Un modèle neuro markovien profond pour l'extraction de séquences dans des documents manuscrits

Simon Thomas, Clément Chatelain, Thierry Paquet, Laurent Heutte

Université de Rouen, LITIS EA 4108
BP 12 - 76801 Saint-Etienne du Rouvray, France
prenom.nom@univ-rouen.fr

RÉSUMÉ. *Dans cet article, nous proposons un système d'extraction de mots clés dans des documents manuscrits. Notre approche est basée sur la reconnaissance des lignes de texte à l'aide d'un modèle HMM capable de rejeter les mots n'appartenant pas à un lexique prédéfini. Afin d'être plus discriminant, nous avons remplacé les mélanges de gaussiennes des HMM par un réseau de neurones profond pour calculer les probabilités a posteriori des observations. Nous montrons sur la base de documents des compétitions ICDAR 2009 l'intérêt de notre approche d'extraction d'information par rapport à une stratégie basée sur la reconnaissance intégrale du document. Les résultats montrent également l'apport de l'architecture profonde par rapport aux mélanges de gaussiennes.*

ABSTRACT. *In this paper, we propose a keyword extraction system able to extract keywords in handwritten documents. The base system rely on a HMM line model made of an Out-Of-KeyWord Vocabulary model and keywords model. In order to be more discriminant at the local level (the frame level), the standard gaussian mixture of the HMM are replaced by a deep neural network (DNN) for computing the observations probabilities. Experimentations are carried out on an unconstrained handwritten document database used for the 2009 ICDAR handwriting recognition competitions. The results demonstrate the interest of the keyword extraction system as opposed to the sequential integration strategy of full text recognition prior to the detection of keywords. We also show the benefit from using the deep architecture instead of the gaussian mixtures.*

MOTS-CLÉS : *reconnaissance de l'écriture, keyword spotting, HMM, modèle hors lexique, architectures profondes, modèle hybride.*

KEYWORDS: *offline handwriting recognition, keyword spotting, HMM, out-of-vocabulary model, deep neural network, neuro markovian model.*

DOI:10.3166/DN.16.2.49-68

1. Introduction

Récemment, plusieurs systèmes de détection de mots clés dans des documents manuscrits non contraints ont été développés (Cao, Govindaraju, 2007; Choisy, 2007; Rodríguez-Serrano *et al.*, 2009; Paquet *et al.*, 2012; Chatelain *et al.*, 2006; Frinken *et al.*, 2012; Fischer *et al.*, 2012). De tels systèmes de détection visent à isoler et reconnaître un ensemble réduit de mots clés dans des images de documents. Ces requêtes peuvent être fournies au système de reconnaissance sous la forme d'image (Rath, Manmatha, 2003; Cao, Govindaraju, 2007; Adamek *et al.*, 2007; Rusiñol *et al.*, 2011), ou bien sous la forme de texte au format ascii (Choisy, 2007; Rodríguez-Serrano *et al.*, 2009; Paquet *et al.*, 2012; Frinken *et al.*, 2012; S.Thomas *et al.*, 2010). Cette seconde catégorie d'approches est basée sur un moteur de reconnaissance d'écriture manuscrite classique, et présente donc des performances limitées. En revanche, elle a l'avantage de ne pas être limitée à un unique scripteur comme les systèmes basés sur des requêtes images.

Généralement, la segmentation d'un document en mots et la reconnaissance de ces mots sont effectuées par deux modules distincts de manière séquentielle : segmentation en mots puis reconnaissance des mots. Dans ces systèmes, les erreurs de segmentation en mots sont souvent irréversibles et la règle de décision permettant d'accepter ou de rejeter une hypothèse de mot, qui est le cœur de l'approche, est très dépendante des données et donc difficile à concevoir. Dans le cas idéal, la segmentation en mots et leur reconnaissance doivent être effectuées conjointement.

C'est ce que proposent les auteurs de (Fischer *et al.*, 2010), où un système de détection de mots clés à base de HMM est appliqué à des documents manuscrits avec un modèle de ligne intégrant deux modèles de remplissage gauche et droite et le modèle de mot clé à détecter. Une normalisation et un seuil sont utilisés pour décider si l'image de ligne contient ou non le mot clé en question. Bien que la reconnaissance et la segmentation soient effectuées conjointement, l'approche est limitée par la recherche d'un unique mot clé. De plus, les HMM fournissent des bonnes capacités en modélisation de séquences mais restent peu discriminants au niveau local, c'est-à-dire que l'estimation de la probabilité d'appartenance d'une trame aux modèles de caractères peut être améliorée.

En effet, il a été montré sur de nombreuses applications que l'ajout d'un étage d'apprentissage discriminant au sein d'un modèle HMM pouvait améliorer très significativement les performances. La capacité de discrimination peut être apportée soit par un apprentissage discriminant des HMM (Woodland, Povey, 2002; Do, Artières, 2009), soit en substituant les mélanges de gaussiennes par un classifieur discriminant tel que SVM (Ganapathiraju *et al.*, 2000a; Huang *et al.*, 2006) ou réseau de neurones (Boquera *et al.*, 2011; Graves *et al.*, 2009; Marukatat *et al.*, 2001). Dans cet article, nous proposons de combiner les capacités de modélisation des HMM avec les capacités de discrimination des architectures profondes pour constituer un système de spotting par requête ASCII. Ces architectures ont montré qu'elles étaient capables d'apprendre automatiquement des caractéristiques à partir des données brutes, par exemple

en reconnaissance de chiffres manuscrits (Ciresan *et al.*, 2011; Niu, Suen, 2012), en reconnaissance de gestes (Le *et al.*, 2011), ou encore en classification audio (Lee *et al.*, 2009). En particulier, s'agissant d'images, cette nouvelle architecture permet un apprentissage automatique de caractéristiques à partir des pixels de l'image, évitant ainsi une étape coûteuse et parfois arbitraire de définition de caractéristiques.

Le système générique d'extraction d'information que nous proposons se base sur deux contributions : premièrement, nous présentons un modèle générique statistique d'extraction d'information intégrant un lexique de mots clés à base de HMM, capable de segmenter et de reconnaître des lignes de texte. Deuxièmement, nous combinons le pouvoir génératif des HMM avec un réseau de neurones profonds (DNN) entraîné pour discriminer les configurations locales des pixels d'écriture.

Cet article est organisé comme suit. Dans la section 2, le modèle générique de ligne de texte est présenté. Dans la section 3 la combinaison des DNN et des HMM est décrite. Ce système est d'abord évalué sur une tâche de reconnaissance de mots isolés dans la section 4, puis sur une tâche d'extraction d'information sur une base de courriers entrants francophones (Grosicki, El-Abed, 2009) dans la section 5. Les conclusions et nos futurs travaux sont présentés dans la section 6.

