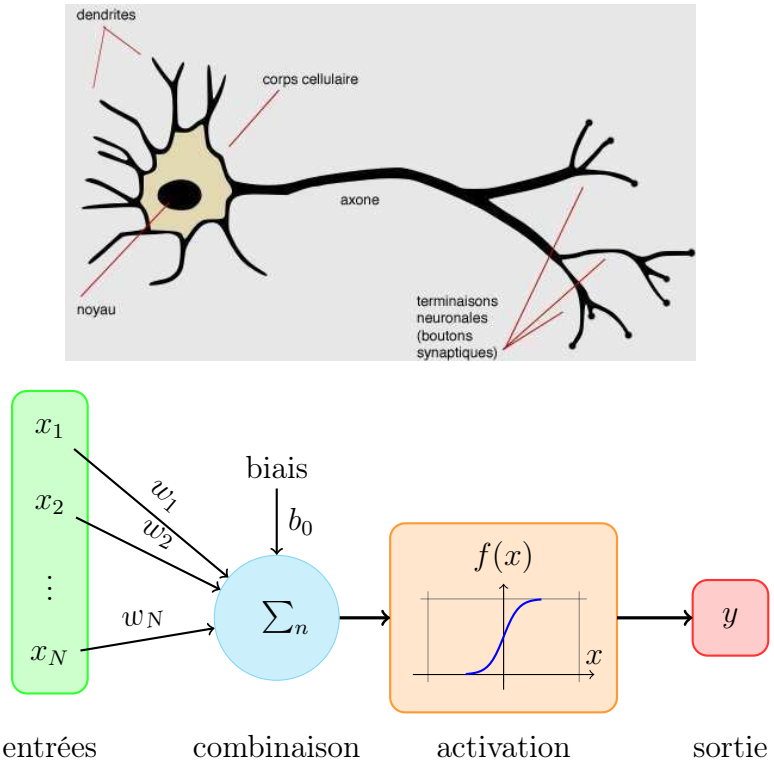


FIG. 4.1: Illustration d'un neurone biologique et de son approximation mathématique avec $\mathbf{x_n}$ les entrées du neurone, $n \in [1, N]$, $\mathbf{w_n}$] les poids relatifs aux entrées, $\mathbf{b_0}$ le biais associé à ce neurone qui caractérise son état au repos, $\mathbf{f(x)}$ la fonction d'activation du neurone, $\mathbf{y}$ la sortie du neurone calculée à l'aide de la fonction d'activation à partir de la combinaison des entrées et de leur poids.

4.2.1 Types de réseaux de neurones

Les types de réseaux dépendent directement de leur topologie, c'est à dire de l'agencement de leurs neurones. On distingue classiquement :

les réseaux de neurones à passe avant (ou *feedforward*) qui sont organisés en couches s'activant séquentiellement. Ils ont été introduits dans [Minsky 1969] sous la forme du perceptron multicouches (*PMC* ou *MLP* pour *MultiLayer Perceptron*) et sont formés d'un empilement de perceptron associés en couches [Rosenblatt 1958]. Les neurones d'une couche sont totalement interconnectés avec ceux des couches précédente et suivante. Le MLP est un des réseaux les plus communs au même titre que le réseau à fonctions de bases radiales ou *RBF* pour *Radial Basis Function*. Les réseaux de neurones profonds [Bengio 2007c, Hinton 2006b] entrent

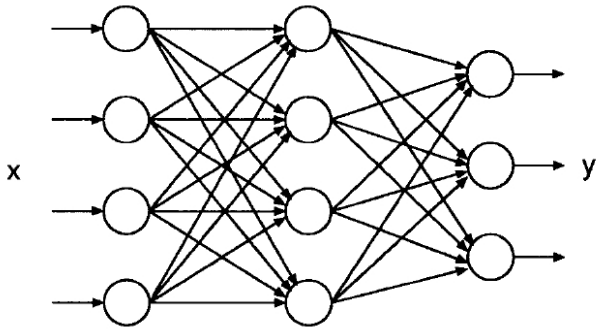


FIG. 4.2: Illustration d'un réseau de neurones multicouches à passe avant. Il se caractérise par la propagation des entrées x vers les sorties y à travers les couches cachées de neurones. Figure extraite de [Cho 1997]

également dans cette catégorie.

les réseaux de neurones récurrents [Poisson 2005] : ils ont la particularité d'avoir des connexions de rebouclage des sorties vers les entrées. Ceci permet de prendre en compte un contexte spatial, comme par exemple avec le *SDNN*[4] [Matan 1992, Bengio 1995] (cf. figure 4.3), un contexte temporel avec le *TDNN*[5] [Lang 1990] ou le *LSTMN* pour *Long-Short Term Memory Network* [Graves 2009], ou bien les deux dans une même architecture avec par exemple le *MS-TDNN*[6] [Jaeger 2000] (cf. figure 4.4).

les réseaux totalement interconnectés de type Hopfield [Hopfield 1982] et machine de Boltzmann [Hinton 1986] (cf. figure 4.5) permettent d'apprendre des répresentations internes pour les données d'entrée et sont donc capables, en théorie, de modéliser n'importe quel problème combinatoire. Cependant, ils n'ont jamais été utilisés avec succès dans une application concrète.

Le dimensionnement de telles architectures (agencement des neurones, nombre de couches, nombre de neurones par couche) ainsi que leur apprentissage sont deux difficultés. S'il n'existe pas de solution automatique en ce qui concerne le premier point [Arnold 2010], il a été montré qu'un réseau à passe avant avec une seule couche cachée permettait d'approximer n'importe quelle fonction sous la contrainte d'un nombre infini de neurones dans la couche cachée et d'une base d'apprentissage infinie [Richard 1991]. Certains résultats

[4] *Space Displacement Neural Network*
[5] *Time Delay Neural Network*
[6] Multi-State Time Delay Neural Networks

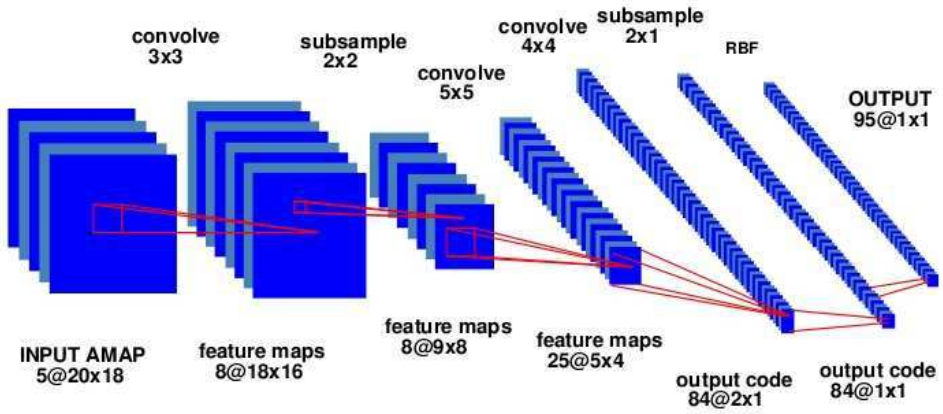


FIG. 4.3: Illustration du SDNN embarqué dans le système de lecture LeRec [Bengio 1995]

mettent toutefois en avant l'intérêt de considérer deux ou plusieurs couches cachées [Bengio 1995, Kenyon 1998, Jaeger 2000, Utgoff 2002]. En effet, les réseaux obtenus en empilant plusieurs couches cachées définissent des frontières de décisions plus précises et plus complexes dans l'espace de réprésentation. Par contre, le choix du nombre de neurones cachés par couche est un problème d'autant plus difficile. Dans le cas d'un réseau à passe avant à une seule couche cachée, le nombre de neurones dans sa couche cachée est généralement déterminé à l'aide d'heuristiques comme par exemple la moyenne arithmétique ou géométrique des nombres d'entrée et de sortie.

En ce qui concerne l'apprentissage d'un réseau, une solution efficace a été proposée dans les années 80 avec l'algorithme de rétropropagation du gradient de l'erreur[7] [Rumelhart 1986, LeCun 1987]. Cet algorithme a pour but de régler les poids d'un réseau *feedforward* de manière itérative en cherchant à minimiser l'erreur globale de classification sur l'ensemble des exemples de la base d'apprentissage en ajustant les poids du réseau afin de faire tendre la sortie vers la valeur désirée au regard de ses entrées. L'apprentissage d'un réseau devient cependant difficile lorsque le nombre de couches cachées est grand. En effet, l'utilisation de la rétropropagation du gradient pour entrainer conjointement un grand nombre de couches de neurones est délicate à cause de la baisse de l'énergie des erreurs tout au long de ce réseau lors de la rétropopagation : plus les couches cachées sont éloignées de la

[7]Les origines des algorithmes d'apprentissage à base de rétropropagation sont assez délicates à établir puisqu'ils n'ont pas connu un succès immédiat. Les travaux de [Werbos 1974] font ainsi partie des précurseurs alors que ceux de [Rumelhart 1986] marque le début de l'utilisation massive de réseaux neuronaux appris à l'aide de l'algorithme de rétropropagation du gradient.

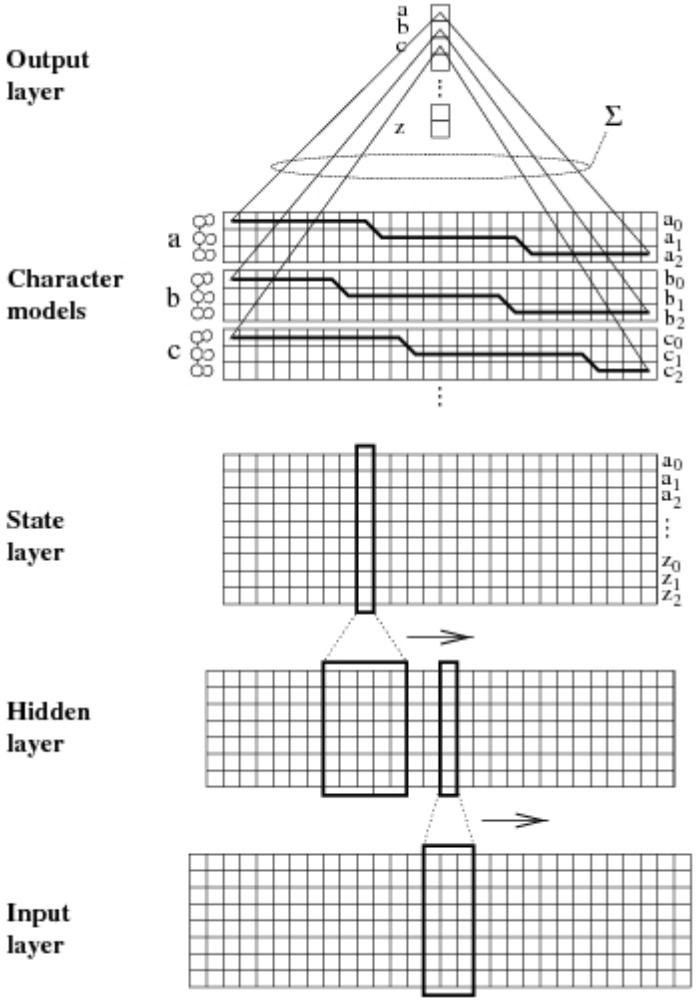


FIG. 4.4: Illustration du MS-TDNN embarqué dans le système de lecture NPen++ [Jaeger 2000]

couche de sortie, moins elles seront affectées par la mise à jour des poids pendant l'apprentissage. L'initialisation des poids du réseau est également un point critique qui peut entrainer le blocage de l'apprentissage dans un minimum local et par conséquent un problème de généralisation [Larochelle 2009].

Les réseaux de neurones à passe avant de type MLP ou RBF sont toujours largement utilisés sur des problèmes classiques de reconnaissance de formes mais sont toutefois surpassés en termes de performances, par les SVM notamment [Camastra 2007]. Les algorithmes d'apprentissage développés récemment pour les architectures profondes [Hinton 2006b, Bengio 2007c]

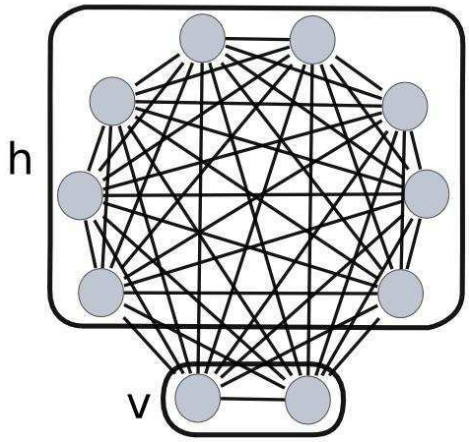


FIG. 4.5: Illustration d'une machine de Bolzmann extraite de [Arnold 2010]. *v* représente les unités visibles et *h* les unités cachées.

ont toutefois permis de surpasser les SVM sur de nombreux problèmes[8] [Xue 2008, Rifai 2011b]. Les réseaux de neurones profonds se basent sur l'idée (assez ancienne) qu'entrainer un réseau de neurones possédant un grand nombre de couches cachées est un bon moyen pour :

i) apprendre des frontières de décision complexes entre les classes d'un problème,

ii) obtenir des représentations cachées intermédiaires de haut niveau des données favorisant leur discrimination.

Nous présentons maintenant plus en détail les réseaux de neurones profonds.

4.2.2 Réseaux de neurones profonds

4.2.2.1 Principe et structure des réseaux profonds

Les architectures profondes possèdent une structure de type *feed-forward* avec de nombreuses couches cachées. La différence avec des réseaux plus classiques réside dans la façon dont les paramètres des couches cachées sont appris. En effet, lorsqu'un grand nombre de couches cachées sont empilées dans un réseau, la rétropropagation classique des erreurs n'a pas assez d'énergie pour affecter les poids du réseau entier, et en particulier ceux des couches proches des entrées. La stratégie d'apprentissage relative aux réseaux de neurones profonds vise donc à pré-apprendre les couches cachées individuellement et de

[8] cf. performances atteintes sur la base MNIST http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations

manière non supervisée. L'apprentissage d'une structure complète à Λ couches se fait ainsi en deux étapes :

- les $\Lambda - 1$ permières couches (les couches cachées) sont successivement apprises de manière non supervisée (les méthodes sont décrites ci-après) et itérativement bloquées puis empilées depuis l'étage d'entrée jusqu'à la couche de sortie du réseau. Cette étape est appelée **pre-training** et les couches sont dites **couches de modélisation**. En prenant directement les pixels en entrée du réseau, des représentations de haut niveau sont obtenues ce qui peut être apparenté à un processus d'inférence de caractéristiques.
- la dernière couche, couche de sortie du réseau, est ensuite ajoutée au réseau afin d'effectuer la discrimination. L'estimation de ses poids se fait en libérant les paramètres de toutes les couches cachées qui la précèdent et en effectuant une descente du gradient sur l'ensemble du réseau. Cette étape permettant d'apprendre la **couche de décision** pour discriminer les données à partir des représentations de haut niveau est appelée **fine-tuning**.

Dans leur forme d'origine, les architectures profondes sont constituées de couches de **machines de Boltzman restreintes** ou *Restricted Boltzman Machine* (RBM) et sont appelées réseaux à croyance profonde ou *Deep Belief Networks* (DBN) [Hinton 2006a, Arnold 2010]. Par la suite, une alternative a été proposée en substituant les RBM par des couches d'**auto-encodeurs** (AE) également appelés auto-associateurs. Un tel réseau est communément dénommé réseau de neurones profond, nous utiliserons l'abbréviation *DNN*[9] [Bengio 2007a, Vincent 2008].

Un AE est composé d'un étage d'entrée et de 2 couches de neurones : l'étage d'entrée a une dimension égale à la dimension des données d'entrée, la seconde couche est cachée, son nombre de neurones est généralement inférieur à la dimension des données d'entrée et la couche de sortie, visible, est constituée d'un nombre de neurones égal à la dimension des entrées. Les connexions entre l'étage d'entrée et la couche cachée permettent d'**encoder** les données d'entrée en une représentation cachée alors que les connexions de la couche cachée vers le couche de sortie permettent de décoder la représentation cachée en cherchant à reconstruire les entrées et donc de vérifier la qualité de l'encodage realisé par l'AE. L'apprentissage d'un auto-encodeur se fait à l'aide de la

[9] de l'anglais *Deep Neural Networks*. Notons que les réseaux à croyance profonde peuvent également être appelés réseaux de neurones profonds. Dans la fin du document, ce que nous appellerons réseaux de neurones profonds fera référence aux DNN tels que nous venons de les définir

rétropropagation du gradient de l'erreur de reconstruction. Nous reviendrons sur sa procédure d'apprentissage après avoir introduit les réseaux de neurones profonds de manière plus formelle.

4.2.2.2 Formalisme d'un DNN

Un réseau de neurones profonds contenant Λ couches d'auto-encodeurs peut être défini ainsi : chaque couche calcule un vecteur de sortie notée $\mathbf{H}^\lambda$, $\lambda \in \{1, \ldots, \Lambda\}$ désignant le numéro de la couche. La première couche prend en considération les entrées du réseau alors que la dernière retourne les sorties $\mathbf{H}^\Lambda$ sous la forme de probabilités a posteriori. Soient $\{N^1, \ldots, N^\lambda, \ldots, N^\Lambda\}$ les nombres de neurones pour chaque couche. Les couches intermédiaires retournent $\mathbf{H}^\lambda = \{h_i^\lambda\}$, $\forall \lambda \in \{1, \ldots, \Lambda\}, i \in \{1, \ldots, N^\lambda\}$, où h_i^λ caractérise la valeur associée au i^{eme} neurone de la couche d'indice λ. Cette sortie est déterminée à l'aide de l'équation 4.1.

$$h_i^\lambda = f^\lambda \left(\sum_{j=1}^{N^{\lambda-1}} \left[w_{i,j}^\lambda \, . \, h_j^{\lambda-1} \right] + b^{\lambda-1} \right) \quad \forall i \in \{1, \ldots, N^\lambda\}, j \in \{1, \ldots, N^{\lambda-1}\} \ , \tag{4.1}$$

avec $w_{i,j}^\lambda \in$ représentant les poids entre les neurones j de la couche $\lambda - 1$ et i de la couche λ, $b^{\lambda-1} \in$ le biais (à raison de un par couche), et f^λ la fonction d'activation non-linéaire sur la somme des poids.

