Détection de mots clés et d'expressions régulières en vue de la reconnaissance d'entités nommées dans des documents manuscrits

Gautier Bideault

Normandie Université

Pour obtenir le grade de Docteur

opéré par l'Université de Rouen

Spécialité Informatique

Détection de mots clés et d'expressions régulières en vue de la reconnaissance d'entités nommées dans des documents manuscrits

Présentée et soutenue publiquement par
Gautier BIDEAULT

Thèse soutenue publiquement le 1er décembre 2015 devant le jury composé de		
Mme Nicole VINCENT	Université Paris Descartes	Rapporteur
Mme Laurence LIKFORMAN-SULEM	Université Télécom ParisTech	Rapporteur
M Jean Marc OGIER	Université de La Rochelle	Examinateur
M Vincent POULAIN D'ANCY	Société ITESOFT	Examinateur
M Thierry PAQUET	Université de Rouen	Directeur de thèse
M Clément CHATELAIN	INSA de Rouen	Encadrant

Thèse dirigée par Thierry PAQUET et Clément CHATELAIN, laboratoire LITIS

UNIVERSITÉ DE ROUEN — SPMII ECOLE DOCTORALE — iTESOFT — litis

Remerciements

Je commencerai par remercier mon directeur de recherche Thierry Paquet pour son temps, ses conseils et sa patience (surtout pendant la rédaction !). J'ai pu ainsi profiter de ton expérience et de tes idées pour me sortir des impasses.

Je tiens également à exprimer ma gratitude à Clément Chatelain qui m'a donné goût au traitement du signal au cours de ma formation INSA et proposé ce sujet de thèse. J'ai pris beaucoup de plaisir à travailler avec toi, je me rappellerai longtemps de nos discussions sur les présentations ("Poses le problème"). Travailler avec vous deux aura été une expérience aussi enrichissante que plaisante.

Je tiens également à remercier Laurence Likforman et Nicole Vincent qui ont accepté d'étudier mon manuscrit en tant que rapporteurs. Les discussions que nous avons pu avoir à propos de mes travaux ont laissé entrevoir de nombreuses pistes à explorer.

Je remercie également Jean Marc Ogier pour m'avoir fait l'honneur d'être le président de mon jury de thèse.

Je tiens aussi à remercier Vincent Poulain D'andecy et Tayeb Benkhelfallah pour la richesse de nos échanges et leur compréhension quant à l'aspect exploratoire de cette thèse.

Je remercie aussi l'ensemble de mes collègues du LITIS et tout particulièrement Luc Mioulet et Philippine Barlas amis et collègues avec qui nous avons travaillé en collaboration sur l'ensemble de cette thèse et durant ma scholarité INSA pour leur bonne humeur, leur humour et leur super taille crayon !

Merci également à Florien Yger pour l'ensemble de nos échanges professionnels ou non (surtout en afterwork) en espérant que nous pourrons bientôt bosser ensemble (avec ta barbe) sur Paris.

Merci aussi à David Hébert pour son temps et sa patience durant mes débuts avec les CRF.

Je n'oublierai pas non plus les matchs de tennis avec Sebastien Adam, j'arriverai un jour à te prendre un set !

Je remercie également l'ensemble des enseignants INSA notamment Gilles Gasso, Alain Rakotomamonjy et Romain Hérault qui m'ont donné goût au machine learning par leur approche très pédagogique illustrée d'exemples plus intéressants les uns que les autres.

Merci à ma famille et mes amis qui m'ont soutenu tout au long de ces travaux de thèse : Dominique, Corinne, Thibaut, Christine, Michel, Stéphane, Yohan, Samuel, Stéphanie, Romain, Julia, Arthur, Anne, Adam, Vincent, Anne-Marie, Adrien, ...

Mes derniers remerciements vont vers Delphine qui m'a supporté et soutenu tout au long de cette thèse malgré les rush nocturnes et les week end très studieux !

J'en oublie certainement encore et je m'en excuse. Encore un grand merci à tous pour m'avoir conduit à ce jour mémorable.

UNIVERSITÉ DE ROUEN

Résumé

Université de Rouen
Équipe Document et Apprentissage

par

Les travaux présentés dans cette thèse concernent la détection de mots clés et d'expressions régulières en vue de la reconnaissance d'entités nommées dans des documents manuscrits non contraints. Les entités nommées telles que les noms et prénoms, les noms de compagnies ou les montants numériques constituent généralement une majeure partie de l'information d'un document. D'un point de vue industriel, la détection et la reconnaissance de ces entités nommées permettrait donc d'avoir une compréhension profonde du document traité.

Les entités nommées sont des informations très variables, dont la définition dépend fortement du problème considéré. Les entités nommées liées à une problématique de tri du courier (nom et prénom de personne, type et nom de voie, nom de ville, code postal) sont par exemple différentes de celles liées à un problème de catégorisation de documents (lexique de mots clefs liés au domaine). Cette variabilité rend la détection des entitées nommées difficile. Lorsque l'on considère des images de documents, la détection et la reconnaissance des entités nommées est également confrontée à la problématique de reconnaissance du texte, perturbée par la variablité de l'écriture (notamment sur les documents manuscrits), ainsi qu'au bruit lié à la numérisation.

La première contribution de cette thèse est un système de reconnaissance de mots isolés basé sur un Champs Aléatoire Conditionnel (CAC), ce qui d'après notre bibliographie n'a pas encore été proposé. La deuxième contribution est un système générique de détection de mots clés et d'expressions régulières permettant de détecter n'importe quelle séquence dans une ligne de texte. Une structure se démarque des autres par ses performances et sa capacité à traiter des requêtes très difficiles, le BLSTM-CTC. Cette dernière semble être la clé de la résolution du problème initial.

Sommaire

Introduction Générale

Avec le développement exponentiel du nombre de documents physiques générés au sein de l'industrie ou dans l'administration et le besoin de gagner en efficacité, de nombreux chercheurs ont travaillé sur le développement de modèles et de méthodes permettant de traiter automatiquement de grands volumes de données.

Tous ces travaux ont permis l'apparition de systèmes de traitement d'informations spécifiques comme la lecture automatique de formulaires, de codes postaux ou de chèques bancaires.

Au delà de ces applications spécifiques, le traitement automatique de documents manuscrits reste un problème difficile et ouvert. En effet, malgré l'évolution des méthodes statistiques et de l'informatique en général, à ce jour, aucun système n'est capable de reconnaître parfaitement un document manuscrit complet non-contraint. Ceci s'explique par la variabilité de l'écriture manuscrite propre au scripteur à laquelle on peut ajouter les problématiques liées à la dégradation du document étudié comme c'est le cas dans le traitement de documents historiques.

En revanche, il est possible d'envisager la mise en place de systèmes traitant des sous-problèmes plus réalistes et correspondant à des besoins industriels spécifiques. C'est le cas de ces travaux qui proposent d'aborder la détection et la reconnaissance d'entités nommées (EN) manuscrites dans des documents non-contraints. Cette étude a été réalisée dans le cadre du projet DOD (Document on Demand) pour le compte de la société ITESOFT.

Nous proposons dans cette étude de décomposer la tâche de détection et reconnaissance des entités nommées selon deux étapes successives. Une première étape permet la détection des zones amorces précurseures de la présence d'entités nommées telles que la présence de mots clés, de sigles, de séquences numériques, etc. Une seconde étape sera dédiée à la reconnaissance des entités nommées ainsi localisées à l'issue de la première étape. Dans ce document, les développements algorithmiques et les expériences concernent la première étape de ce système dédié à la détection de zones amorces

précurseures de la présence d'entités nommées dans le document à traiter. L'ensemble de nos systèmes est évalué sur les bases publiques Rimes mots isolés 2009 [1] et lignes 2011 [2] afin de pouvoir être comparé à d'autres approches de la littérature.

Le premier chapitre de cette thèse retrace l'historique de la détection et de la reconnaissance d'entités nommées. Nous commençons par définir les entités nommées, nous montrons qu'elles sont complexes, variables et propres à chaque problème à résoudre. Nous mettons donc en avant l'importance de bien les définir, puis nous évoquons le cycle des conférences MUC qui a permis de construire la tâche d'extraction d'information comme on la connait aujourd'hui. Nous axons notre étude sur le traitement des entités nommées. Ce chapitre se termine sur un passage en revue de plusieurs systèmes mis en place à la suite du cycle de conférences MUC et sur l'impact sur les performances du niveau de bruit dans les documents à traiter.

La détection et la reconnaissance des entités nommées dans un document ASCII est déjà à lui seul un problème complexe, l'ajout de la variabilité et du bruit de l'écriture manuscrite nécessite l'emploi de modèles statitistiques très performants. C'est pourquoi le chapître 2 présente différents modèles graphiques séquentiels, génératifs et discriminants. Nous passons en revue des méthodes comme les Modèles de Markov Cachés (MMC), les Champs Aléatoires Conditionnels (CAC) ainsi que des structures hybrides comme des réseaux neuro-markoviens ou le BLSTM-CTC (Bidirectionnal Long Short Term Memories - Connectionist Temporal Classification).

Le chapitre 3 présente notre première contribution, une structure hybride CAC-MMC pour la reconnaissance de mots isolés dirigée par un lexique. Ce système est opposé à 4 autres systèmes : MMC standard, Perceptron Multi-Couche-MMC, Réseaux de neurones recurrents-MMC, BLSTM-CTC-MMC. Le premier étage discriminant traite les informations de bas niveau, alors que le deuxième étage modélisant intègre des informations de haut niveau (lexique, modèle de langage). Les performances sont évaluées sur la base Rimes mots isolés 2009 [1].

Le chapitre 4 est consacré à notre deuxième contribution, une structure hybride pour la détection de mots clés et d'expressions régulières dans des documents manuscrits non-contraints. Ces structures s'appuient sur les précédentes décrites dans le chapitre 3 en remplaçant le modèle de mot par un modèle de ligne, permettant ainsi aux systèmes de traiter des lignes de textes complètes. Le traitement des expressions régulières est un problème complexe car on ne peut effectuer de reconnaissance diriger par un lexique. Les performances sont évaluées sur la base Rimes lignes 2011 [2].

Nous observons qu'une architecture se démarque des autres, le BLSTM-CTC-MMC qui semblent pouvoir traiter des requêtes très peu contraintes comme des séquences

de majuscules ou de chiffres non contraintes. Cela nous permet donc d'apporter des premiers éléments de réponse au traitement des entités nommées dans des documents manuscrits.

Chapitre 1

Historique de l'extraction d'entités nommées

1.1 Introduction

La problématique d'extraction d'entités nommées (EN) est un thème de recherche du Traitement Automatique de la Langue (TAL). Il consiste par exemple à extraire des blocs adresse, de noms/prénoms, de numéros de références, etc. Ces entités permettent de classer en catégories (numéros de téléphone, entreprises pharmaceutiques, adhérents, etc.) les noms propres, les noms d'entreprises, les nombres et chiffres divers, elles constituent pour la plupart des informations essentielles à la bonne compréhension d'un document textuel. C'est pourquoi, leur extraction a motivé de nombreux travaux au cours des dernières décennies. Les applications sont multiples (étiquetage des mots d'une phrase, classification de documents, etc.) et avec la nécessité de gagner toujours plus de temps et d'automatiser un maximum d'opérations, l'industrie finance de nombreux projets nécessitant la résolution de problèmes difficiles, comme la classification de factures, le traitement automatique de courriers, etc.

La détection et la reconnaissance d'entités nommées s'inscrit dans le cadre de ces recherches. Cela reste un problème épineux dû à la variabilité des informations à détecter (noms, prénoms, villes, compagnies, etc) et aux ambiguïtés de certaines classes ("Orange" est une entreprise et un fruit). Des solutions à ce problème basées sur des règles de détection (présence de majuscules, suite de majuscules précédé d'un mot clé comme "Monsieur", "Général" ont été mises en place, les meilleurs systèmes actuels de détection d'EN (Entités Nommées) ont des performances très proches de celles d'un être humain sur des documents électroniques (texte ASCII).

Cependant, les solutions actuelles ne peuvent être transposées directement aux transcriptions issues de la reconnaissance de données bruitées comme les documents imprimés ou manuscrits. En effet, dans ce cas il est nécessaire d'obtenir une transcription ASCII du document. Cette transcription étant source d'erreurs et d'ambiguïtés, les méthodes classiques de recherche d'EN n'obtiennent pas les mêmes performances que sur les documents électroniques. Des travaux ont été effectués en traitement automatique du langage et sur des documents imprimés. Nous l'évoquons plus tard dans ce chapitre. La littérature concernant la recherche d'EN dans des documents manuscrits est très pauvre. Avant de chercher à reconnaître ces informations il faut les définir. Notre définition est basée sur les travaux de Maud Erhman [3] sur la définition des EN et sur l'extraction d'information en générale avec les travaux de Guillaume Koch [4]. Nous commençons par donner une définition des entités nommées, puis nous évoquons des difficultés rencontrées lors de l'extraction de ce type d'information. Finalement, nous passons en revue les différents systèmes utilisés pour détecter et reconnaître ces EN sur des documents électroniques et sur des transcriptions de paroles [5].

1.2 Définition

Une entité nommée est une appellation générique pour la catégorisation d'un certain nombre d'objets textuels rencontrés dans un document [6]. Les entités nommées incluent traditionnellement quatre grandes classes [6] :

- Les noms
- Les quantités
- Les dates
- Les durées

Cependant, certains problèmes nécessitent l'ajout d'autres classes ou de sous classes. C'est le cas par exemple si l'on veut classifier des entités nommées de type noms de personnes, adresse, etc (cf Figure 1.1).

Le nombre de classes et de sous classes dépendra de la difficulté du problème.

La détection de ces EN permet d'extraire l'information essentielle du texte à étudier, d'effectuer une classification, une détection de mots clés, ... La reconnaissance de ces informations peut-être mise en œuvre dans tout type de texte. Si historiquement ce processus a été appliqué sur des corpus journalistiques traitant de sujets géopolitiques au cours des compagnes d'évalutation de MUC-4 et MUC-5 (cf section 1.2.1.1), il est aujourd'hui également appliqué sur d'autres types de corpus portant sur des domaines plus spécifiques. La biologie et la médecine sont très demandeurs de ce genre d'analyse. La reconnaissance des noms de gènes, de protéines ou de maladies correspondant à

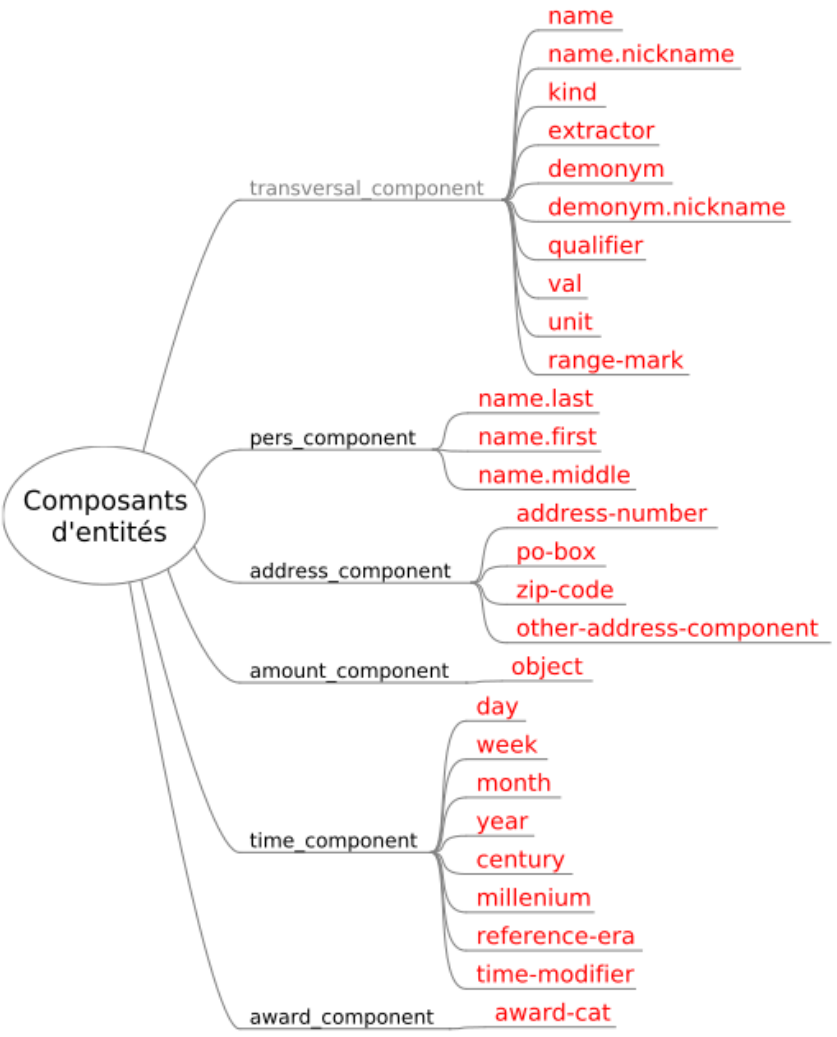


FIGURE 1.1: Exemples de classes et de sous classes d'EN [6]

certains traitements nécessitent des méthodes statistiques pour être traités efficacement vu l'importante quantité d'information produite par ces communautés.

Les entités nommées représentent donc une information importante à identifier dans le document à traiter. Elles doivent être composées d'autant de types et de sous-types nécessaires pour couvrir l'ensemble du problème posé. Il est donc impératif de prendre le temps de bien les définir dès le départ d'un projet.

1.2.1 Historique des campagnes sur les entités nommées

La détection et la reconnaissance des entités nommées est apparue avec l'évolution de la tâche d'extraction d'information. L'extraction d'information consiste à détecter dans les documents des groupes de mots ou des mots clés qui apportent une information précise

que l'on recherche automatiquement. Pour cela, il existe de types de méthodes : les méthodes basées sur un ensemble de règles syntaxiques et celles basées sur un apprentissage statistique. Les règles syntaxiques sont définies par des experts-linguistes suite à la lecture du corpus. Généralement, il s'agit de règles contextuelles indiquant ou non la présence de l'élément à trouver (présence de majuscule, présence d'un mot particuliers avant ou après, etc.). L'inconvénient majeur de ce type de méthode est que la mise en place du jeux de règles est fastidieuse. Les méthodes à apprentissage statistique apprendront à sélectionner l'information pertinente parmi un vecteur de caractéristiques extrait du texte. Dans ce cas, l'humain n'a pas beaucoup d'impact sur l'agencement interne du système et la base de données d'apprentissage doit-être proche de celle de test. Quelle que soit la méthode utilisée, le but est ici d'identifier des occurences d'éléments particuliers (champs), d'en extraire les arguments impliqués pour ensuite en donner une représentation structurée. Ces éléments peuvent être regroupés par paire, triplet, etc. de manière à conserver les relations existantes entre ces derniers. On parle alors d'extraction multi-champs. L'analyse s'effectue au niveau local et seule une partie du texte est considérée. Cette tâche doit absolument être automatique car le but est d'appliquer ce traitement sur un très grand nombre de documents et d'avoir un résultat rapidement. Suivant la nature des documents, différentes techniques d'extraction sont utilisées. Nous allons passer en revue un certain nombre de projets et de conférences qui ont amené à la mise en place des systèmes d'extraction d'entités nommées actuels.