2. Un modèle générique de ligne de texte

La conception d'un système d'extraction d'information requiert la modélisation de l'information pertinente à extraire (mots clés, séquences numériques ...), et de l'information non pertinente (mots vides, mots hors lexique, bruit, par exemple) au sein d'une ligne. Pour obtenir un modèle probabiliste efficace de reconnaissance de l'écriture, toutes les connaissances *a priori* doivent être prises en compte dans le modèle de ligne : modèles de caractères, modèles de langage, proportion d'information pertinente dans les documents, etc. D'un point de vue « extraction d'information », une ligne peut être vue comme une succession de séquences du lexique et d'information non pertinente séparée par des espaces. Par conséquent, le modèle doit mettre en relation ces deux types d'information dans une ligne de texte :

– l'information pertinente formée des séquences d'un lexique spécifique à l'application considérée ;

– l'information non pertinente formée des séquences hors lexique, de la ponctuation, du bruit représenté par un modèle de remplissage (de l'anglais *filler* model).

La figure 2 représente le modèle de ligne manuscrite. Il prend en compte les deux types d'informations qui peuvent être rencontrées au sein d'une ligne de texte. D'une part, le lexique de mots clés est représenté dans la partie haute du schéma. C'est un modèle parallèle de mots-clés caractérisés par leurs séquences de caractères. D'autre part, l'information non pertinente est représentée dans la partie inférieure du schéma. Elle prend en compte n'importe quel mot qui n'appartient pas au lexique (KL pour Keyword Lexicon), nous noterons ces mots hors lexique OKL (*Out of Keyword Lexicon*). Il s'agit d'un modèle ergodique qui prend en compte n'importe quelle séquence

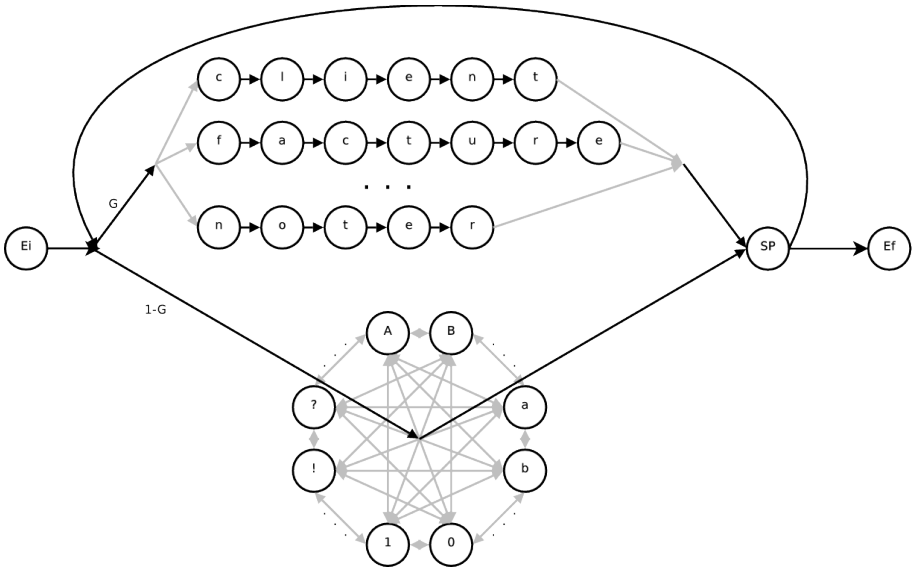


Figure 1. Modèle de ligne global

de caractères alphanumériques, y compris les lettres majuscules et minuscules, les chiffres et les signes de ponctuation. Les mots clés dans KL peuvent apparaître dans le texte avec une fréquence plus basse que les mots dans OKL. C'est pourquoi les deux modèles sont mis en compétition en parallèle et pondérés par le paramètre G qui représente la proportion d'information pertinente pouvant apparaître dans le texte. Enfin, le modèle de la ligne permet la succession de mots KL ou OKL séparés par des espaces (SP) jusqu'à la fin de la ligne.

Nous avons choisi de modéliser ces lignes d'écriture par des modèles de Markov cachés. En effet, les HMM forment un outil probabiliste de référence en ce qui concerne la modélisation de séquences (Rabiner, 1990). Grâce à la méthode d'apprentissage efficace embarqué de l'algorithme de Baum-Welch, les HMM ont été très largement utilisés pour la reconnaissance de mots manuscrits (Kessentini *et al.*, 2010) ou de phrases (Vinciarelli *et al.*, 2004). Cependant, malgré leur bonnes capacités de modélisation, les HMM possèdent un faible pouvoir local discriminant. En effet, les probabilités d'émission $P(o|s_k)$ des observations o lorsque le modèle est dans un état s_k sont estimées en utilisant des mélanges de gaussiennes (GMM). Les GMM étant modélisantes, des gaussiennes appartenant à plusieurs états peuvent se chevaucher dans l'espace des représentations, rendant leur discrimination difficile. Par ailleurs, l'estimation d'un GMM en grande dimension nécessite une grande quantité de données. Pour prendre en compte cette difficulté, on limite généralement la dimension de l'espace des caractéristiques. Pour surmonter ces limitations, de nombreux travaux ont proposé de remplacer avec succès les GMM par un classificateur discriminant (réseau de neurones (Bengio *et al.*, 1995; Knerr, Augustin, 1998; Marukatat *et al.*, 2001),

SVM (Ganapathiraju *et al.*, 2000b)). Dans cet article, nous proposons de combiner la capacité générative et modélisante des HMM avec la capacité discriminante d'un nouveau classifieur de la littérature, à savoir un réseau de neurones profond (DNN).

La mise en place du modèle présenté en figure 2 nécessite l'apprentissage de deux types d'informations : les modèles de caractères (cercles noirs) et les probabilités de transitions (flèches grises et noires). Les probabilités de transitions entre les modèles à l'intérieur du lexique sont à 1. Les probabilités de transition à l'intérieur du modèle de remplissage sont fixées par un modèle de surface du langage. Ce modèle est appris sur la base d'apprentissage. Les modèles de caractères hybrides DNN-HMM sont des modèles HMM gauche-droite dont les probabilités a posteriori $p(s_k|o)$ sont estimées par le DNN. Ils nécessitent une phase d'apprentissage discriminante sur les données localement étiquetées (au niveau trame) en ce qui concerne le DNN alors que les probabilités de transition entre les états peuvent être apprises comme c'est le cas pour des HMM classiques, en utilisant l'algorithme de Baum-Welch. La difficulté majeure dans l'apprentissage d'une architecture profonde est de fournir un ensemble de données d'apprentissage étiquetées au niveau des trames. Nous discuterons ce point dans la section suivante.

3. Combinaison DNN-HMM

Dans cette section, nous présentons l'architecture générale à base de HMM et d'un DNN et nous détaillons ensuite le paramétrage et l'apprentissage de la combinaison.