Typiquement, cette fonction d'activation est représentée par :
- la fonction sigmoïde pour les couches cachées, soit $f^\lambda(x) = \frac{1}{1+\exp(-x)}$
- la fonction softmax pour la couche Λ de sortie, soit $f^\Lambda(x) = \frac{\exp(x)}{\sum_i \exp(x)}$, permettant d'estimer des probabilités *a posteriori*

Le réseau de neurones profond calcule itérativement les sorties $\mathbf{H}^1$ à $\mathbf{H}^\lambda$ à l'aide de l'équation 4.1 qui peut être réécrite sous la forme matricielle suivante :

$$\mathbf{H}^\lambda = f^\lambda(\ \mathbf{W}^\lambda \ . \ \mathbf{H}^{\lambda-1} + \mathbf{b}^{\lambda-1} \ ) \quad \forall \lambda \in \{1, \ldots, \Lambda\} \ , \tag{4.2}$$

avec $\mathbf{W}^\lambda = \{w_{i,j}^\lambda\} \in \ ^{N^\lambda \times N^{\lambda-1}}$ la matrice des poids à apprendre pour la couche λ.

4.2.2.3 Apprentissage d'un DNN

Apprentissage des couches cachées

Le formalisme mathématique des auto-associateurs permet d'apprendre de manière non supervisée des fonctions non linéaires.

Un auto-associateur apprend deux fonctions :

– une fonction d'encodage e, qui transforme la donnée d'entrée x en une représentation cachée $e(x)$,
– une fonction de décodage d, permettant de retrouver la représentation d'entrée $\hat{x} = d(e(x))$. L'estimation $\hat{x}$ est appelée la reconstruction de l'entrée x à travers e et d.

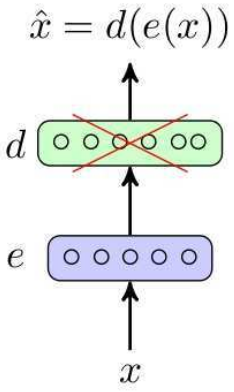


FIG. 4.6: Illustration de l'apprentissage d'une couche d'auto-encodeur pour l'apprentissage des poids des couches cachées d'un réseau profond. $e(x)$ est la représentation cachée de l'entrée x inférée par l'encodeur e et reconstruite par le décodeur d telle que $\hat{x} = d(e(x))$. Quand la couche est initialisée, le décodeur est supprimé et $e(x)$ est utilisé comme entrée de l'étage supérieur.

Les paramètres de l'encodeur et du décodeur sont partagés : la matrice des poids du décodeur est fixée à la transposée de la matrice des poids de l'encodeur [Bengio 2007b]. Ils sont itérativement appris pour minimiser l'erreur de reconstruction $|| \hat{x} - x ||^2$ sur la base d'apprentissage en utilisant la rétropropagation du gradient. Lorsque le critère d'arrêt de l'apprentissage est atteint, l'encodeur est empilé dans la structure profonde et forme une nouvelle couche de modélisation. Le décodeur est supprimé et les poids de l'encodeur sont bloqués. Ses sorties sont utilisées comme entrées pour l'initialisation des paramètres de la couche suivante. Ceci est illustré par les schémas des figures 4.6 et 4.7.

L'apprentissage non-supervisé des couches cachées permet d'extraire de nouvelles représentations cachées des formes d'entrées et donc d'inférer des caractéristiques discriminantes [Bengio 2007b]. L'apprentissage séparé des couches cachées permet également d'apprendre différents niveaux de caractéristiques : représentation de bas niveau pour la première couche puis des représentations de plus haut niveau sur les couches suivantes [Ranzato 2006]. Cette procédure d'apprentissage non-supervisé d'un grand nombre de couches cachées permet donc :

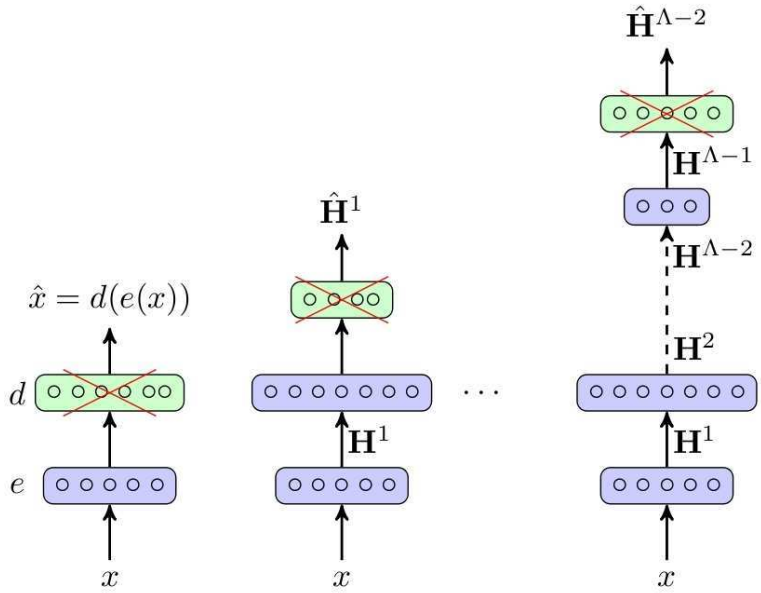


FIG. 4.7: Illustration de l'apprentissage de $\Lambda - 1$ couches cachées d'un réseau profond par empilement d'auto-encodeurs

- de travailler directement sur les pixels des images.
- de se passer de la complexe étape d'ingénerie consistant à définir un jeu de primitives pour caractériser les formes à reconnaitre.
- d'initialiser de manière pertinente et automatique les poids des couches cachées du réseau.

L'apprentissage supervisé permet d'affiner l'estimation des paramètres du réseau en combinant les représentations de haut niveau inférées par les couches cachées. Il permet donc de constituer les frontières de décision entre les classes du problème [Xue 2008, Bengio 2007a].

Fine-Tuning par rétropropagation du gradient

Le processus de correction des poids se fait en 3 passages dans le réseau.

Premier temps Un exemple de la base d'apprentissage est présenté au réseau et ses caractéristiques sont associées aux entrées du réseau. Elles sont propagées à travers les différentes couches pour obtenir une estimation des sorties qui sont comparées aux sorties désirées.

Second temps Cette comparaison permet de calculer les erreurs locales relatives aux neurones de la couche de sortie. Ces erreurs locales permettent d'estimer les compensations à appliquer aux poids des connections entrants sur ces neurones afin d'obtenir les sorties désirées. Ces compensations sont mémorisées et les erreurs locales sont ensuite propagées vers

les couches inférieures. La rétropropagation des erreurs locales est réitérée de proche en proche (des sorties vers les entrées d'où le terme *rétro*) pour calculer les termes de correction des poids de toutes les couches.

Troisième temps De nouveau dans le sens de la décision, tous les poids du réseau sont mis à jour à l'aide de la règle de correction classique des méthodes à gradient et rappelé par l'équation 4.3.

Comme toute méthode à gradient, le réglage des paramètres - ici les poids $w_{j,i}^{\lambda+1}|n$ du réseau - sont ajustés itérativement à l'aide de la règle de correction suivante :

$$w_{j,i}^{\lambda+1}|_{n+1} = w_{j,i}^{\lambda}|_n - \mu_n \times \frac{\partial E(w_{j,i}^{\lambda}|_n)}{\partial w_{j,i}^{\lambda}|_n} \tag{4.3}$$

où $E(w_{j,i}^{\lambda+1})$ est l'erreur locale répercutée par le poids $w_{j,i}^{\lambda}$ et μ_n le pas d'apprentissage lors de l'itération n de l'algorithme.

L'apprentissage est stoppé lorsque l'erreur globale sur l'ensemble de la base d'apprentissage est inférieure à un seuil fixe ϵ. Ce critère d'arrêt est essentiel : il doit permettre d'éviter le phénomène de sur-apprentissage qui se caractérise par une modélisation trop fine des frontières de décision au détriment d'une bonne généralisation. Ce phénomène peut être contrôlé sur une base de validation. L'itération d'apprentissage à partir de laquelle les performances en décision commencent à se déteriorer sur cette base de validation indique que le réseau entre dans une phase de sur-apprentissage.

Nous venons de présenter les réseaux de neurones profonds et les algorithmes d'apprentissage qui permettent de fixer efficacement leurs paramètres, nous présentons maintenant les travaux visant à combiner ces classifieurs discriminants avec les MMC modélisants afin de tirer le meilleur parti de leur avantage respectif.

4.3 Combinaisons neuro-markoviennes

Nous souhaitons intégrer un réseau de neurones profond comme discriminateur d'observation en les substituant des mélanges de Gaussiennes classiquement embarqués dans les états des MMC.

Les combinaisons neuro-markoviennes sont d'abord apparues en traitement de la parole continue [Bottou 1991, Knerr 1998, Bourlard 1998, Trentin 2001, Gorski 2001]. Elles ont pour objectif de combler les lacunes des MMC en matière de discrimination locale en tirant avantage des qualités des réseaux neuronaux. Ces combinaisons ont peu à peu été étendues à la reconnaissance

de l'écriture manuscrite en-ligne [Poisson 2005, Bengio 1995, Caillault 2007] mais surtout hors-ligne [Marukatat 2001, Morita 2003, Espana-Boquera 2011, Bengio 1995, Koerich 2002b, Gorski 2001].

Mis à part quelques travaux précurseurs [Bengio 1995, LeCun 1998, Waibel 1989, Lang 1990] où les MMC ont été combinés avec des réseaux de neurones à convolution[10], à notre connaissance aucun travaux ne combinent DNN et MMC. Cette section a pour but de présenter le principe de la combinaison neuro-markovienne, ses différents mécanismes d'association de modèles de Markov cachés et de réseaux de neurones et enfin les différentes méthodes d'apprentissage.

4.3.1 Principe des combinaisons neuro-markoviennes

Dans les MMC classiques qui sont des modèles génératifs, les mélanges de Gaussiennes permettent d'estimer la probabilité $p(O_t|q_i)$ qu'un état q_i ait généré une observation O_t. On peut chercher à estimer cette quantité à l'aide d'un réseau de neurones qui lui estime une probabilité *a posteriori*. C'est l'idée de base des combinaisons hybrides neuro-markoviennes, qui permet ainsi de bénéficier du fort caractère discriminant des réseaux de neurones et des bonnes capacités de modélisation des MMC (cf. illustration des figures 4.8 et 4.9). Dans le cas de la reconnaissance de l'écriture, les observations sont des pixels ou des caractéristiques provenant de trames ou de graphèmes. Le décodage des séquences d'observations est toujours supporté par le formalisme des MMC et la reconnaissance locale est laissée au réseau de neurones.

D'un côté, les MMC sont connus pour être efficaces en ce qui concerne la modélisation de séquences de symboles. Comme nous l'avons déjà observé, ils sont largement utilisés en reconnaissance de mots manuscrits isolés et en reconnaissance de séquence de mots. Ils permettent en effet de modéliser des séquences d'observations générées par des processus stochastiques tels que ceux liés à l'écriture manuscrite [Koerich 2003] ou à la parole [Rabiner 1990]. Ils possèdent l'avantage d'être relativement robustes aux variations de l'écriture manuscrite et s'adaptent donc bien à ses différents styles.

De l'autre côté, les réseaux de neurones présentent de très bonnes performances en discrimination. Dans le cadre de la reconnaissance de l'écriture manuscrite, ils sont plutôt utilisés pour la reconnaissance de caractères isolés car ils s'avèrent très efficaces lorsqu'il s'agit de discriminer des formes telles que des chiffres ou des caractères [Casey 1996, LeCun 1995]. Ceci s'explique

[10] qui peuvent être apparentés à des réseaux profonds de par leur structure

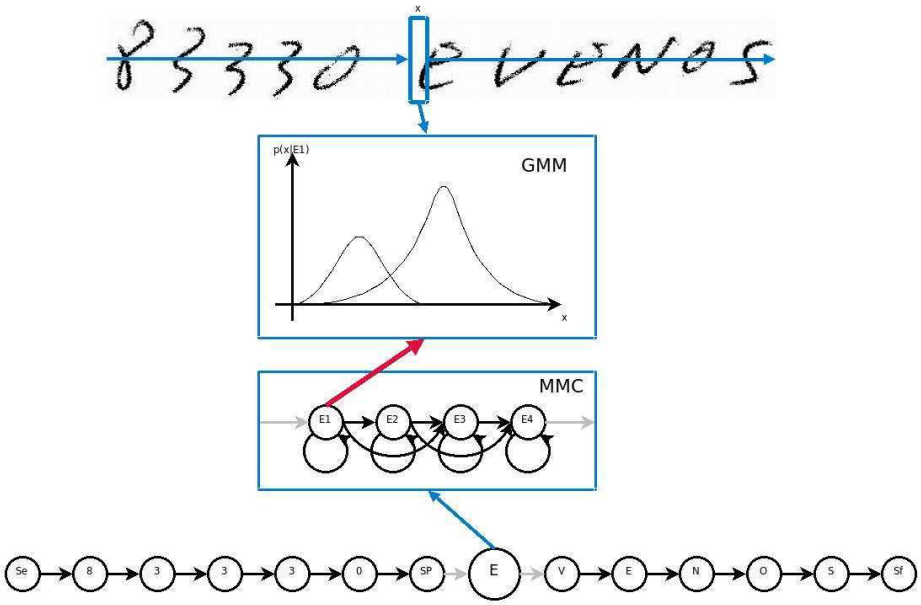


FIG. 4.8: MMC intégrant les classiques mélanges de Gaussiennes (GMM) dont le but est de calculer la vraisemblance d'un état d'un MMC par rapport à une observation.

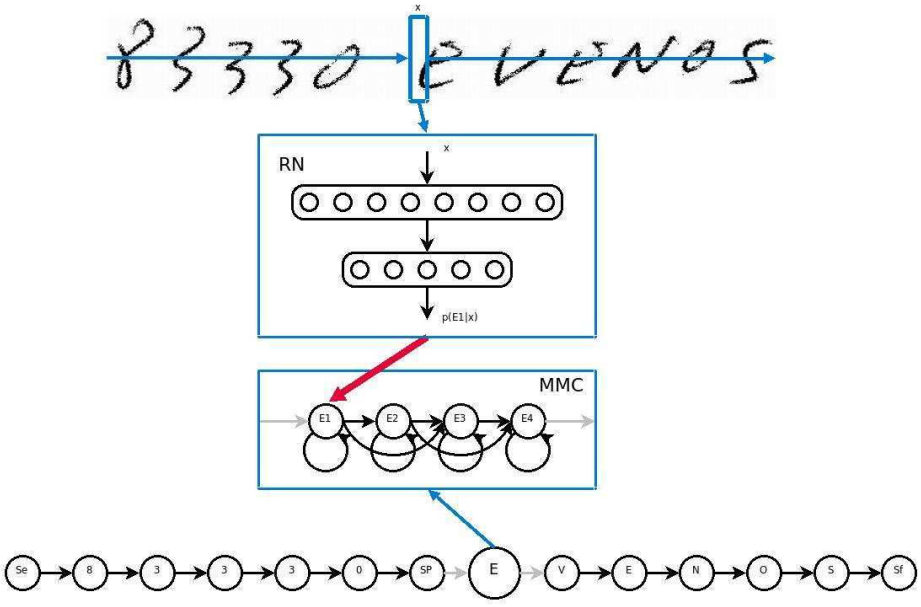


FIG. 4.9: On a remplacé les classiques mélanges de Gaussiennes par un réseau de neurones (RN) qui estime les probabilités *a posteriori* d'une observation

par le fait que leur apprentissage cherche à définir les meilleures frontières de décision entre les classes d'un problème dans leur espace de représentation. Ils ont également l'avantage de bien supporter les hautes dimensions contrairement aux mélanges de Gaussiennes classiquement embarquées dans les MMC. En effet, les performances des MMC chutent fortement au délà de quelques dizaines d'entrées, à cause de la difficulté à estimer correctement les Gaussiennes dans un espace à haute dimension, *a fortiori* lorsque la base d'apprentissage est de taille limitée.

Formellement, à partir d'une séquence d'observations, les réseaux de neurones calculent les probabilités *a posteriori* $p(q_i|O_t)$ des états q_i sachant la trame O_t. Or les MMC considèrent des vraisemblances $p(O_t|q_i)$ c'est à dire le score qu'un état q_i d'un MMC ait pu générer l'observation O_t à l'instant t. Ces dernières sont estimées à l'aide de la règle de Bayes par [Bourlard 1998, Knerr 1998, Espana-Boquera 2011] :

$$p(O_t|q_i) = \frac{p(q_i|O_t)p(O_t)}{p(q_i)} \tag{4.4}$$

où $p(q_i)$ est la probabilité *a priori* de l'état q_i, estimé sur la base d'apprentissage. Dans cette même équation 4.4, le terme $p(O_t)$ correspondant à la probabilité *a priori* d'observer la trame O_t est difficile à estimer et ne dépend pas des états. En effet, étant donné une trame, cette quantité est la même pour tous les états et n'influence pas la comparaison de leur score par rapport à cet trame. Il peut donc être supprimé dans l'estimation de la vraisemblance, soit :

$$p(O_t|q_i) \approx \frac{p(q_i|O_t)}{p(q_i)} \tag{4.5}$$

Il a ainsi été montré à de nombreuses reprises qu'une telle combinaison permet d'obtenir de meilleurs résultats qu'avec des MMC classiques (embarquant des modèles de mélanges de Gaussiennes) pour estimer les vraisemblances locales [Poisson 2005, Bengio 1995, Caillault 2007, Marukatat 2001, Espana-Boquera 2011, Bengio 1995, Gorski 2001]. Cependant, la mise en œuvre d'une telle architecture n'est pas immédiate, notamment à cause de l'apprentissage des paramètres des deux classifieurs.

4.3.2 Apprentissage neuro-markovien

L'apprentissage d'une structure hybride neuro-markovienne est le point critique de ces combinaisons. Il s'agit d'estimer les paramètres des deux classi-

fieurs, tâche délicate lorsqu'ils sont considérés indépendamment l'un de l'autre mais qu'il l'est d'autant plus lorsque l'on souhaite les utiliser dans un même système.

Nous pouvons distinguer principalement deux types d'apprentissage :

- les apprentissages disjoints des réseaux de neurones et des modèles de Markov cachés
- les apprentissages conjoints des deux classifieurs

L'apprentissage disjoint, également qualifié d'apprentissage local, consiste à estimer séparément les paramètres des deux classifieurs [Trentin 2001]. En reconnaissance de l'écriture, les réseaux de neurones sont appris sur des caractères isolés alors qu'une annotation au niveau mot voire au niveau ligne est suffisante pour l'apprentissage des MMC puisqu'il se fait de manière embarquée. Cet apprentissage séparé présente donc l'inconvénient de nécessiter une grande quantité de données annotées : des images de mots pour les MMC et de caractères pour les réseaux de neurones.

Il peut être envisagé d'automatiser l'annotation des données d'apprentissage pour les réseaux de neurones en utilisant les MMC déjà appris. Un alignement forcé du MMC d'un mot sur son image à l'aide de l'algorithme de Viterbi permet en effet d'estimer les frontières entre les caractères de manière précise. On peut également utiliser directement les trames de la séquence annotée au niveau des états des MMC et ainsi utiliser le réseau de neurones appris sur ces données produites en temps qu'estimateur local pour les MMC [Trentin 2001].