1.2.1.1 Cycle des conférences MUCs

La série de conférences MUC (*Message Understanding Conferences*) a permis à l'extraction d'informations de gagner en maturité en faisant travailler ensemble des scientifiques du monde entier. Ce cycle de conférences, organisé en Californie par le NOSC (*Naval Ocean Systems Center*) et financé par la DARPA (*Defense Advanced Research Projects Agency*), s'est déroulé de 1987 à 1998, motivant de nombreuses équipes de recherche pendant plus d'une décennie. L'objectif de ces conférences était à l'origine d'encourager la recherche autour de la compréhension automatique de messages militaires. Ces conférences sont en réalité des campagnes d'évaluation au cours desquelles les participants se voient remettre un échantillon de textes (corpus d'entraînement) ainsi que des instructions précises sur les informations à extraire automatiquement (scenarii) quelques mois avant les conférences (entre un et six mois). Les organisateurs ajoutent un corpus supplémentaire, appelé corpus de test sur lequel tous les participants doivent appliquer leur système afin que ces derniers soient sur un pied d'égalité pour pouvoir être comparés. Les résultats sont ensuite évalués et présentés lors de la conférence finale, à laquelle seuls les participants à l'évaluation ont le droit d'assister.

Il est possible de distinguer trois cycles au sein des 7 conférences qui se sont succédés, chacune de ces conférences ayant apporté des problématiques de plus en plus difficiles à traiter (taille et nature des corpus à analyser, degré d'aboutissement du processus d'évaluation, etc.).

Les deux première conférences MUC-1 (1987) et MUC2 (1989) peuvent être considérées comme un premier cycle exploratoire. Les corpus sont composés de messages de la *Navy* de style télégraphique. Au cours de la première conférence de 1987, aucune instruction précise quant aux données à extraire n'avait été fournies aux participants. Il faut attendre la seconde conférence de 1989 pour voir apparaître un premier formulaire répertoriant les champs pré-remplis correspondant à l'information à extraire (template). A l'époque les deux indicateurs de performance choisis pour évaluer les systèmes de recherche d'informations sont :

1. Le Rappel qui détermine le taux d'informations correctes extraites par rapport au nombre total d'informations pertinentes disponibles dans le texte analysé.

$$Rappel = \frac{NbInformationsCorrectes}{NbTotalAExtraire}$$

   où $NbTotalAExtraire$ correspond au nombre total d'informations attendues (nombre total d'information pertinentes dans le texte à étudier) et $NbInformationsCorrectes$ au nombre d'informations correctes extraites par le système.

2. La Précision qui calcule le rapport entre le nombre d'informations extraites correctes et le nombre total d'informations collectées par le système (correctes et incorrectes).

$$Precision = \frac{NbInformationsCorrectes}{NbInformationsRapportees}$$

   où $NbInformationsRapportees$ correspond au nombre d'informations correctes et incorrectes extraites par le système.

D'après notre bibliographie, il semble qu'aucun article détaillant les systèmes utilisés à l'époque ne soit accessible. Cependant comme l'atteste les travaux de *Chinchor et al.* [7] sur les systèmes de MUC-3, ils semblent que les systèmes proposés lors des deux premières conférences soient tous composés d'un ensemble de régles linguistiques.

Les conférences MUC-3 (1991), MUC-4 (1992) et MUC-5 (1993) constituent le second cycle. Au cours de celui-ci, la tâche d'extraction d'information est devenue plus précise mais également plus complexe.

Les participants de MUC-3 et MUC-4 ont travaillé sur des corpus de nature journalistique, traitant d'événements ou d'actes terroristes en Amérique Centrale et du Sud. Les

templates de ce deuxième cycle comportent alors jusqu'à 24 champs (cf Figure 1.2). La tâche à effectuer sur ce corpus était d'extraire de ces dépêches le type d'incident, le lieu, la date, les exécutants, la cible ainsi que les effets sur cette dernière. Cette augmentation conséquente du nombre d'informations différentes à reconnaître va amener les chercheurs à revoir leur propre approche du problème. Si les textes sont mieux écris qu'au cours des conférences précédentes (rédaction plus soignée et plus homogène) leur analyse est elle plus complexe. Ils sont également plus longs et l'information à extraire est plus difficile à identifier car plus ambiguë. Même si beaucoup de systèmes sont encore basés sur des règles synthaxique et sémantiques [7], l'apport de bases d'apprentissage étiquetées plus conséquentes permet l'apparition des premiers systèmes à base d'automates [8].

MUC-4 introduit également un nouvel indicateur de performance : la F-mesure. Elle combine la précision et le rappel et rend ainsi plus facile la comparaison des systèmes entre eux.

$$F_{mesure} = \frac{2 * (precision * rappel)}{(precision + rappel)}$$

MUC-5 (1993) suit de près ces deux conférences et gagne encore en complexité. Au cours de cette dernière conférence du deuxième cycle, deux domaines sont proposés : technologique (microélectronique) et commercial (ventes d'entreprises), pour deux langues : anglais et japonais. Cette diversification correspond à une volonté d'améliorer la portabilité des systèmes en vue d'applications industrielles. Néanmoins, les temps de développement de ces derniers sont extrêmement longs (environ 6 mois) et les niveaux de performances ne dépassent pas ceux des systèmes précédents. Vue par certains comme un échec, cette dernière conférence de 1993 met en évidence des problématiques encore étudiées actuellement telles que le traitement multi-domaines et multi-lingues. Ces problématiques obligent les participants à repenser leurs architectures pour les rendre plus génériques et plus portables. C'est ainsi que les systèmes développés commencent à prendre la forme d'agrégats de modules génériques indépendants, se rapprochant de la structure des systèmes actuels.

Ce deuxième cycle de conférences a donc mis au grand jour la nécessité de fragmenter une tâche d'extraction d'information devenue trop complexe en une série de fonctionnalités indépendantes, concept phare des systèmes actuels.

Le troisième et dernier cycle est composé des conférences MUC-6 (1995) et MUC-7 (1998). MUC-6 marque un profond tournant pour ces campagnes d'évaluation. En effet, trois nouvelles taches sont introduites et la tâche initiale est simplifiée afin de répondre aux nouveaux objectifs fixés par le comité scientifique de l'époque. Retenant les leçons des cycles précédents, MUC-6 se donne pour programme de concevoir des systèmes

19 March – A bomb went off this morning near a power tower in San Salvador leaving a large part of the city without energy, but no casualties have been reported. According to unoffial sources, the bomb – allegedly detonated by urban guerrilla commandos – blew up a power tower in the northwestern part of San Salvador at 0650 (1250 GMT).

INCIDENT TYPE	bombing
DATE	March 19
LOCATION	El Salvador : San Salvador (city)
PERPETRATOR	urban guerrilla commandos
PHYSICAL TARGET	power tower
HUMAN TARGET	-
EFFECT ON PHYSICAL TARGET	destroyed
EFFECT ON HUMAN TARGET	no injury or death
INSTRUMENT	bomb

FIGURE 1.2: Exemple de formulaire d'extraction au cours de la conférence MUC-3[3]

indépendants, portables et faciles d'utilisation. Au cours de cette conférence, les organisateurs veulent encourager la production de travaux pouvant contribuer à des modules de compréhension profonde des documents [9, 10]. Concernant cette problématique spécifique, trois tâches sont envisagées : la résolution de coréférences, la désambiguisation lexicale et la détection de structures prédicat-argument. Baptisé *semEval*, ce pôle de recherche devant regrouper ces trois tâches ne présentera que la résolution des coréférences lors de la conférence, les problématiques étant encore trop récentes pour être traitées en temps et en heure par les participants. Concernant l'aspect portabilité et facilité d'utilisation, un *mini-MUC* est mis en place en parallèle. Au cours de cette compétition le formulaire traditionnel d'extraction d'information est simplifié. Le nombre de champs passe de 24 à 6, un formulaire additionnel d'entitées est également proposé (*template Element*). Toujours dans l'esprit de créer des modules de traitement de plus en plus indépendants, on voit alors apparaître les premiers systèmes de reconnaissance d'entités nommées (EN).

MUC-6 est donc composé au final des tâches suivantes :

— Entités Nommées
— Coréférence
— Formulaire des entités (Template Element)
— Formulaire des scénarios (Scenario Template, correspondant au template traditionnel des MUCS)

Ce programme certes ambitieux pour l'époque se focalise sur la nécessité de séparer les tâches en modules indépendants, chacun spécialisé dans l'extraction d'un type d'information précis. Cette organisation de la tâche d'extraction d'information sous la forme de modules constitue la principale innovation de cette sixième conférence.

Les corpus fournis sont de taille inférieure et mis à disposition des participants plus tardivement avant le début de la compétition. De ce fait, cette conférence voit se multiplier les méthodes probabilistes et les systèmes à base d'apprentissage (automates à états-finis, réseaux de neurones, ...). Les niveaux de performances atteints pour chacune des tâches définies commencent à être intéressants. Pour la tâche sur les entités nommées, des F-mesures supérieures à 90% sont atteintes par plusieurs systèmes. Concernant le formulaire des entités les performances sont de l'ordre de 75 % pour le rappel et de 86 % pour la précision, pour le formulaire scénario (ou *mini-MUC*) le rappel et la précision sont respectivement de l'ordre de 50% et de 70 %. Ces résultats prometteurs, au sortir de MUC-6, attestent de la nécessité de décomposer la tâche d'extraction d'information pour d'une part obtenir des résultats probants et d'autre part pour faciliter la conception de systèmes génériques indépendants. Si d'importants progrès sont encore à réaliser pour permettre la portabilité des systèmes, le bilan de MUC-6 est en grande partie conforme aux objectifs fixés préalablement par le comité scientifique de l'époque.

Organisé trois ans plus tard, MUC-7 ne fera pas autant d'émules que MUC-6, peu de choses nouvelles sont à noter. Parallèlement à MUC-6 et MUC-7, une campagne d'évaluation multilingue en extraction d'information a été organisée : MET (*Mutlilingual Entity Task*). Cette conférence a permis, entre autres, le développement de systèmes de reconnaissance d'entités nommées pour l'espagnol, le japonais et le chinois, expatriant avec succès cette tâche du monde anglophone vers les autres langues.

Ce dernier cycle de la série des MUC, même s'il s'achève brutalement, n'en reste pas moins fondamental au regard de l'évolution de la tâche d'extraction d'information et plus précisement de la tâche de reconnaissance des entités nommées. Après ce rapide survol des conférences MUC, considérons à présent la tâche de reconnaissance sur les entités nommées de plus près, puisqu'elle est au coeur de notre problématique.

1.2.2 Reconnaissance d'entités nommées au sein des conférences MUCs

La tâche spécifique de reconnaissance des entités nommées est apparue lors du troisième cycle des conférences MUC (MUC-6) et sera également présente dans la septième et dernière conférence MUC. L'intérêt pour ce type d'information s'explique par le fait qu'elles sont présentes dans tout type de texte, quel que soit le domaine (généricité). La détection et la reconnaissance de ces entités constituent donc un point de passage obligatoire pour tout système cherchant à extraire l'information pertinente contenue dans un texte. La tâche d'extraction et de reconnaissance d'entités nommées lors de MUC-6 s'est focalisée sur trois types d'entités nommées.

ENAMEX : pour les noms de personnes, d'organisations et de lieux. Les sous-types sont : PERSON, ORGANISATION et LOCATION

TIMEX : pour les expressions temporelles. Les sous types sont : DATE et TIME.

NUMEX : pour les expressions numériques. Les sous-types sont : MONEY et PERCENT.

Lors de la sixième conférence MUC, la tâche d'extraction d'entités nommées consistait à annoter les entités nommées de type ENAMEX, TIMES et NUMEX comme illustré dans la Figure 1.3.

Mr. <ENAMEX TYPE=« PERSON » > Dooner </ENAMEX> met with <ENAMEX TYPE=« PERSON » > Martin Puris </ENAMEX>, president and chief executive officer of <ENAMEX TYPE=« ORGANIZATION » > Ammirati & Puris </ENAMEX>, about <ENAMEX TYPE=« ORGANIZATION » > McCann </ENAMEX>'s acquiring the agency with billings of <NUMEX TYPE=« MONEY » > $400 million </NUMEX>, but nothing has materialized.

FIGURE 1.3: Exemple d'annotation d'entités nommées (MUC-6)[3]

Les performances atteintes lors de ces deux campagnes d'évaluation furent remarquables. Lors de la conférence MUC-6, les documents à étudier étaient des dépêches portant sur des changements de position dans les entreprises. Lors de la compétition, vingt systèmes furent testés sur cette base de données (15 participants). Sur la totalité des systèmes, plus de la moitié affichent des taux combinés de rappel et de précision supérieurs à 90%. Le meilleur d'entre eux obtient une F-mesure de 96%. Ces résultats inattendus sont d'autant plus impressionnants qu'ils approchent des performances humaines. Néanmoins, comme le font remarquer B. Sundheim [11] et R. Grishman [10], ces performances sont à apprécier en tenant compte du fait que les corpus d'évaluation sont homogènes, de très bonne qualité rédactionnelle, et en nombre relativement restreint (30 exemples). Des performances du même ordre de grandeur furent constatées lors de la conférence MUC-7. Toutefois, les performances pâtirent un peu de l'augmentation de la complexité des documents (le meilleur système n'obtient que 92 % de rappel et 95 % de précision). Ceci s'explique par l'ajout des expressions temporelles relatives dans la catégorie TIMEX, et par le fait que le corpus d'entrainement soit d'un autre domaine (accidents d'avions) que celui de test (lancements de satellites).

Force est donc de constater que la tâche de reconnaissance des entités nommées fut un réel succès pour les conférences MUC. Les résultats étant très vite devenus prometteurs, cette tâche suscita beaucoup d'engouement de la part des participants et des utilisateurs potentiels. Les systèmes devenant de plus en plus génériques et de plus en plus faciles d'utilisation, les entreprises et le gouvernement commencent à s'intéresser de près à ces

derniers. Tant et si bien qu'à l'issue de MUC-6, deux systèmes sont commercialisés et d'autres intégrés dans des systèmes gouvernementaux d'analyse de textes.

Ces dix années de Conférences américaines Message Understanding ont permis de faire considérablement évoluer la définition de la tâche d'extraction d'information ainsi que les systèmes de traitement associés. Ceci a permis de sensibiliser les entreprises et les gouvernenements sur l'importance de traiter ce genre de tâche afin de faire évoluer le traitement automatique de documents.

La définition de l'extraction d'information continue d'évoluer, elle tend à devenir de plus en plus spécifique et à s'intéresser à des éléments de plus en plus précis dans le texte. Cette évolution ne simplifie en rien le problème initial et les systèmes deviennent de plus en plus spécifiques. Le fait de limiter la diversité des types d'information à reconnaître n'empêche pas les systèmes produits d'être complexes car l'information reste difficile à extraire. C'est ainsi que, compte tenu de ces dernières évolutions et de la nécessité de prendre en compte la réalité des applications industrielles, l'extraction d'information a progressivement pris un caractère modulaire, se décomposant en plusieurs fonctionnalités autonomes.

Suite à cette série de conférences, de nombreuses autres campagnes d'évaluation ont été mises en place permettant d'avancer un peu plus dans le développement de cette tâche nouvelle.

1.2.2.1 Les autres conférences et projets traitant des entités nommées

Nous avons déjà évoqué les deux campagnes MET, organisées parallèlement à MUC-6 puis MUC-7, et dédiées à la reconnaissance des entités nommées dans d'autres langues que l'anglais en l'occurence l'espagnol, le chinois et le japonais [12]. Ces deux campagnes bien que peu médiatisées et informelles ont motivé diverses expérimentations et favorisé la conception de systèmes indépendants de la langue employée. On voit donc apparaître les premiers systèmes proposant des performances intéressantes sur la détection et la reconnaissance des entités nommées dans des corpus de nature journalistiques espagnol, chinois et japonais. Immédiatement après ce premier pas vers des systèmes multi-lingues, le projet d'évaluation IREX (Information Retrieval and Extraction Exercise) fait son apparition au Japon. L'objectif des organisateurs [13] était de réaliser des conférences proches du fonctionnement des cycles MUC, en évaluant des systèmes d'extraction d'entités nommées sur des bases communes et en partageant données et expériences. Une quinzaine de systèmes ont été mis en compétitions sur des textes extraits de quotidients nippons. Les scores obtenus ont été très encourageants (certains systèmes atteignent des

F-mesures supérieures à 80% [14]). En 2002 et 2003, CoNLL (Conference on Computational Natural Language Learning) a proposé une tâche de reconnaissance d'entités nommées, pour l'espagnol et le hollandais dans un premier temps, puis pour l'anglais et l'allemand. Cette conférence a réunit plus d'une dizaine de participants à chaque fois [15].

En France, la campagne ESTER (2002-2006) intégrée au projet EVALDA a proposé une tâche d'évaluation de systèmes de transcription d'émissions radiophoniques en langue française. Ces transcriptions ayant été au préalable enrichies par un ensemble d'informations annexes dont le marquage des entités nommées.

S'attachant encore d'avantage à l'aspect multilingue, la campagne HAREM proposa quant à elle une évaluation pour le portugais (portugais du Portugal, du Brésil d'Afrique et d'Asie). Elle réunit près de 10 participants autour de corpus de natures diverses, issus de journaux, de courrier électroniques, de fictions, de rapports techniques ou encore de pages web [16].

Il importe également d'évoquer le programme ACE, mis en œuvre de 2000 à 2004. Ce programme de recherche entend poursuivre des travaux dans la même direction que les conférences MUC, à savoir la détection des entités nommées, des événements et des relations entre ces éléments sur la langue anglaise. Cependant l'accent est plus centré sur la mise au point de technologies que sur le traitement d'applications spécifiques. En ce qui concerne la tâche d'analyse de texte en elle-même, la problématique est un peu différente de celle de MUC. L'objectif dans ACE est plus complexe et plus précis, la compréhension des éléments au sein des entités nommées est devenue essentielle. Il ne s'agit plus ici d'extraire une chaîne de caractères mais bien d'essayer d'extraire le concept d'entité nommée en elle-même. Cela implique donc une annotation beaucoup plus pointue des entités nommées. Néanmoins, le nombre de types d'entités nommées différentes reste le même (PERSONNE, ORGANISATION et LIEU). Certaines modifications ont toutefois été apportées afin que ces groupes couvrent un plus grand ensemble de cas. Comparé aux conférences MUC, le niveau d'abstraction de compréhension du texte s'est élévé d'un cran.