3.1. Modèle général de la ligne

Considérons la séquence d'observation $O = \{o_1, \ldots, o_g\}$ extraite d'une ligne de texte L, quelle que soit la nature de l'information codée (une séquence de vecteurs de caractéristiques continues, ou tout simplement une séquence d'images en niveau de gris comme décrit plus loin). Notons L_{opt} la séquence optimale d'étiquettes formée de mots du lexique ou de séquences hors lexique, qui est associée à O. Elle peut être calculée grâce à l'équation 1 :

$$L_{opt} = \arg\max_{L}\{P(L|O)\} \quad (1)$$

$$= \arg\max_{L}\left\{\frac{P(O|L)P(L)}{P(O)}\right\} \quad (2)$$

$$= \arg\max_{L}\{P(O|L)P(L)\} \quad (3)$$

où $P(O|L)$ est la probabilité d'une séquence d'observations O étant donnée la séquence de mots L. Même si la probabilité $P(O)$ est difficile à calculer, elle est constante pour une trame donnée et n'influence pas la recherche de la meilleure séquence

d'états. Ce terme est donc supprimé des calculs. Dans l'équation 3 , $P(L)$ est la probabilité a priori d'une séquence de mots. Elle permet de pénaliser les séquences de mots peu probables par l'intermédiaire de modèles de langages au niveau lettre et/ou mots. Nous avons choisi d'utiliser des bigrammes de caractères, appris sur la base d'apprentissage, intégrés au modèle de ligne sous la forme de probabilités de transitions entre les différents modèles.

L_{opt} est déterminé à l'aide du *Time Synchronous Beam Search algorithm* introduit dans (Moore, 2002). Supposons que L contienne N séquences de caractères K_n appartenant à KL, M séquences de caractères W_m appartenant à OKL, et P espaces S_p. G représentant la proportion d'information pertinente dans le texte, $P(O|L)$ s'écrit alors comme suit :

$$P(O|L) = G^N \prod_{n=1}^{N} P(O_n|K_n) \times (1-G)^M \prod_{m=1}^{M} P(O_m|W_m) \times \prod_{p=1}^{P} P(O_p|S_p) \quad (4)$$

et ainsi :

$$P(O|L_{opt}) = \max_{n,m,p} \{P(O_n|K_n) \times P(O_m|W_m) \times P(O_p|S_p)\} \quad (5)$$

Notons que l'utilisation d'un classifieur de type réseau de neurones pour estimer la vraisemblance des observations O nécessite un changement d'échelle de ses sorties. En effet, un réseau de neurones renvoie en sortie des probabilités a posteriori alors que les HMM doivent être alimentés avec des vraisemblances locales. Par conséquent, le théorème de Bayes est appliqué pour obtenir des vraisemblances normalisées à partir des probabiltiés a posteriori. Nous présentons maintenant l'architecture profonde.

3.2. Architecture profonde discriminante

Récemment, une nouvelle famille de réseaux de neurones a été proposée : les architectures profondes. Elle se base sur l'idée (ancienne) que l'entraînement d'un réseau avec de nombreuses couches cachées est un bon moyen de *(i)* gérer les grandes dimensions en entrée, ce qui permet l'utilisation de données d'entrées brutes plutôt que d'extraire un vecteur de caractéristiques, et *(ii)* apprendre des frontières de décision complexes entre les classes. La structure de ces architectures est donc classique (voir figure 2), la nouveauté réside dans la procédure d'entraînement des couches cachées sont entraînées. En effet, en utilisant l'algorithme classique de rétropropagation du gradient sur de nombreuses couches, l'énergie de l'erreur est trop faible pour apprendre convenablement l'ensemble des poids du réseau. Par conséquent, pour l'apprentissage d'un réseau de neurones profond avec Λ couches cachées, un apprentissage en deux étapes a été proposé (Hinton *et al.*, 2006; Bengio *et al.*, 2007) :

- les $\Lambda - 1$ premières couches cachées sont entraînées de manière non supervisée en utilisant des auto-associateurs (AA) ou auto-encodeurs [1], successivement verrouillés et empilés de l'entrée jusqu'à la dernière couche. Ces couches sont appelées *couches de modèle* et l'estimation des poids d'une couche cachée est qualifiée de *préapprentissage*. Une représentation de haut niveau des données peut ainsi être produite automatiquement à partir des pixels en entrée du réseau. Cette étape peut être apparentée à un processus automatique de conception d'un vecteur de caractéristiques.

- ensuite, la dernière couche (appelée *couche de décision*) est ajoutée au réseau. Pour son apprentissage, tous les poids des couches du modèle sont déverrouillés et une rétropropagation classique est effectuée sur l'ensemble du réseau. Cette opération permet d'apprendre la fonction de décision discriminante et d'affiner les paramètres du réseau : il s'agit de l'étape de *fine-tuning*.

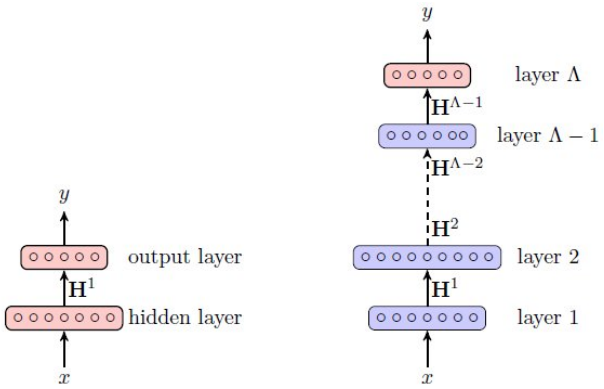


Figure 2. Un MLP classique à une couche cachée et une couche de sortie, et un réseau de neurones profond avec plusieurs couches de modèles en bleu (entraînées de manière non supervisée) et une couche de décision en rouge (entraînée de manière supervisée)

3.2.1. Definition d'un réseau profond

Plus formellement, nous définissons un réseau de neurones profond contenant Λ couches cachées, où chaque couche calcule $\mathbf{H}^{\lambda}$, $\lambda \in \{1, \ldots, \Lambda\}$. La première couche prend en considération les entrées globales du réseau alors que la dernière renvoie les sorties sous la forme de probabilités a posteriori $\mathbf{H}^{\Lambda}$. 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} représente la sortie du i^{ieme} neurone de la couche λ. Cette sortie est déterminée à l'aide de l'équation suivante :

1. À noter qu'une machine de Boltzman restreinte (RBM) peut également être utilisée, on parle alors de *réseaux de croyance profonde* (Hinton *et al.*, 2006)

$$h_i^\lambda = f^\lambda \left(\sum_{j=1}^{N^{\lambda-1}} \left[w_{i,j}^\lambda \,.\, h_j^{\lambda-1} \right] + \mathbf{b}_i^{\lambda-1} \right) \quad \forall i \in \{1, \ldots, N^\lambda\} \tag{6}$$

où $w_{i,j}^\lambda \in \mathbb{R}$ sont les poids entre les couches $\lambda - 1$ et λ, $\mathbf{b}_i^{\lambda-1} \in \mathbb{R}$ les biais (un par neurone), et f^λ une fonction non linéaire sur la somme des poids ; typiquement, la fonction sigmoïde pour les $\Lambda - 1$ premières couches, et la fonction softmax pour la couche de sortie. Ceci permet de sommer les sorties du réseau à 1 et donc d'approximer des probabilités a posteriori. En considérant une entrée fictive $h_0^{\lambda-1} = 1$ à chaque neurone de la couche λ, les biais peuvent être intégrés aux $w_{i,j}^\lambda$, ce qui simplifie les notations :

$$h_i^\lambda = f^\lambda \left(\sum_{j=0}^{N^{\lambda-1}} \left[w_{i,j}^\lambda \,.\, h_j^{\lambda-1} \right] \right) \quad \forall i \in \{1, \ldots, N^\lambda\} \tag{7}$$

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

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

où $\mathbf{W}^\lambda = \{w_{i,j}^\lambda\}, \lambda \in \{1, \ldots, \Lambda\}$ est la matrice des poids à apprendre.