L'apprentissage joint ou global permet quant à lui l'estimation itérative des paramètres des réseaux de neurones et des MMC [Poisson 2005, Espana-Boquera 2011]. À l'image de l'algorithme EM permettant d'apprendre conjointement et itérativement les distributions de mélanges de Gaussiennes et les paramètres des MMC, les frontières entre les observations de différentes classes sont peu à peu affinées : MMC et réseaux de neurones convergent conjointement en alternant rétropropagation du gradient de l'erreur et alignement des MMC à l'aide de l'algorithme de Viterbi. Un critère d'arrêt global est utilisé pour stopper l'apprentissage. Les critères d'arrêt les plus courants sont [Poisson 2005] :

- le maximum de vraisemblance ou MLE pour *Maximum Likelihood Estimation*
- le MMI pour *Maximum Mutual Information*
- le MAP pour *Maximum a Posteriori*

Lors de l'apprentissage, ces fonctions de coût sont partagées par l'apprentis-

sage des deux classifieurs, le gradient de l'erreur est alors calculé à partir de la dérivée de ces fonctions de critère [Trentin 2001].

Nous venons de présenter les différents types d'apprentissages hybrides entre modèles de Markov cachés et réseaux de neurones artificiels. Dans la section suivante, nous montrons comme nous avons fusionné les MMC avec un réseau de neurones profond récemment introduit dans la littérature. Nous présentons également les expérimentations menées avec différentes configurations de combinaison hybride neuro-markovienne.

4.4 Intégration de réseau de neurones profond dans notre système de détection d'information

4.4.1 Expérimentations

Afin de comparer les performances de notre approche hybride combinant MMC et DNN, le système présenté au chapitre précédent sera utilisé en référence. Celui-ci intègre des mélanges de Gaussiennes pour estimer les vraisemblances locales des observations. Une combinaison classique de MLP et MMC a également été apprise et tiendra le rôle de comparateur. Les MMC et ses transitions probabilisées étant déjà apprises, une base d'apprentissage pour les réseaux de neurones est d'abord créée. Nous abordons ce point avant d'exposer les expérimentations conduites.

4.4.1.1 Annotations des observations et entrées des réseaux de neurones

Afin d'apprendre les paramètres des réseaux de neurones, nous devons lui fournir des trames annotées. Celles-ci ont été annotées de manière automatique par les MMC entrainés pour les expérimentations décrites dans le chapitre précédent (cf. section 3.3). Les images de lignes de texte de la base d'apprentissage des MMC sont utilisées à cet effet.

Pour chaque trame, les caractéristiques du système de base sont extraites et mémorisées. Nous rappelons qu'il s'agit d'un vecteur combinant 26 caractéristiques statistiques et structurelles et dépendant des lignes de base de l'écriture. Afin de prendre en compte le contexte de la trame courante, un autre vecteur est formé à partir des caractéristiques des trames précédente et suivante qui sont associées à celles de la trame courante pour former un

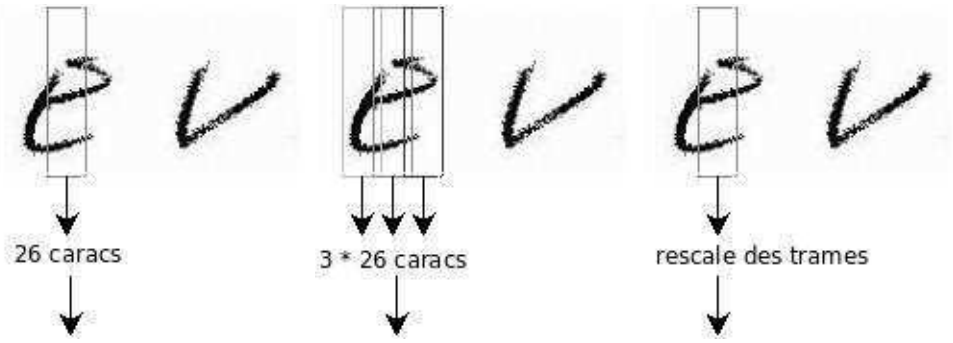


FIG. 4.10: Illustration des 3 jeux de caractéristiques utilisés pour l'apprentissage des réseaux de neurones. Le premier a 26 caractéristiques, le second en a 78 et le dernier, considérant directement les pixels des trames, nécessite un *rescale* pour obtenir 432 caractéristiques.

vecteur de dimension 78 à l'image des travaux de [Johansen 1996]. Les cas limites en début et fin de ligne sont respectivement gérés par duplication du premier et du dernier jeu de caractéristiques de cette ligne. Nous avons également entrainé un système sur les pixels de chaque trame. Les réseaux de neurones étant de taille fixe, les trames doivent être adaptées afin de remplir cette contrainte. Elles sont ainsi redimensionnées par rapport aux lignes de base. La partie supérieure contenant hampes et majuscules et la partie inférieure contenant les jambages sont redimensionnées à une hauteur fixe de 12 pixels. La partie médiane renfermant le corps de texte est fixé à 30 pixels soit une fenêtre de taille fixe de 54 pixels. L'extraction de ces 3 jeux de caractéristiques est illustrée par la figure 4.10

Pour résumer, les réseaux suivants seront appris :

MLP-26 un MLP entrainé sur le jeu de caractéristiques décrit dans le chapitre précédent

DNN-26 un DNN entrainé sur ce même jeu de caractéristiques

MLP-78 un MLP entrainé sur le jeu de caractéristiques contextualisé pour prendre en compte le voisinage gauche - droite d'une trame

DNN-78 un DNN entrainé sur ce même jeu de caractéristiques

DNN-432 un DNN entrainé sur les pixels d'écriture directement. Cela permet de tester la capacité du DNN à appréhender les espaces de hautes dimensions et surtout sa capacité à inférer et apprendre de nouvelles représentations des formes d'entrée et donc des caractéristiques.

4.4.1.2 Apprentissage des réseaux

Stratégie d'apprentissage

Étant donné que nous avons à notre disposition des MMC déjà appris et intégrés dans le système de référence présenté dans le chapitre précédent, nous avons choisi dans un premier temps un apprentissage séparé des deux classifieurs : nous avons utilisé les MMC pour annoter les observations au niveau état afin de créer une base d'apprentissage pour le réseau de neurones profond (comme décrit précédemment au début de cette section).

Pour étendre l'apprentissage des paramètres de la combinaison hybride à un apprentissage conjoint, correspondant à un apprentissage itératif des paramètres des MMC et du réseau de neurones profond, il nous faudrait recréer des bases d'apprentissage à partir des combinaisons neuro-markoviennes déjà apprises. À chaque itération, les bases doivent être successivement réannotées à l'aide d'un alignement par la combinaison hybride avec l'algorithme de Viterbi. Sachant que la procédure d'initialisation des couches cachées d'une architecture profonde se fait de manière non supervisé (sans fournir au réseau les étiquettes des formes qui lui sont présentées), nous pouvons considérer que ses paramètres sont déjà correctement estimés à la fin de la première itération. Pour affiner les paramètres lors des itérations suivantes, il nous faudrait uniquement effectuer l'étape supervisée de *Fine-Tuning* du réseau après l'alignement des observations réalisé par la combinaison hybride neuro-markovienne. La mise en place de cet apprentissage conjoint constitue une perspective d'amélioration du système.

Une des difficultés avec un réseau de neurones, quelque soit sa nature, réside dans le choix des hyperparamètres d'apprentissage. Nous évoquons maintenant ce point.

Hyperparamètres d'apprentissage

Pour l'apprentissage des DNN et des MLP, des hyperparamètres sont à définir : le pas d'apprentissage et le critère d'arrêt. Le pas d'apprentissage initial a été fixé empiriquement entre 0.05 et 0.0005 et est dégressif avec le nombre d'itérations. La décroissance du pas est linéaire et se fait selon l'équation 4.6. Ceci permet de converger de plus en plus précisément vers le meilleur réseau.

$$\mu_n \quad = \frac{\mu_0}{n.\nu+1} \tag{4.6}$$

où μ_0 est le pas d'apprentissage initial, μ_n est le pas d'apprentissage considéré lors de l'itération n et ν le paramètre permettant de diminuer le pas. Il a été fixé empiriquement à $\nu = \frac{\mu_0}{10}$.

Le critère d'arrêt de l'apprentissage est également à définir. La différence ϵ de l'erreur quadratique moyenne entre les entrées et leur reconstruction est utilisée pour stopper l'apprentissage des couches de modèles des DNN. Une valeur relativement forte de $0,01$ est considérée dans le cas des DNN pour éviter le phénomène de sur-apprentissage des couches cachées. L'expérience nous a montré qu'une valeur très petite de ce critère pouvait être un frein à l'apprentissage des couches de modèles dans le DNN. À titre de comparaison, [Bengio 2009] utilise dans ses expérimentations un pas relativement faible et un nombre d'itérations d'apprentissage fixé à 10 pour chaque couche cachée. Pour ce qui est de l'apprentissage de la dernière couche, le même critère de différence d'erreur quadratique est utilisé en substituant la reconstruction avec l'étiquette. Le sur-apprentissage est contrôlé *a posteriori* sur une base de validation de mots manuscrits isolés : un lexique de 100 mots est constitué pour chaque image correspondant à l'étiquette du mot à reconnaitre à laquelle 99 mots sont ajoutés de manière aléatoire par rapport au vocabulaire complet de la base de données. La figure 4.11 illustre les taux d'erreurs obtenus sur la base de validation en fonction du nombre d'itération. Un critère d'arrêt plus strict est considéré pour garantir l'obtention du meilleur réseau.

Avec le choix des hyperparamètres d'apprentissage, le dimensionnement des réseaux de neurones est un autre point sensible.

Dimensionnement des réseaux

Les réseaux de neurones ont été dimensionnés de la manière suivante : dans le cas des MLP, le nombre de neurones de la couche cachée est calculé par la moyenne des entrées et des sorties. Dans le cas des réseaux profonds, nous avons fait le compromis d'un réseau à 3 couches cachées comme dans [Bengio 2009]. Des structures pyramidales complétement interconnectées ont été choisies pour limiter le nombre d'hyperparamètres à apprendre. Par exemple, dans les travaux de [Larochelle 2009], le nombre de neurones par couches est fixé arbitrairement entre 300 et 1500 et est identique entre les couches cachées d'un même réseau. N'ayant pas un nombre de données d'apprentissage très important (voir ci-après) et travaillant avec plus de classes (10 pour [Larochelle 2009] et [Bengio 2009] qui testent leurs architectures profondes sur *MNIST* contre 284 dans notre cas, cf. section 3.2.4.2), nous avons fait les choix récapitulés dans le tableau 4.12. Les 284 classes de sortie

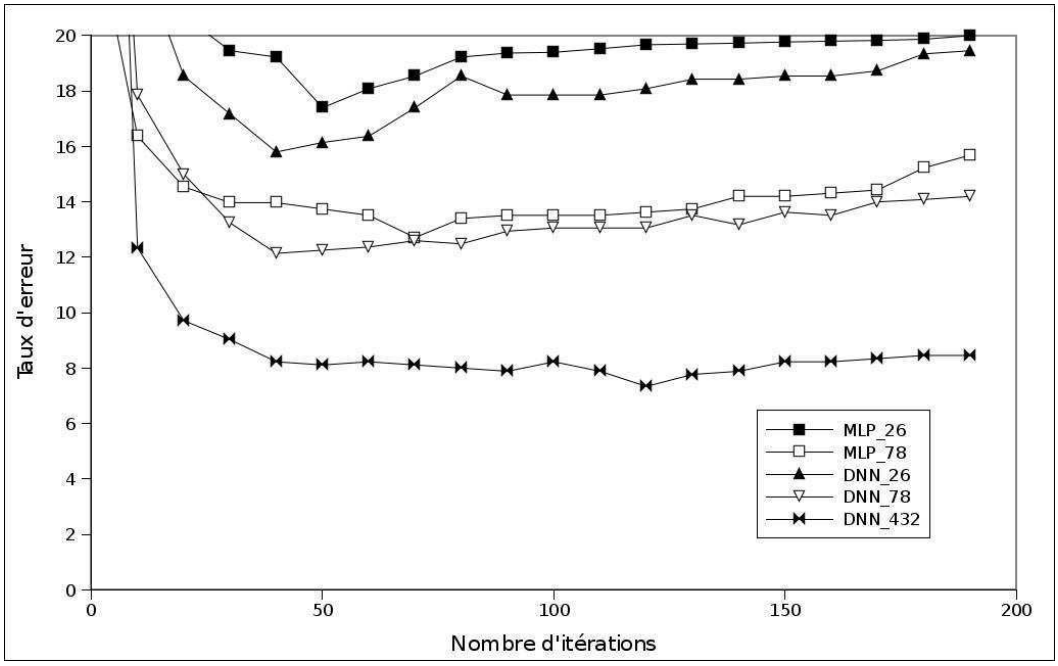


FIG. 4.11: Taux d'erreurs sur la base de validation mots pour les différentes combinaisons de classifieurs entrainées

pour les réseaux de neurones correspondent aux états des MMC appris pour notre problème d'extraction, soient 71 modèles à 4 états considérés lors des expérimentations du chapitre précédent.

Réseau	entrées (N^0)	N^1	N^2	N^3	sorties (N^4)
DNN-26	26	200	225	250	284
DNN-78	78	200	225	250	284
DNN-432	432	400	350	300	284
MLP-26	26	155	$\emptyset$	$\emptyset$	284
MLP-78	78	181	$\emptyset$	$\emptyset$	284

FIG. 4.12: Effectifs des couches des différents réseaux sélectionnés

Pour apprendre les paramètres de ces réseaux de neurones, ils nous faut des données d'apprentissage. Nous abordons ce point dans la section suivante.

Bases d'apprentissage

Trois bases d'apprentissage de trames annotées au niveau état de MMC ont été créées. Celles-ci ont été équilibrées i.e. le nombre de représentants par classe est le même pour toutes les classes. Pour les classes n'ayant pas assez de représentants, une duplication aléatoire des éléments existants pour ces classes est réalisée. Dans le cas inverse, un tirage aléatoire sans remise est effectué. Les effectifs de ces 3 bases sont donnés par le tableau de la figure 4.13.

Types d'entrées	Carac	Context	Img
Nombre d'entrées	26	78	432
Nombre d'éléments par classe	1500	1200	900
Nombre d'éléments total	426000	340800	255600

FIG. 4.13: Effectifs des différentes bases de données pour l'apprentissage des réseaux de neurones

Les bases ne sont pas de tailles identiques pour plusieurs raisons. L'apprentissage des réseaux de neurones nécessite la sauvegarde en mémoire de données intermédiaires durant l'apprentissage comme par exemple les évaluations du gradient des erreurs locales pour tous les poids des réseaux. Dans le cas des DNN, les représentations cachées des données en sortie des couches intermédiaires sont également mémorisées dans un souci de rapidité et nécessitent un important espace mémoire, tout comme les variables intermédiaires pour le calcul des gradients.

En utilisant ces bases de données pour apprendre nos réseaux de neurones, nous pouvons confronter les combinaisons neuro-markoviennes imaginées à notre système de référence présenté au cours du chapitre précédent.

4.4.2 Résultats

4.4.2.1 Tests de la modélisation neuro-markovienne en reconnaissance de mots

La base de test utilisée est la base de mots *RIMES*. Elle a fait l'objet d'une compétition de reconnaissance de mots isolés [Grosicki 2009], nous pouvons ainsi comparer notre approche aux systèmes présentés dans cette compétition en notant toutefois que les lexiques de 100 mots que nous avons utilisés ne sont pas les mêmes que ceux considérés lors de cette compétition. Les résultats de reconnaissance sont illustrés par les courbes de la figure 4.14 et le tableau de la figure 4.15 et récapitulés pour toutes nos implémentations.

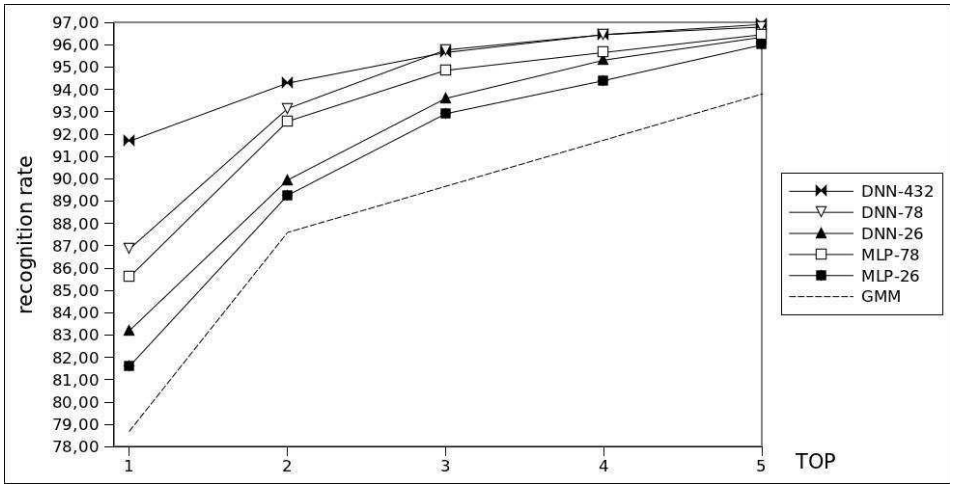


FIG. 4.14: Taux de reconnaissance de mots isolés pour les différentes configurations neuro-markoviennes

	MLP-26	DNN-26	MLP-78	DNN-78	DNN-432	GMM
Top 1	81,6	83,2	85,6	86,9	91,7	78,7
Top 2	89,3	89,9	92,6	93,1	94,3	87,6
Top 5	96,0	96,3	96,5	96,8	96,9	93,8
Top 10	98,4	98,2	98,3	98,6	98,3	97,2

FIG. 4.15: Taux de reconnaissance pour différents *Top* sur une tâche de reconnaissance de mots manuscrits isolés avec un dictionnaire fixé à 100 mots

Les performances obtenues par les combinaisons neuro-markoviennes sont meilleures que pour les MMC classiques intégrant des mélanges de Gaussiennes (nous appelons ce système de référence GMM[11]). Plus en détail, nous pouvons noter que :

- les modélisations neuro-markoviennes (MLP ou DNN) sont au dessus des performances du moteur de reconnaissance de base utilisant les MMC seuls pour tous les *Tops*.
- les combinaisons des MMC avec les réseaux profonds permettent d'atteindre de meilleures performances qu'avec les MLP pour des entrées identiques.

[11] pour notre système à base de MMC embarquant des GMM *Gaussian Mixture Models*

– notre meilleure combinaison est obtenue en prenant en entrée du réseau de neurones profond les pixels des trames redimensionnées.
– cette combinaison réduit l'écart avec les meilleurs systèmes de reconnaissance présentés pour la compétition RIMES [Grosicki 2009] par rapport à notre système de référence décrit au chapitre précédent (cf. tableau 4.16).