Pour finir, on ne peut passer sous silence, le projet Quaero de 2008 à 2013 [6]. Ce grand projet européen est fondé sur un consortium public-privé unique de 32 partenaires franco-allemands. Il est composé de groupes industriels, de PME, de laboratoires et d'institutions publiques, à la pointe dans leur domaine de recherche. Coordonné par Technicolor, Quaero mobilise plus de 1000 doctorants, chercheurs, ingénieurs et administratifs. Quaero fédère un grand nombre de partenaires de tous horizons. Au sein de cet immense projet européen, l'extraction des entités nommées a bien entendu une place de choix. La définition des entités nommées a une nouvelle fois évolué pour devenir des

entités nommées étendues permettant de couvrir tous les cas nécessaires. C'est pourquoi elles contiennent désormais de nouveaux types (civilisations, fonctions, ...) et des expressions construites autour de noms communs. Nous sommes donc bien loin des 3 types d'entités nommées définis au cours des conférences MUC.

1.3 Descriptions de quelques systèmes d'extraction d'EN existants

Placées sur le devant de la scène à l'occasion des conférences MUC, les entités nommées n'ont depuis cessé de motiver d'autres projets ou campagnes d'évaluation, favorisant ainsi de nombreux travaux. Beaucoup de systèmes ont vu le jour depuis, les plus performants ont été répertoriés dans plusieurs études [14, 17–20].

Dans cette partie, nous allons dans un premier temps nous atteler à détailler les différents indices et règles possibles pour reconnaître ces informations si utiles aux systèmes d'extraction d'information. Puis nous passerons rapidement en revue quelques systèmes d'extraction appliqués à différents domaines. Finalement nous ferons un point sur les éléments et aspects essentiels à la reconnaissance d'entités nommées.

1.3.1 Reconnaissance d'entités nommées : indices et règles

Comment reconnaître des entitées nommées dans des textes ? voila l'essence même du problème. Avant de se lancer dans la conception d'un système de reconnaissance EN, il nous faut d'abord passer en revue les différents indices ou règles afin de faciliter le travail de nos futurs systèmes. Ces règles ou indices peuvent être très intuitifs ou basés sur une étude approfondie de la sémantique du document.

1.3.1.1 Comment identifier une entité nommée ?

D.McDonald propose, dans un article abondamment cité depuis sa parution [21], les indices internes (internal evidence) et indices externes (external evidence). Les indices internes concernent tous les éléments se trouvant à l'intérieur d'une unité lexicale ou d'un syntagme (groupe d'unité lexicales) pouvant amener à la détection d'une entitée nommée. En général, ces derniers permettent dans 9 cas sur 10 de trouver directement l'entité nommée. Il faut cependant les utiliser avec parcimonie car ils peuvent entraîner dans certains cas des confusions majeures menant à d'importantes baisses de performance des systèmes. Afin d'illustrer ce problème, prenons un exemple simple d'indice :

la présence ou non de majuscule dans le mot. Les noms de famille, les prénoms, les noms d'entreprise contiennent souvent des majuscules. Cependant, en français le premier mot de chaque phrase contient également une majuscule qu'il s'agisse d'une EN ou non. La majuscule manifeste donc la présence potentielle d'une EN mais ne permet pas de la catégoriser à 100%.

— **Jessica** Alba
— **J.** ALBA
— le **roi Louis XV**
— Microsoft **Inc.**

En plus des indices internes, on peut ajouter les indices externes aux EN ou preuves externes. Il s'agit dans ce cas là de mots introduisant des EN, l'exemple le plus simple est **Monsieur, Madame**. Ces éléments appelés communement *amorces* (trigger words) peuvent être des mots, des abbréviations précédant ou suivant régulièrement une entité nommée. Ils ne font pas partie de l'EN en elle-même mais leur présence indique la possibilité d'avoir une entité nommée juste après, par exemple :

— Le **président** Hollande, **Mme** Martin
— **Général** De Gaulle
— l'**entraîneur** Claude Onesta
— the Coca-cola **compagny**

La frontière entre les deux types d'indices est fine. Elle dépend de la limite que l'on souhaite pour l'entité nommée notamment quand il s'agit d'un titre (général, président, etc.). Qu'elles soient internes ou externes, ces preuves ou indices constituent des informations primordiales pour le bon fonctionnement d'un système de reconnaissance d'EN. D'autres informations peuvent également être prises en compte, l'utilisation d'un lexique par exemple. Un lexique ou base de connaissance lexicale a pour but de décrire le sens, la relation et les emplois d'un ensemble de mots. Il peut prendre différentes formes en fonction du niveau d'abstraction nécessaire : dictionnaire, grammaire, thésaurus ou terminologie [22]. Il s'agit d'apporter au système des connaissances *a priori* sur des informations à reconnaître dans le document. Dans le cadre de la reconnaissance d'EN, les lexiques sont généralement des listes de mots associés à des catégories sémantiques (personnes, lieux, ...). L'utilisation de lexiques a été initiée dès MUC-6 et bien que controversée demeure très répandue à l'heure actuelle.

La plupart des systèmes actuels d'extraction d'EN se sont basés sur la combinaison des connaissances contenues dans ces informations contextuelles et lexicales. Elles constituent les deux sources de connaissances les plus fiables pour ce type de tâche.

H. Shinyama et S. Sekine ont cherché à ajouter d'autres types d'éléments informatifs dans leur système d'annotation d'EN afin de traiter les EN rares pour lesquelles il est

difficile d'avoir accès aux informations citées plus haut [23]. Dans ce cas, ils proposent d'exploiter des corpus de même nature afin d'ajouter de l'information supplémentaire sur l'apparition des mots recherchés. Leur hypothèse est la suivante : étant donnés deux corpus de textes journalistiques alignés sur la base de leur date de parution (articles publiés le même jour), un mot pour lequel on observe un pic de fréquence similaire entre les deux corpus a de fortes chances d'être une entitée nommée. En effet, une même entité nommée ne peut varier de façon conséquente entre deux articles traitant du même sujet au même instant. Utilisant deux corpus journalistiques de 1995, *Los Angeles Times et Reuters*, les auteurs ont observé la fréquence d'apparition de deux mots "killed" et "yigal", soit un verbe et le nom de l'assassin du premier ministre israelien Yitzhak Rabin, décédé le 7 novembre 1995. Le verbe "killed" apparait de manière régulière dans l'ensemble du corpus étudié tout au long de l'année, on en déduit qu'il s'agit d'un mot commun. Concernant le nom de l'assassin "yigal" sa fréquence a fortement augmenté sur une très courte période, ce comportement conduit à penser que ce mot est probablement une entité nommée.

Les auteurs ont renforcé leur approche en ajoutant des paramètres supplémentaires comme la variation du nombre d'articles publiés par jour, le délai de publication, etc. Ils sont arrivés à la conclusion que l'aspect temporel est un indice important pour le repérage des entités nommées. Cependant, cette technique comporte quelques défauts. Elle ne permet pas de reconnaître un grand nombre d'entités nommées et surtout elle ne permet en aucun cas de les classer. Les auteurs insistent sur le fait que cette approche doit être utilisée en complément des indices internes et externes et des lexiques définis plus haut afin d'assurer une performance correcte des systèmes.

Nous avons mis en avant les différentes informations à prendre en compte afin de produire un système de reconnaissance d'EN performant. La partie suivante est consacrée à la description des différentes manières d'acquérir et d'exploiter les EN.

1.3.1.2 Comment décrire les entités nommées ?

En TAL, on distingue traditionnellement deux grandes approches, l'approche dite linguistique ou symbolique, et l'approche dite statistique.

La première est basée sur l'intuition humaine, l'opérateur construit manuellement des modèles d'analyse, sous la forme de patrons d'extraction. Ces patrons sont généralement composés d'un ensemble de règles morpho-syntaxique et de diverses ressources annexes comme des lexiques ou des grammaires. La reconnaissance d'entités nommées a majoritairement été effectuée suivant cette approche dans les années 1990 notamment au cours des conférences MUCs. En effet, les EN ont des caractéristiques particulières bien

adaptées à la reconnaissance par règles linguistiques. Par exemple, si un prénom connu (connaissance issue d'un lexique) précède un mot inconnu commençant par une majusule, alors l'entité nommée peut être étiquétée comme un nom de personne ; ou bien : si un mot inconnu est suivi du mot (ou la forme) Inc., alors il s'agit d'un nom d'organisation. Ici, les règles sont grandement simplifiées car la segmentation précise de l'entité nommée nécessite des règles beaucoup plus complexes mais l'essentiel de l'approche y est résumé. Cette approche permet de gérer des cas difficiles et peu présents dans la base, elle est également plus précise que son homologue probabiliste. Le problème de ce type d'approche est bien entendu le temps de mise en place. En effet, un expert linguiste-informaticien met plusieurs mois à produire un ensemble de règles assez robustes pour avoir des performances correctes sur le corpus à étudier. De plus, il n'est pas dit que l'expert ne passe pas à coté de certaines règles pouvant apporter des informations discriminantes supplémentaires.

Dans la deuxième approche, l'opérateur a beaucoup moins d'impact sur le fonctionnement interne du système. L'objectif est de créer des modèles d'analyse à partir de volumes importants de données. Ces données sont appelées données d'apprentissage, le volume doit être assez conséquent pour couvrir le maximum de variations au sein du corpus. Plus la base d'apprentissage est grande plus les modèles seront longs à être mis en place mais plus ils seront robustes. Les méthodes employées sont diverses et variées, elles peuvent prendre la forme d'arbres de décisions, de modèles probabilistes, de Modèles de Markov Cachées, de Champs Aléatoires Conditionnels, etc. Avec le développement important de l'apprentissage statistique, ces méthodes sont de plus en plus utilisées en TAL et dans beaucoup de domaines voisins (reconnaissance de la parole, reconnaissance d'écriture manuscrite, etc). Elles tendent à remplacer les systèmes à base de règles. Ce type de système présente l'avantage de pouvoir être facilement transposé à des domaines voisins ou à des problèmes légèrement différents, il est également plus robuste au bruit. La problème de cette approche réside dans la nécessité de posséder un corpus annoté.

La solution idéale semble être comme souvent un compromis entre ces deux méthodes. Ces approches de types hybrides ou mixtes sont rendues possibles grâce à la maturité acquise à la suite des conférences MUCs.

La partie suivante détaille des exemples concrets de systèmes permettant la reconnaissance des EN.

1.3.2 Quelques systèmes de référence

N. Friburger passe en revue dans son manuscrit de thèse [20] un grand nombre de systèmes de reconnaissance d'entités nommées, qu'ils soient symboliques, à base d'apprentissage statistique ou hybrides. Dans cette partie, nous présenterons un système par approche en TAL puis un système dans un domaine transversal la reconnaissance de la parole.

1.3.2.1 Un exemple de systèmes symbolique : laSIE

Développé à l'Université de Sheffield dans les années 90, le système LaSIE [24] fut initialement conçu dans le cadre d'un projet de recherche autour de l'extraction d'information et plus précisement du traitement naturel de la langue. L'objectif était bien sur de participer à la compétition de la conférence MUC-6. Les auteurs ont choisi de mettre en place un système intégré contenant au sein d'une seule et même architecture plusieurs modules capables de répondre aux différentes tâches proposées. Ils imbriquent trois étapes successives, premièrement une analyse lexicale, puis une analyse sémantique et enfin une classification des éléments à trouver. Se conformant aux exigences de la conférence MUC-6, ce système possède une chaîne de traitement modulaire prenant en entrée un document et produit en sortie un fichier respectant les templates de l'évaluation.

La première étape est divisée en un ensemble d'opérations toutes modulaires permettant de répondre au mieux à la problématique. On y trouve une segmentation en mots (transformation du flux de caractères en chaîne d'unités ou tokenization), un étiquetage des parties du discours à l'aide de la méthode *Transformation-Based Error-Driven Learning* d'E Brill [25] (méthode à base de règles couplée à une phase d'apprentissage), une lemmatisation et l' étiquetage d'entités nommées sur la base de lexiques de noms propres et de lexiques spécialisés comprenant des mots amorces. Cette phase correspond en quelques sorte à l'étape de compréhension globale du document. Seule l'étape d'étiquetage des parties du discours basée sur la méthode d'E Brill contient une partie apprentissage statistique, l'étiquetage des EN est réalisé par une simple interrogation de lexique (look-up).

La deuxième étape collecte les informations récupérées précédemment et effectue une analyse sémantique par le biais de l'utilisation de grammaires. Au cours de cette analyse une recherche des entités nommées "évidentes" est effectuée. Suivant un ensemble de règles définies au préalable (environ 200 dont presque la moitié pour les noms d'organisations), le système extrait les EN et les classifie. La complexité de ces règles diffère en fonction du type d'EN, elles vont d'une simple suite de majuscules à l'apparition de

certains mots clés discriminants ("Inc", "Monsieur", etc.). Ci-après un exemple de règle de reconnaissance d'entité nommée utilisée dans LaSIE :

```
ORGAN_NP -> NAMES_NP '&' NAMES_NP
```

Cette règle ([24]) signifie que la séquence désigne un syntagme *nom d'organisation*

La troisième et dernière étape procède à l'analyse finale du discours, elle effectue l'annotation complète des entités nommées à partir des différentes étapes préalables . C'est à cette étage que les problèmes de coréférence sont réglés à l'aide d'un ensemble d'heuristiques définies au préalable. Ces heuristiques prennent en compte le contexte de décision par exemple si dans un texte un nom d'organisation est reconnu correctement, et que des abbréviations ou des variantes de ce même nom d'organisation sont reconnues, elles se verront attribuer la même classe d'appartenance (l'organisation redondante). Ces règles sont construites manuellement, aucun module d'apprentissage n'est utilisé, il s'agit ici d'une phase de look-up permettant d'enrichir les annotations du système.

Au final, LaSIE est un système intégré d'analyse de textes entièrement fondé sur une approche linguistique ; les résultats obtenus lors de MUC-6 pour la tâche de reconnaissance des entités nommées furent de 0.91 de F-mesure, et pour MUC-7 de 0.85 (performance réalisée par LASIE-II, pour plus de précision, se reporter à [26]. A la suite de cette description d'un système de reconnaissance d'entités nommées de type symbolique, considérons à présent un exemple à base d'apprentissage.

1.3.2.2 Un exemple de système à base d'apprentissage : MENE

Le système MENE [1] a été présenté pour la première fois à MUC-7 (1998) par l'Université de New-York. Conçu par [27], MENE est un système de reconnaissance d'entités nommées basé sur le principe d'entropie maximale. À l'origine grandeur thermodynamique mesurant le degré de désordre de la matière, le principe de l'entropie a été adapté à la théorie de l'information par C. Shannon. Cette mesure correspond à l'incertitude liée à la distribution d'un événement aléatoire. Transposée à la reconnaissance d'entités nommées, un classifieur à entropie maximale permet d'obtenir une sortie probabiliste informant sur la probabilité d'apparition des caractères. En utilisant ce principe, on doit au cours du cycle d'apprentissage parvenir à réaliser l'étiquetage souhaité (correspondant à la vérité terrain) tout en restant le moins spécifique possible afin que le système soit générique. Il est impératif de garder le plus d'indécisions possibles dans le système tout en essayant de maximiser la probabilité de la bonne solution. Comme tout système

1. Maximum Entropy Named Entity

d'apprentissage statistique, il nécessite la mise en place d'un ensemble de caractéristiques correspondant à la représentation machine des documents à analyser. A. Borthwick et al. ont utilisé un ensemble de descripteurs divisés en 5 sous ensembles :

Descripteurs Binaires : descripteurs pouvant prendre 2 valeurs 0 ou 1 représentant la présence ou non d'un ensemble d'événements (majuscules,caractères numériques, etc.).

Descripteurs lexicaux : descripteurs issus du contexte lexical de l'unité considérée. Ils correspondent à un ensemble de règles basées sur la présence ou non de préfixes, suffixes ou mots amorces. Par exemple si un mot est précédé du mot "Mrs" ou "Mr" il y a une forte probabilité pour que le mot suivant soit une EN (ces descripteurs sont également binaires).

Descripteurs textuels : descripteurs indiquant la position spatiale de l'information dans la structure du texte, c'est-à-dire la localisation de l'unité dans le titre, le résumé, le corps du document, etc.

Descripteurs issus de dictionnaires : descripteurs signalant l'appartenance d'une unité à l'un de ces cinq états possibles, "start, continue, end, unique, other", sur la base d'une récupération des informations contenues dans des dictionnaires de noms propres. Ainsi, à partir de la connaissance de l'unité British Airways contenue dans le dictionnaire, si l'expression *on British Airways Flight 962* est rencontrée, alors elle se voit attribuer la caractéristique "other start end other other". Cela signifie que l'EN commence au deuxème mot de l'expression et finie au troisième (British Airways). MENE exploite de la sorte différentes ressources comme des dictionnaires de noms de personnes, d'organisations, d'universités, de régions, ainsi que des sigles d'entreprises. Les auteurs insistent sur l'importance de ce type de caractéristiques pour leur système.

Descripteurs externes : descripteurs issus d'autres systèmes de reconnaissance d'entités nommées (entités déjà reconnues avec un score de confiance élevé, etc).

Une fois l'ensemble de ces descripteurs collecté, MENE utilise un classifieur à maximum d'entropie (équivalent de la régression logistique cf section 2.3.4.1) pour l'annotation d'entités nommées dans la base de test. Lors de la compétition MUC-7, ce système afficha 0.92 de F-mesure sur le corpus d'entrainement et 0.84 sur le corpus de test.

Rappelons que durant cette conférence, le changement de domaine entre le corpus d'entraînement (sur les accidents d'avion) et le corpus de test (sur les lancements des satellites), ne fut pas annoncé par les organisateurs de la compétition et prit par suprise la quasi totalité des systèmes à apprentissage statistique participants. Nous avons ici l'illustration de l'un des points faibles de ce type de système, nécessitant de nouveau corpus

d'apprentissage pour pouvoir fonctionner sur des domaines très différents, contrairement aux systèmes symboliques qui si leur règles sont bien structurées [19] sont facilement adaptables à d'autres dictionnaires de mots à reconnaître.