Notons que les couches de modèle et de décision calculent les mêmes quantités, elles ne diffèrent que dans la facon dont elles ont été apprises. Nous décrivons maintenant les deux principales étapes de la procédure d'apprentissage du DNN.

3.2.2. Apprentissage d'une couche de modèle

Le formalisme mathématique des auto-associateurs permet d'apprendre de manière non supervisée des extracteurs de caractéristiques non linéaires. Un auto-associateur apprend une fonction d'encodage e, qui transforme la donnée d'entrée x en une représentation cachée $e(x)$, et une fonction de décodage d, permettant de retrouver la représentation d'entrée $\hat{x} = d(e(x))$. L'encodeur et le décodeur sont constituée de neurones classiques, et $\hat{x}$ est ainsi calculé à l'aide de l'équation de propagation 7. L'estimation $\hat{x}$ est la reconstruction de x à travers la couche considérée. Les paramètres de l'encodeur et du décodeur sont appris pour minimiser une erreur de reconstruction sur la base d'apprentissage, de manière similaire à l'apprentissage d'un perceptron multicouche, en utilisant la rétropropagation.

Une fois apprise, la fonction d'encodage e est empilée en tant que nouvelle couche de modèle dans l'architecture profonde, la fonction décodeur d est oubliée. La nouvelle couche empilée est ensuite utilisée comme entrée pour l'apprentissage d'un nouvel auto-encodeur. Il est possible d'empiler de nombreuses couches (voir figure 3) en fonction de la complexité de l'espace d'entrée. Enfin, une couche de sortie est ajoutée

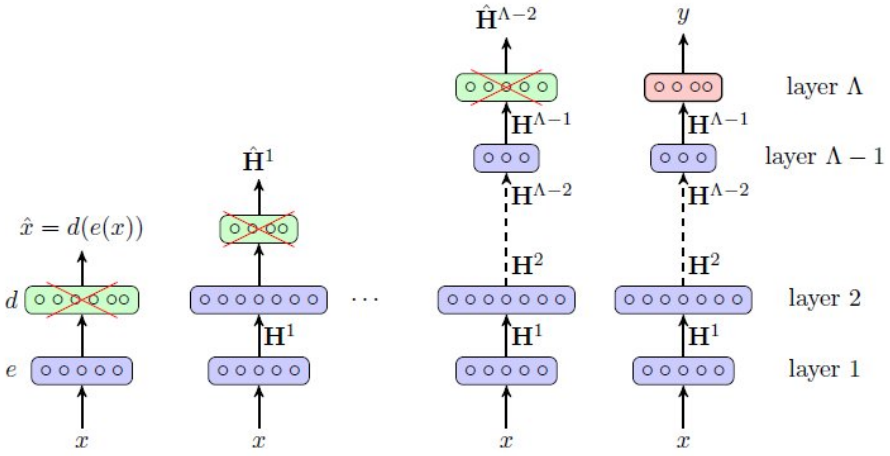


Figure 3. i) Un auto-encodeur (e, d) est appris sur les entrées $\mathbf{x}$. La fonction e caractérisant l'encodeur est empilée pour produire une représentation cachée $\mathbf{H}^1$ des données, et la fonction d n'est pas utilisée. ii) Un nouvel auto-encodeur est alors appris sur les unités cachées $\mathbf{H}^1$ pour apprendre $\hat{\mathbf{H}}^1$. iii) Ce processus est répété pour obtenir une représentation de haut niveau des données. iv) Enfin, un dernier étage est ajouté et une rétropropagation permet d'ajuster les poids du réseau entier (fine-tuning)

après la dernière couche de modèle et un algorithme de rétropropagation est utilisé pour affiner tous les poids du réseau complet en fonction de la sortie y désirée (les états s_k dans le cas de notre combinaison). Afin de maximiser les capacités du réseau, tous les paramètres du réseau profond sont dévérouillés pour que la rétropropagation puisse les atteindre.

3.2.3. Apprentissage de la combinaison DNN-HMM

Le DNN et les HMM doivent être appris pour que les DNN fournissent les vraisemblances des observations au HMM. L'apprentissage d'une combinaison réseau de neurones-HMM peut être réalisé à l'aide d'un apprentissage conjoint (Bengio *et al.*, 1995), ou indépendamment l'un de l'autre. L'apprentissage conjoint affine itérativement les paramètres du DNN pendant l'apprentissage embarqué des HMM, à l'image de l'algorithme EM pour l'apprentissage des GMM. Dans le cas d'un apprentissage indépendant, les HMM intégrant des GMM sont appris, puis utilisés pour réaliser un alignement forcé sur la base d'apprentissage. Cela fournit un étiquetage de la base au niveau trame, permettant l'apprentissage indépendant du réseau de neurones. Dans ce cas, comme des GMM fournissent des vraisemblances aux HMM, la probabilité a posteriori donnée par le réseau de neurones doit être redimensionnée en vraisemblance en utilisant la règle de Bayes. Nous avons choisi cette deuxième solution pour des raisons de simplicité d'implémentation. Dans la prochaine section, nous présen-

tons les combinaisons DNN-HMM apprises pour une tâche de reconnaissance de mots isolés.

4. Experimentations en reconnaissance de mots isolés

Afin de comparer notre approche, plusieurs systèmes ont été appris : un HMM standard intégrant des GMM, une combinaison hybride MLP-HMM, et enfin la combinaison DNN-HMM.

Le jeu de caractéristiques est inspiré de (Al-Hajj *et al.*, 2007), qui a montré son efficacité dans la compétition de reconnaissance de mots manuscrits isolés de la conférence ICDAR 2009 (Kessentini *et al.*, 2010). Il est extrait depuis une fenêtre glissante de 8 pixels de largeur, et contient 26 caractéristiques basées sur les densités de pixels d'écriture et les concavités.

4.1. Apprentissage des HMM

Comme déjà évoqué, un apprentissage indépendant des HMM et des réseaux de neurones a été effectué. Tous les apprentissages ont été réalisés sur une partie de la base de données Rimes (Grosicki, El-Abed, 2009). Nous avons pris en compte $N_c = 71$ caractères : les 26 lettres minuscules, les 26 lettres majuscules, les 10 chiffres, les caractères accentués les plus présents dans la base (é, è, ê, à) et des signes de ponctuations (.,', -, /). Les hyperparamètres des HMM ont été fixés expérimentalement sur une base de validation. Le nombre d'états par modèle de caractère a été fixé à 4, la largeur de la fenêtre glissante à 8 pixels et le recouvrement entre deux fenêtres successives à 5.