Système	Top 1	Top 10
TUM	98,4	99,9
UPV	96,4	99,6
Siemens	95,3	99,7
DNN-432	**91,7**	**98,3**
LITIS	91,4	98,3
IRISA	91,2	96,3
Paris Tech (système 1)	86,4	88,8
Paris Tech (système 2)	82,0	87,2
Paris Tech (système 3)	79,9	88,0
GMM	**78,7**	**97,2**
ITESOFT	74,6	85,3

FIG. 4.16: Comparaison des résultats obtenus avec les systèmes ayant pris part à la compétition de reconnaissance de mots isolées RIMES. Le lexique de reconnaissance est fixé à 100 mots mais rappelons que les lexiques que nous avons utilisés ne sont pas ceux qui l'ont été lors de cette compétition.

Les tests en reconnaissance de mots semblent prometteurs pour les configurations testées. Les résultats obtenus par notre système sur une tâche de détection d'information sont maintenant présentés.

4.4.2.2 Tests du système en requêtage (*keyword-spotting*)

Nous commencons par comparer le système de détection lorsque nous l'interrogeons sur des requêtes. Pour cela, nous avons mené les mêmes expérimentations que dans la section 3.4.2. Rappelons que 10 séquences appartenant à un courrier sont successivement recherchées dans leur image et ce, sur l'ensemble de la base de test composé de 100 courriers. Nous comparons ici uniquement la combinaison ayant obtenu les meilleurs résultats en reconnaissance de mots afin de ne pas surcharger les illustrations. Les résultats comparatifs sont récapitulées sur la figure 4.17 et dans le tableau de la figure 4.19. À titre de comparaison, le tableau regroupant les résultats obtenus avec notre système de référence à base de GMM est donné par la

figure 4.18.

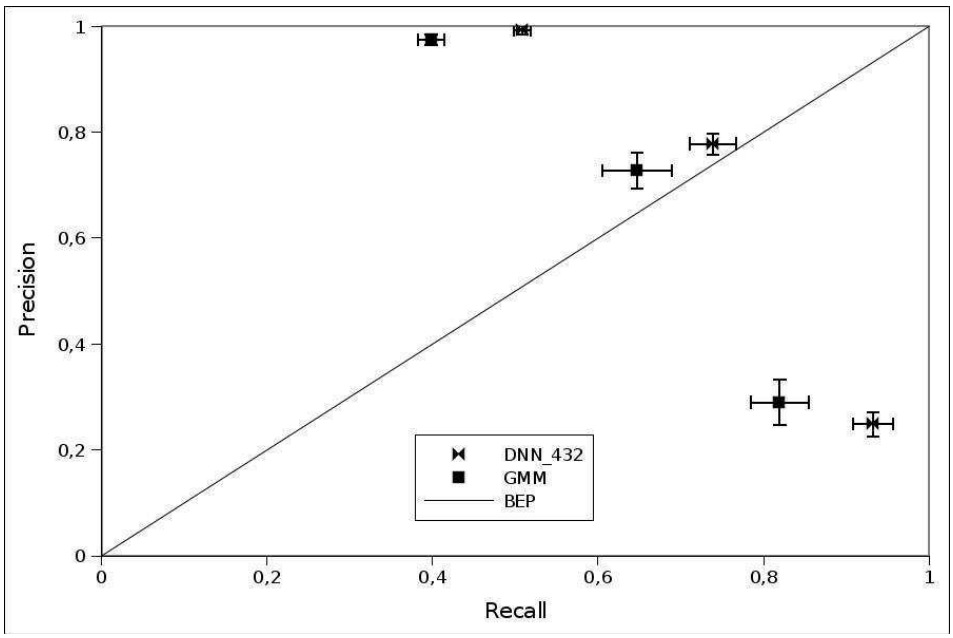


FIG. 4.17: Rappel / Précision pour 3 points de fonctionnement du système en requêtage. Comparaison entre le système à base de MMC purs (GMM) et la combinaison neuro-markovienne profonde travaillant au niveau pixel (DNN-432).

Pour des configurations identiques à celles utilisées par notre système à base de MMC classiques, les points de fonctionnement atteints sont globalement meilleurs. Notamment, une séquence recherchée sur 2 est retrouvée de manière quasi certaine (plus de 99% de précision), un BEP proche de 75% est atteint ainsi qu'un rappel supérieur à 90% pour la troisième configuration choisie. Les écarts-types sont également en baisse et illustrent une meilleure stabilité du système pour ces différents points de fonctionnement. À titre indicatif, nous rappelons dans le tableau de la figure 4.20 les perfor-

Point de fonctionnement	R	P	$\sigma(R)$	$\sigma(P)$
$G \to 0$	39,8%	97,5%	1,6%	0,9%
G tel que $R \approx P$	64,7%	72,8%	4,1%	3,4%
$G \to 1$	81,9%	29,0%	3,5%	4,2%

FIG. 4.18: Performances obtenues par notre système de détection pour 3 points de fonctionnement en ne considérant qu'une requête à la fois.

Point de fonctionnement	R	P	$\sigma(R)$	$\sigma(P)$
$G \to 0$	50,8%	99,2%	1,0%	0,8%
G tel que $R \approx P$	73,9%	77,8%	2,8%	2,1%
$G \to 1$	93,2%	25,3%	2,4%	2,3%

FIG. 4.19: Performances obtenues par notre système de détection intégrant la combinaison neuro-markovienne profonde *DNN_432* pour 3 points de fonctionnement.

Système	BDD	nb requêtes	BEP
Fischer	IAM database (929 lignes)	3421	$\approx 30\%$
	GW database (675 lignes)	1067	$\approx 40\%$
Rodriguez	Privée (630 courriers)	1	$\approx 75\%$
Frinken	IAM database (929 lignes)	2807	$\approx 72\%$
	GW database (675 lignes)	1067	$\approx 65\%$

FIG. 4.20: BEP obtenus par 3 systèmes de *keyword-spotting* de la littérature avec leurs différentes conditions expérimentales. Fischer fait référence à [Fischer 2010], Rodriguez à [Rodríguez-Serrano 2009a] et Frinken à [Frinken 2012].

mances obtenues en *keyword-spotting* pour différents systèmes de la littérature [Fischer 2010, Rodríguez-Serrano 2009a, Frinken 2012]. Nous renvoyons à la section 3.4.2 du chapitre précédent pour la comparaison entre les protocoles expérimentaux.

Il est intéressant de noter que dans [Frinken 2012], les auteurs relèvent également un point de fonctionnement autour de 50% de rappel pour plus de 95% de précision ainsi qu'un BEP de 72%. Notre système se positionne donc favorablement en termes de performances par rapport à celui-ci malgré les conditions favorables dans lesquelles se placent les auteurs (segmentation en lignes parfaite, mot à rechercher présent dans la ligne).

4.4.2.3 Tests des modélisations en détection de mots clés

Afin de tester le système complet de détection avec la combinaison HMM/DNN mise en place, les expérimentations similaires à celles présentées dans le chapitre précédent ont été conduites. Le protocole décrit en section 3.3.4 est suivi et toutes les configurations neuro-markoviennes apprises sont comparées au système de détection de référence à base de MMC présenté au chapitre précédent. Nous rappelons que différentes tailles de lexique sont utilisées pour tester la détection d'information à savoir 10, 100 et 500 requêtes.

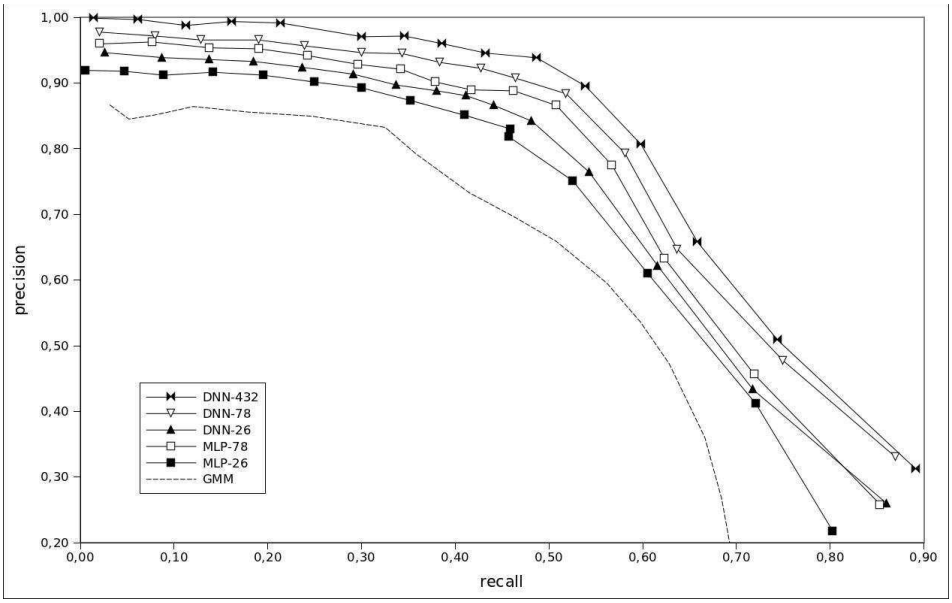


FIG. 4.21: Rappel / Précision pour toutes les configurations sur un lexique de 10 mots

Rappelons que ces ensembles ont été construits de manière dynamique pour chaque document de la base de test en fonction de leur vocabulaire et de celui de la base de test complète. Les résultats correspondant sont illustrés par les figures 4.21, 4.22 et 4.23. Le tableau de la figure 4.24 met en avant les précisions moyennes obtenues lors de ces expérimentations ainsi que les Break-Even Point (BEP).

Un gain significatif est obtenu par rapport au système de référence à base de MMC classique pour toutes les tailles de lexique de mots clés testées avec les modélisations neuro-markoviennes. Ces modélisations s'avèrent plus efficaces et plus stables lorsque le nombre de requêtes augmente, notamment en précision. En ce qui concerne le rappel, même si une baisse générale est observée avec l'augmentation du nombre de requêtes, de bons taux sont atteints, au délà de 80%, pour toutes les modélisations. Pour un lexique de 10 mots clés à détecter, des *BEP* au délà de 60% sont obtenus correspondant à des gains allant jusqu'à 10% par rapport au système de référence.

Pour les trois configurations de lexique, la supériorité de la modélisation profonde utilisant directement les pixels en entrée est observée. L'utilisation des caractéristiques présentées dans le chapitre précédent avec les réseaux de neurones donne de moins bons résultats que les pixels mais la contextualisa-

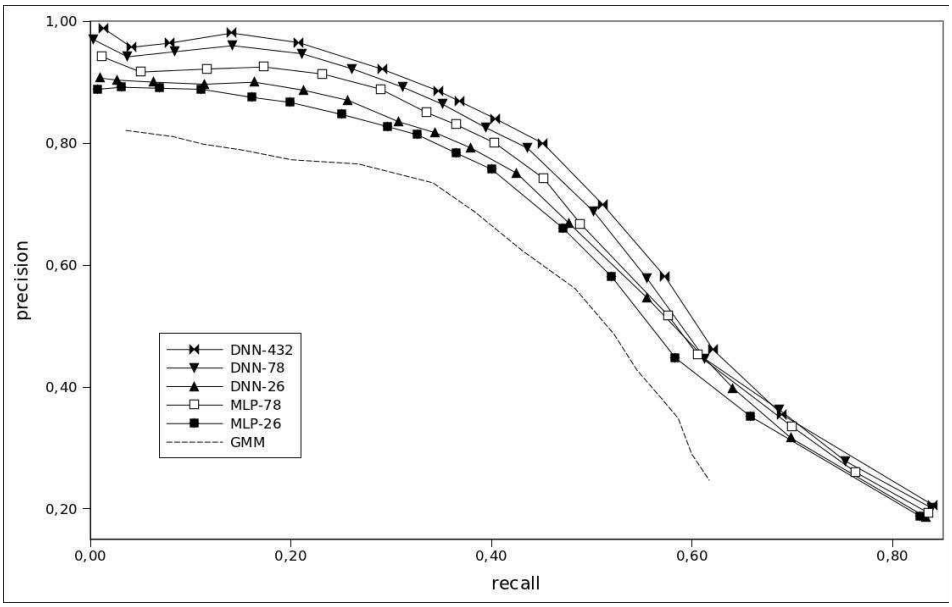


FIG. 4.22: Rappel / Précision pour toutes les configurations sur un lexique de 100 mots

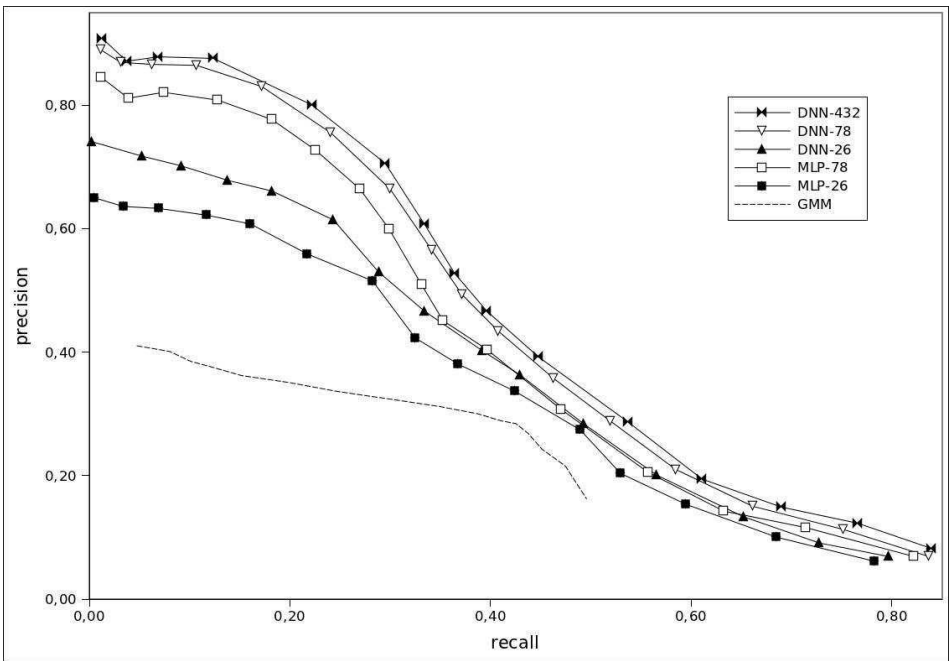


FIG. 4.23: Rappel / Précision pour toutes les configurations sur un lexique de 500 mots

Lexique	10 requêtes		100 requêtes		500 requêtes	
	P_{moy}	BEP	P_{moy}	BEP	P_{moy}	BEP
MLP-26	78,2%	60,6%	69,7%	53,9%	38,5%	37,6%
DNN-26	80,3%	61,8%	72,3%	55,2%	41,6%	39,7%
MLP-78	81,9%	62,7%	71,6%	55,5%	49,2%	39,9%
DNN-78	84,1%	64,1%	72,2%	56,3%	51,3%	41,0%
DNN-432	86,0%	65,8%	72,6%	57,5%	51,6%	42,4%
GMM	61,1%	57,5%	57,1%	50,9%	27,3%	28,2%

FIG. 4.24: Performances du système de détection intégrant les différentes combinaisons neuro-markoviennes pour des ensembles de 10, 100 et 500 requêtes de type mot clé.

tion de ces caractéristiques en y ajoutant les jeux correspondant aux trames précédentes et suivantes tend à se rapprocher des performances obtenues avec les pixels. Pour ces deux types d'entrées, on constate que les différences sont de plus en plus faibles lorsque l'on force le système à extraire des mots clés (valeurs de G proches de 1). Sur la figure 4.23 où l'on représente les résultats en détection sur des ensembles de 500 requêtes, le modèle de ligne semble prendre le pas sur la modélisation locale de l'écriture. Ce phénomène peut donc s'apparenter à une saturation du modèle de ligne.

Au delà des entrées des réseaux de neurones, la supériorité des DNN sur les MLP est observée pour toutes les tailles de lexique. Ceci renforce l'idée que les réseaux de neurones profonds sont capables d'inférer des représentations robustes des données d'entrée équivalant à un apprentissage efficace des caractéristiques.

Dans la section suivante, nous étendons ces expérimentations pour différents types de requêtes (alphabétiques, numériques et alphanumériques) en nous concentrant d'avantage sur la meilleure combinaison neuro-markovienne travaillant directement sur les pixels des trames.

4.4.2.4 Tests de la modélisation en détection d'information de différents types

En suivant le protocole décrit dans la section 3.4.5, nous testons maintenant le système de détection d'information implémentant la meilleure combinaison neuro-markovienne (travaillant directement sur les pixels des images de lignes) sur des lexiques de différents types. Les figures 4.25, 4.26 et 4.27 pré-

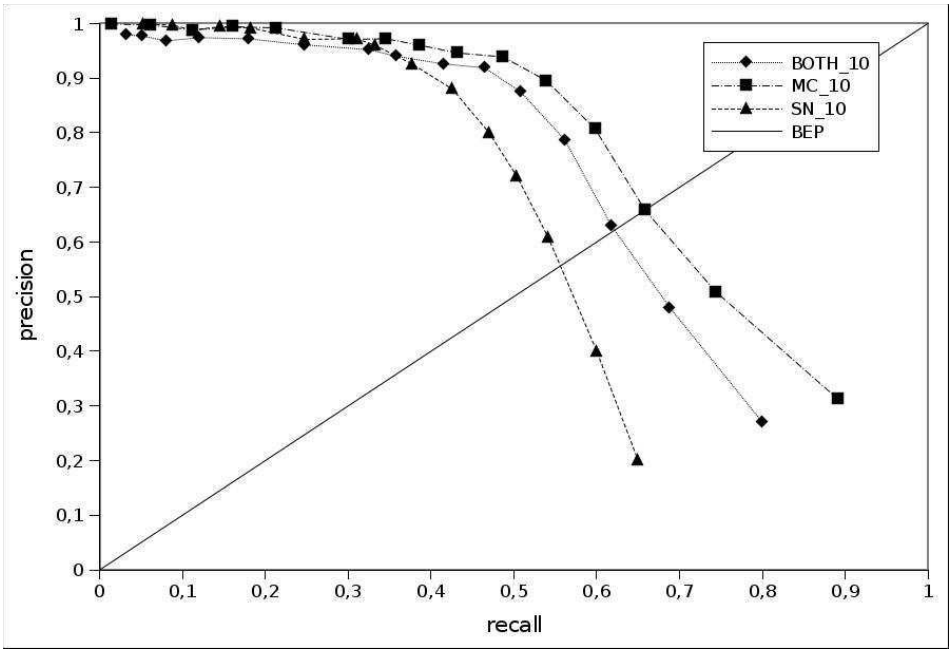


FIG. 4.25: Rappel / Précision en détection d'information de différents types avec des ensembles de 10 requêtes.

sentent les résultats correspondant pour des ensembles de données à détecter de respectivement 10, 100 et 500 requêtes.