Quoi qu'il en soit, les résultats obtenus par MENE sont très intéressants. Ces derniers varient bien sûr en fonction de la quantité de données disponibles pour l'apprentissage : avec 20 documents MENE propose une F-mesure de 0.8 mais monte à 0.92 avec 425 documents. Toutefois, le système maximise sa performance quand il est combiné avec d'autres systèmes, de type symbolique, tels que IsoQuest [28] , Manitoba [29] ou Proteus [30]. Dans ce cas de figure, MENE affiche alors, sur le corpus d'entraînement de MUC-7, 0.97 de F-mesure. Ainsi, A. Borthwick et al. confirme l'intérêt de l'utilisation de méthodes de type hybride combinant une méthode symbolique et une à apprentissage statistique, l'une pouvant pallier les défauts ou insuffisances de l'autre. C'est ce postulat que D. Lin s'applique à mettre en œuvre.

1.3.2.3 Un exemple de système mixte : travail de D. Lin.

Pour la compétition MUC-7, D.Lin [29] a proposé une extension du système symbolique réalisé par l'Université de Manitoba lors de MUC-6, extension consistant à ajouter un module d'enrichissement et de génération automatique de règles de reconnaissance d'entités nommées sur la base de données quantitatives au système initial. Le processus d'annotation des entités nommées opère en deux passes : la première construit à partir d'un corpus traité par le système intial (annotation complète et analyse syntaxique), une base de données de bigrammes de mots à partir de laquelle sont ensuite générées de nouvelles règles. L'annotation des EN à cette étape n'est pas encore définitive, elle sera validée par la deuxième passe qui exploite les informations issues de la première passe. La construction de la base de bigrammes de mots est effectuée grâce à un parseur (MINIPAR), chaque bigramme est stocké avec sa fréquence d'apparition. L'auteur propose de concaténer ces informations contextuelles au sein d'un même vecteur appelé "dependency triple". Ce vecteur prend la forme suivante, (word, relation, relative) où les champs "word", "relation" et "relative" correspondent respectivement au mot du corpus concerné, son bigramme et la relation grammaticale reliant les deux champs précédents. Chaque mot du corpus est stocké dans la base suivant ce protocole :

review, V :comp1 :N, acquisition=3 signifie que le verbe to review a eu trois fois le nom acquisition en relation objet dans le corpus analysé

review, N :nn :N, admission=2 signifie que le nom review a eu deux fois le nom admission en relation modifieur de nom dans le corpus analysé

La base de données contient au total 22 millions de mots. Une fois construite, cette base de données est exploitée par deux mécanismes :

1. Un mécanisme de génération automatique de règles d'annotation d'entités nommées évidentes ;
2. Un mécanisme d'annotation des entités nommées inconnues.

Rappelons avant tout que le système initial proposé par l'Université de Manitoba est purement linguistique (basé sur des règles). Il exploite des patrons à base d'automates à états finis, tirant profit de ressources lexicales tout comme des formes de surface du texte à traiter. D.Lin part du principe que le contexte d'apparition d'une entité nommée constitue une information vitale pour sa catégorisation (indice externe). Le contexte ainsi que la fréquence ont été résumés pour l'ensemble du corpus dans la base de données, il est alors facile d'observer la répartition de ces informations pour en déduire des schémas d'annotations. L'auteur explique par exemple que sur 33 occurences de noms propres précédés de la séquence *managing director*, 26 sont classées comme faisant partie de la catégorie d'organisation. Grâce à cette étude, 3600 règles permettant d'annoter des entités nommées contenant ou étant des noms propres ont pu être mise en place. Cependant, certaines observations ne semblent pas être conditionnées par des règles évidentes, dans ce cas aucune classification sûre n'est déduite. A ce stade, l'on dispose donc d'un ensemble de règles permettant de détecter les informations souhaitées, cependant, la complexité de certaines entités nommées (variabilité, etc) rend la mise en place de règles robustes très difficile manuellement. Afin de palier ce problème, l'idée est de laisser la méthode définit ses propres règles à l'aide d'un apprentissage statistique.

C'est ici qu'intervient le second mécanisme qui a pour but de classifier ces entités nommées "difficiles". Ce système est composé d'un classifieur bayésien naif qui permet de calculer des probabilités conditionnelles.

L'objectif final est de prédire la catégorie d'une entité nommée à partir d'un ensemble de caractéristiques. Ces caractéristiques sont très simples, elles sont la concatenation de tous les vecteurs de dépendance triple en rapport avec la séquence à analyser. L'humain ne pouvant en tirer une règle logique, on laisse l'algorithme le faire lui-même. Il n'y a aucune assurance que toutes les entités nommées soient bien classifiées à l'issue de l'apprentissage, il est possible que l'ensemble des caractéristiques fournit ne soit pas assez discriminant pour obtenir des frontières parfaites entre les classes. Il est donc nécessaire de faire des tests de performance sur une base de validation afin de vérifier le bon fonctionnement du système. Une fois les deux mécanismes réalisés, le système atteint une F-mesure de 86% performance saluée par la conférence MUC-7.

Nous venons de décrire trois systèmes proposant trois approches différentes pour la reconnaissance d'entités nommées. Ces trois systèmes reposent sur les mêmes types d'indices mais les exploitent de manières différentes. Toutefois, ces derniers parviennent tous à passer les 85% de F-mesure lors des compétitions MUC. Ces trois systèmes fonctionnaient sur des documents electroniques, il n'y avait donc pas besoin d'étage de transcription comme c'est le cas par exemple dans des domaines voisins comme la reconnaissance de la parole ou la reconnaissance d'écriture imprimée. Qu'en-est-il de ces autres domaines ? Dans quelle mesure la difficulté est-elle augmentée ? C'est ce que nous allons examiner dans le paragraphe suivant.

1.3.3 Domaine Transversaux : Extraction d'entités nommées à partir de données bruitées

Miller et al. [5] analyse la dégradation des systèmes d'extraction d'entités nommées en fonction des erreurs de reconnaissance des systèmes de transcription en traitement automatique de la parole (journaux télévisés) et de l'écriture imprimée (journaux). Ils étudient également l'influence de la taille de la base d'apprentissage sur des systèmes d'extraction issus de données propres et bruitées.

Avec le développement des médias dans le monde, les volumes de données à traiter sont devenus de plus en plus importants, trop importants pour que des opérateurs classent manuellement les documents concernés. Des systèmes automatiques de traitement ont alors vu le jour pour des signaux audios (ASR : Automatic Speech Recogntion) et pour les documents imprimés ou manuscrits (OCR : Optical Character Recognition). Ces systèmes ont évolué depuis la date de publication (2000), à l'époque les ASR sur des journaux télévisés avaient encore un taux d'erreurs assez élévé entre 14 et 28% notament à cause de la présence importante de mots hors vocabulaire dans ce type de signal. Les OCR étaient eux déjà robustes à l'époque sur des documents imprimés notamment lorsque l'impression était en haute résolution. Lorsque cela n'est pas le cas, les erreurs étaient de l'ordre de 20%. Les auteurs disposaient d'une base audio comprenant 100 heures de l'émission *Broadcast News* divisées en 114 épisodes. 98 épisodes ont été sélectionnés pour la base d'apprentissage (640 000 mots) et 16 épisodes pour le test (160 000 mots). Pour la base de documents imprimés, il s'agissait du corpus *University of Washington English Image Database* ajouté au corpus de la conférence MUC-6. 690 000 mots ont été utilisés pour l'apprentissage et 20 000 pour le test.

L'idée centrale de cette publication est qu'un système de transcription peut ne pas reconnaître des informations très utiles pour la reconnaissance d'EN, typiquement la casse et la ponctuation.

Afin de valider cette intuition les auteurs proposent plusieurs expériences. Une première basée uniquement sur la présence ou non d'éléments jugés important comme la ponctuation et les majuscules que les systèmes de transcription automatique ont tendance à mal reconnaître (cette étude est uniquement effectuée sur la base audio). Dans un deuxième temps, une étude quantitative de l'influence du taux d'erreur de l'étage de transcription en fonction de la F-Mesure de l'extracteur d'EN pour la base audio et imprimés est réalisée. Finalement, la dernière analyse concerne l'influence de la taille de la base d'apprentissage pour 2 transcriptions ayant respectivement 0% et 15% d'erreur mot.

1.3.3.1 Expérience 1 : Influence des majuscules, minuscules et de la ponctuation

Dans un premier temps à partir d'une transcription parfaite, les auteurs ont créé trois bases aux propriétées particulières. La première ne comporte aucune minuscule, la deuxième aucune ponctuation et la dernière ni l'une ni l'autre.

TABLE 1.1: Influence de la présence de majusucles et de la ponctuation [5]

	Majuscule/Minuscule (F-mesure)	Majuscule seulement (F-mesure)
Avec ponctuation	92.4	90.1
Sans ponctuation	91.0	89.0

On observe que l'absence de différenciation majuscule/minuscule entraine une baisse de 2.3%, et que l'absence de ponctuation entraine une diminution de 1.4%. Les auteurs mettent en avant le fait que la somme des pertes individuelles est supérieure à la perte globale ce qui indique que dans certain cas l'absence complète des deux paramètres résoud le problème.

1.3.3.2 Expérience 2 : Influence de l'erreur mot

Sur cette même base, les auteurs ont effectué des tests de performances à partir de l'extracteur d'EN **IdentiFinder** le système le plus performant à l'époque de la publication prenant en entrée les sorties de tous les systèmes de transcription automatique de la langue de la campagne d'évaluation Hub-4 en 1998 [31]. A l'aide d'une interpolation linéaire, les auteurs ont pu mettre en avant qu'en moyenne 1% d'erreur mot supplémentaire entraine une perte de 0.7% de F-mesure.

Le même test a été effectué sur la reconnaissance d'écriture imprimée à l'aide du système décrit par [32] sur la base du *Wall Street Journal* (1.3 millions de lettres). Afin d'évaluer

l'influence de la dégradation de la transcription initiale, un OCR est utilisé sur les documents en haute définition (2.7% d'erreur mot), sur des images dégradées (13.7% d'erreur mot) et sur des images dégradées sans modèle interne de langage (19.1 % d'erreur mot). L'interpolation linéaire effectuée sur les résultats obtenus montre une diminution de 0.6% de F-mesure pour 1% d'erreur mot supplémentaire.

1.3.3.3 Expérience 3 : Influence de la taille de la base d'apprentissage

Les auteurs ont également effectué une autre expérience permettant d'observer l'influence de la taille de la base d'apprentissage sur les deux tâches citées plus haut. Pour la reconnaissance de la parole, ils ont sélectionné deux vérités terrain une à 0% d'erreur mot et une à 15%, le système d'extraction est entrainé sur des bases d'apprentissage allant de 30 000 à 1 200 00 mots. Une fois l'interpolation effectuée, on remarque que plus la taille de la base d'apprentissage est grande plus la F-mesure est élevée, cependant l'écart entre les deux courbes reste quasi identique. On peut en conclure que l'écart entre les deux courbes est lié à des erreurs sur des mots peu fréquents dans la base, l'augmenter ne change pas le problème initial.

Pour la reconnaissance d'écriture imprimée, l'expérience a été effectuée selon les mêmes transcriptions que précédemment, la transcription parfaite (0% WER), la transcription à partir d'images hautes définition (2.7 % d'erreur mot), celle sur des images dégradées (13.7% d'erreur mot) et finalement celle sur des images dégradées sans modèle de langage (19.1 % d'erreur mot). Les résultats obtenus ont la même tendance que pour la reconnaissance de la parole, l'augmentation de la base d'apprentissage augmente les performances globales des systèmes mais jamais les erreurs initiales de transcription ne sont compensées.

Dans cette partie, nous avons mis en avant les difficultés apportées par le bruit dans les signaux à reconnaître. Nous avons vu que pour la reconnaissance de la parole et de l'écriture manuscrite la performance du système et le taux d'erreur mot de la transcription semblent être liés par une fonction linéaire. Cependant, ces deux tâches ne proposent pas des documents aussi bruités que les travaux sur l'écriture manuscrite. Dans quelle mesure les performances vont-elles être affectées ? Est-ce la raison pour laquelle très peu voir aucune étude n'a été publiée sur la reconnaissance d'EN dans ce type de document ?

1.3.4 Conclusion

Dans ce chapitre, nous avons vu l'importance des EN dans l'extraction d'informations pertinentes pour la classification de documents ou la compréhension profonde de ces

derniers. Les EN sont en effet l'essence même du contenu d'un document. Le cycle des conférences MUC a permis de mettre en commun les ressources de plusieurs laboratoires de recherche afin de faire évoluer l'extraction d'informations automatique dans des documents. Les EN ont été au centre des problématiques de MUC-6, cette conférence a mis en avant la complexité de la détection de ce type d'informations. Depuis, beaucoup de projets se sont succédés exigeant des performances de plus en plus grandes sur des corpus de plus en plus difficiles. En effet, de par la nécessité de traiter des corpus de plus en plus grands et variés, le nombre d'entités nommées différentes n'a cessé d'augmenter, complexifiant ainsi le problème. Nous avons également présenté les différents éléments permettant de créer un système d'extraction d'entités nommées performant. Les indices internes et externes de D. McDonald [21] sont des éléments incontournables pour paramétrer des systèmes performants dans la réalisation de cette tâche. La détection de ces *patterns* ou mots *amorces* facilite énormément l'extraction des EN. Ils nous faudra donc construire un système robuste de détections de ces éléments dans des documents manuscrits. Cela implique la mise en place d'outils permettant la détection de mots clés (Monsieur, Madame), d'expressions alpha-numérique (numéros de référence, numéros de téléphone, etc.). La détection de l'ensemble de ces éléments permettra alors d'avoir les localisations probables des entités nommées dans le document. L'ensemble de ces positions sera alors exploité dans un deuxième étage qui effectuera une reconnaissance mot afin d'extraire les EN. Nos travaux se plaçant dans le cadre de la détection et la reconnaissance d'EN dans des documents manuscrits non-contraints, nous ajoutons à un problème déjà complexe les difficultés liés à la reconnaissance de l'écriture manuscrite (bruit, variabilité, etc.). Afin de palier l'ensemble des difficultés, deux types de systèmes s'opposent : les systèmes à base de règles et les systèmes à base d'apprentissage statistique. Dans ces travaux, nous proposons d'utiliser des méthodes à apprentissage statistique. Le second chapitre de cette thèse sera consacré entièrement à une vue d'ensemble des méthodes statistiques utilisées pour traiter des documents manuscrits. Nous détaillerons bien évidemment les méthodes à l'état de l'art, comme les Champs de Markov Cachés ou les systèmes hybrides à base de réseaux de neurones récurrents (BLSTM-CTC).

Chapitre 2

Modèles séquentiels pour la reconnaissance d'écriture manuscrite

2.1 Introduction

Pour proposer des éléments de réponse à la détection et la reconnaissance des entités nommées, nous avons besoin d'un système effectuant la reconnaissance d'écriture manuscrite. Nous avons opté pour l'utilisation de méthodes à apprentissage statistique. C'est-à-dire des méthodes qui vont être entraînées à traiter une tâche spécifique grâce à un processus d'apprentissage sur de nombreux exemples étiquetés. L'écriture manuscrite étant séquentielle (séquence de caractères), nous allons utiliser des modèles statistiques dynamiques capables de traiter des séquences. La modélisation de l'écriture manuscrite nécessite des outils statistiques performants en raison de sa grande variabilité et du niveau élevé de bruit dans les images. Beaucoup de travaux ont été publiés sur ce sujet [33–40], les systèmes utilisés sont variés. Parmis ceux-ci, les Modèle de Markov Cachés (MMC) sont des modèles génératifs encore très prisés des chercheurs actuellement. On trouve également des méthodes discriminantes comme les Réseaux de Neurones Artificiels (RNA) et les Champs Aléatoires Conditionnels (CAC). Les méthodes hybrides combinant des étages discriminants et des étages génératifs ou de programmation dynamique ont apporté des perspectives nouvelles en reconnaissance de l'écriture en améliorant significativement les performances [41–45]. Récemment, les BLSTM (Bidirectionnal Long Short Term Memory) combinés au CTC [46] (Connectionist Temporal Classification) ont prouvé qu'ils étaient la structure hybride la plus performante en classification de séquences [1, 47].

Dans cette partie, nous commençons par poser le problème de la reconnaissance d'écriture manuscrite, puis nous décrivons les modèles séquentiels génératifs (MMC [48]) et discriminants (CAC [49–51], Champs Aléatoires Conditionnels Cachés (CACC) [52]) après quelques rappels fondamentaux. Nous terminons ce chapitre par la présentation des méthodes hybrides alliant les qualités des modèles génératifs et discriminants qui permettent à l'heure actuelle d'avoir les meilleures performances dans le domaine de la reconnaissance de l'écriture manuscrite.

2.2 Modélisation de l'écriture manuscrite

2.2.1 Modélisation du problème

L'objectif est de donner une transcription ASCII d'une image contenant un ou des mots manuscrits.

Afin d'obtenir cette transcription il faut contourner plusieurs problèmes, comme la variabilité de l'écriture. En effet, la forme des lettres diffère d'une personne à une autre (cf Figure 2.1).

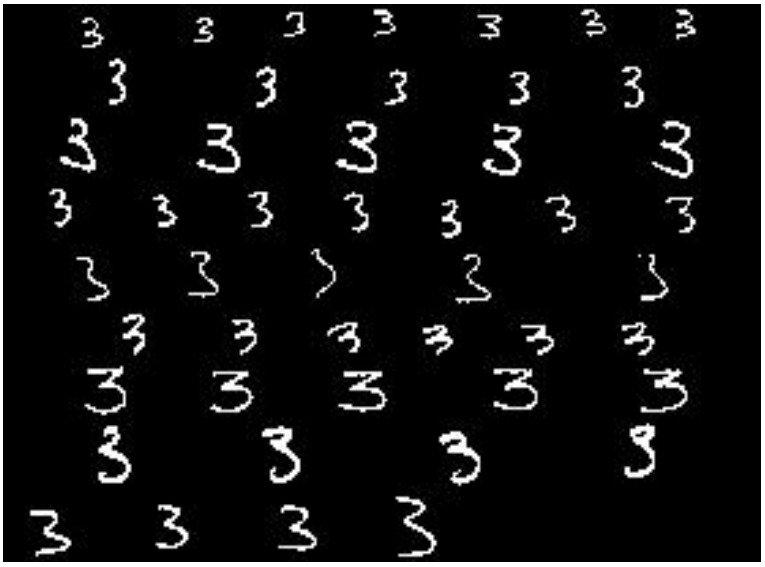


FIGURE 2.1: Exemple de variations de l'écriture manuscrite : le chiffre 3

Étant donné que nous travaillons sur l'écriture cursive, la seconde difficulté est de trouver le début et la fin de chacun des caractères du mot présents dans l'image afin de pouvoir les reconnaître (paradoxe de Sayre [53]). Cette tâche, appelée segmentation, est difficile même pour un être humain. En effet, en écriture cursive par exemple, une ligature entre deux lettres est difficile à classifier car on ne peut pas dire précisement quand finit le premier caractère et quand débute le deuxième.