4.2. Apprentissage des réseaux de neurones

Les MLP et DNN ont été appris sur les mêmes 26 caractéristiques que pour les GMM, mais aussi sur un ensemble de caractéristiques contextualisées (jeu constitué de trois jeux de caractéristiques extraits depuis la trame courante et les trames suivante et précédente soient $3 \times 26 = 78$ caractéristiques). Un DNN a également été entraîné directement sur les pixels de la trame (de dimensions 54×8 pixels), soient 432 valeurs correspondant à des niveaux de gris, afin d'évaluer sa capacité à apprendre les caractéristiques de haut niveau à partir de données brutes. Les réseaux sont appelés par la suite **DNN-26**, **DNN-78** et **DNN-432**, **MLP-26** et **MLP-78**.

Le dimensionnement des DNN n'est pas une tâche aisée à cause d'un certain nombre d'hyperparamètres à fixer en particulier le nombre de couches cachées et le nombre de neurones par couche cachée. Le nombre de couches cachées a été expérimentalement fixé à 3. Le nombre d'unités cachées pour chaque couche a été fixé pour augmenter ou diminuer le nombre d'entrées par rapport au nombre de sorties, comme indiqué dans le tableau 1. Le nombre de sorties de chaque réseau est le nombre d'états du HMM.

Tableau 1. Nombres d'entrées, nombres d'unités cachées dans les couches cachées h_i, et nombre de sorties pour les 3 DNN et les 2 MLP. Le nombre de sorties des réseaux est égal au nombre d'états des HMM : 4 états 71 caractères = 284*

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

Afin d'entraîner les réseaux, une base de données étiquetées au niveau des observations a été construite en utilisant la base de données de mots isolés en alignant les observations sur la séquence de lettres à l'aide des HMM. $300\ 000$ observations ont ainsi été étiquetées. Le surapprentissage du réseau a été contrôlé en utilisant une base de données séparée. Nous présentons maintenant les résultats en reconnaissance de mots isolés.

4.3. Résultats en reconnaissance de mots isolés

La figure 4 illustre les résultats en reconnaissance de mots sur la base Rimes mots isolés avec un lexique à 100 mots. Les résultats sont donnés pour les combinaisons de HMM avec chacun de nos réseaux de neurones. On peut noter que les résultats obtenus avec un réseau de neurones surpassent ceux obtenus avec les classiques GMM, quel que soit le jeu de caractéristiques pris en entrée et quel que soit le type de réseau. On peut également noter que les DNN donnent de meilleurs résultats que les MLP. Ceci est particulièrement le cas lorsque le DNN est utilisé sur les pixels bruts. Il confirme la capacité du DNN à apprendre des caractéristiques de haut niveau à partir de données brutes et de discriminer efficacement les formes.

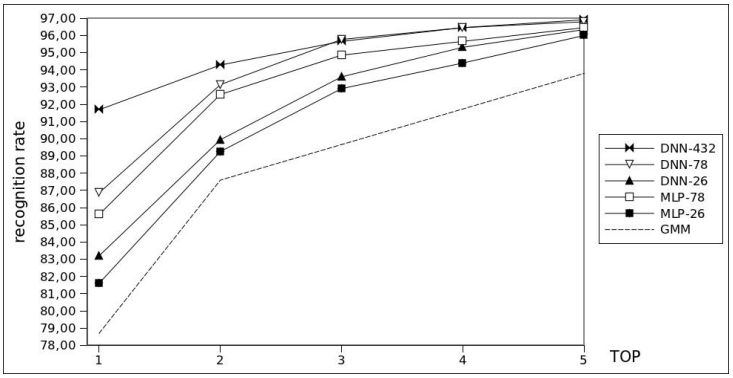


Figure 4. Taux de reconnaissance pour les HMM seuls et pour les combinaisons markoviennes avec les 5 types de réseaux

Nous présentons maintenant les résultats des combinaisons en extraction d'information.

5. Résultats en extraction d'information

Dans cette section, la base de données RIMES utilisée pour nos expérimentations est présentée ainsi que le protocole expérimental que nous avons retenu.

5.1. Base de données

La base de données de RIMES comprend 1 150 courrier entrants francophones (voir figure 5) de différents auteurs (Grosicki, El-Abed, 2009). 950 d'entre eux, contenant environ 36 000 mots, sont utilisés pour l'apprentissage des modèles de caractères et des transitions dans le modèle de ligne. 200 documents sont répartis entre bases de test et de validation.

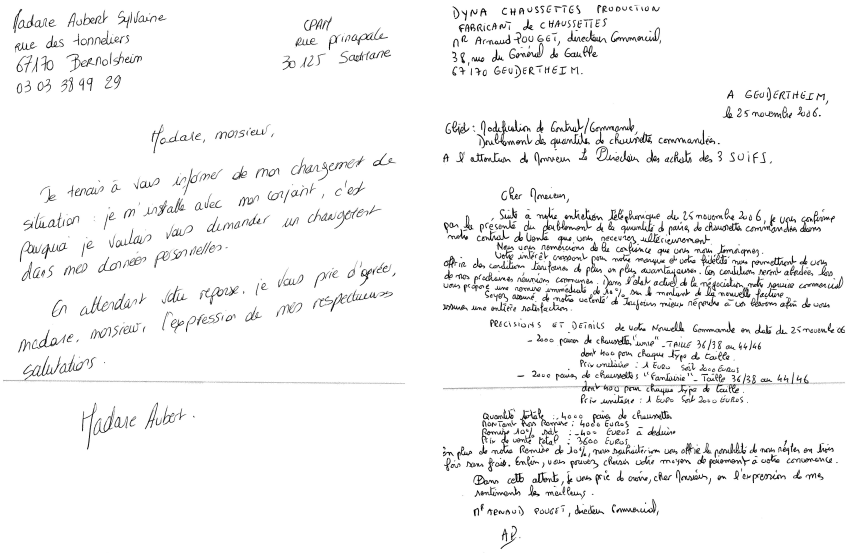


Figure 5. Courriers entrants de la base RIMES

5.2. Prétraitements

Notre modèle de reconnaissance étant orienté « ligne de texte », une étape de segmentation en lignes est nécessaire pour l'appliquer. Des prétraitements sont ensuite effectués afin de diminuer la variabilité de l'écriture.

L'étape de segmentation en lignes est importante et difficile, à cause de l'inclinaison variable des lignes au sein d'un même document, et des lignes adjacentes.

Pour y remédier, nous avons appliqué une méthode robuste basée sur une analyse des composantes connexes. Des lignes sont d'abord constituées à partir des plus grandes composantes, puis les plus petites sont rattachées. Nous renvoyons le lecteur à (Paquet *et al.*, 2012) pour une description plus précise de l'approche.