En comparant ces résultats avec ceux obtenus lors des expérimentations du chapitre précédent (section 3.4.5), nous dressons le tableau de la figure 4.28. Nous pouvons ainsi comparer les résultats de la combinaison neuro-markovienne profonde avec notre système de référence à base de GMM. Nous constatons une nette hausse des performances de manière générale (les gains par rapport au système de référence pour des expérimentations et des lexiques identiques, sont donnés en pourcentage dans le tableau ci-après) pour tous les types de requête et pour toutes les tailles de lexique. Nous remarquons que les résultats de détection obtenus avec des requêtes numériques (**SN**) sont toujours plus faibles que pour les autres types de lexique mais tendent à se rapprocher des résultats sur des données alphabétiques (**MC**) et alpha-numériques (**BOTH**). Le système apparait plus stable face à l'augmentation de la taille du lexique considéré et ce, quelque soit le type d'information qu'il contient. Rentrons dans le détail de ces résultats avec le tableau de la figure 4.28.

Lorsque la taille des lexiques de différent type est de 10 (figure 4.25), les précisions maximales sont élevées et proches de 100%. La précision moyenne

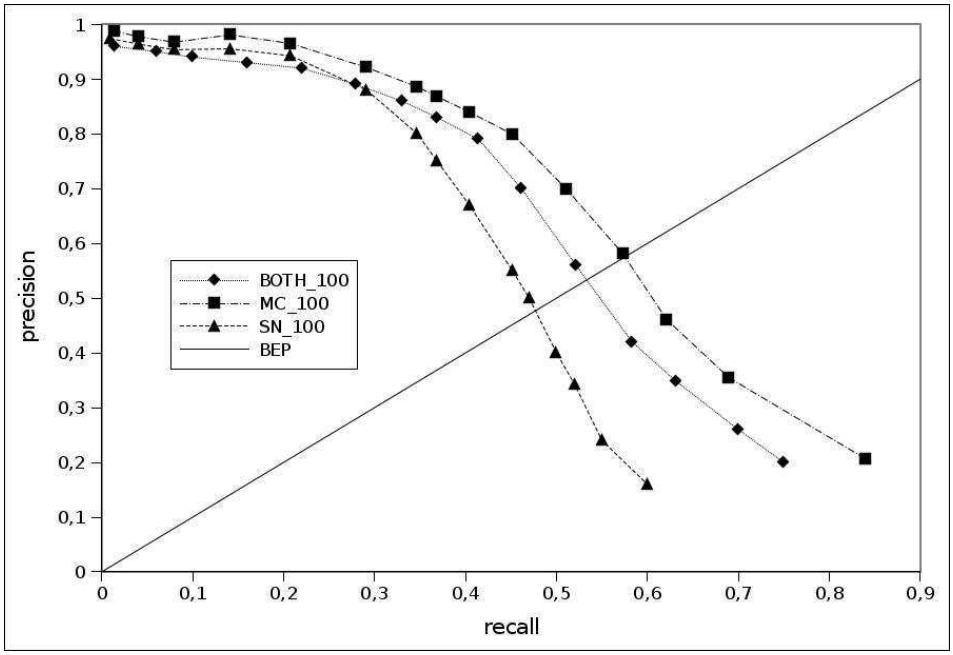


FIG. 4.26: Rappel / Précision en détection d'information de différents types avec des ensembles de 100 requêtes.

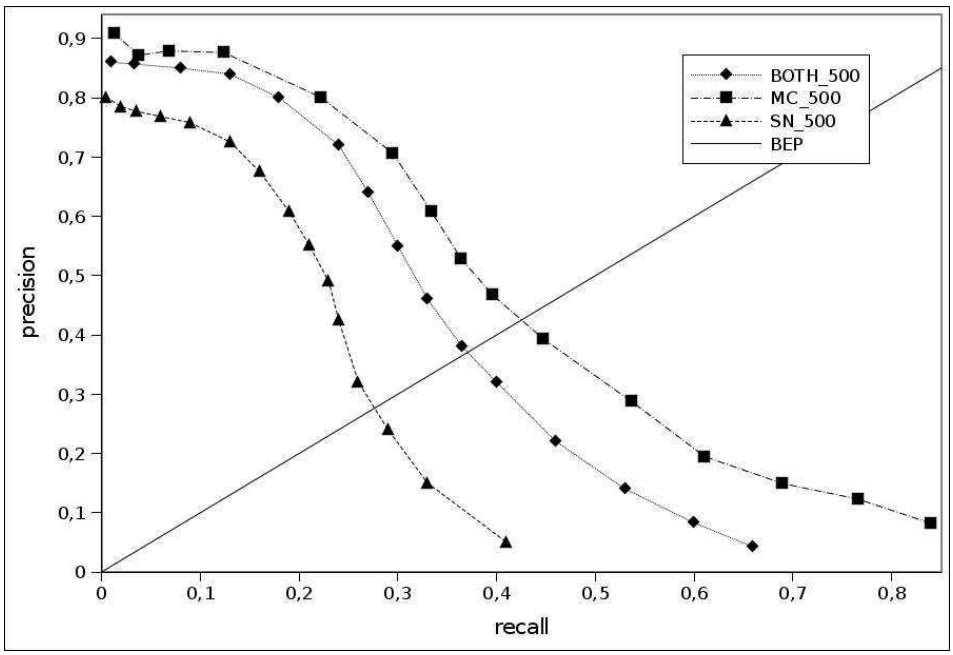


FIG. 4.27: Rappel / Précision en détection d'information de différents types avec des ensembles de 500 requêtes.

du système de détection est supérieure à 80% représentant une hausse de 40% par rapport au système de référence. Les rappels sont également supérieurs à ceux obtenus par le système pour tous les types : ils dépassent 80% pour les courbes **MC_10** et **BOTH_10** et 65% pour la courbe **SN_10**. Enfin, les BEP sont en hausse de respectivement 15%, 31% 44% pour les expérimentations **MC_10**, **BOTH_10** et **SN_10** soit des points de fonctionnement à $65,8\%$, $62,1\%$ et $55,6\%$. Nous constatons que les gains les plus significatifs sont obtenus pour l'expérimentation concernant les séquences numériques ce qui met d'ores et déjà en avant l'apport de l'architecture profonde.

En portant le nombre de requêtes à 100 (figure 4.26), nous constatons de nouveau une stabilité dans les précisions maximales atteintes pour les différentes configurations par rapport aux expérimentations menées avec ce même système sur des ensembles de 10 requêtes. Les précisions moyennes sont en légère baisse mais restent proches ou supérieures à 70%. Les rappels maximum atteints sont également en légère baisse par rapport aux précédentes expérimentations mais en forte hausse pour les configurations similaires avec le système de référence. Ceci met en avant la stabilité du système face à l'augmentaton de la taille du lexique, tout comme la position des BEP autour de 50% avec respectivement $57,5\%$, $53,3\%$ et $47,7\%$ pour les courbes **MC_100**, **BOTH_100** et **SN_100**.

Pour finir, nous avons porté le nombre de requêtes à 500 (cf. figure 4.27). Les précisions maximales restent hautes et proches de 100% sauf pour la courbe **SN_500**. Les précisions moyennes sont en plus forte baisse mais restent supérieures à 50% pour tous les types de requêtes. Les rappels maximum sont relativement stables pour les courbes **MC_500** et **BOTH_500** mais baissent autour de 40% pour la courbe **SN_500**. Enfin, en ce qui concerne les BEP, nous constatons des gains de plus de 30% pour tous les types de requêtes soient $42,4\%$, 37% et $27,8\%$ pour les courbes **MC_500**, **BOTH_500** et **SN_500** respectivement.

De manière générale, l'ajout d'un pouvoir discriminant permet d'obtenir des performances bien supérieures à celles obtenues par les MMC intégrant des GMM. Bien que très proches en termes de résultats de détection, les DNN permettent de mieux discriminer les formes locales comparativement aux MLP. Enfin, l'utilisation des pixels en entrée du DNN apporte un plus confirmant les propriétés de ce type de réseaux à savoir la gestion des hautes dimensions en entrée et en sortie mais surtout la possibilité de se passer d'une étape difficile d'ingénierie correspondant à la définition et l'implémentation d'un jeu

Lexique	R_{max}	P_{max}	P_{moy}	BEP
MC_10	89,1% (+27,8%)	99,9% (+15,2%)	86,2% (+41,1%)	65,8% (+14,4%)
SN_10	65,2% (+58,6%)	99,8% (+11,1%)	82,7% (+32,1%)	55,6% (+48,3%)
BOTH_10	80,1% (+25,4%)	97,9% (+14,1%)	84,1% (+49,1%)	62,1% (+20,8%)
MC_100	84,2% (+35,4%)	98,8% (+20%)	76,6% (+34%)	57,5% (+13%)
SN_100	60,1% (+75,2%)	97,4% (+21,7%)	68,2% (+24,7%)	47,7% (+46,3%)
BOTH_100	74,9% (+42,1%)	96,2% (+23,2%)	70,4% (+30,6%)	53,3% (+12,7%)
MC_500	83,9% (+69,5%)	90,1% (+119%)	54,2% (+75,2%)	42,4% (+32,5%)
SN_500	41% (+62%)	80,1% (+153,5%)	51,7% (+150,8%)	27,8% (+44,8%)
BOTH_500	66,6% (+43,8%)	86% (+114%)	52,5% (+92,3%)	37% (+31,2%)

FIG. 4.28: Points de fonctionnement du système pour différents types et tailles de lexique 3.42. Nous notons entre parenthèses les gains obtenus (en pourcentage) par rapport aux expérimentations menées sur des lexiques identiques avec le système de référence présenté au chapitre précédent.

de caractéristiques. Par contre, le choix des hyperparamètres d'apprentissage des réseaux de neurones reste un problème complexe et incontournable.

4.5 Conclusion

Au cours de ce chapitre, nous avons présenté une amélioration de notre système d'extraction consistant à remplacer des mélanges de Gaussiennes embarqués dans les MMC par des réseaux de neurones profonds discriminants. L'amélioration des capacités de reconnaissance locale a permis d'améliorer le pouvoir de détection global de notre modèle de ligne comme nous en avions fait l'hypothèse. Les résultats expérimentaux obtenus en détection d'information confirment cette tendance et illustrent l'intérêt de l'apprentissage hybride, même séparé.

Les DNN apportent au système global une stabilité en précision, ils sont supérieurs aux MLP lorsque les mêmes types d'entrées leur sont présentées et sont particulièrement intéressants lorsqu'ils sont utilisés directement sur les pixels des trames. En effet, l'apprentissage non-supervisé des couches cachées permet d'extraire des représentations complexes et donc d'inférer des caractéristiques robustes de manière automatique. Le gain de la modélisation neuro-markovienne en général et plus précisément des DNN est intéressant lorsque la taille du lexique de mots augmente. Les performances baissent en rappel mais ne chutent pas et sont relativement stables en termes de précision illustrant l'intérêt du modèle de ligne utilisé ainsi que la modélisation

neuro-markovienne implémentée.

Bien que les résultats en détection soient toujours un peu plus faibles en ce qui concerne les séquences numériques et alphanumériques, ils ont également été grandement améliorés avec notre combinaison hyride profonde illustrant une nouvelle fois les capacités à inférer des représentations cachées pertinentes. Le test de différentes structures de réseaux de neurones profonds est une piste d'amélioration sérieuse pour notre système. Ceci peut être envisagé à deux niveaux, en ce qui concerne les hyperparamètres des réseaux (nombre de couches, nombre de neurones par couches, arrêt de l'apprentissage, etc.) dans un premier temps. En effet, il n'existe pas à ce jour de travaux concernant le choix de ces hyperparamètres et, comme nous ne disposions pas de serveurs de calcul très performants, cela a limité les tests d'un éventail important de jeux d'hyperparamètres. Ceci nous a obligé à faire des choix peut être condamnables et a contribué à rendre cette tâche très difficile. L'exploration et l'automatisation du choix des hyperparamètres constitue donc une piste d'investigation importante nécessitant un temps de calcul considérable. Une étape de *cross-validation* peut être envisagée mais nécessite une quantité importante de données d'apprentissage que nous n'avons pas à notre disposition.

Dans un second temps, l'amélioration de la combinaison hybride neuro-markovienne peut être envisagée en testant différents types d'auto-encodeurs. Développés récemment, les *Denoising Auto-Encoders* (DAE) et les *Contractive Auto-Encoders* (CAE) proposent des extensions aux auto-encodeurs classiques :

les DAE [Vincent 2008, Larochelle 2009, Erhan 2009, Erhan 2010, Bastien 2010, Bengio 2011, Vincent 2010] se démarquent des auto-encodeurs en cherchant à reconstruire les entrées qui ont au préalable été bruitées. Ceci permet d'apprendre des représentations plus robustes aux variations des entrées. Dans notre cas, on peut imaginer automatiser le redressement de l'écriture manuscrite en utilisant cette procédure.

les CAE [Rifai 2011b, Rifai 2011c, Rifai 2011d, Rifai 2011a] se démarquent des DAE en pénalisant la reconstruction plutôt que les entrées. Cela permet d'encourager les représentations cachées à contracter l'espace d'entrée dans des régions (*manifold*) séparées par des zones de faibles densités. Cette procédure permettrait donc de rendre la discrimination des observations encore plus robuste sans toutefois avoir à bruiter les données d'apprentissage.

Enfin, la mise en place d'un apprentissage conjoint des MMC et des réseaux profonds est à envisager pour itérativement améliorer la combinaison hybride neuro-markovienne. Les couches cachées étant initialisées de manière non su-

pervisée, nous pouvons considérer qu'elles sont convenables pré-entrainées lors de la première passe d'apprentissage. L'apprentissage conjoint peut donc être realisé en effectuant alternativement l'étape de *fine-tuning* de la couche de décision du réseau de neurones profond et un alignement des modèles sur les données d'apprentissage avec la combinaison hybride neuro-markovienne.

Conclusion Générale et Perspectives

Tout au long de ce manuscrit nous avons proposé des solutions pour accéder à l'information alphanumérique pertinente présente dans des documents manuscrits non contraints. Nous avons mis en avant les difficultés rencontrées lorsque l'on cherche à traiter des images de document manuscrit non contraint et en particulier des courriers entrants manuscrits. Afin d'accéder à une partie précise de leur contenu, nous avons justifié une approche de détection d'information au profit d'une lecture complète de document particulièrement lourde en termes de connaissances à prendre en compte et de temps de traitement. Notre approche se démarque des stratégies classiques de détection visant à segmenter un document en mots avant de les reconnaitre, en abordant conjointement les trois points clés que nous avons identifiés lorsque l'on travaille sur des images de document manuscrit non contraint : la segmentation, la reconnaissance et le rejet.

Pour pouvoir intégrer ces trois traitements en parallèle dans notre système, une première contribution a été présentée dans le second chapitre sous la forme d'un modèle de ligne de texte pour la détection d'information. Cette modélisation se base sur les modèles de Markov cachés et met en concurrence deux types d'information à discriminer : *(i)* l'information pertinente caractérisée par un ensemble de requêtes alphabétiques, numériques et alphanumériques à détecter ; ces requêtes sont modélisées de manière classique par la concaténation des MMC de caractères qui les composent ; *(ii)* l'information non pertinente caractérisée par un modèle de rejet sous la forme d'un modèle de remplissage (tous les MMC de caractères sont interconnectés). Ce modèle absorbe toutes les portions du signal d'écriture ne correspondant par à l'information à détecter. Ces transitions sont apprises sur la base d'apprentissage sous la forme d'un modèle de bigrammes de caractères.

Dans un troisième chapitre, le système complet a été décrit. La reconnaissance est supportée par un algorithme de type Beam Search contraint par le modèle de ligne développé. Nous avons montré le potentiel de notre modèle et sa généricité à travers des exemples de détection d'information de types variés : mots clés, séquences numériques, séquences alphanumériques, séquences de mots, etc. Les résultats obtenus par rapport au protocole expérimental mis en place ont montré l'intérêt de notre approche par rapport

à une lecture complète de document en permettant de détecter la présence de séquences de n'importe quel type. Cependant, une baisse importante de la précision du système est observée lorsque la taille du lexique en entrée du système augmente. Une baisse plus importante des performances est notamment remarquée sur des données de type numérique leur syntaxe n'étant pas intégrée aux connaissances *a priori*.

Afin d'améliorer le pouvoir de détection du système et sa précision, une seconde contribution a consisté à mettre en œuvre une combinaison hybride neuro-markovienne profonde que nous avons présentée dans le quatrième chapitre. Cette combinaison hybride permet de mieux discriminer les formes locales présentes dans les trames des lignes de texte et contribue à améliorer le système de détection tout en stabilisant la précision et le rappel du système lorsque le nombre de mots clés augmente. Les points forts des DNN et en particulier leur capacité à considérer des espaces de haute dimension en entrée comme en sortie ont été vérifiés. En effet, les meilleurs résultats ont été obtenus en considérant directement les pixels en entrée du réseau mettant en exergue les propensions de l'architecture profonde à inférer elle même des représentations cachées discriminantes. Les nettes améliorations observées en détection des séquences numériques et alphanumériques confirment ces propos. Nous avons également montré la supériorité de notre combinaison hybride neuro-markovienne profonde par rapport à des combinaisons classiques de type MLP/MMC.

Les perspectives d'évolution de ces travaux sont nombreuses et peuvent être regroupées en deux axes : le moteur de reconnaissance et donc la combinaison hybride neuro-markovienne profonde ainsi que la modélisation de l'information.

Premièrement, le moteur de reconnaissance qui est au cœur du système de détection, peut être amélioré suivant plusieurs pistes. L'optimisation des hyperparamètres, évoquée au cours du chapitre précédent, en est une première au même titre que la mise en œuvre d'un apprentissage conjoint des MMC et du réseau de neurones profond. Une seconde piste consiste à explorer les réseaux de neurones profonds développés très récemment, à savoir les auto-encodeurs débruiteurs [Bengio 2011] qui cherchent à reconstruire les données d'entrée bruitées afin d'apprendre des représentations cachées plus robustes aux variations, ou encore les auto-encodeurs contracteurs [Rifai 2011a] qui visent à pénaliser les reconstructions plutôt que les entrées afin de les regrouper dans des zones de l'espace de représentations séparées par des régions de faible densité. La robustesse face aux variabilités permise par les algorithmes d'apprentissage de ces nouveaux auto-encodeurs peut être bénéfique

à l'amélioration de notre système de détection. Il pourrait être intéressant par exemple de tester un système de détection sans prétraitement et de voir comment se comporte un réseau profond à base d'auto-encodeurs débruiteurs face à la variabilité de l'écriture.