On peut ajouter à cela la qualité du document qui peut contenir plus ou moins de bruit nuisant au bon déroulement de la reconnaissance (rature, écriture très serrée, ect.), voir la Figure 2.2.

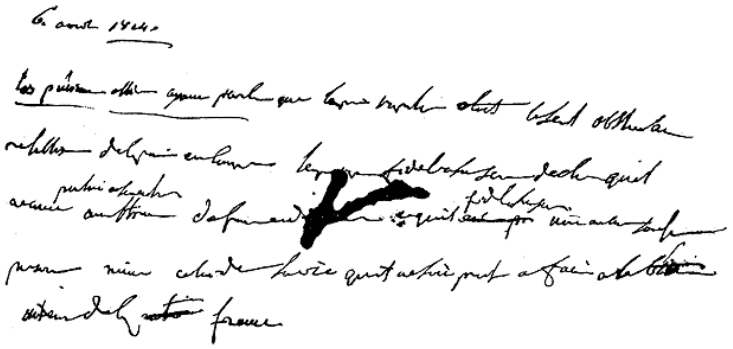


FIGURE 2.2: Exemple de document très bruité : lettres de Gutenberg

Dans des documents manuscrits, la prise en compte des dépendances entre les pixels (ou contexte) est essentielle pour fournir de bonnes décisions. Dans ces travaux, nous proposons de distinguer trois niveaux de dépendances (cf Figure 2.3) : Premièrement les dépendances proches entre les pixels qui forment un morceau de caractère comme la boucle d'un g, la barre d'un t, etc Deuxièmement les dépendances moyennes entre les pixels qui forment un caractère complet. Finalement les dépendances lointaines qui lient les caractères entre eux ou les mots entre eux si l'on travaille sur de la reconnaissance de lignes manuscrites complètes.

Dans le cadre de cette thèse nous avons opté pour des approches basées sur des modèles graphiques. Généralement, ces méthodes sont des modèles statistiques capables de modéliser des dépendances complexes entre des ensembles variables à l'aide d'une étape dîte d'apprentissage. Comme un enfant apprend à lire, ces méthodes "apprennent" à l'aide d'un corpus étiqueté (images de mots dont on connait la transcription) à reconnaître les lettres. Dans la plupart des cas, elles suivent un processus itératif. A chaque itération une image et sa transcription lui sont présentées, la méthode va alors modifier ses paramètres internes de façon à ce que la fonction de décision corresponde à la décision souhaitée. Cette étape est répétée autant de fois que nécessaire jusqu'à convergence des paramètres.

Pour utiliser ce type de méthodes, il faut d'abord poser le problème de façon mathématique.

L'objectif est de reconnaître les caractères d'une séquence (Y) présente dans l'image, on les appelle les labels ou étiquettes y_j avec $y_j \in Y$ où $Y = y_1, y_2, ..., y_n$ où n est le nombre de caractères différents dans la séquence. Afin de pouvoir trouver la position des lettres et les reconnaître, dans notre cas, nous avons utilisé ce que l'on appelle une

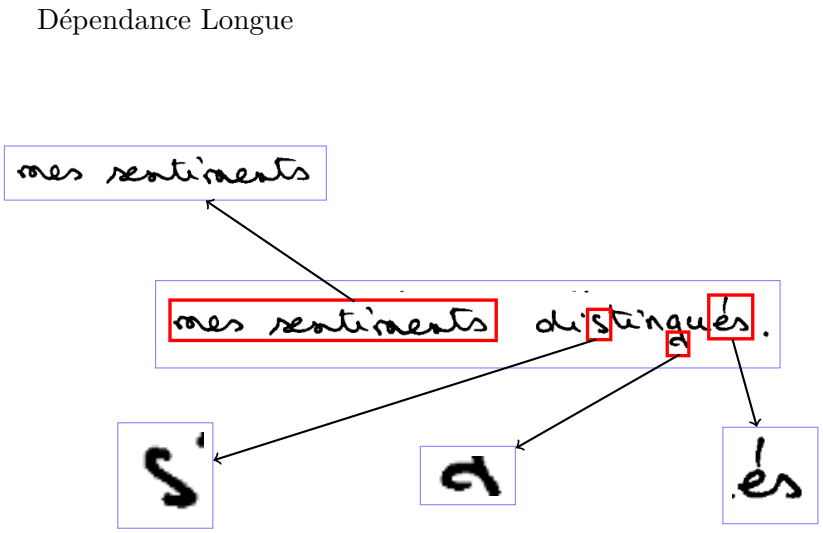


FIGURE 2.3: Schéma des trois niveaux de dépendance dans une image de mots manuscrits.

segmentation implicite c'est-à-dire une approche par fenêtre glissante. Il s'agit d'une division de l'image en un ensemble de sous-images suffisament petites pour qu'une sous-image contienne au maximum un caractère du mot (cf Figure 2.4). Ces sous-images sont appelées trames.

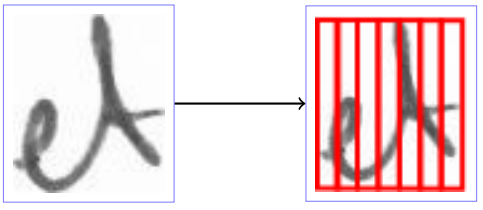


FIGURE 2.4: Schéma explicatif de la segmentation en trames du mot **et**.

Nous avons choisi d'extraire un ensemble de descripteurs (caractéristiques) de chacune de ces trames afin de fournir un représentation numérique compacte de l'image. Ces caractéristiques doivent être le plus robuste possible vis à vis de la variation de l'écriture manuscrite. L'idéal est qu'une lettre ait les mêmes valeurs de caractéristiques quelle que soit sa forme.

Formellement, on dispose des observations $X = (x_1, ..., x_m)$ où m est le nombre de trames dans l'image, où chaque observation x_i est définie par un vecteur de caractéristiques.

On cherche à reconnaître la séquence de caractères située dans l'image, cela revient à modéliser la probabilité que la séquence Y apparaisse sachant l'ensemble des observations disponibles X, soit $P(Y|X)$. Pour modéliser ce problème, il existe un très grand nombre de méthodes. Dans le cadre de nos travaux, nous proposons d'utiliser des modèles statistiques graphiques de séquences. Les différents types et propriétés de ces derniers sont présentés dans la suite de ce chapitre.

2.2.2 Généralités sur les modèles graphiques de séquences

Un modèle graphique de séquences est une notion très visuelle. Afin de donner une définition à cet élément fondamental, nous nous appuyons sur [51, 54–56]. Pour faciliter la visualisation graphique d'une séquence, on la représente dans la majorité des cas par un graphe. Un graphe est un ensemble de nœuds et de liens modélisant les dépendances entre ces nœuds. Un nœud modélise une variable d'un système, que l'on désigne comme un état du modèle. Il est représenté graphiquement sous forme d'un cercle. Les états sont divisés en deux types : les états dits *observables* et *non-observables*. On appelle état observable un état dont on connait la valeur, par exemple un vecteur de caractéristiques extrait d'une image. Un état non-observable ou état caché modélise une variable inconnue (étiquette de la séquence à trouver, variables internes du modèle, etc.). La dépendance entre deux états (cercles) est représentée par un lien joignant les deux cercles. Si le lien est agrémenté d'une flèche, la dépendance modélisée par ce dernier est dîte orientée (cf Figure 2.5). Une dépendance orientée est associée à une dépendance probabiliste entre deux variables, une probabilité conditionnelle (cf Figure 2.5). Si le lien ne contient aucune flèche, la dépendance modélisée est dîte non-orientée (cf Figure 2.14). Elle peut-être caractérisée par un potentiel (noté ici $\phi(X, y)$) lié aux valeurs des deux nœuds connectés par ce lien. Le fait de passer d'un type de lien à l'autre (orienté et non-orienté) impacte fortement la modélisation réalisée par le modèle graphique.

Une clique est un ensemble de noeuds tous connectés entre eux. Une clique est dîte maximale quand on ne peut ajouter de noeud supplémentaire sans compromettre l'interconnectivité de tous les nœuds présents dans la clique et rompre ainsi le postula de départ (interconnectivité de tous les noeuds).

Dans la cadre de nos travaux, nous cherchons à modéliser une séquence d'observations extraite des données fournies. Cette séquence d'observations ou ensemble d'états observables est liée à des réalisations du processus modélisé. Le graphe représente les données disponibles, visibles ou mesurables sur les données d'entrée du système. Le lien entre ces

observations et la modélisation de la séquence n'est pas direct, il est souvent nécessaire de passer par un niveau d'interprétation sous-jacent composé d'états cachés. Cet ensemble d'états cachés correspond à des états dont les valeurs sont déduites des valeurs des états observables et des dépendances/indépendances entre états du modèle. Le graphe intervient alors pour proposer une factorisation de la distribution d'un ensemble X d'états observables et d'un ensemble d'étiquettes Y non observées que l'on souhaite déterminer. Le type de liens (orienté ou non) entre ces deux ensembles définit les deux grandes familles d'outils statistiques utilisés actuellement par la communauté : les modèles orientés correspondant principalement aux méthodes génératives et les modèles non orientés corerspondant aux modèles discriminants.

Dans la suite de ce chapitre, nous présenterons dans le cadre de la reconnaissance d'écriture manuscrite, deux modèles graphiques de séquences, les MMC (Modèle de Markov Cachés) [48] et les CAC (Champs Aléatoires Conditonnels) [50] respectivement génératif et discriminant. Puis nous détaillerons les approches hybrides mélant les avantages des méthodes génératives et discriminantes, notamment le BSLTM-CTC, système actuellement à l'état de l'art en classification de séquence [46].

2.3 Modèles graphiques séquentiel génératifs : Les Modèles de Markov Cachés

Dans cette partie, nous commençons par présenter des généralités sur les modèles graphiques génératifs afin d'introduire les concepts de cette catégorie de modèle. Puis nous détaillerons l'un des modèles graphiques séquentiels génératifs les plus utilisés : les MMC.

2.3.1 Généralités sur les modèles graphiques génératifs

Dans la suite de ce chapitre, une observation définie par une unique valeur est notée χ. Une observation définie par un vecteur d'observation est notée $x = \{\chi_1, ..., \chi_T\}$. Finalement, une séquence d'observations composée d'une séquence de vecteurs d'observations est notée $X = (x_1, ..., x_T)$ où $x_i = \{\chi_{i1}, ..., \chi_{iT}\}$.

2.3.1.1 Modèle génératif élémentaire

Les modèles graphiques génératifs modélisent une observation conditionnellement à l'état caché qui lui est associé. Graphiquement cela se traduit par un lien orienté entre l'état

caché y et l'observation χ (cf Figure 2.5). Ce lien correspond à la probabilité conditionnelle $P(observation|etat_{cache})$. Cette modélisation conduit à écrire la probabilité jointe de l'observation et de l'état caché sous la forme factorisée suivante :

$$P(\chi, y) = P(y)P(\chi|y) \tag{2.1}$$

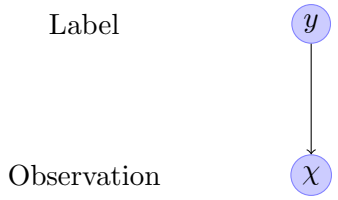


FIGURE 2.5: Schéma d'un modèle génératif élémentaire modélisant le lien entre l'étiquette y à l'observation χ : $P(\chi, y) = P(y)P(\chi|y)$

En général, on est plutôt intéressé par la tâche de classification, c'est-à-dire par le calcul de la probabilité *a posteriori* $P(y|\chi)$. Afin d'obtenir la probabilité voulue on utilise la formule de Bayes :

$$P(y|\chi) = \frac{P(\chi, y)}{P(\chi)} = \frac{P(\chi|y)P(y)}{P(\chi)} \tag{2.2}$$

On voit sur cette exemple simple que la règle de décision de classification conduit à inverser le modèle génératif. Quand on généralise ce modèle à un ensemble d'observations $x = \{\chi_1, .., \chi_n\}$, on parle alors de classifieur Bayésien Naïf.

2.3.1.2 Le classifieur Bayésien Naïf

On dispose de plusieurs observations, on peut alors modéliser le lien entre le vecteur d'observations $x = \{\chi_1, .., \chi_n\}$ et l'état caché y en utilisant un classifieur Bayésien Naïf (cf Figure 2.6).

Selon ce modèle, chaque observation χ_i dépend de l'état caché y :

$$P(y, x) = P(y)P(x|y) = P(y)P(\chi_1, ..., \chi_n|y) \tag{2.3}$$

Les observations χ_i sont supposées indépendantes entre elles conditionnellement à l'état caché y ce qui se traduit sur la graphe par l'absence de lien entre les observations. Ainsi la probabilité d'apparition du vecteur d'observations x conditionné à l'état caché y est

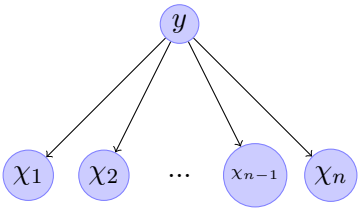


FIGURE 2.6: Schéma du classifieur bayésien naïf modélisant le lien entre l'étiquette y et l'ensemble d'observation $x = \{\chi_1, .., \chi_n\}$: $P(y, x)$

le produit des probabilités élémentaires. Finalement le graphe de la Figure 2.6 permet d'écrire la probabilité jointe $P(y, x)$ selon la formule 2.4.

$$P(x, y) = P(y)P(x|y) = \prod_{i=1}^{n} P(\chi_i|y)P(y) \tag{2.4}$$

On peut alors généraliser ce modèle à une séquence de labels et à une séquence d'observations. Pour cela on utilise les Modèles de Markov Cachés.

2.3.2 Modèles de Markov Cachés : Définition et structure

2.3.2.1 Introduction

Les MMC sont nés en 1966 au sein d'une équipe de recherche dirigée par L.E Baum [57] [58]. Leurs travaux proposaient une méthode de maximisation réalisant l'apprentissage de modèles MMC à partir d'une seule observation. Il faut attendre 1977 et les travaux de Dempster, Laird et Rubin [59] proposant l'algorithme Expectation-Maximization (EM) pour apporter un cadre théorique solide à l'estimation des paramètres de ces modèles à partir d'un seul exemple. La généralisation de cette méthode d'apprentissage à un ensemble de séquences voit le jour en 1983 avec les travaux de Levinson, Rabiner et Sondhi [60]. Dans cet article, les auteurs proposent une méthode d'apprentissage des MMC basée sur l'estimation par maximum de vraisemblance. À partir des années 1980, la popularité des MMC n'a cessé de croitre, leur capacité à modéliser des séquences couplée à leur base théorique solide leur a même permis de s'imposer dans certains domaines d'applications comme la reconnaissance de la parole [48], la reconnaissance de gestes ou encore la modélisation de séquences ADN [61]. Aujourd'hui encore les MMC restent un modèle de référence en reconnaissance de l'écriture manuscrite [33–36].

Les MMC sont des modèles graphiques probabilistes séquentiels et génératifs. Les liens entre les états cachés et les observations (approche générative) et les liens entre les états

cachés (séquentialité) sont orientés. Ces deux ensembles de liens définissent deux processus stochastiques, nous appelerons $q_t \in Y$ le processus représentant l'évolution des états cachés (non-observables) avec $Y = \{y_1, ...y_N\}$ et $x_t \in X$ le processus représentant la séquence d'observations générée par le MMC (vecteur de caractéristiques), avec $X = (x_1, ..., x_T)$. Les probabilités conditionnelles régissant les émissions des observations en fonction des états sont notées : $P(x_t|q_t = y_i)$, la probabilité d'apparition d'une observation x_t sachant que l'état caché q_t est égal au label y_i. Un même modèle peut modéliser des séquences d'observations de longueurs différentes. À chaque instant, l'état caché q_t peut prendre l'une des N valeurs (N classes) possibles. Dans le cas discret $x_t \in \{v_1, ..., v_M\}$ est un ensemble discret de M observations tandis que dans le cas continu $x_t \in \mathbb{R}^M$ où M est la dimension du vecteur de caractéristiques.

Afin d'illustrer le fonctionnement d'un MMC, nous nous appuyons sur deux schémas. Le premier schéma (Figure 2.7) représente la relation entre la séquence d'observations X et la séquences d'états cachés Q au cours du temps.

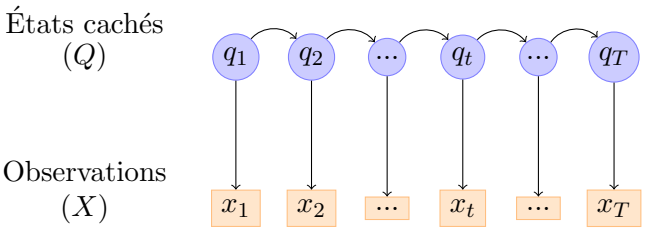


FIGURE 2.7: Schéma d'un modèle de Markov Cachés déroulé dans le temps

Le deuxième schéma (Figure 2.8) représente le modèle graphique du MMC du mot **mot**. Sachant que nous n'avons aucune information sur la durée en nombre de trames de chacune des lettres du mot à reconnaître, il est nécessaire de permettre au système de pouvoir s'aligner sur des plages de données de longueur variable. Chacun des états pourra être choisi plusieurs fois (probabilité d'auto-transition). Ainsi on espère que chaque état sera spécialiste d'un morceau de caractère (graphème) composant le caractère à modéliser (début, milieu, fin du caractère).

2.3.2.2 Définition

Dans leur forme initiale, les MMC considèrent les observations discrètes, des ensembles de symboles. Ils sont utilisés notamment sous cette forme en TAL [62].