Une fois les lignes de texte extraites du document, celles ci sont prétraitées pour faciliter l'extraction de caractéristiques. Nous procédons successivement à une correction de l'inclinaison de la ligne et à une correction de l'inclinaison du texte pour normaliser les images de lignes, comme présenté sur les figures 6 et 7 (Kimura *et al.*, 1994). Un lissage est ensuite effectué afin d'éliminer les petites imperfections dues aux corrections.

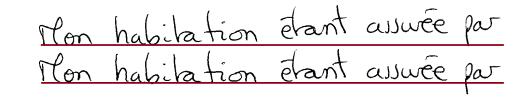


Figure 6. Correction de l'inclinaison de la ligne

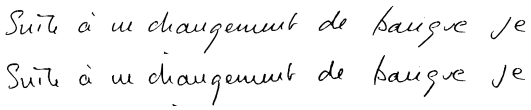


Figure 7. Correction de l'inclinaison du texte

5.3. Protocole expérimental

Afin d'évaluer l'extraction d'information sur une base de D documents, le rappel et la précision doivent être calculés. L'hyperparamètre G (voir figure 2) permet d'obtenir différents points de fonctionnement en favorisant plus ou moins les mots du lexique par rapport aux mots hors lexique dans le modèle. Il permet de décrire la courbe rappel / précision : une valeur de G près de 1 donne un avantage à l'information pertinente au détriment du modèle de remplissage et donc favorisera le rappel au détriment de la précision. A contrario, les valeurs de G proches de 0 se traduiront par l'amélioration de la précision du système. En cas de déploiement d'un tel système, la valeur de G peut être réglée selon les besoins de l'application.

Étant donné un document d, soient $N_{ok}(d)$ le nombre de séquences correctement détectées dans ce document, $N_{fa}(d)$ le nombre de fausses alarmes et $N(d)$ le nombre de séquences à extraire, les moyennes du rappel et de la précision pour une base de données contenant D documents sont calculées comme suit :

$$R = \frac{1}{D}\sum_{d}\frac{N_{ok}(d)}{N(d)} \qquad P = \frac{1}{D}\sum_{d}\frac{N_{ok}(d)}{N_{ok}(d)+N_{fa}(d)}$$

Afin de calculer des résultats pertinents en termes de rappel et la précision, la même quantité d'information à extraire doit être présente dans chaque document pour

une expérience donnée. Pour ce faire, nous avons décidé de générer un lexique d'extraction par document en tirant au hasard 10 séquences dans le vocabulaire du document considéré.

5.4. Résultats en extraction d'information avec un lexique

Nous présentons les résultats pour les 3 différentes tailles de lexique : 10, 100 et 500. Le protocole expérimental est le suivant : un lexique de L sequences est généré en tirant aléatoirement 10 séquences d'au moins 5 caractères de long. Ces 10 séquences sont complétées de $L-10$ séquences apparaissant dans le vocabulaire des autres documents, mais pas dans le document considéré. Les figures 8, 9 et 10 illustrent les résultats en rappel-précision pour des lexiques de respectivement 10, 100 et 500 mots clés.

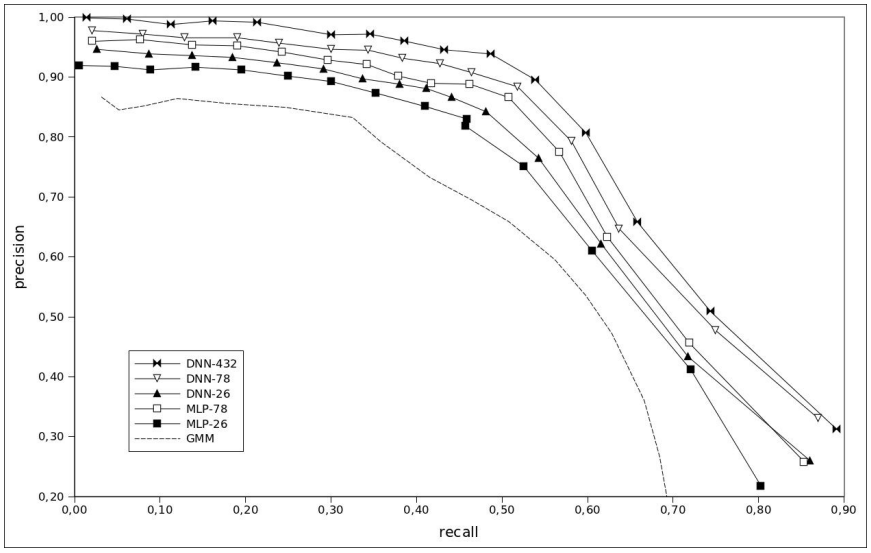


Figure 8. Courbes rappel-précision pour toutes les combinaisons avec un lexique de 10 mots clés

Le premier commentaire est que le comportement du système est le même quelle que soit la taille du lexique : les réseaux de neurones permettent d'atteindre de meilleurs résultats que les GMM. Comme lors de la tâche de reconnaissance de mots isolés, le DNN travaillant sur les pixels présente les meilleurs résultats en extraction d'information. Cette combinaison permet d'atteindre des break-even point [2] de 66 %, 57 % and 43 % pour des lexiques de 10, 100 et 500 mots clés respectivement. Il est à noter qu'un simple MLP appris avec un contexte (observations précédente et suivante) se comporte

2. Le break even point est défini par l'intersection entre la courbe et la première bissectrice dans le plan rappel-précision ; soit le point où le rappel et la précision ont la même valeur.

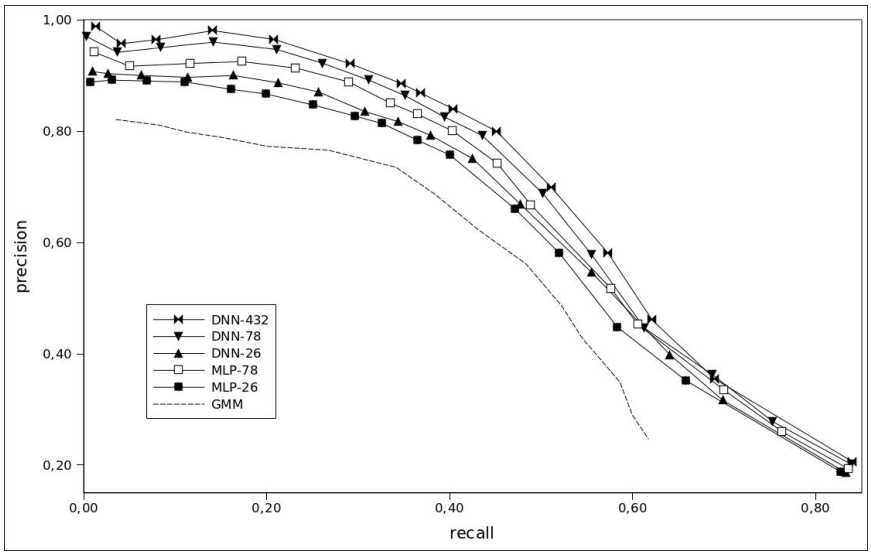


Figure 9. Courbes rappel-précision pour toutes les combinaisons avec un lexique de 100 mots clés