Un second axe d'amélioration concerne la modélisation de l'information à détecter. Nous avons présenté un système générique de détection d'information qui peut être aisément spécialisé pour détecter des données contraintes dans leur syntaxe et dans leur forme. Une modélisation plus fine de l'information pertinente à l'aide de modèle de remplissage plus spécifique peut être par exemple considérée en lieu et place des classiques concaténation de modèles de caractères correspondant à des requêtes *ascii* rigides :

- l'extraction d'un numéro de téléphone peut être guidée par la détection de la séquence « *Tél :* » suivi d'un modèle de remplissage intégrant uniquement les 10 chiffres.
- l'extraction d'un nom de personne peut être guidée par la détection des séquences Mr ou Mme suivi d'un modèle de remplissage n'intégrant que des lettres.
- l'extraction de noms de communes, de noms de rue, de dates peut également être envisagée de manière similaire en détectant les mots clés spécifiques à ces entités : rue, boulevard, bvd, avenue, ave pour l'extraction de nom de rue, lundi, mardi, etc., janvier, jan., février, fév., etc. pour les dates.

Ces quelques exemples sont un premier pas vers la modélisation et la recherche d'expressions régulières. Pour chaque cas décrit ci-dessus, la partie variable de la requête est décrite et sous-entend des connaissances *a priori* sur ces requêtes. La difficulté résiderait donc dans la modélisation de leurs parties variables en fonction de la requête. À l'image des moteurs de recherche d'information, la construction du modèle d'une requête générique pourrait par exemple être guidée par des connaissances apprises sous la forme d'un profil utilisateur et ainsi s'améliorer dans le temps en proposant des suggestions de recherche personnalisées et évolutives. La souplesse, la robustesse et la généricité en recherche comme en extraction d'information dans des documents manuscrits non contraints représentent pour nous les véritables challenges de demain.

Bibliographie

[Adamek 2007] T. Adamek, N. E. Connor et A. F. Smeaton. *Word matching using single closed contours for indexing handwritten historical documents*. IJDAR, vol. 9, no. 2, pages 153–165, 2007. 24, 56, 57, 58

[Ait-Mohand 2011] K. Ait-Mohand. *Techniques d'adaptation de modèles Markoviens. Application à la reconnaissance de documents anciens*. PhD thesis, Université de Rouen, 2011. 54, 84, 93

[Al-Hajj 2007] R. Al-Hajj. *Reconnaissance hors ligne de textes manuscrits cursifs par l'utilisation de systèmes hybrides et de techniques d'apprentissage automatique*. PhD thesis, ENST de Paris, 2007. 87

[Altun 2003] Y. Altun, I. Tsochantaridis et T. Hofmann. *Hidden markov support vector machines*. Machine Learning, vol. 20, no. 1, page 3, 2003. 50

[Andrew 2006] G. Andrew. *A hybrid markovsemi-markov conditional random field for sequence segmentation*. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pages 465–472, 2006. 50

[Arivazhagan 2007] M. Arivazhagan, H. Srinivasan et S. Srihari. *A statistical approach to line segmentation in handwritten documents*. Document Recognition and Retrieval XIV, vol. 6500, no. 1, 2007. 76

[Arnold 2010] L. Arnold, H. Paugham-Moisy et M. Sebag. *Optimisation de la Topologie pour les Réseaux de Neurones Profonds*. RFIA 2010, pages 1–4, 2010. 143, 146, 147

[Aubert 2002] X. L. Aubert. *An overview of decoding techniques for large vocabulary continuous speech recognition*. Computer Speech & Language, vol. 16, no. 1, pages 89–114, 2002. 25, 35, 53

[Baeza-Yates 1999] R. A. Baeza-Yates et B. Ribeiro-Neto. Modern information retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. 9, 11, 12, 13

[Bastien 2010] F. Bastien, Y. Bengio, A. Bergeron, N. Boulanger-Lewandowski, T. M. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache, X. Glorot, X. Muller, S. Pannetier Lebeuf, R. Pascanu, S. Rifai, F. Savard et G. Sicard. *Deep Self-Taught Learning for Handwritten Character Recognition*. CoRR, vol. abs/1009.3589, 2010. 173

[Basu 2007] S. Basu, C. Chaudhuri, M. Kundu, M. Nasipuri et D. K. Basu. *Text line extraction from multi-skewed handwritten documents*. Pattern Recogn., vol. 40, no. 6, pages 1825–1839, 2007. 76

[Baum 1966] L. E. Baum et T. Petrie. *Statistical Inference for Probabilistic Functions of Finite State Markov Chains*. The Annals of Mathematical Statistics, vol. 37, no. 6, pages 1554–1563, 1966. 94

[Baum 1972] L. E. Baum. *An inequalty and associated maximization technique in statistical estimation of probabilistic functions of a Markov process*. Inequalities, vol. 3, pages 1–8, 1972. 49

[Bazzi 2000] I. Bazzi, J. Glass et A. C. Smith. *Modeling Out-Of-Vocabulary Words For Robust Speech Recognition*. PhD thesis, Massachusetts Institute of Technology, 2000. 56, 62, 140

[Bellman 1962] R. E. Bellman et S. E. Dreyfus. Applied Dynamic Programming. Princetown University Press, 1962. 44, 58

[Bengio 1995] Y. Bengio, Y. LeCun, C. Nohl et C. Burges. *LeRec : a NN/HMM hybrid for on-line handwriting recognition*. Neural Comput., vol. 7, pages 1289–1303, November 1995. 50, 51, 140, 143, 144, 152, 154

[Bengio 2007a] Y. Bengio. *Learning deep architectures for AI*. Rapport technique, Dept. IRO, Universite de Montreal, 2007. 147, 150

[Bengio 2007b] Y. Bengio, P. Lamblin, D. Popovici et H. Larochelle. *Greedy layer-wise training of deep networks*. In NIPS, pages 153–160, 2007. 149

[Bengio 2007c] Y. Bengio et Y. Lecun. *Scaling learning algorithms towards AI*. Large-Scale Kernel Machines, pages 1–41, 2007. 140, 142, 145

[Bengio 2009] Y. Bengio. *Learning Deep Architectures for AI*. Found. Trends Mach. Learn., vol. 2, pages 1–127, January 2009. 159

[Bengio 2011] Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. M. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache, X. Glorot, X. Muller, S. Pannetier Lebeuf, R. Pascanu, S. Rifai, F. Savard et G. Sicard. *Deep Learners Benefit More from Out-of-Distribution Examples*. Journal of Machine Learning Research - Proceedings Track, vol. 15, pages 164–172, 2011. 173, 176

[Bertolami 2006] R. Bertolami, M. Zimmermann et H. Bunke. *Rejection strategies for offline handwritten text line recognition*. Pattern Recognition Letters, vol. 27, no. 16, pages 2005–2012, 2006. 56, 59, 61, 84, 128

[Blum 1967] H. Blum. *A transformation for extracting new descriptors of shape.* Models for the perception of speech and visual form, vol. 19, no. 5, pages 362–380, 1967. 81, 82

[Bottou 1991] L. Bottou et P. Gallinari. A framework for the cooperation of learning algorithms. Citeseer, 1991. 151

[Bourlard 1998] H. Bourlard et N. Morgan. *Hybrid HMM/ANN systems for speech recognition : Overview and new research directions.* Adaptive Processing of Sequences and Data Structures, pages 389–417, 1998. 50, 140, 141, 151, 154

[Brakensiek 2004] A. Brakensiek et G. Rigoll. *Handwritten Address Recognition Using Hidden Markov Models.* Reading and Learning, pages 103–122, 2004. 61

[Breiman 2001] Leo Breiman. *Random Forests.* Machine Learning, vol. 45, no. 1, pages 5–32, 2001. 140

[Brodic 2011] D. Brodic, D.R. Milivojevic et Z.N. Milivojevic. *An Approach to a Comprehensive Test Framework for Analysis and Evaluation of Text Line Segmentation Algorithms.* Sensors, vol. 11, no. 9, pages 8782–8812, 2011. 75

[Caillault 2007] É. Caillault et C. Viard-Gaudin. *Mixed Discriminant Training of Hybrid ANN/HMM Systems for Online Handwritten Word Recognition.* IJPRAI, vol. 21, no. 1, pages 117–134, 2007. 152, 154

[Camastra 2007] F. Camastra. *A SVM-based cursive character recognizer.* Pattern Recognition, vol. 40, no. 12, pages 3721–3727, 2007. 145

[Casey 1996] R. G. Casey et E. Lecolinet. *A Survey of Methods and Strategies in Character Recognition.* IEEE Trans. on PAMI, vol. 18, no. 7, pages 690–706, 1996. 43, 152

[Chatelain 2006a] C. Chatelain. *Extraction de séquences numériques dans les documents manuscrits quelconques.* PhD thesis, Université de Rouen, 2006. 132, 134

[Chatelain 2006b] C. Chatelain, L. Heutte et T. Paquet. *Segmentation-driven recognition applied to numerical field extraction from handwritten incoming mail documents.* Document Analysis System, Nelson, New Zealand, LNCS 3872, Springer, pages 564–575, 2006. 28, 42, 87, 133

[Chen 2000] Y.-K. Chen et J.-F. Wang. *Segmentation of Single- or Multiple-Touching Handwritten Numeral String Using Background and Foreground Analysis.* IEEE Trans. on PAMI, vol. 22, no. 11, pages 1304–1317, 2000. 41, 42, 43, 81

[Chinchor 1998] N. A. Chinchor. *Named Entity Task Definition.* Proceedings of the 7th Message Understanding Conference (MUC-7), pages 1–21, 1998. 14, 17, 18

[Cho 1997] S. B. Cho. *Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals.* Neural Networks, vol. 8, no. 1, pages 43–53, Janvier 1997. 143

[Choi 2005] Y. Choi, C. Cardie, E. Riloff et S. Patwardhan. *Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns.* Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing, pages 355–362, 2005. 16

[Choisy 2007] C. Choisy. *Dynamic Handwritten Keyword Spotting Based on the NSHP-HMM.* Proc. ICDAR, vol. 1, pages 242–246, 2007. 24, 25, 56, 59, 60

[Collobert 2002] R. Collobert, S. Bengio et J. Mariéthoz. *Torch : a modular machine learning software library.* IDIAP Research Report 02-46, 2002. 90

[Collobert 2008] R. Collobert et J. Weston. *A unified architecture for natural language processing : deep neural networks with multitask learning.* Proceedings of the 25th international conference on Machine learning, pages 160–167, 2008. 141

[Congedo 1995] G. Congedo, G. Dimauro, S. Impedovo et G. Pirlo. *Segmentation of numeric strings.* Proceedings of the Third International Conference on Document Analysis and Recognition, vol. 2, page 1038, 1995. 43

[Dempster 1977] A. P. Dempster, N. M. Laird et D. B. Rubin. *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society, vol. 39, pages 1–38, 1977. 50

[Dimauro 1997] G. Dimauro, S. Impedovo, G. Pirlo et A. Salzo. *Automatic Bankcheck Processing : A New Engineered System.* IJPRAI, vol. 11, no. 4, pages 467–504, 1997. 23, 24

[Do 2006] T.M.T. Do et T. Artières. *Champs de Markov conditionnels pour le traitement de séquences.* EGC 2006, 2006. 50

[Do 2009] T.M.T. Do et T. Artières. *Large margin training for hidden markov models with partially observed states.* Proceedings of the 26th Annual International Conference on Machine Learning, pages 265–272, 2009. 50

[Do 2011] T.M.T. Do et T. Artières. *Modèle hybride champ markovien conditionnel et réseau de neurones profond*. Document numérique, vol. 14, no. 2, pages 11–27, 2011. 48, 50, 83

[Duda 1972] R. O. Duda et P. E. Hart. *Use of the Hough transformation to detect lines and curves in pictures*. Communication of the ACM, vol. 15, pages 11–15, January 1972. 77

[Dupont 2011] G. Dupont. *Apprentissage implicite pour la recherche d'information*. PhD thesis, Université de Rouen, 2011. 10, 12

[El-Yacoubi 1999] A. El-Yacoubi, R. Sabourin, C. Y. Suen et M. Gilloux. *An HMM-Based Approach for Off-Line Unconstrained Handwritten Word Modeling and Recognition*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 8, pages 752–760, 1999. 43, 44

[El-Yacoubi 2002] M. A. El-Yacoubi, M. Gilloux et J.-M. Bertille. *A Statistical Approach for Phrase Location and Recognition within a Text Line : An Application to Street Name Recognition*. IEEE Trans. on PAMI, vol. 24, no. 2, pages 172–188, 2002. 22, 25, 38, 62, 63, 64, 84

[Elnagar 2003] A. Elnagar et R. Alhajj. *Segmentation of connected handwritten numeral strings*. Pattern Recognition, vol. 36, no. 3, pages 625–634, 2003. 43, 81

[Erhan 2009] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio et P. Vincent. *The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training*. Journal of Machine Learning Research - Proceedings Track, vol. 5, pages 153–160, 2009. 173

[Erhan 2010] D. Erhan, Y. Bengio, A. C. Courville, P.-A. Manzagol, P. Vincent et S. Bengio. *Why Does Unsupervised Pre-training Help Deep Learning ?* Journal of Machine Learning Research, vol. 11, pages 625–660, 2010. 173

[Espana-Boquera 2011] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya et F. Zamora-Martinez. *Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, pages 767–779, April 2011. 24, 25, 48, 80, 81, 83, 85, 140, 152, 154, 155

[Favata 1998] J. Favata, S. Srihari et V. Govindaraju. *Off-line handwritten sentence recognition*. Proc. IWFHR, vol. 1, pages 171–176, 1998. 25, 60

[Feng 2006] S. Feng, R. Manmatha et A. McCallum. *Exploring the Use of Conditional Random Field Models and HMMs for Historical Handwritten Document Recognition*. Proceedings of the Second International

Conference on Document Image Analysis for Libraries, pages 30–37, 2006. 83

[Ferilli 2011] S. Ferilli. Automatic digital document processing and management : Problems, algorithms and techniques. Springer Publishing Company, Incorporated, 1st édition, 2011. 13, 14, 17

[Fernández 2007] S. Fernández, A. Graves et J. Schmidhuber. *An application of recurrent neural networks to discriminative keyword spotting.* Artificial Neural Networks–ICANN 2007, pages 220–229, 2007. 51, 141

[Fetter 1998] P. Fetter. *Detection and Transcription of OOV Words.* PhD thesis, Berlin Technical University, 1998. 62

[Finn 2004] A. Finn et N. Kushmerick. *Multi-level Boundary Classification for Information Extraction.* European Conference on Machine Learning, pages 111–122, 2004. 16

[Fischer 2010] A. Fischer, A. Keller, V. Frinken et H. Bunke. *HMM-based Word Spotting in Handwritten Documents Using Subword Models.* International Conference on Pattern Recognition, pages 3416–3419, 2010. 25, 62, 63, 64, 65, 84, 87, 123, 134, 165

[Flusser 2000] J. Flusser. *On the independence of rotation moment invariants.* Pattern Recognition, vol. 33, no. 9, pages 1405–1410, 2000. 85

[Fornay 1973] G. D. Fornay. *The Viterbi Algorithm.* Proceedings of the IEEE, vol. 61, no. 3, pages 268–278, 1973. 49

[Frinken 2010] V. Frinken, A. Fischer et H. Bunke. *A Novel Word Spotting Algorithm Using Bidirectional Long Short-Term Memory Neural Networks.* Artificial Neural Networks in Pattern Recognition, vol. 5998, pages 185–196, 2010. 25, 51, 56, 62, 64

[Frinken 2012] V. Frinken, A. Fischer, R. Manmatha et H. Bunke. *A Novel Word Spotting Method Based on Recurrent Neural Networks.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, pages 211–224, 2012. 54, 64, 123, 124, 134, 140, 165

[Gatos 2010] B. Gatos, N. Stamatopoulos et G. Louloudis. *ICFHR 2010 Handwriting Segmentation Contest.* ICFHR, pages 737–742, 2010. 76, 77

[Gorski 1997] N. Gorski. *Optimizing Error-Reject Trade off in Recognition Systems.* Proceedings of the 4th International Conference on Document Analysis and Recognition, pages 1092–1096, 1997. 61

[Gorski 2001] N. Gorski, V. Anisimov, E. Augustin, O. Baret et S. Maximov. *Industrial bank check processing : the A2iA CheckReader^{TM}.* IJDAR, vol. 3, no. 4, pages 196–206, 2001. 24, 39, 140, 151, 152, 154

[Govindaraju 1996] V. Govindaraju et R. K. Krishnamurthy. *Holistic handwritten word recognition using temporal features derived from off-line images*. Pattern Recognition Letters, vol. 17, no. 5, pages 537–540, 1996. 39

[Graves 2005] A. Graves et J. Schmidhuber. *Framewise phoneme classification with bidirectional LSTM and other neural network architectures*. Neural Networks, vol. 18, no. 5-6, pages 602–610, 2005. 51

[Graves 2006] A. Graves, S. Fernández, F. Gomez et J. Schmidhuber. *Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks*. Proceedings of the 23rd international conference on Machine learning, pages 369–376, 2006. 51, 83

[Graves 2009] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke et J. Schmidhuber. *A Novel Connectionist System for Unconstrained Handwriting Recognition*. IEEE Trans. on PAMI, vol. 31, no. 5, pages 855–868, 2009. 24, 25, 36, 48, 51, 52, 65, 83, 140, 143

[Grishman 1995] R. Grishman et B. Sundheim. *Design of the MUC-6 evaluation*. Proceedings of the 6th Message Understanding Conference, pages 1–11, 1995. 14

[Grishman 1997] R. Grishman. *Information extraction : Techniques and challenges*. Information Extraction A Multidisciplinary Approach to an Emerging Information Technology, pages 10–27, 1997. 9, 14

[Grosicki 2009] E. Grosicki et H. El-Abed. *ICDAR 2009 Handrwriting Recognition Competition*. ICDAR, pages 1398–1402, 2009. 26, 27, 28, 30, 34, 39, 41, 51, 87, 95, 101, 120, 121, 161, 163

[Gu 2006] Z. Gu et N. Cercone. *Segment-based hidden Markov models for information extraction*. Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pages 481–488, 2006. 16, 84

[Guillevic 1994] D. Guillevic et C. Y. Suen. *Cursive Script Recognition : A Sentence Level Recognition Scheme*. In International Workshop on Frontiers of Handwriting Recognition, pages 216–223, 1994. 40

[Ha 1996] T. M. Ha, G. Kaufmann et H. Bunke. *Text Localisation and Handwriting Recognition*. Rapport technique, Université de Berne, pages 1–23, 1996. 48

[Ha 1997] T. M. Ha et H. Bunke. *Off-Line Handwritten Numeral String Recognition*. Rapport technique, Université de Berne, pages 1–13, 1997. 48, 132