Posons q_t une variable discrète, $Y = \{y_1, ..., y_N\}$ l'ensemble d'états cachés de cardinal N et l'ensemble de M symboles observés $V = \{v_1, ..., v_M\}$. On définit la probabilité de

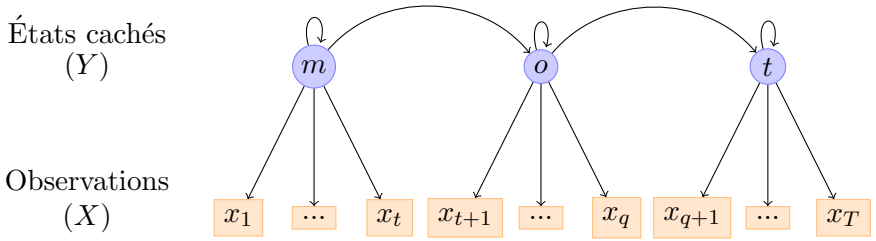


FIGURE 2.8: Schéma d'un modèle de Markov Cachés Gauche-Droite permettant d'effectuer la reconnaissance du mot **mot**

transition de l'état q_{t-1} vers l'état q_t comme la probabilité :

$$a_{ji} = P(q_t = y_i | q_{t-1} = y_j) \forall (i,j) \in Y^2 \tag{2.5}$$

La probabilité que le premier état observé soit l'état y_i est donnée par la probabilité initiale π_i :

$$\pi_i = P(q_1 = y_i) \tag{2.6}$$

En écriture matricielle ces deux ensembles de probabilités peuvent être représentés de la façon suivante :

$$A = \begin{pmatrix} P(y_1|y_1) & \dots & P(y_N|y_1)) \\ \vdots & \vdots & \vdots \\ P(y_1|y_N) & \dots & P(y_N|y_N)) \end{pmatrix} = \begin{pmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \dots & \vdots \\ a_{N1} & \dots & a_{NN} \end{pmatrix} \tag{2.7}$$

$$\Pi = \begin{pmatrix} P(q_1 = y_1) \\ \vdots \\ P(q_1 = y_N) \end{pmatrix} \tag{2.8}$$

Les matrices A et Π sont respectivement appelées matrice de transition et matrice des probabilités intiales. On définit également les probabilités d'émissions de l'ensemble du modèle par une matrice d'émission B de dimension MxN composée des probabilités conditionnelles d'émissions des symboles v_k sachant l'état caché y_i :

$$B_j(k) = P(v_k|q_t = y_i) = \begin{pmatrix} P(v_1|y_1) & \dots & P(v_M|y_1)) \\ \vdots & \dots & \vdots \\ P(v_1|y_N) & \dots & P(v_M|y_N)) \end{pmatrix} \tag{2.9}$$

Lorsque les observations sont binaires, des modèles de Bernouilli peuvent-être utilisés.

En reconnaissance d'écriture, la plupart des travaux considèrent des observations continues car les descripteurs d'images utilisés ont des valeurs réelles [33–40]. Dans ce cas les mélanges de Gaussiennes sont généralement utilisés pour modéliser les probabilités d'émissions des observations.

On associe un mélange de gaussiennes $G = \{g_1, ..., g_n\}$ à chaque état caché y_i du modèle MMC concerné. Ainsi, le mélange modèlise la densité de probabilité conditionnelle de l'apparition de l'observation x_t sachant l'état y_i, $P(x_t|q_t = y_i)$ à l'aide du mélange :

$$P(x_t|q_t = y_i) = \sum_{g=1}^{n} w_{i,g}\mathcal{N}(x_t; \mu_{i,g}, \Sigma_{i,g}) \tag{2.10}$$

où $\mathcal{N}(X; \mu_{i,g}, \Sigma_{i,g})$ est la loi gaussienne de moyenne $\mu_{i,g}$ et de matrice de covariance $\Sigma_{i,g}$ et $w_{i,g}$ représente la probabilité qu'une observation soit produite en moyenne par la g^{ieme} Gaussienne du mélange associée à l'état caché y_i. La matrice de covariance $\Sigma_{i,g}$ contient n paramètres qu'il sera nécessaire d'estimer lors de la phase d'apprentissage du modèle. Dans la pratique, on utilise des matrices Σ diagonales, on réduit ainsi le nombre de paramètres de n^2 à n.

2.3.2.3 Topologie du MMC

Le choix de la topologie du MMC se fait en fonction du problème à traiter. La topologie d'un MMC est structurée par les contraintes des valeurs des transitions entre les états et les probabilités initiales. Il existe plusieurs topologies de MMC, chacune ayant des propriétés particulières. Pour la reconnaissance d'écriture manuscrite, on utilise principalement trois types de topologie : ergodique, gauche-droite et bakis gauche-droite.

Dans les trois prochains schémas, l'état grisé représente l'état non-émetteur d'entrée et de sortie du modèle MMC.

Un modèle ergodique est un modèle MMC sans contrainte au sein duquel toutes les transitions entre états sont possibles : $a_{ij} > 0 \; \forall (i,j) \in [1, N]$. Un exemple d'un tel modèle est donné en Figure 2.9). Il permet de construire toutes les séquences de caractères contenant les lettres *n*, *o* et *m* (nom,mon,omm,ooommmnnnnnn...).

Un modèle gauche-droite est un modèle fortement contraint par les propriétés suivantes :

- $a_{ij} = 0 \; \forall j < i$ seules les transitions vers un état d'indice supérieur à l'état actuel sont autorisées ;
- $a_{ij} = 0 \; \forall j > i + \delta$ il n'est possible de transiter que vers des états dont l'indice est proche (inférieur à $i + \delta$) de celui de l'état actuel ;

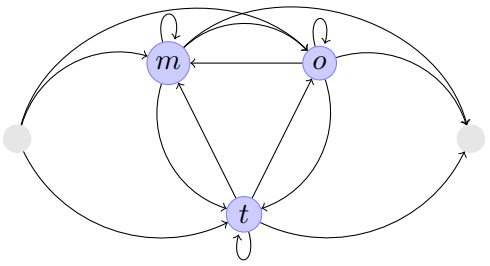


FIGURE 2.9: Exemple d'un MMC ergodique

— Dans le cas d'un modèle gauche-droite classique $\delta = 1$.

— La séquence d'état doit toujours débuter par le premier état du modèle ($\Pi_i = 1$).

Dans la Figure 2.10 et 2.11, ces modèles permettent uniquement de construire le mot *mot*. Le second comporte un saut supplémentaire qui permet d'absorber l'absence d'un caractère dans la séquence (mt,ot).

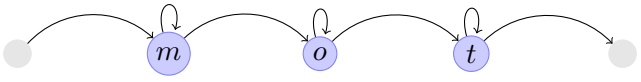


FIGURE 2.10: Schéma d'un MMC gauche-droite

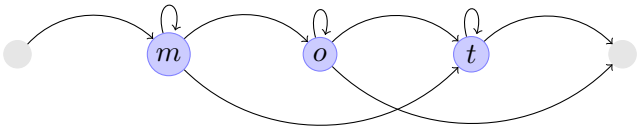


FIGURE 2.11: Schéma d'un MMC de Bakis gauche-droite

Le modèle de Bakis gauche-droite est le plus courant dans la modélisation de données séquentielles et continues comme la reconnaissance de la parole ou de l'écriture. Dans le cas de ce modèle, $\delta = 2$. Il peut être inversé (droite-gauche) si l'écriture se lit dans le sens inverse de l'écriture latine, c'est le cas en arabe par exemple.

2.3.3 Modèles de Markov Cachés : Apprentissage et Décision

Afin d'appliquer les MMC à un problème de reconnaissance d'écriture manuscrite, il faut résoudre les trois problèmes classiques pour l'apprentissage et la décision. Ces derniers ont été formalisés dans [48] :

Problème 1 : Sachant la séquence d'observations $X = (x_1, ..., x_T)$ et le modèle $\lambda = (A, B, \pi)$ comment peut-on calculer $P(X|\lambda)$ la probabilité d'apparition de la séquence d'observations X efficacement ?

Problème 2 : Sachant la séquence d'observations $X = (x_1, ..., x_T)$ et le modèle $\lambda = (A, B, \pi)$ comment trouve-t-on la séquence d'états cachés $Y = (y_1, ..., y_n)$ optimale (qui génère le mieux la séquence d'observations X) ?

Problème 3 : Comment optimiser les paramètres du modèle $\lambda = (A, B, \pi)$ afin de maximiser $P(X|\lambda)$?

Nous donnons maintenant les solutions algorithmiques permettant de résoudre ces trois problèmes. Nous nous appuyerons une fois de plus sur l'article de *Rabiner* [48].

2.3.3.1 Problème 1 : Calcul de la probabilité d'émission d'une séquence

On cherche à calculer la probabilité d'émission de la séquence d'observations X par un modèle λ : $P(X|\lambda)$ où $X = (x_1, ..., x_T)$. Si l'on suppose connue la séquence d'états cachés $Y = (y_1, ..., y_T)$, la probabilité jointe s'écrit alors :

$$P(X, Y|\lambda) = P(X|Y, \lambda)P(Y) \tag{2.11}$$

où $P(X|Y, \lambda)$ est le modèle d'attache aux données et $P(Y)$ le modèle des connaissances. En posant l'hypothèse d'indépendance conditionnelle des observations, on obtient la factorisation de la formule 2.11 qui ne fait intervenir que des termes de la matrice B :

$$P(X|Y, \lambda) = P(x_1|y_1)P(x_2|y_2)...P(x_T|y_T) \tag{2.12}$$

On applique alors l'hypothèse de markov d'ordre 1 pour la dépendance des états entre eux ($p(y_t|y_{t-1}, y_{t-2}, ..., y_1) = p(y_t|y_{t-1})$) :

$$P(Y) = \prod_t P(y_t|y_{t-1}) \tag{2.13}$$

Grâce aux deux hypothèses précédentes, on réécrit l'équation 2.11 en 2.14 :

$$P(X,Y|\lambda) = \pi_{y_1} P(x_1|y_1)P(y_2|y_1)P(x_2|y_2)...P(y_T|y_{T-1})P(x_T|y_T) \tag{2.14}$$

Il existe en tout N^T séquences d'états de longueur T possibles. La probabilité que la séquence d'observations X apparaisse quelle que soit la séquence d'état cachés Y qui l'ait généré s'écrit :

$$P(X) = P(X|\lambda) = \sum_{Y} P(X,Y|\lambda) \tag{2.15}$$

Ce calcul est très coûteux en temps : $2\text{x}T\text{x}N^T$ multiplications car il y a N^T séquences possibles et $2T$ multiplications à effectuer pour chaque séquence. Mais on peut recourir à l'utilisation de la programmation dynamique afin d'en diminuer la complexité.

L'algorithme forward permet de calculer efficacement le score en tenant compte de tous les chemins d'états possibles. Dans cette section nous allons détailler son fonctionnement.

La variable forward $\alpha_t(i)$ s'écrit :

$$\alpha_t(i) = P(x_1 x_2 ... x_t, q_t = y_i|\lambda) \tag{2.16}$$

C'est la probabilité que la séquence partielle $X_t = x_1 x_2 ... x_t$ entraine l'apparition de l'état y_i à l'instant t sachant les paramètres du modèle λ.

Sachant que l'on parcourt l'ensemble des chemins possibles, on remarque que la variable forward s'écrit en fonction des variables forward à l'instant précédent :

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(x_{t+1}) \text{ avec } 1 \leq t \leq T-1 \text{ et } 1 \leq j \leq N \tag{2.17}$$

Cette factorisation permet alors de calculer $P(X|\lambda)$ efficacement selon l'algorithme forward ci-dessous :

Initialisation :

$$\alpha_1(i) = \pi_i b_i(x_1) \text{ avec } 1 \leq i \leq N \tag{2.18}$$

Récurrence :

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(x_{t+1}) \text{ avec } 1 \leq t \leq T-1 \text{ et } 1 \leq j \leq N \tag{2.19}$$

Fin :

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \tag{2.20}$$

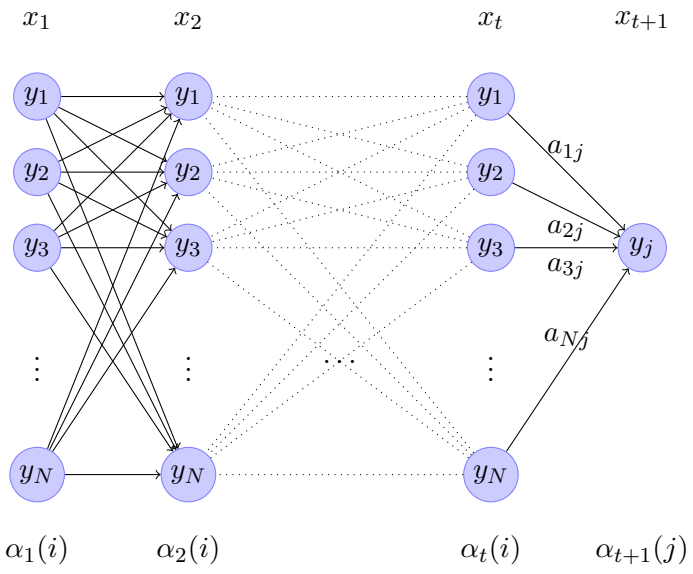


FIGURE 2.12: Description de l'algorithme forward

Cet algorithme a une complexité en $O(TN^2)$ nettement inférieure aux $2\text{x}T\text{x}N^T$ de la formule directe (cf section 2.3.3.1).

Cependant, il se peut que dans certains cas il ne soit pas nécessaire d'avoir les scores de l'ensemble des séquences possibles. Par exemple durant la phase de décodage, où l'on veut que le système nous donne la séquence la plus probable ou si l'on souhaite segmenter la séquence en caractères. On utilise alors l'algortihme de Viterbi [63]. Son fonctionnement est très proche de l'algorithme Forward à la différence près qu'à chaque instant t on n'effectue pas une somme mais une recherche de maximum.

2.3.3.2 Problème 2 : Algorithme de Viterbi

On cherche à déterminer la séquence d'états cachés, c'est à dire trouver la séquence d'états optimale Y^* ayant le plus probablement généré la séquence d'observations X :

$$Y^* = \underset{Y}{argmax}\left[P(Y|X,\lambda)\right] = \underset{Y}{argmax}\left(\frac{P(X,Y|\lambda)}{P(X)}\right) = \underset{Y}{argmax}\left[P(X,Y|\lambda)\right] \tag{2.21}$$

L'algorithme de Viterbi [63] effectue une recherche du meilleur chemin dans l'ensemble des séquences Y possibles. Cet ensemble de séquences est couramment représenté sous forme de treillis (cf Figure 2.13). Contrairement à l'algorithme forward, cet algorithme permet d'identifier la séquence d'états cachés qui maximise la probabilité jointe de la séquence d'observations et la séquence d'états cachés. Chaque déplacement dans le treillis est associé à un coût correspondant à la probabilité de transition entre les deux états concernés. Ces probabilités sont cumulées (multiplication des probabilités entre elles) afin d'obtenir le score global du chemin. Le chemin choisi sera celui qui maximise ce score.

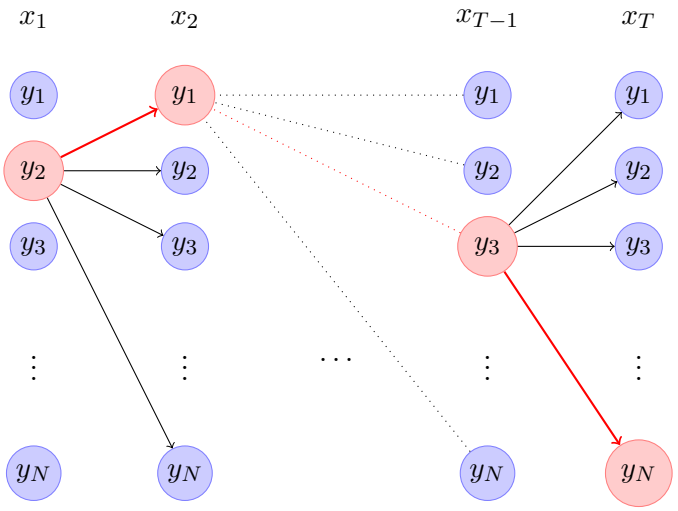


FIGURE 2.13: Schéma représentant le fonctionnement de l'algorithme de Viterbi, en rouge le chemin optimal de viterbi

La recherche du chemin optimal peut se mettre en œuvre par un algorithme de programmation dynamique. En effet, il s'agit de rechercher l'optimum d'une fonction décomposable et monotone. Donc pour résoudre la recherche de l'optimum il suffit d'appliquer une politique de recherche de solutions partielles elles-mêmes optimum car il n'y a pas d'optimum locaux sur la fonction à optimiser.

Posons $\delta_t(i)$, la probabilité du meilleur sous-chemin conduisant à l'état i à l'instant t en ayant pris un certain chemin d'états :

$$\delta_t(i) = max_{q_1 q_2 q_3 ... q_t = y_i}(P(x_1 x_2 x_3 ... x_t, q_1 q_2 q_3 ... q_t = y_i|\lambda)) \tag{2.22}$$

et $\Psi_t(i)$, la variable qui mémorise le chemin localement optimal emprunté :

$$\Psi_t(i) = \underset{j}{argmax}(\delta_{t-1}(j)a_{ji}) \tag{2.23}$$

L'algorithme d'optimisation de Viterbi est alors le suivant :

Initialisation :

- $\delta_1(i) = \pi_i b_i(x_1)$ avec $1 \leq i \leq N$
- $\Psi_1(i) = 0$

Récursion :

- $\delta_t(i) = \underset{j}{max}(\delta_{t-1} a_{ji}) b_i(x_t)$ avec $1 \leq i \leq N$ et $2 \leq t \leq T$
- $\Psi_t(i) = \underset{j}{argmax}(\delta_{t-1} a_{ji})$ avec $1 \leq i \leq N$ et $2 \leq t \leq T$

Terminaison :

- $P* = \underset{j}{max}(\delta_T(j))$
- $y_T^* = \underset{i}{argmax}(\delta_T(i))$

Calcul du meilleur chemin :

- $y_t* = \Psi_{t+1}(y_{t+1}*)$ avec $t = T-1, T-2, ..., 1$

La dernière étape consiste en la recherche du meilleur chemin, elle est couramment appelée *backtrack*. Elle permet d'avoir tous les éléments du chemin optimal à chaque instant t, autrement dit la séquence d'états optimale.

De par sa structure en forme de treillis, l'algorithme de Viterbi peut être utilisé sur des séquences de grandes tailles. En effet, il existe des solutions d'élagage efficaces de façon à réduire la complexité, l'algorithme beam-search [64, 65] est l'un des plus utilisé. Cette solution consiste à faire une sélection de chemins à chaque étape, éliminant ainsi les chemins trop éloignés de la solution optimale (hypothèses localement peu probables) en fonction d'un paramètre de tolérance. Sachant que le chemin optimal global ne correspond pas toujours à la somme des maximums locaux (à chaque instant t), ce paramètre doit être choisi soigneusement pour éviter de supprimer des solutions potentiellement intéressantes.

Maintenant que nous avons résolu les deux premiers problèmes, il nous faut réussir à optimiser les paramètres du modèle $\lambda = (A, B, \pi)$ afin de maximiser $P(O|\lambda)$.