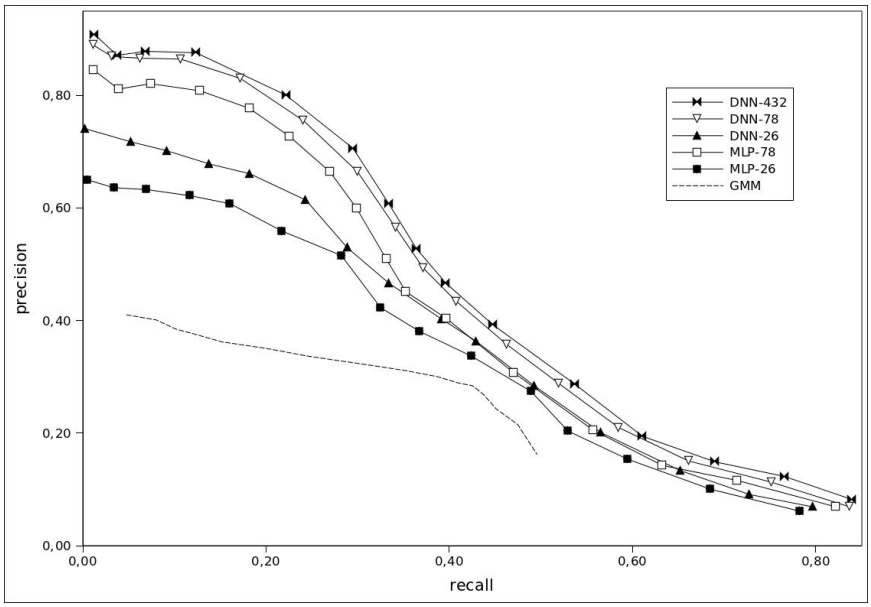


Figure 10. Courbes rappel-précision pour toutes les combinaisons avec un lexique de 500 mots clés

Tableau 2. BEP de plusieurs systèmes pour le spotting d'un seul mot. "nb de mots" représente le nombre de répétition de l'expérience avec des mots différents

Système	base	nb de mots	Break-even-point
Fischer (Fischer *et al.*, 2012)	IAM (929 lignes)	3421	≈ 45 %
	GW (675 lignes)	1067	≈ 60 %
Rodriguez (Rodríguez-Serrano, Perronnin, 2009)	base privée (630 courriers)	10	≈ 75 %
Frinken (Frinken *et al.*, 2012)	IAM (929 lignes)	2807	≈ 72 %
	GW (675 lignes)	1067	≈ 65 %
GMM-HMM	**RIMES (780 lignes)**	**2000**	$\approx$ **69** %
DNN-HMM	**RIMES (780 lignes)**	**2000**	$\approx$ **76** %

également bien. En conclusion, nous pouvons dire que les architectures profondes sont capables de considérer des entrées en grandes dimensions mais aussi d'inférer des caractéristiques de haut niveau plus pertinentes que les jeux de caractéristiques. Le comportement du système lorsque la taille du lexique augmente est un point intéressant : les performances décroissent mais ne s'effondrent pas. La figure 11 illustre le comportement du système avec un lexique de 10 mots.

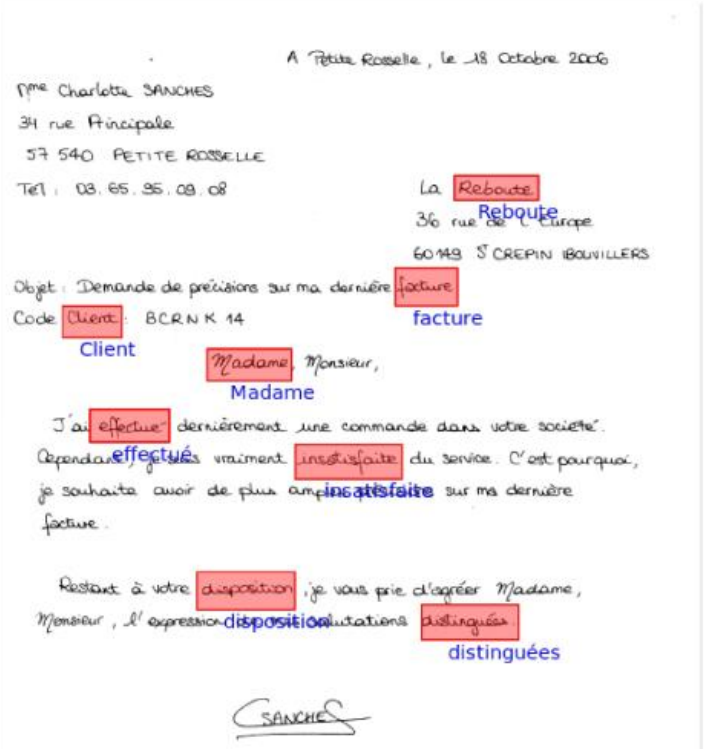
5.5. Résultats en détection avec un mot unique

Nous donnons dans cette section les performances du système en détection avec un mot unique afin de nous comparer avec les autres approches de la littérature [3]. De plus, dans la section précédente nous avons pu voir que les résultats du système dépendaient fortement de la taille du lexique.

Les résultats obtenus confirment la supériorité de l'architecture DNN-HMM sur un HMM classique. Les break-even points de nos différentes architectures sont présentés dans le tableau 2, et comparés à ceux obtenus par les autres approches de la littérature. Il est important de signaler que la comparaison est uniquement à titre informatif car les bases (IAM, GW, RIMES) sont différentes, et les protocoles ne sont pas tout à fait comparables ((Fischer *et al.*, 2012; Frinken *et al.*, 2012) travaillent sur des lignes déjà segmentées, le choix des mots à spotter diffère, etc.).

On remarque que le système proposé semble bien se comporter par rapport aux travaux de référence de la littérature (Fischer *et al.*, 2012; Rodríguez-Serrano, Perronnin, 2009; Frinken *et al.*, 2012), qui sont tous basés sur une modélisation caractères et mots de type HMM, mis à part (Frinken *et al.*, 2012) qui repose sur un système

3. Ce type de tâche est généralement appelé *word spotting* dans la littérature anglophone.



Keyword	found/present
Reboute	1/1
insatisfaite	1/1
effectué	1/1
facture	1/2
client	1/1
Madame	1/2
distinguées	1/1
disposition	1/1
lampadaire	0/0
novembre	0/0

Figure 11. Exemple de résultats d'extraction de mots clés. Le lexique est listé sous la figure, avec deux mots n'appartenant pas au vocabulaire du document ("lampadaire" et "novembre"). Aucune confusion n'est constatée mais deux mots (Madame et facture) ne sont pas détectés. Dans cet exemple, le rappel est de $6/8 = 75\ \%$ *et la précision de* $8/8 = 100\ \%$

à base de réseaux de neurones récurrents BLSTM/CTC (Bidirectionnal Long Short Time Memory/Connectionist Temporal Classification).