[Harris 1954] Z. Harris. *Distributional structure.* Word, pages 146–162, 1954. 11

[Heutte 1997] L. Heutte, P. Barbosa-Pereira, O. Bougeois, J.-V. Moreau, B. Plessis, P. Courtellemont et Y. Lecourtier. *Multi-Bank Check Recognition System : Consideration on The Numeral Amount Recognition Module.* IJPRAI, vol. 11, no. 4, pages 595–618, 1997. 23, 38

[Heutte 1998] L. Heutte, T. Paquet, J.V. Moreau, Y. Lecourtier et C. Olivier. *A structural-statistical feature based vector for handwritten character recognition.* Pattern Recognition Letters, vol. 19, pages 629–641, 1998. 47, 87

[Hinton 1986] G. E. Hinton et T. J. Sejnowski. *Learning and relearning in Boltzmann machines.* Parallel distributed processing : explorations in the microstructure of cognition, vol. 1, pages 282–317, 1986. 143

[Hinton 2006a] G. E. Hinton, S. Osindero et Y.-W. Teh. *A Fast Learning Algorithm for Deep Belief Nets.* Neural Comp., vol. 18, no. 7, pages 1527–1554, Juillet 2006. 140, 147

[Hinton 2006b] G. E. Hinton et R. R. Salakhutdinov. *Reducing the Dimensionality of Data with Neural Networks.* Science, vol. 313, no. 5786, pages 504–507, Juillet 2006. 142, 145

[Hobbs 2010] J. Hobbs et E. Riloff. *Handbook of Natural Language Processing, chapter 21 : Information Extraction.* Chapman and Hall/CRC Press, 2010. 14, 15, 16, 17, 18, 84

[Hopfield 1982] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Sciences, vol. 79, no. 8, pages 2554–2558, Avril 1982. 143

[Hu 1962] M.-K. Hu. *Visual pattern recognition by moment invariants.* IEEE Trans. on Information Theory, vol. 8, no. 2, pages 179–187, 1962. 85

[Huffman 1996] S. B. Huffman. *Learning information extraction patterns from examples.* Learning for Natural Language Processing, pages 246–260, 1996. 16

[Jaeger 2000] S. Jaeger, S. Manke et A. Waibel. *Npen++ : An On-Line Handwriting Recognition System.* in 7th International Workshop on Frontiers in Handwriting Recognition, pages 249–260, 2000. 143, 144, 145

[Jain 2000] A. K. Jain. *Statistical Pattern recognition : a review.* IEEE Trans. on PAMI, vol. 22, no. 1, pages 4–37, 2000. 46, 141

[Joachims 1998] T. Joachims. *Text categorization with support vector machines : Learning with many relevant features.* Machine Learning : ECML-98, pages 137–142, 1998. 9, 13

[Johansen 1996] F. T. Johansen. *A Comparison of hybrid HMM architectures using global discriminative training.* Speech and Audio Processing, IEEE Transactions on, pages 1–4, 1996. 157

[Juang 1992] B.H. Juang et S. Katagiri. *Discriminative learning for minimum error classification [pattern recognition].* Signal Processing, IEEE Transactions on, vol. 40, no. 12, pages 3043–3054, 1992. 50

[Kapoor 2004] R. Kapoor, D. Bagai et T. S. Kamal. *A new algorithm for skew detection and correction.* Pattern Recognition Letters, vol. 25, no. 11, pages 1215–1229, 2004. 80

[Kavallieratos 1997] E. Kavallieratos, N. Antoniades, N. Fakotakis et G. Kokkinakis. *Extraction and recognition of handwritten alphanumeric characters from application forms.* International Conference on Digital Signal Processing Proceedings, vol. 2, pages 695–698, 1997. 20, 38

[Keaton 1997] P. Keaton, H. Greenspan et R. Goodman. *Keyword Spotting for Cursive Document Retrieval.* DIA '97 : Proceedings of the 1997 Workshop on Document Image Analysis, pages 74–82, 1997. 81

[Kenyon 1998] Claire Kenyon et Hèléne Paugam-Moisy. *Multilayer neural networks and polyhedral dichotomies.* Annals of Mathematics and Artificial Intelligence, vol. 24, no. 1-4, pages 115–128, Novembre 1998. 144

[Kessentini 2010] Y. Kessentini, T. Paquet et A. Benhamadou. *Off-line Handwritten Word Recognition Using Multi-Stream Hidden Markov Models.* Pattern Recognition Letters, vol. 31, pages 60–70, 2010. 87

[Khurshid 2008] K. Khurshid, C. Faure et N. Vincent. *Recherche de mots dans des images de documents par appariement de caractères.* Colloque International Francophone sur l'Ecrit et le Document, CIFED 2008, France, vol. 1, October 2008. 35, 36

[Kim 1993] J. Kim et D Moldovan. *Acquisition of Semantic Patterns for Information Extraction from Corpora.* Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications, pages 171–176, 1993. 16

[Kim 1996] G. Kim et V. Govindaraju. *Recognition of Handwritten Phrases as Applied to Street Name Images.* Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition, page 459, 1996. 81

[Kim 1997] G. Kim et V. Govindaraju. *Bankcheck Recognition Using Cross Validation Between Legal and Courtesy Amounts.* IJPRAI, vol. 11, no. 4, pages 657–674, 1997. 24

[Kimura 1994] F. Kimura, S. Tsuruoka, Y. Miyake et M. Shridhar. *A Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten*

Words. IEICE trans. on Information and Systems, vol. 77, no. 7, pages 785–793, 1994. 47, 57, 85

[Kimura 1997] F. Kimura, N. Kayahara, Y. Miyake et M. Shridhar. *Machine and Human Recognition of Segmented Characters from Handwritten Words*. Proceedings of the 4th International Conference on Document Analysis and Recognition, page 866, 1997. 80

[Knerr 1997] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price et J.-C. Simon. *The A2iA Intercheque System : Courtesy Amount and Legal Amount Recognition for French Checks*. IJPRAI, vol. 11, no. 4, pages 505–548, 1997. 23, 38, 39

[Knerr 1998] S. Knerr et E. Augustin. *A neural network-hidden markov model hybrid for cursive word recognition*. ICPR, vol. 2, pages 1518–1520, 1998. 50, 151, 154

[Koch 2004] G. Koch, L. Heutte et T. Paquet. *Combination of Contextual Information for Handwritten Word Recognition*. IWFHR2004, 2004. 40, 43

[Koch 2006] G. Koch. *Catégorisation automatique de documents manuscrits : application aux courriers entrants*. PhD thesis, Université de Rouen, 2006. 25, 28, 29, 44, 45, 62, 63, 64, 78, 80, 82, 87, 133, 134, 140

[Koerich 2002a] A. Koerich. *Large vocabulary off-line handwritten word recognition*. PhD thesis, Ecole de Technologie Superieure de Montréal, Canada, 2002. 43, 44

[Koerich 2002b] A. L. Koerich, Y. Leydier, R. Sabourin et C. Y. Suen. *A Hybrid Large Vocabulary Handwritten Word Recognition System using Neural Networks with Hidden Markov Models*. Proc. 8th International Workshop on Frontiers in Handwriting Recognition, pages 99–104, 2002. 25, 93, 152

[Koerich 2003] A.L. Koerich, R. Sabourin et C.Y. Suen. *Large vocabulary off-line handwriting recognition : A survey*. Pattern Analysis and Applications, vol. 6, pages 97–121, 2003. 40, 52, 84, 152

[Kolcz 2000] A. Kolcz, J. Alspector, M. Augusteijn, R. Carlson et G.V. Popescu. *A Line-Oriented Approach to Word Spotting in Handwritten Documents*. PAA, vol. 3, no. 2, pages 153–168, 2000. 59

[Korfhage 2008] R. R. Korfhage. Information storage and retrieval. Wiley, 1st édition, 2008. 12

[Krogh 1994] A. Krogh, I. S. Mian et D. Haussler. *A hidden Markov model that finds genes in E.coli DNA*. Nucleic Acids Research, vol. 22, no. 22, pages 4768–4778, 1994. 84

[Kuropka 2004] D. Kuropka. Modelle zur reprasentation naturlichsprachlicher dokumente - information-filtering und -retrieval mit relationalen datenbanken. Advances in Information Systems and Management Science, 2004. 11

[Lafferty 2001] John D. Lafferty, Andrew McCallum et Fernando C. N. Pereira. *Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data*. Proceedings of the Eighteenth International Conference on Machine Learning, pages 282–289, 2001. 48, 50, 51, 83

[Lang 1990] K. J. Lang, A. H. Waibel et G. E. Hinton. *A time-delay neural network architecture for isolated word recognition.* Neural networks, vol. 3, no. 1, pages 23–43, 1990. 143, 152

[Larochelle 2009] H. Larochelle, Y. Bengio, J. Louradour et P. Lamblin. *Exploring Strategies for Training Deep Neural Networks*. J. Mach. Learn. Res., vol. 10, pages 1–40, June 2009. 145, 159, 173

[Lavrenko 2004] V. Lavrenko, T. M. Rath et R. Manmatha. *Holistic Word Recognition for Handwritten Historical Documents*. Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04), pages 278–286, 2004. 40, 55

[LeCun 1987] Y. LeCun. *Modèles connexionnistes de l'apprentissage (connectionist learning models)*. PhD thesis, Université P. et M. Curie (Paris 6), June 1987. 141, 144

[LeCun 1990] Y. LeCun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel et H. S. Baird. *Handwritten Zip Code Recognition with Multilayer Networks*. Proc. of the International Conference on Pattern Recognition, vol. II, pages 35–40, 1990. 47, 141

[LeCun 1995] Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard et V. Vapnik. *Comparison of learning algorithms for handwritten digit recognition*. International Conference on Artificial Neural Networks, pages 53–60, 1995. 141, 152

[LeCun 1998] Y. LeCun, L. Bottou, Y. Bengio et P. Haffner. *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE, vol. 86, no. 11, pages 2278–2324, November 1998. 152

[Lee 1997] L. Lee, M. Lizarraga, N. Gomes et A. Koerich. *A Prototype for Brazilian Bankcheck Recognition*. IJPRAI, vol. 11, no. 4, pages 549–570, 1997. 23, 39

[Leydier 2009] Y. Leydier, A. Ouji, F. Lebourgeois et H. Emptoz. *Towards an omnilingual word retrieval system for ancient manuscripts.* Pattern Recognition, vol. 42, no. 9, pages 2089–2105, 2009. 57

[Lii 1995] J. Lii. *Use of character recognition and syntax in locating address paragraphs in complex documents.* PhD thesis, State University of New York at Buffalo, Buffalo, NY, USA, 1995. 22

[Likforman-Sulem 1994] L. Likforman-Sulem et C. Faure. *Extracting text lines in handwritten documents by perceptual grouping.* Advances in handwriting and drawing : a multidisciplinary approach, pages 117–135, 1994. 76, 77

[Likforman-Sulem 1995] L. Likforman-Sulem et C. Faure. *Une méthode de résolution des conflits d'alignements pour la segmentation des documents manuscrits.* Traitement du Signal, vol. 12, pages 541–549, 1995. 76, 77, 78

[Likforman-Sulem 2007] L. Likforman-Sulem, A. Zahour et B. Taconet. *Text line segmentation of historical documents : a survey.* International Journal on Document Analysis and Recognition, vol. 9, no. 2, pages 123–138, 2007. 75

[Liu 2004] C.-L. Liu, H. Sako et H. Fujisawa. *Effects of Classifier Structures and Training Regimes on Integrated Segmentation and Recognition of Handwritten Numeral Strings.* IEEE Trans. on PAMI, vol. 26, no. 11, pages 1395–1407, 2004. 44

[Lorette 1999] G. Lorette. *Handwritting Recognition or reading ? What is the situation at the dawn of the 3rd Millenium ?* IJDAR, vol. 2, no. 1, pages 2–12, 1999. 25, 35, 53, 83

[Louloudis 2009] G. Louloudis, B. Gatos, I. Pratikakis et C. Halatsis. *Text line and word segmentation of handwritten documents.* Pattern Recogn., vol. 42, pages 3169–3183, December 2009. 35, 36, 77

[Lüthy 2007] F. Lüthy, T. Varga et H. Bunke. *Using Hidden Markov Models as a Tool for Handwritten Text Line Segmentation.* ICDAR, pages 8–11, 2007. 35, 36

[Madasu 2003] V. K. Madasu, M. Hafizuddin, M. Yusof, M. Hanmandlu et K. Kubik. *Automatic Extraction of Signatures from Bank Cheques and Other Documents.* DICTA, pages 591–600, 2003. 23

[Manmatha 1996] R. Manmatha, C. Han et E. M. Riseman. *Word Spotting : A New Approach to Indexing Handwriting.* CVPR, pages 631–637, 1996. 25, 57

[Manos 1997] A. S. Manos, R. S. Manos et V. W. Zue. *A Study On Out-Of-Vocabulary Word Modeling For A Segment-Based Keyword Spotting System*. Masters Thesis, MIT, 1997. 56, 62, 99

[Marti 2000] U. Marti et H. Bunke. *Handwritten sentence recognition*. Proc. ICDAR, vol. 3, pages 467–470, 2000. 25, 36, 53, 60, 63

[Marti 2001a] U. Marti et H. Bunke. *Text line segmentation and word recognition in a system for general writer independent handwriting recognition*. In Sixth International Conference on Document Analysis and Recognition, vol. 1, pages 159–163, 2001. 35, 36, 48, 49, 53, 54, 55, 76, 83, 87, 93

[Marti 2001b] U.-V. Marti et H. Bunke. *Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System*. IJPRAI, vol. 15, no. 1, pages 65–90, 2001. 37, 54, 84, 85, 86

[Marti 2002] U.-V. Marti et H. Bunke. *The IAM-database : an English sentence database for offline handwriting recognition*. IJDAR, vol. 5, no. 1, pages 39–46, 2002. 34, 54

[Marukatat 2001] S. Marukatat, T. Artires, P. Gallinari et B. Dorizzi. *Sentence Recognition through hybrid neuro-markovian modeling*. In International Conference on Document Analysis and Recognition (ICDAR, pages 731–735, 2001. 36, 50, 53, 152, 154

[Marukatat 2002] S. Marukatat, T. Artieres, P. Gallinari et B. Dorizzi. *Rejection Measures for Handwriting Sentence Recognition*. In 8th Int. Workshop on Frontiers in Handwriting Recognition, pages 24–29, 2002. 61

[Matan 1992] O. Matan, H. S. Baird, J. Bromley, C. J. C. Burges, J. S. Denker, L. D. Jackel, Y. LeCun, E. P. D. Pednault, W. Satterfield, C. E. Stenard et T. J. Thompson. *Reading Handwritten Digits : A ZIP Code Recognition System*. IEEE Computer, vol. 25, no. 7, pages 59–63, 1992. 143

[McCallum 2005] A. McCallum. *A conditional random field for discriminatively-trained finite-state string edit distance*. Technical Report,University of Massachusetts Amherst, 2005. 50

[McCulloch 1943] W. McCulloch et W. Pitts. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biology, vol. 5, no. 4, pages 115–133, 1943. 141

[Minsky 1969] M. Minsky et S. Papert. Perceptrons : An introduction to computational geometry. The MIT Press, 1969. 142

[Moore 2002] D. Moore. *TODE : A Decoder for Continuous Speech Recognition.* IDIAP Research Report 02-09, 2002. 90, 92

[Morita 2003] M. Morita, R. Sabourin, F. Bortolozzi et Y. Suen. *Segmentation and recognition of handwritten dates : an HMM-MLP hybrid approach.* Int. J. Doc. Anal. Recognit., vol. 6, no. 4, pages 248–262, 2003. 23, 24, 44, 152

[Myers 1981] C. Myers et L. Rabiner. *A Level-Building Dynamic Time Warping Algorithm for connected word recognition.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, pages 284–297, 1981. 90

[Nicolas 2006] S. Nicolas. *Graphes aléatoires conditionnels pour l'extraction d'information dans les données complexes.* PhD thesis, Université de Rouen, 2006. 76

[Nikolaou 2010] N. Nikolaou, M. Makridis, B. Gatos, N. Stamatopoulos et N. Papamarkos. *Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths.* Image and Vision Computing, vol. 28, pages 590–604, Avril 2010. 35, 36, 76, 77

[Niyogi 1996] D. Niyogi, S. Srihari et V. Govindaraju. *Analysis of printed forms.* Handbook on Optical Character Recognition and Document Image Analysis, pages 330–342, 1996. 20, 38

[Oliveira 2002] L. Oliveira, R. Sabourin, F. Bortolozzi et C. Y. Suen. *Automatic Recognition of Handwritten Numerical Strings : A Recognition and Verification Strategy.* IEEE Trans. on PAMI, vol. 24, no. 11, pages 1438–1454, 2002. 40, 43, 81, 82

[Oliveira 2004] L. S. Oliveira et R. Sabourin. *Support Vector Machines for Handwritten Numerical String Recognition.* Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition, pages 39–44, 2004. 44

[Pal 2003] U. Pal, A. Belaïd et Ch. Choisy. *Touching numeral segmentation using water reservoir concept.* Pattern Recognition Letters, vol. 24, no. 1-3, pages 261–272, 2003. 42, 43

[Parada 2010] C. Parada, M. Dredze, D. Filimonov et F. Jelinek. *Contextual information improves OOV detection in speech.* Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 216–224, 2010. 62

[Patwardhan 2007] S. Patwardhan et E. Riloff. *Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions.* Conference

on Empirical Methods in Natural Language Processing, pages 717–727, 2007. 16

[Perraud 2003] F. Perraud, C. Viard-Gaudin, E. Morin et P.-M. Lallican. *N-Gram and N-Class Models for On line Handwriting Recognition*. Proceedings of the Seventh International Conference on Document Analysis and Recognition, page 1053, 2003. 53, 61

[Pfister 2000] M. Pfister, S. Behnke et R. Rojas. *Recognition of Handwritten ZIP Codes in a Real-World Non-Standard-Letter Sorting System*. Applied Intelligence, vol. 12, no. 1-2, pages 95–115, 2000. 22, 38

[Plamondon 2000] R. Plamondon et S. N. Srihari. *On-line and off-line handwritting recognition : A comprehensive survey*. IEEE Trans. on PAMI, vol. 22, no. 1, pages 63–84, 2000. 35, 49, 53, 83

[Poisson 2004] E. Poisson, C. Viard-Gaudin et P.M. Lallican. *Système TDNN / HMM de reconnaissance de mots cursifs en ligne à apprentissage simplifié*. Conférence Internationale Francophone sur l'Ecrit et le Document, 2004. 48, 50, 51, 83

[Poisson 2005] E. Poisson. *Architecture et Apprentissage d'un Système Hybride Neuro-Markovien pour la Reconnaissance de l'Écriture Manuscrite En-Ligne*. PhD thesis, École polytechnique de Nantes, 2005. 140, 143, 152, 154, 155