2.3.3.3 Problème 3 : Apprentissage d'un MMC

Apprendre un MMC revient à répondre à la question suivante : Comment déterminer les paramètres du HMM ($\lambda = (A, B, \pi)$) à partir d'un ensemble d'apprentissage étiqueté ?

Pour cela, on cherche à partir de l'ensemble d'exemples de séquences d'observations fourni, à "entraîner" le modèle à générer de mieux en mieux les exemples fournis. Pour aider notre modèle à converger vers une génération de plus en plus performante, on doit

disposer d'un critère permettant de chiffrer l'aptitude du modèle à générer une séquence. On utilise pour cela la vraisemblance :

$$L(\lambda) = \sum_{i=1}^{N} log(P(X^{(n)}, Y^{(n)}|\lambda)) \tag{2.24}$$

où $X^{(n)}$ et $Y^{(n)}$ désignent respectivement la séquence d'observations et la séquence d'états du nième exemple. Sachant que l'on ne connait pas la séquence d'états cachés au niveau trames $Q = q_1 q_2 ... q_T$ (car on ne connaît pas la segmentation en caractères) il nous faut estimer les paramètres en examinant l'ensemble des séquences d'états possibles que l'on peut associer à chaque séquence d'observations. On utilise pour cela l'algorithme de Baum-Welch [66].

L'algorithme Baum-Welch est un algorithme forward-backward EM (Expectation-Maximization), c'est à dire que l'optimisation se fait en deux temps, la partie *Expectation* au cours de laquelle les données manquantes sont estimées, et la partie *Maximization* où les nouveaux paramètres sont calculés en cherchant à maximiser le critère d'apprentissage.

Un algorithme forward-backward permet d'estimer la probabilité marginale que l'état q_t à l'instant t prenne la valeur y_i sachant l'ensemble des observations X : $P(q_t = y_i|X)$. Elle est obtenue par une combinaison des probabilités forward et backward. La probabilité forward, $\alpha_t(i) = P(x_1 x_2 ... x_t, q_t = y_i|\lambda)$ a été défini en section 2.3.3.1. La probabilité backward est la probabilité que l'état q_t prenne la valeur y_i quel que soit le chemin d'états cachés qui sera emprunté après cet état jusqu'à la fin de la séquence :

$$\beta_t(i) = P(x_{t+1} ... x_T, q_t = y_i) = \sum_j \beta_{t+1}(j) a_{i,j} b_j(x_{t+1}) \tag{2.25}$$

avec $\beta_T(i) = 1$.

On parcourt donc l'ensemble des observations deux fois dans deux sens différents : forward ($0 \rightarrow T$) et backward ($T \rightarrow 0$). On obtient ainsi une probabilité locale à contexte très large (toute la séquence) :

$$\gamma_t(i) = P(q_t = y_i|X) = \frac{p(q_t = y_i, X)}{P(X)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} \tag{2.26}$$

Dans la cadre d'un apprentissage MMC cela se déroule de la façon suivante :

Étape E : Estimation des variables manquantes : $\alpha_t(i), \beta_t(i), \gamma_t(i)$

Étape M : Calcul des nouveaux paramètres qui maximisent la vraisemblance :

— $\overline{a_{ij}} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T-1}\alpha_t^{(n)}(i)a_{j,i}b_j(x_{t+1}^{(n)})\beta_{t+1}^{(n)}(i)}{\sum_{n=1}^{N}\sum_{t=1}^{T-1}\alpha_t^{(n)}(i)\beta_t^{(n)}(i)}$

— $\overline{b_i(x)} = \frac{\sum_{n=1}^{N}\sum_{t=1,x_t=k}\gamma_t^{(n)}(i)}{\sum_{n=1}^{N}\sum_{t=1}\gamma_t^{(n)}(i)}$

— $\overline{\pi_i} = \frac{1}{N}\sum_{n=1}^{N}\gamma_t^{(n)}(i)$

— Calcul du critère : $L(\lambda_{j+1}) = \sum_{i=1}^{N} log(P(X^{(n)}, Y^{(n)}|\lambda))$

Boucle : Itérer E puis M tant que la vraisemblance augmente.

Cet algorithme converge vers un optimum local. Comme tout algorithme d'apprentissage, l'algorithme de Baum-Welch est sensible au sur-apprentissage, il est nécessaire de bien contrôler la progression de l'apprentissage à l'aide d'une base de validation afin d'éviter que le modèle produit soit trop proche de la base d'apprentissage.

L'un des avantages majeurs des MMC est qu'ils peuvent être appris suivant une méthode d'apprentissage embarquée. C'est-à-dire qu'ils ne nécessitent pas de connaître la segmentation des observations. C'est le cas par exemple dans la plupart des bases d'écriture manuscrite où l'on dispose de la transcription du mot contenu dans l'image mais où la position des différents caractères n'est pas renseignée. Les modèles de caractères concaténés pour former des modèles de mots permettent aux MMC de s'aligner sur une séquence d'observations du mot correspondant ce qui permet de fournir une estimation statistique de la présence des caractères sur chacune des trames de l'image. Au fur et à mesure de l'avancement de l'apprentissage, cette estimation sera de plus en plus précise jusqu'à convergence de l'algorithme. A la convergence, les modèles de caractères ont été appris sans jamais avoir procédé à la segmentation du mot en caractères.

2.3.3.4 MMC pour la reconnaissance d'écriture manuscrite

Grâce aux mélanges de Gaussiennes, les MMC continus possèdent une haute capacité modélisante leur permettant de classifier des séquences temporelles très variables. Cependant, ils possèdent certaines limites qui peuvent dégrader leurs performances. Outre le fait que le critère d'apprentissage maximise une probabilité jointe ($P(X, Y)$) et non le critère de décision qui sera utilisé au moment de la reconnaissance ($P(Y|X)$), les MMC se basent sur des calculs de vraisemblances qui ne sont pas normalisés et sont donc difficilement interprétables. De plus, la modélisation locale sur une seule observation à chaque instant t (à cause de l'hypothèse d'indépendance des observations) constitue une limite importante pour la modélisation séquentielle car on ne prend en compte qu'un contexte temporel limité. On sait également que la modélisation des données à l'aide de distributions gaussiennes ne permet pas l'usage de vecteurs de caractéristiques de taille trop importante ce qui conduit à une estimation peu précise des probabilités locales d'apparition.

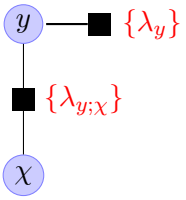


FIGURE 2.14: Schéma d'un modèle discriminant élémentaire liant l'étiquette y et l'état observé χ : $\phi(\chi, y)$.

Des modèles discriminant tels que les Champs Aléatoires Conditionnels (CAC) [49, 50] semblent posséder des propriétés théoriques paliant les limitations des MMC. Ils sont détaillés dans la section suivante.

2.3.4 Modèle graphique séquentiel discriminant : Champs Aléatoires Conditionnels

Dans cette section, nous commençons par présenter des généralités sur les modèles discriminants, puis nous détaillons la régression logistique afin d'introduire la présentation des champs aléatoires conditionnels.

2.3.4.1 Généralités sur les modèles discriminants

Contrairement à leurs homologues génératifs, les modèles discriminants permettent de modéliser les frontières entre les classes du problème donné (dans notre cas les caractères). De ce fait, ils se focalisent directement sur la fonction de décision qui nous intéresse : la probabilité conditionnelle de l'étiquette y sachant l'observation χ, $P(y|\chi)$. La Figure 2.14 représente un modèle graphique discriminant élémentaire, on parle alors de régression logistique élémentaire. Ici le lien entre l'étiquette y et l'observation χ n'est pas orienté, il est caractérisé par une valeur numérique, un potentiel noté $\phi(\chi, y)$ qui est lié aux valeurs des noeuds χ, y et le poids asssocié λ_y avec $\phi(\chi, y) = exp(\lambda_y.\chi)$:

$$P(y|\chi) = \frac{1}{Z(\chi)}\phi(\chi, y)) = \frac{\phi(\chi, y)}{\sum_i \phi(\chi, y_i)} \tag{2.27}$$

Ce modèle élémentaire se généralise aisement au cas à plusieurs observations ($x = \{\chi_1, \chi_2, ..., \chi_n\}$) pour donner le modèle de classification élémentaire, appelé couramment régression logistique représentée par la Figure 2.15. Les liens de ce graphe étant

non-orientés, la caractérisation des dépendances entre l'état y et les observations $x = \{\chi_1, \chi_2, ..., \chi_n\}$ est réalisée par les poids $\lambda_{y,i}$ qui pondèrent chaque arc.

Finalement la contribution de chaque observation à l'état y résulte de la somme des contribution, soit :

$$C_{y,x} = \sum_i \lambda_{y,i} \cdot \chi_i \tag{2.28}$$

On peut dériver cette formule pour se conformer à un modèle multiplicatif et décrire la contribution de chaque observation sous la forme suivante :

$$\phi(y, x) = \prod_i exp(\lambda_{y,i} \cdot \chi_i) \tag{2.29}$$

On déduit alors la probabilité conditionnelle de l'état y par le rapport des fonctions :

$$P(y|x) = \frac{\phi(y, x)}{\phi(x)} = \frac{\phi(y, x)}{\sum_{y'} \phi(y', x)} = \frac{\phi(y, x)}{Z(x)} \tag{2.30}$$

On obtient donc directement la probabilité conditionnelle (modèle discriminant) contrairement aux modèles génératifs qui fournit la probabilité jointe (cf equation 2.4). Afin de simplifier les notations et de faire le lien avec les modèles présentés dans la section suivante, plutôt que de définir des paramètres λ pour chacune des valeurs possibles de l'étiquette y, il est possible de définir des fonctions de caractéristiques $\{f_1, ..., f_K\}$ (*feature functions*) non nulles uniquement pour une seule valeur de cette étiquette, on obtient alors :

$$P(y|X) = \frac{1}{Z(x)} exp(\sum_{k=1}^{K} \lambda_k f_k(y, \chi_k)) \tag{2.31}$$

2.3.4.2 Champs Aléatoire Conditionnels : Définition

Les Champs Aléatoires Conditionnels (CAC) [49, 50] sont la généralisation séquentielle de la régression logistique. Un modèle graphique de type CAC est composé d'un ensemble de régressions logistiques contraintes par une hypothèse de Markov d'ordre 1 (cf Figure 2.16).

À l'inverse du MMC, le CAC modélise directement la probabilité conditionnelle $P(Q = Y|X)$. La probabilité que la séquence d'états $Q = q_1 q_2 ... q_T$ prenne pour valeur une

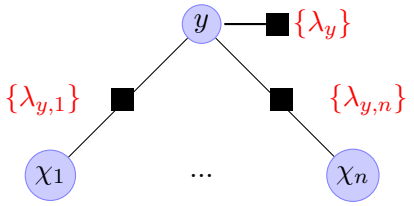


FIGURE 2.15: Schéma d'un modèle de régression logistique élémentaire liant l'étiquette y à l'ensemble d'états observés $x = \{\chi_1, ..., \chi_n\}$: $P(y|x)$

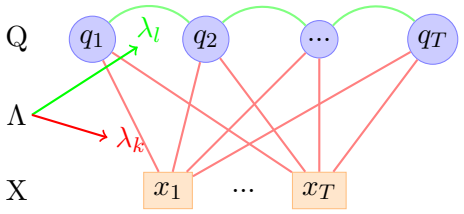


FIGURE 2.16: Représentation graphique d'un Champs Aléatoire Conditionnel

certaine séquence de labels dans l'ensemble $\{y_1 y_2 ... y_n\}$ des labels possibles est donnée par la formule suivante :

$$P(Y|X) = \frac{1}{Z(X)} exp\left(\sum_t \sum_m \lambda_m f_m(X, q_{t-1} = y_j, q_t = y_i)\right)$$

$$P(Y|X) = \frac{exp\left(\sum_t \sum_m \lambda_m f_m(X, q_{t-1} = y_j, q_t = y_i)\right)}{\sum_{y,y'} exp\left(\sum_t \sum_m \lambda_m f_m(X, q_{t-1} = y', q_t = y)\right)} \tag{2.32}$$

Le dénominateur $Z(X) = \sum_{y,y'} exp\left(\sum_t \sum_m \lambda_m f_m(X, q_{t-1} = y', q_t = y)\right.$ agit ici comme facteur de normalisation afin d'obtenir une probabilité. C'est la fonction de partition. On distingue deux types de fonctions de caractéristiques :

— Les fonctions de caractéristiques d'émissions $f(X, q_t = y_i)$, comme pour la régression logistique (cf section 2.3.4.1), matérialisées en rouge sur la Figure 2.16.
— Les fonctions de caractéristiques de transitions $f(q_{t-1} = y_j, q_t = y_i)$ associées aux poids λ_l qui traduisent le modèle markovien d'ordre 1 modélisant la dépendance entre l'occurence de deux états consécutifs q_t et q_{t-1}, matérialisées en vert sur la Figure 2.16.

On peut donc écrire :

$$\sum_{m}^{M} \lambda_m f_m(X, q_{t-1} = y_j, q_t = y_i) = \sum_{k=1}^{K} \lambda_k f_k(X, q_t = y_i) + \sum_{l=K+(i-1).n+1}^{K+i.n} \lambda_l f_l(q_{t-1} = y_j, q_t = y_i) \tag{2.33}$$

où l comptabilise l'ensemble des n transitions depuis chacun des n labels précédents possibles vers le label courant y_i, et où $M = K + n^2$ désigne le nombre total de caractéristiques (de transition et d'émission)

Les poids λ sont les paramètres du modèle, ils régissent la structure du modèle et devront être optimisés au cours de la phase d'apprentissage.

Les fonctions de caractéristiques peuvent être réelles ou binaires, en fonction de la nature des données d'entrée.

Les fonctions de caractéristiques de transitions sont toujours binaires car il y a autant de caractéristiques de transition que de transitions possibles, soit n^2 lorsqu'il y a n labels distincts. Elles sont définies par

$$f_l(q_{t-1} = y_j, q_t = y_i) = \begin{cases} 1 \text{ si } q_{t-1} = y_j \text{ ET } q_t = y_i \\ 0 \text{ sinon} \end{cases} \tag{2.34}$$

où $l = K + 1, \ldots, K + n^2$ On retrouve ici le parallèle avec les fonctions de transitions des MMC et l'hypothèse de Markov d'ordre 1 (cf section 2.3.2).

Concernant les fonctions de caractéristiques d'émissions $f_k(X, q_t = y_i)$, l'indice k permet d'indexer un couple $< y_i, b >$ où y_i correspond à une valeur particulière de l'état q_t et où b est une fonction de description des observations. Cette fonction peut renvoyer une valeur numérique réelle, discrète ou binaire en fonction de la séquence d'observations choisie (complète ou partielle) :

$$f_{i,b}(q_t = y_i, X) = \begin{cases} b(X) \text{ si } q_t = y_i \\ 0 \text{ sinon} \end{cases} \tag{2.35}$$

Dans le cas binaire, la fonction de caractéristique est activée pour un état particulier y_i à la position t dans la séquence et la réalisation d'une condition booléenne sur la séquence d'observations :

$$f_{i,b}(q_t = y_i, X) = \begin{cases} b(X) \text{ si } q_t = y_i \text{ ET } b(X) = \text{"}vrai\text{"} \\ 0 \text{ sinon} \end{cases} \tag{2.36}$$

Un CAC est un modèle discriminant par construction. En effet, un CAC modélise directement la probabilité conditionnelle $P(Y|X)$. L'absence d'hypothèse d'indépendance des observations permet au CAC de prendre des décisions locales avec un contexte très large sur les observations, potentiellement l'ensemble de la séquence d'observations car les fonctions de caractéristiques peuvent correspondre à l'occurence de motifs sur des durées de plusieurs trames. On parle alors de caractéristique n-gram. En TAL par exemple, certains auteurs manipulent des vecteurs de caractéristiques qui s'étendent sur toute la phrase ce qui représente plusieurs milliers de caractéristiques. Les paramètres λ ne sont pas normalisés, de ce fait un poids mal appris faute d'exemples d'apprentissage aura peu d'impact sur la décision finale (valeur proche de 0). De plus, en cas d'apparition d'éléments très discriminants mais peu présents, leurs poids ne pâtiront pas d'une normalisation globale.

2.3.4.3 Champs Aléatoires Conditionnels : Apprentissage

Les paramètres du modèle $\Lambda = \{\lambda_k\}$ sont estimés sur une base d'apprentissage suivant un critère discriminant à maximiser. Le CAC étant discriminant l'objectif est de maximiser le score de la bonne séquence d'états par rapport à toutes les autres séquence possibles (y compris la bonne séquence). Cela se traduit par un ratio entre ces deux scores pour chaque séquence d'apprentissage indicée par n :

$$L(\Lambda) = \sum_{n=1}^{N} log \left[\frac{exp(\sum_{t=1}^{T} \sum_{m=1}^{M} \lambda_m f_m(X^{(n)}, q_{t-1}^{(n)} = y_j^{(n)}, q_t^{(n)} = y_i^{(n)}))}{\sum_{y,y'} exp(\sum_{t=1}^{T} \sum_{m=1}^{M} \lambda_m f_m(X^{(n)}, q_{t-1} = y', q_t = y))} \right] \tag{2.37}$$

L'optimisation du critère $L(\Lambda)$ va conduire à maximiser le score des bonnes séquences d'étiquettes (numérateur) tout en minimisant la somme totale des scores (dénominateur). Le ratio tendra alors vers 1.