6. Conclusion

Dans cet article, nous avons présenté un système d'extraction d'information dans des documents manuscrits non contraints. Il se base sur une modélisation globale de la ligne de texte à l'aide de HMM permettant de considérer deux types d'information : l'information à extraire et l'information non pertinente à rejeter, et sur une classification locale efficace des observations à l'aide d'un réseau de neurones profond. Les résultats sur la base de courriers manuscrits RIMES illustre le pouvoir de ce schéma de combinaison. Les meilleurs résultats ont été obtenus avec l'architecture profonde travaillant directement au niveau pixel, ce qui met en avant les capacités d'inférence de représentations de haut niveau des réseaux profonds sur les observations.

Bibliographie

Adamek T., Connor N., Smeaton A. (2007). Word matching using single closed contours for indexing handwritten historical documents. *IJDAR*, vol. 9, n° 2, p. 153–165.

Al-Hajj R., Mokbel C., Likforman-Sulem L. (2007). Combination of hmm-based classifiers for the recognition of arabic handwritten words. *ICDAR* , p. 959–963.

Bengio Y., Lamblin P., Popovici D., Larochelle H. (2007). Greedy layer-wise training of deep networks. *NIPS* , p. 153–160.

Bengio Y., LeCun Y., Nohl C., Burges C. (1995). Lerec: a nn/hmm hybrid for on-line handwriting recognition. *Neural Comput.*, vol. 7, p. 1289–1303.

Boquera S. E., Bleda M. C., J.Gorbe-Moya, Zamora-Martínez F. (2011). Improving offline handwritten text recognition with hybrid hmm/ann models. *IEEE Trans. PAMI*, vol. 33, n° 4, p. 767-779.

Cao H., Govindaraju V. (2007). Template-free word spotting in low-quality manuscripts. *ICDAR* , p. 392–396.

Chatelain C., Heutte L., Paquet T. (2006). Discrimination between digits and outliers in handwritten documents applied to the extraction of numerical fields. In *Iwfhr, la baule, france*, p. 475-480.

Choisy C. (2007). Dynamic handwritten keyword spotting based on the nshp-hmm. *Proceedings of the Ninth ICDAR* , vol. 1, p. 242–246.

Ciresan D., Meier U., Gambardella L., Schmidhuber J. (2011). Convolutional neural network committees for handwritten character classification. *ICDAR* , p. 1135-1139.

Do T., Artières T. (2009). Maximum margin training of gaussian hmms for handwriting recognition. *ICDAR* , p. 976-980.

Fischer A., Keller A., Frinken V., Bunke H. (2010). Hmm-based word spotting in handwritten documents using subword models. *ICPR* , p. 3416–3419.

Fischer A., Keller A., Frinken V., Bunke H. (2012). Lexicon-free handwritten word spotting using character hmms. *Pattern Recognition Letters*, vol. 33, n° 7, p. 934 - 942.

Frinken V., Fischer A., Manmatha R., Bunke H. (2012). A novel word spotting method based on recurrent neural networks. *IEEE Trans. PAMI*, vol. 34, n° 2, p. 211-224.

Ganapathiraju A., Hamaker J., Picone J. (2000a). Hybrid svm/hmm architectures for speech recognition. *Interspeech* , p. 504-507.

Ganapathiraju A., Hamaker J., Picone J. (2000b). Hybrid svm/hmm architectures for speech recognition. *Speech Transcription Workshop* , p. 504–507.

Graves A., Liwicki M., Fernandez S., Bertolami R., Bunke H., Schmidhuber J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. PAMI*, vol. 31, n° 5, p. 855-868.

Grosicki E., El-Abed H. (2009). Icdar 2009 handwriting recognition competition. *ICDAR* , p. 1398–1402.

Hinton G. E., Osindero S., Teh Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, vol. 18, n° 7, p. 1527–1554.

Huang B., Du C. J., Zhang Y. B., Kechadi M. T. (2006). A hybrid hmm-svm method for online handwriting symbol recognition. *International Conference on Intelligent Systems Design and Applications* , p. 887-891.

Kessentini Y., Paquet T., Benhamadou A. (2010). Off-line handwritten word recognition using multi-stream hidden markov models. *Pattern Recognition Letters*, vol. 31, p. 60-70.

Kimura F., Tsuruoka S., Miyake Y., Shridhar M. (1994). A lexicon directed algorithm for recognition of unconstrained handwritten words. *IEICE Trans. on Information & Syst.*, vol. E77-D, n° 7, p. 785–793.

Knerr S., Augustin E. (1998). A neural network-hidden markov model hybrid for cursive word recognition. *ICPR* , vol. 2, p. 1518–1520.

Le Q., Zou W., Yeung S., Ng A. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. *CVPR* , p. 3361-3368.

Lee H., Pham P., Y.Largman, Ng A. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *NIPS* , p. 1096-1104.

Marukatat S., Artières T., Gallinari P., Dorizzi B. (2001). Sentence recognition through hybrid neuro-markovian modeling. *ICDAR* , p. 731-735.

Moore D. (2002). TODE: A Decoder for Continuous Speech Recognition. *IDIAP Research Report 02-09*.

Niu X., Suen C. (2012). A novel hybrid cnn-svm classifier for recognizing handwritten digits. *Pattern Recognition*, vol. 45, n° 4, p. 1318 - 1325.

Paquet T., Heutte L., Koch G., Chatelain C. (2012). A categorization system for handwritten documents. *International Journal on Document Analysis and Recognition*, vol. 15, n° 4, p. 315-330.

Rabiner L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition* , p. 267–296.

Rath T., Manmatha R. (2003). Features for word spotting in historical manuscripts. *ICDAR* , p. 218–222.

Rodríguez-Serrano J. A., Perronnin F. (2009). Handwritten word-spotting using hidden markov models and universal vocabularies. *Pattern Recognition*, vol. 42, n° 9, p. 2106–2116.

Rodríguez-Serrano J. A., Perronnin F., Llados J. (2009). A similarity measure between vector sequences with application to handwritten word image retrieval. *CVPR* , p. 1722-1729.

Rusiñol M., Aldavert D., Toledo R., Lladós J. (2011). Browsing heterogeneous document collections by a segmentation-free word spotting method. *ICDAR* , p. 63-67.

S.Thomas, C.Chatelain, L.Heutte, T.Paquet. (2010). An information extraction model for unconstrained handwritten documents. *ICPR , Istanbul, Turkey*, p. 4.

Vinciarelli A., Bengio S., Bunke H. (2004). Offline recognition of unconstrained handwritten texts using hmms and statistical langage models. *IEEE Trans. on PAMI*, vol. 26, n° 6, p. 709–720.

Woodland P., Povey D. (2002). Large scale discriminative training of hidden markov models for speech recognition. *Computer Speech and Language*, vol. 16, n° 1, p. 25 - 47.