[Quattoni 2007] A. Quattoni, S. Wang, L.P. Morency, M. Collins et T. Darrell. *Hidden conditional random fields*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, no. 10, pages 1848–1852, 2007. 50

[Quiniou 2007] S. Quiniou. *Intégration de connaissances linguistiques pour la reconnaissance de textes manuscrits en-ligne*. PhD thesis, INSA de Rennes, Rennes, France, 2007. 37, 53, 61

[Rabiner 1990] L. R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Readings in speech recognition, pages 267–296, 1990. 48, 83, 84, 89, 90, 94, 152

[Rabiner 1993] L. Rabiner et B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall PTR, April 1993. 49, 53, 83, 90

[Ranzato 2006] M. A. Ranzato, C. Poultney, S. Chopra et Y. Lecun. *Efficient Learning of Sparse Representations with an Energy-Based Model*. in NIPS, pages 1137–1144, 2006. 149

[Rath 2002] T. M. Rath, S. Kane, A. Lehman, E. Partridge et R. Manmatha. *Indexing for a Digital Library of George Washington's Manuscripts : A Study of Word Matching Techniques*. Technical Report MM-34, 2002. 13, 25, 34, 56, 57

[Rath 2003a] T. Rath et R. Manmatha. *Features for Word Spotting in Historical Manuscripts*. Proc. ICDAR, pages 218–222, 2003. 57

[Rath 2003b] T. M. Rath et R. Manmatha. *Word Image Matching Using Dynamic Time Warping*. CVPR, pages 521–527, 2003. 24, 57, 58

[Rath 2004] T. Rath, R. Manmatha et V. Lavrenko. *A Search Engine for Historical Manuscript Images*. SIGIR 04, pages 369–376, 2004. 25, 39, 58

[Rey-Debove 2006] J. Rey-Debove et A. Rey. Petit robert. Dictionnaires Le Robert, 2006. 7, 38

[Richard 1991] M.D. Richard et R.P. Lippmann. *Neural network classifiers estimate Bayesian a posteriori probabilities*. Neural computation, vol. 3, no. 4, pages 461–483, 1991. 143

[Rifai 2011a] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin et X. Glorot. *Higher Order Contractive Auto-Encoder*. In ECML/PKDD (2), pages 645–660, 2011. 173, 176

[Rifai 2011b] S. Rifai, X. Muller, X. Glorot, G. Mesnil, Y. Bengio et P. Vincent. *Learning invariant features through local space contraction*. CoRR, vol. abs/1104.4153, 2011. 140, 146, 173

[Rifai 2011c] S. Rifai, P. Vincent, X. Muller, X. Glorot et Y. Bengio. *Contractive Auto-Encoders : Explicit Invariance During Feature Extraction*. In ICML, pages 833–840, 2011. 173

[Rifai 2011d] Salah Rifai, Yann N. Dauphin, Pascal Vincent, Yoshua Bengio et Xavier Muller. *The Manifold Tangent Classifier*. Advances in Neural Information Processing Systems 24, pages 2294–2302, 2011. 173

[Riloff 1993] E. Riloff. *Automatically Constructing a Dictionary for Information Extraction Tasks*. Proceedings of 11th National Conference on Artificial Intelligence, pages 811–816, 1993. 16

[Riloff 1996] E. Riloff. *Automatically Generating Extraction Patterns from Untagged Text*. Proceedings of 13th National Conference on Artificial Intelligence, pages 1044–1049, 1996. 16

[Rodríguez-Serrano 2008] J. A. Rodríguez-Serrano, F. Perronnin et J. Llados. *Score normalization for HMM-based word spotting using a universal background model*. ICFHR 08, July 2008. 24, 59, 60

[Rodríguez-Serrano 2009a] J. A. Rodríguez-Serrano et F. Perronnin. *Handwritten word-spotting using hidden Markov models and universal vocabularies*. Pattern Recognition, pages 2106–2116, February 2009. 54, 56, 59, 61, 123, 124, 140, 165

[Rodríguez-Serrano 2009b] J. A. Rodríguez-Serrano, F. Perronnin et J. Llados. *A similarity measure between vector sequences with application to handwritten word image retrieval*. CVPR, August 2009. 24, 56, 58

[Rodriguez 2008] J. A. Rodriguez et F. Perronnin. *Local gradient histogram features for word spotting in unconstrained handwritten documents*. Proceedings of the 1st International Conference on Handwriting Recognition (ICFHR'08), August 2008. 25, 84, 85, 86, 87

[Rosenblatt 1958] F. Rosenblatt. *The Perceptron : A probabilistic model for information storage and organization in the brain*. Psychological Review, vol. 65, pages 386–408, 1958. 141, 142

[Rothfeder 2003] J. L. Rothfeder, S. Feng et T. M. Rath. *Using Corner Feature Correspondences to Rank Word Images by Similarity*. DIAR, pages 30–35, 2003. 57

[Rumelhart 1986] D. E. Rumelhart, G. E. Hinton et R. J. Williams. *Learning representations by back-propagating errors*. Nature, vol. 323, no. Oct, pages 533–536+, 1986. 144

[Sadri 2007] J. Sadri, C. Y. Suen et T. D. Bui. *A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings*. Pattern Recognition, vol. 40, no. 3, pages 898–919, 2007. 40, 43, 44, 45

[Sakoe 1978] H. Sakoe. *Dynamic programming algorithm optimization for spoken word recognition*. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, pages 43–49, 1978. 44, 58

[Sanchis 2000] A. Sanchis, V. Jimenez et E. Vidal. *Efficient Use of the Grammar Scale Factor to Classify Incorrect Words in Speech Recognition Verification*. Proceedings of the International Conference on Pattern Recognition - Volume 3, pages 3278–3281, 2000. 61

[Sankar 2007] K. Sankar et C. Jawahar. *Probabilistic reverse annotation for large scale image retrieval*. CVPR, pages 1–6, 2007. 58

[Saon 1997] G. Saon et A. Belaïd. *Off-line Handwritten Word Recognition Using A Mixed HMM–MRF Approach*. Fourth International Conference on Document Analysis and Recognition, vol. 1, pages 118–122, 1997. 59

[Saykol 2004] E. Saykol, A. Kemal Sinop, U. Güdükbay, Ö. Ulusoy et A. Enis Çetin. *Content-based retrieval of historical Ottoman documents stored as textual images*. IEEE Transactions on Image Processing, vol. 13, no. 3, pages 314–325, 2004. 56, 57

[Sayre 1973] K. M. Sayre. *Machine recognition of handwritten words : A project report.* Pattern Recognition, vol. 5, no. 3, pages 213–228, 1973. 38

[Schuster 1997] M. Schuster et K.K. Paliwal. *Bidirectional recurrent neural networks.* Signal Processing, IEEE Transactions on, vol. 45, no. 11, pages 2673–2681, 1997. 51

[Sebastiani 2002] F. Sebastiani. *Machine learning in automated text categorization.* ACM Comput. Surv., vol. 34, pages 1–47, March 2002. 13

[Senior 1998] A. W. Senior et A. J. Robinson. *An Off-Line Cursive Handwriting Recognition System.* IEEE Trans. on PAMI, vol. 20, no. 3, pages 309–321, 1998. 25, 51, 60, 81

[Sha 2007] F. Sha et L.K. Saul. *Large margin hidden Markov models for automatic speech recognition.* Advances in neural information processing systems, vol. 19, page 1249, 2007. 50

[Shetty 2007] S. Shetty, H. Srinivasan et S. Srihari. *Handwritten Word Recognition Using Conditional Random Fields.* Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 2, pages 1098–1102, 2007. 36, 43, 50

[Sicard 2006] R. Sicard, T. Artières et E. Petit. *Modélisation de l'écrit en ligne à l'aide de réseaux bayésiens et de caractéristiques relationnelles.* RFIA, 2006. 50

[Slavík 2001] P. Slavík et V. Govindaraju. *Equivalence of Different Methods for Slant and Skew Corrections in Word Recognition Applications.* IEEE Trans. on PAMI, vol. 23, no. 3, pages 323–326, 2001. 79, 80

[Soullard 2011] Yann Soullard et Thierry Artières. *Hybrid HMM and HCRF model for sequence classification.* ESANN, pages 1–6, 2011. 48, 50, 83

[Srihari 1997] S. N. Srihari et E. J. Kuebert. *Integration of hand-written address interpretation technology into the United States Postal Service Remote Computer Reader system.* Proceedings of the 4th International Conference on Document Analysis and Recognition, pages 892–896, 1997. 22

[Suen 1999] Ching Y. Suen, Q. Xu et L. Lam. *Automatic recognition of handwritten data on cheques : Fact or fiction ?* Pattern Recognition Letters, vol. 20, no. 11-13, pages 1287–1295, 1999. 23, 39

[Terasawa 2005] K. Terasawa, T. Nagasaki et T. Kawashima. *Eigenspace Method for Text Retrieval in Historical Document Images.* ICDAR, pages 437–441, 2005. 57, 58

[Trentin 2001] E. Trentin et M. Gori. *A survey of hybrid ANN/HMM models for automatic speech recognition.* Neurocomputing, vol. 37, no. 1, pages 91–126, 2001. 50, 140, 141, 151, 155, 156

[Trier 1996] O. Trier, A. Jain et T. Taxt. *Feature extraction methods for character recognition - A survey.* Pattern Recognition, vol. 29, pages 641–662, 1996. 46

[Utgoff 2002] P. E. Utgoff et D. J. Stracuzzi. *Many-layered learning.* Development and Learning, 2002. Proceedings. The 2nd International Conference on, pages 141–146, 2002. 144

[van der Zant 2008] T. van der Zant, L. Schomaker et K. Haak. *Handwritten-Word Spotting Using Biologically Inspired Features.* IEEE Trans. on PAMI, vol. 30, no. 11, pages 1945–1957, 2008. 24, 25, 59

[Vapnik 1998] V. N. Vapnik. Statistical Learning Theory. Wiley-Interscience, Septembre 1998. 47, 140

[Viard-Gaudin 1999] C. Viard-Gaudin, P. M. Lallican, P. Binter et S. Knerr. *The IRESTE On/Off (IRONOFF) Dual Handwriting Database.* Proc. ICDAR, page 455, 1999. 94

[Vincent 2008] P. Vincent, H. Larochelle, Y. Bengio et P.-A. Manzagol. *Extracting and composing robust features with denoising autoencoders.* Proceedings of the 25th international conference on Machine learning, pages 1096–1103, 2008. 147, 173

[Vincent 2010] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio et P.-A. Manzagol. *Stacked Denoising Autoencoders : Learning Useful Representations in a Deep Network with a Local Denoising Criterion.* Journal of Machine Learning Research, vol. 11, no. 3371–3408, Décembre 2010. 173

[Vinciarelli 2002] A. Vinciarelli. *A survey on off-line cursive word recognition.* Pattern Recognition, vol. 35, pages 1433–1446, 2002. 40, 46, 52, 81, 82, 84

[Vinciarelli 2003] A. Vinciarelli. *Offline Cursive Handwriting : From Word to Text Recognition.* IDIAP-RR, vol. 24, 2003. 48, 49, 55, 83, 84, 85, 86, 87

[Vinciarelli 2004] A. Vinciarelli, S. Bengio et H. Bunke. *Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models.* IEEE Trans. on PAMI, vol. 26, no. 6, pages 709–720, 2004. 25, 36, 37, 53, 54, 60, 81, 84, 87, 93

[Vinel 2011] A. Vinel, T.M.T. Do et T. Artières. *Neuro HCRFs pour l'étiquetage de signaux.* 7e Plateforme AFIA : Association Française pour

l'Intelligence Artificielle, Chambéry, 16 au 20 mai 2011, page 409, 2011. 50

[Vintsyuk 1968] T. K. Vintsyuk. *Speech discrimination by dynamic programming.* Kibnernetika, vol. 4, pages 81–88, Jan Feb 1968. 58

[Waibel 1989] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano et K.J. Lang. *Phoneme recognition using time-delay neural networks.* Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 37, no. 3, pages 328–339, 1989. 152

[Weliwitage 2003] C. Weliwitage, A. Harvey et A. Jennings. *Whole of word recognition methods for cursive script.* WDIC, pages 111–116, 2003. 39, 55

[Werbos 1974] P. Werbos. *Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, Cambridge, MA, 1974. 144

[Wessel 2001] F. Wessel, R. Schluter, K. Macherey et H. Ney. *Confidence measures for large vocabulary continuous speech recognition.* Speech and Audio Processing, IEEE Transactions on, vol. 9, no. 3, pages 288–298, 2001. 56

[Wollmer 2010] M. Wollmer, F. Eyben, A. Graves, B. Schuller et G. Rigoll. *Bidirectional LSTM networks for context-sensitive keyword detection in a cognitive virtual agent framework.* Cognitive Computation, vol. 2, no. 3, pages 180–190, 2010. 52

[Wong 1982] K. Y. Wong, R. G. Casey et F. M. Wahl. *Document analysis system.* IBM J. Res. Dev., vol. 26, pages 647–656, November 1982. 76, 77

[Woodland 2002] P.C. Woodland et D. Povey. *Large scale discriminative training of hidden Markov models for speech recognition.* Computer Speech & Language, vol. 16, no. 1, pages 25–47, 2002. 50

[Xu 2003] Q. Xu, L. Lam et C. Y. Suen. *Automatic Segmentation and Recognition System for Handwritten Dates on Canadian Bank Cheques.* Proceedings of the Seventh International Conference on Document Analysis and Recognition, page 704, 2003. 23

[Xue 2008] G.-R. Xue, D. Xing, Q. Yang et Y. Yu. *Deep classification in large-scale text hierarchies.* Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages 619–626, 2008. 140, 146, 150

[Ye 1999] X. Ye, M. Cheriet, C. Y. Suen et K. Liu. *Extraction of bankcheck items by mathematical morphology.* IJDAR, vol. 2, no. 2-3, pages 53–66, 1999. 24

[You 2003] D. You et G. Kim. *An approach for locating segmentation points of handwritten digit strings using a neural network*. Proceedings of the Seventh International Conference on Document Analysis and Recognition, page 142, 2003. 43

[Young 1989] S.J. Young, N.H. Russell et J.H.S Thornton. *Token Passing : a Simple Conceptual Model for Connected Speech Recognition Systems*. Mathematics of Control, Signals, and Systems (MCSS), pages 1–8, 1989. 84, 90

[Young 1999] S. Young, J. Jansen, J. Odell, D. Ollason et P. Woodland. *The HTK Handbook*, 1999. 90

[Zeppenfeld 1997] T. Zeppenfeld, M. Finke, K. Ries, M. Westphal et A. Waibel. *Recognition of Conversational Telephone Speech using the Janus Speech Engine*. Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 3, pages 1815–1818, 1997. 61

[Zheng 2005] Y. Zheng, H. Li et D. Doermann. *A Parallel-Line Detection Algorithm Based on HMM Decoding*. IEEE Trans. on PAMI, vol. 27, no. 5, pages 777–792, 2005. 76

[Zimmermann 2003] M. Zimmermann et H. Bunke. *Parsing n-best lists of handwritten sentences*. ICDAR, pages 572–576, 2003. 59, 61

[Zimmermann 2004a] M. Zimmermann, R. Bertolami et H. Bunke. *Rejection Strategies for Offline Handwritten Sentence Recognition*. Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02, pages 550–553, 2004. 56

[Zimmermann 2004b] M. Zimmermann et H. Bunke. *Optimizing the Integration of a Statistical Language Model in HMM based Offine Handwritten Text Recognition*. ICPR (2), pages 541–544, 2004. 54, 84

[Zimmermann 2006] M. Zimmermann et J.-C. Chappelier. *Offline Grammar-Based Recognition of Handwritten Sentences*. IEEE Trans. on PAMI, vol. 28, no. 5, pages 818–821, 2006. 59, 61, 84

Extraction d'information dans des documents manuscrits non contraints : application au traitement automatique des courriers entrants manuscrits

Résumé :

Malgré l'entrée récente de notre monde dans l'ère du *tout numérique*, de nombreux documents manuscrits continuent à s'échanger, obligeant nos sociétés et administrations à faire face au traitement de ces masses de documents. Le traitement automatique de ces documents nécessite d'accéder à un échantillon inconnu mais pertinent de leur contenu et implique de prendre en compte trois points essentiels : la segmentation du document en entités comparable à l'information recherchée, la reconnaissance de ces entités recherchées et le rejet des entités non pertinentes. Nous nous démarquons des approches classiques de lecture complète de documents et de détection de mots clés en parallélisant ces trois traitements en une approche d'extraction d'information. Une première contribution réside dans la conception d'un modèle de ligne générique pour l'extraction d'information et l'implémentation d'un système complet à base de modèles de Markov cachés (MMC) construit autour de ce modèle. Le module de reconnaissance cherche, en une seule passe, à discriminer l'information pertinente, caractérisée par un ensemble de requêtes alphabétiques, numériques ou alphanumériques, de l'information non pertinente, caractérisée par un modèle de remplissage. Une seconde contribution réside dans l'amélioration de la discrimination locale des observations des lignes par l'utilisation d'un réseau de neurones profond. Ce dernier permet également d'inférer une représentation de haut niveau des observations et donc d'automatiser le processus d'extraction des caractéristiques. Il en résulte un système complet, générique et industrialisable, répondant à des besoins émergents dans le domaine de la lecture automatique de documents manuscrits : l'extraction d'informations complexes dans des documents non-contraints.

Mots clés : Reconnaissance de l'écriture manuscrite, Extraction d'information, Modèles de Markov Cachés, Réseaux de neurones profonds, détection de séquences alphanumériques

Information extraction in unconstrained handwritten documents : application to automatic processing of handwritten incoming mail

Abstract :

Despite the avenment of our world into the *digital era*, a large amount of handwritten documents continue to be exchanged, forcing our companies and administrations to cope with the processing of masses of documents. Automatic processing of these documents requires access to an unknown but relevant part of their content, and implies taking into account three key points : the document segmentation into relevant entities, their recognition and the rejection of irrelevant entities. Contrary to traditional approaches (full documents reading or keyword detection), all processes are parallelized leading to an information extraction approach. The first contribution of the present work is the design of a generic text line model for information extraction purpose and the implementation of a complete system based on Hidden Markov Models (HMM) constrained by this model. In one pass, the recognition module seeks to discriminate relevant information, characterized by a set of alphabetic, numeric or alphanumeric queries, with the irrelevant information, characterized by a filler model. A second contribution concerns the improvement of the local frame discrimination by using a deep neural network. This allows one to infer high-level representation for the frames and thus automate the feature extraction process. These result is a complete, generic and industrially system, responding to emerging needs in the field of handwritten document automatic reading : the extraction of complex information in unconstrained documents.

Keywords : Offline Handwriting Recognition, Information Extraction, Hidden Markov Models, Deep Neural Networks, Out-Of-KeyWord Vocabulary Model