Comme indiqué précédement, ce critère est un critère convexe, on peut ainsi appliquer un algorithme classique et efficace de descente de gradient, sans risque de converger vers un minimul local :

$$\begin{aligned} \frac{\partial L(\Lambda)}{\partial \lambda_m} = & \sum_{n=1}^{N} \sum_{t=1}^{T} [f_m(X^{(n)}, q_{t-1}^{(n)} = y_j^{(n)}, q_t^{(n)} = y_i^{(n)}) \\ & - \sum_{y,y'} P(q_t = y, q_{t-1} = y'|X^{(n)}) f_m(X^{(n)}, q_{t-1} = y', q_t = y)] \end{aligned} \tag{2.38}$$

où $P(q_t = y^{(n)}, q_{t-1} = y'^{(n)}|X^{(n)})$ est la probabilité *a posteriori* locale d'obtenir le couple de labels $< y_i, y_j >$ sachant l'observation $x_t^{(n)}$. Cette probabilité s'obtient grâce à une propagation forward-backward, on la note $\gamma_t(i)$:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} \tag{2.39}$$

où la variable forward est définie par :

$$\alpha_t(i) = P(x_1 x_2 ... x_t, q_t = y_i) = \sum_j \alpha_{t-1}(j)\Phi(q_t = y_i, q_{t-1} = y_j, X) \tag{2.40}$$

et où la variable backward est définie par :

$$\beta_t(i) = P(x_{t+1} x_{t+2} ... x_T, q_t = y_i) = \sum_j \beta_{t+1}(j)\Phi(q_t = y_i, q_{t+1} = y_j, X) \tag{2.41}$$

avec

$$\Phi(q_t = y_i, q_{t-1} = y_j, X) = exp\left(\sum_k \lambda_k f_k(X, q_t = y_i) + \sum_l \lambda_l f_l(q_t = y_i, q_{t-1} = y_j)\right) \tag{2.42}$$

Comme le montre ce critère, il est nécessaire d'avoir la vérité terrain au niveau trame (en rouge dans la formule 2.38), ce qui rend l'apprentissage embarqué impossible. En effet, les CAC ne disposent pas de modèles de durées comme dans les MMC leur permettant de passer d'une vérité terrain de haut niveau à une vérité terrain de niveau trames. Dans l'équation 2.38, on observe l'opposition entre le score de la bonne séquence d'étiquettes et ceux des autres séquences. Cette dernière est propre aux méthodes discriminantes, elle est toujours visible après la dérivée (en rouge les termes relatifs aux étiquettes de la bonne séquence opposés à la partie bleue composée des termes relatif aux étiquettes de l'ensemble des séquences d'étiquettes possibles). Ce problème est convexe à grande dimension. [67] et [68] proposent l'algorithme L-BFGS pour le résoudre. Cet algorithme est dit quasi-Newton car il s'agit d'un algorithme de descente de gradient où le calcul et le stockage de l'inverse de la matrice Hessienne sont approximés afin de pouvoir travailler en grandes dimensions.

Une alternative est d'utiliser des algorithmes de descente de gradient dits stochastiques (SGD). Proposé par [69], cette méthode effectue la mise à jour des paramètres du modèle à partir d'un sous ensemble de la base d'apprentissage tiré aléatoirement. Chaque

itération est donc beaucoup moins coûteuse en temps de calcul et on observe une convergence plus rapide vers une solution proche de l'optimale, le hasard permettant de gagner en généralisation.

Dans un papier de 1999 [70], les auteurs proposent un terme de régularisation gaussien pour les méthodes à entropie maximum, typiquement les CAC. Ce terme de régularisation est basé sur une norme Euclidienne des paramètres pondérée par le terme $1/2\sigma^2$ afin de contrôler l'amplitude de cette pénalité :

$$L(\Lambda) = \sum_{n=1}^{N} log \left[\frac{exp(\sum_{t=1}^{T} \sum_{m=1}^{M} \lambda_m f_m(X^{(n)}, q_{t-1}^{(n)} = y_j^{(n)}, q_t^{(n)} = y_i^{(n)}))}{\sum_{y,y'} exp(\sum_{t=1}^{T} \sum_{m=1}^{M} \lambda_m f_m(X^{(n)}, q_{t-1} = y', q_t = y))} \right] - \sum_m \frac{\lambda_m^2}{2\sigma^2 ||\Lambda||^2} \tag{2.43}$$

Les CAC sont des outils combinatoires extrêmement performants, capables de passer à l'échelle en très grandes dimensions. Ils sont très utilisés dans le domaine du TAL pour résoudre des problèmes de coréférence [71], la reconnaissance d'entités nommées [72] et plus généralement l'extraction d'information [73]. Leur capacité discriminante et leur aptitude à prendre en compte des dépendances à contexte large permet de prendre des décisions locales en ayant observé (en théorie) toute la séquence d'observations. Cet ensemble de propriétés semble permettre au CAC de traiter des données très bruitées et variables comme l'écriture manuscrite.

2.3.4.4 Espace de représentation des caractéristiques

Lorsque l'on veut traiter un problème de reconnaisance d'écriture manuscrite les descripteurs utilisés sont dans la plupart des cas réels. Or les CAC dans leur forme orginale [50] ne sont pas capables de modéliser correctement les dépendances entre des variables réelles car leur critère d'apprentissage est basé sur une somme pondérée des fonctions de caractéristique. Si la fonction de caractéristique prend ses valeurs dans un intervalle réel large alors il se peut que cette caractéristique ait un pouvoir discriminant faible. En effet, on ne peut faire évoluer la valeur du poids sur un sous-intervalle de cette caractéristique. C'est pourquoi au sein des systèmes traitant des données réelles, les CAC sont souvent utilisés pour modéliser les dépendances entre les sorties (probabilités *a posteriori* d'apparition des classes) d'un premier étage qui fournit des décisions locales (réseaux de neurones, réseaux de neurones récurrents, etc), comme c'est le cas en traitement automatique de la parole [74, 75]. Les CAC n'ont pas la capacité d'embarquer des informations de haut niveau comme des lexiques ou des modèles de langage comme c'est le cas pour les MMC. L'autre limite majeure des CAC est la nécessité d'avoir une vérité terrain niveau trame pour effectuer l'apprentissage du modèle comme le montre

l'équation 2.38. On ne peut donc pas appliquer un apprentissage embarqué comme pour les MMC.

Afin de palier cette série de problèmes, une évolution des CAC a été proposée dans [52], les Champs Aléatoires Conditionnels Cachés (CACC). Il s'agit d'un CAC muni d'une couche d'états cachés permettant ainsi d'avoir un niveau d'abstraction supplémentaire dans la modélisation. On parvient ainsi à modéliser un espace de caractéristiques continues directement sans avoir à passer pour une étape de discrétisation. Cette méthode est décrite dans la section suivante.

2.3.4.5 Champs Aléatoires Conditionnels Cachés : Définition

En 2007, Quattoni [52] propose un modèle de Champs Aléatoires Conditionnels Cachés : un modèle graphique discriminant alliant un CAC classique à une couche cachée comme dans les MMC (cf Figure 2.17).

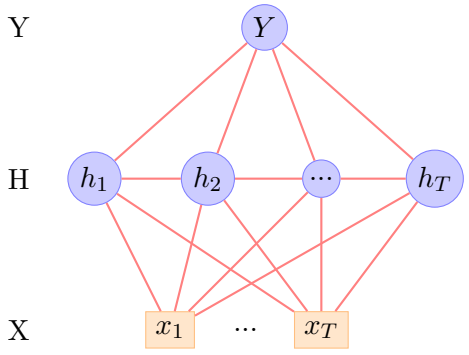


FIGURE 2.17: Représentation graphique d'un CACC

Un CACC modélise la distribution $P(Y, H|X, \Lambda)$ où Y est la classe recherchée, $H = (h_1 h_2, ..., h_T)$ est la séquence d'états cachés intermédiaires et $X = (x_1, x_2, ..., x_T)$ la séquence d'observations. Contrairement aux MMC, les paramètres du modèle sont entrainés de façon discriminante sur l'optimisation de la probabilité d'apparition de la classe désirée $P(Y|X)$.

Durant l'apprentissage, les paramètres $\Lambda*$ vont être estimés suivant un critère discriminant classique [49, 50]. Le but est toujours de maximiser la probabilité d'apparition de l'étiquette recherchée y sachant un ensemble d'observation X et les paramètres du modèle Λ $(\underset{Y}{argmax}(P(Y^{(n)}|X^{(n)}, \Lambda*)))$:

$$L(\Lambda) = \sum_{n=1}^{N} log(P(Y^{(n)}|X^{(n)}, \Lambda)) \tag{2.44}$$

On recherche alors à l'aide d'une méthode de descente de gradient les paramètres optimaux maximisant la fonction objectif $L(\Lambda)$: $\Lambda* = \underset{\Lambda}{argmax}(L(\Lambda))$ où $\Lambda =< \lambda_k, \lambda_l >$. L'ajout d'états cachés complique le problème d'optimisation en le rendant non-convexe, c'est pourquoi la recherche du meilleur minimum local se fait à partir de plusieurs initialisations aléatoires. Comme pour un CAC, un CACC est composé de deux types de fonctions de caractéristiques :

Fonctions de caractéristiques d'émissions : $f_k(Y, h_t, X)$

Fonctions de caractéristiques de transitions : $f_l(Y, h_t, h_{t-1}, X)$

Par rapport aux CAC classiques, on voit que les fonctions de caractéristiques possèdent une dépendance supplémentaire liée aux états cachés. On définit Φ les fonctions de caractéristique globales telles que :

$$\Phi(y, h, X; \Lambda) = \sum_{t=1}^{T} \sum_{k \in K} \lambda_k f_k(Y, h_t = s, X) + \sum_{(s,s') \in E} \sum_{l \in L^2} \lambda_l f_l(Y, h_{t-1} = s', h_t = s, X) \tag{2.45}$$

La fonction objectif devient alors :

$$L(\Lambda) = \sum_{n} log(P(Y^{(n)}|X^{(n)}, \Lambda)) = log\left(\frac{\sum_h exp(\Phi(Y^{(n)}, h, X^{(n)}; \Lambda))}{\sum_{Y',h} exp(\Phi(Y', h, X^{(n)}; \Lambda))}\right) \tag{2.46}$$

Si on dérive cette fonction objectif par rapport aux paramètres λ_l (paramètres des fonctions de caractéristiques d'émissions) pour une seule séquence, on obtient :

$$\begin{aligned}
\frac{\partial L(\Lambda)^{(n)}}{\partial \lambda_k} &= \sum_h (h|Y^{(n)}, X^{(n)}, \Lambda) \frac{\partial \Phi(Y^{(n)}, h, X^{(n)}; \Lambda)}{\partial \lambda_k} - \sum_{Y',h} P(Y', h|X^{(n)}, \Lambda) \frac{\partial \Phi(Y', h, X^{(n)}; \Lambda)}{\partial \lambda_k} \\
&= \sum_h P(h|Y^{(n)}, X^{(n)}, \Lambda) \sum_{t=1}^{T} f_k(Y^{(n)}, h_t, X^{(n)}) \\
&\quad - \sum_{Y',h} P(Y', h|X^{(n)}, \Lambda) \sum_{t=1}^{T} f_k(Y', h_t, X^{(n)}) \\
&= \sum_{t,s} P(h_t = s|Y^{(n)}, X^{(n)}, \Lambda) f_k(Y^{(n)}, h_t = s, X^{(n)}) \\
&\quad - \sum_{Y',s} P(h_t = s, Y'|X^{(n)}, \Lambda) f_k(Y', h_t = s, X^{(n)})
\end{aligned} \tag{2.47}$$

De même pour les paramètres λ_l (paramètres des fonctions de caractéristique de transitions) :

$$\begin{aligned}
\frac{\partial L(\Lambda)^{(n)}}{\partial \lambda_l} &= \sum_{j,k,a,b} P(h_j = a, h_k = b|Y^{(n)}, X^{(n)}, \Lambda) f_l(j, k, a, b, X^{(n)}) \\
&\quad - \sum_{Y',j,k,a,b} P(h_j = a, h_k = b, Y'|X^{(n)}, \Lambda) f_l(j, k, Y', a, b, X^{(n)})
\end{aligned} \tag{2.48}$$

On observe que les dérivées des fonctions objectifs d'un CACC sont des fonctions objectifs d'un CAC (terme en rouge) avec un terme de propagation des états cachés (termes en bleu). Les CACC étant des modèles discriminants, on conserve toujours l'opposition entre les termes de la bonne séquence et les termes représentant toutes les séquences possibles du problème 2.46.

Le problème de ce type d'approche reste la lourdeur du processus d'apprentissage. La structure étant devenue très complexe en raison des dépendances importantes entre les observations et les états, les auteurs préfèrent limiter la complexité des dépendances [76] pour que les temps de calcul soient raisonnables. On revient à des structures proches d'un MMC avec une observation conditionnée par un seul état caché. On se prive ainsi de la possibilité de prendre en compte le contexte au niveau de la séquence d'observations et on se retrouve avec une partie des limites des MMC (cf section 2.3.2).

Une approche différente a été proposée dans les années 1990, les modèles hybrides [77]. Ces derniers sont des systèmes à deux étages composés d'une méthode discriminante et d'une méthode générative. Ainsi, on bénéficie des avantages des deux méthodes sans en subir les inconvénients. Ces approches hybrides sont détaillées dans la section suivante.

2.4 Modèles hybrides pour la reconnaissance d'écriture manuscrite

2.4.1 Introduction

Le début des années 90 voit apparaître les premières méthodes hybrides combinant les approches génératives (MMC, Automates) et les approches discriminantes (Réseaux de neurones, Réseaux de neurones récurrents, etc.). En général, le modèle discriminant a pour but d'analyser les informations locales et fournit des probabilités a posteriori locales à l'étage génératif. Ce dernier embarque des connaissances de haut niveau comme un lexique ou un modèle de langage. Ces systèmes furent d'abord utilisés pour la reconnaissance de la parole [78–81], puis en reconnaissance de l'écriture manuscrite [42, 82–84].

Récemment, l'architecture BLSTM/CTC proposée dans [46] a considérablement augmenté les performances des systèmes de reconnaissance d'écriture [47]. Depuis les années 90, il semble que les modèles hybrides soient l'alternative la plus efficace pour la classification de séquences. Dans la plupart des cas, le premier étage est composé d'une méthode de type réseaux de neurones récurrents (RNN) ou feedforward (RNA) couplé avec une étage modélisant (MMC ou programmation dynamique comme le CTC) [47, 78–82]. Il existe également certains travaux utilisant les SVM comme alternative aux réseaux de neurones [85, 86]. Dans cette section, nous commençons par faire une rapide introduction sur les réseaux de neurones puis nous présentons les premiers modèles hybrides composés d'un réseau de neurones et d'un MMC. Par la suite, nous détaillons l'architecture BLSTM/CTC, actuellement méthode à l'état de l'art en classification de séquences.

2.4.2 Réseaux de Neurones Artificiels/Modèles de Markov Cachés

2.4.2.1 Réseaux de Neurones Artificiels

Un réseau de neurones articifiels (RNA) [87, 88] est un classifieur statique qui se représente sous la forme de couches successives complètement interconnectées. Chaque sortie de neurone de la couche i est reliée à toutes les entrées des neurones de la couche $i+1$ (cf Figure 2.18). On distingue trois types de couches :

La couche d'entrée : Le vecteur d'observations ;

La/les couche(s) cachée(s) : L'ensemble des neurones et leurs poids associés ;

La couche de sortie : L'ensemble des classes (les caractères en reconnaissance d'écriture) du problème à traiter.

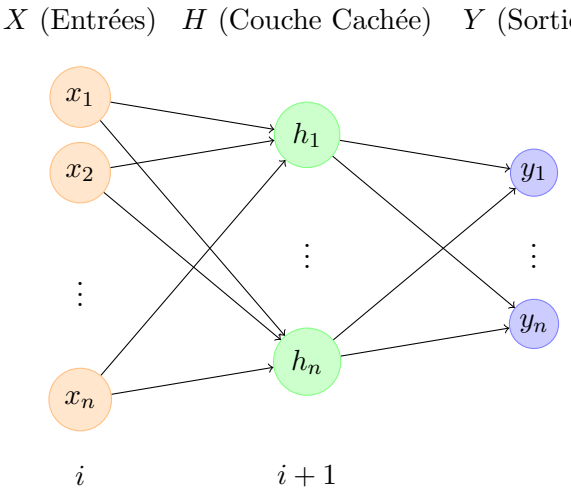


FIGURE 2.18: Représentation graphique d'un réseau de neurones articiels feed-forward

Un neurone n de la couche t est donc connecté à un neurone j de la couche $t+1$ par une liaison pondérée par w_{ji}. Ces poids sont les paramètres du modèle qui seront appris suivant une méthode de rétropropagation du gradient [42]. Chaque neurone est décrit par une fonction de transfert ou fonction d'activation ($f(x_n^t)$), généralement une sigmoïde. En sortie de chaque neurone, on obtient une somme pondérée des entrées de ce dernier $x_n^h = \sum_{j=1}^{J}(z_j^{h-1}w_{nj})$ sur laquelle on applique la fonction de transfert :

$$f(x_n^h) = f(\sum_{j=1}^{J} z_j^{h-1}w_{nj}) \tag{2.49}$$

La fonction de transfert doit être choisie en fonction du problème à traiter (linéaire, non-linéaire, ...). Plus le nombre de couches d'un réseau de neurones artificiels est élevé plus il est capable de résoudre des problèmes complexes mais plus la phase d'apprentissage est difficile. Au-delà de 3 couches cachées, on parle de réseaux de neurones profonds. A partir de cette limite une simple méthode de rétropropagation du gradient n'est plus suffisante pour entraîner le réseau, on observe le phénomène de *perte du gradient* [89]. L'apprentissage se fait alors en deux temps, un premier apprentissage non-supervisé pour apprendre les couches basses puis un apprentissage supervisé pour les couches hautes. Quel que soit le nombre de couches utilisé, les RNA passent très bien à l'échelle même en très grandes dimensions. Cependant, ces structures souffrent d'un nombre d'hyper-paramètres élévé (nombre de couches, nombres de neurones sur chaque couche, pas d'apprentissage,etc.). Très peu de méthodes sont proposées pour déterminer ces hyper-paramètres. Ils devront être estimés par l'expérimentation sur une base de validation. Dans le cadre d'applications concrètes comme la reconnaissance de l'écriture manuscrite,

les RNA sont souvent couplés avec des étages de modélisation séquentielle comme les MMC. Les mélanges de gaussiennes internes aux MMC sont remplacés les RNA. Ce couplage permet à la fois de palier les lacunes de modélisation temporelle des RNA et les lacunes en discrimination des modèles génératifs (Mélanges de Gaussiennes).

2.4.2.2 Inférence au sein d'un système hybride

Bourlard et al. [80, 81] proposent un système hybride RNA/MMC pour la reconnaissance de la parole. Dans ce système le réseau de neurones doit estimer les probabilités d'apparition des états du HMM, nous appelerons H l'ensemble des séquences possibles d'états cachés, Y la séquence d'étiquettes et X la séquence d'observations. La fonction objectif doit maximiser la probabilité d'apparition de la séquence d'étiquettes $Y = (y_1, ..., y_Q)$ sachant la séquence d'observations $X = (x_1, ..., x_T)$:

$$\begin{aligned} P(Y|X) &= \sum_H P(H, Y|X) \\ &= \sum_H P(H|X)P(Y|H, X) \\ &= \sum_H P(H|X)P(Y|H) \\ &= \sum_H P(h_1|X)P(h_2|X, h_1)...P(h_L|X, h_1, ..., h_{L-1})P(Y|H) \\ &= \sum_H \left\{ \prod_{l=1}^{L} P(h_l|X, H) \right\} P(Y|H) \end{aligned} \tag{2.50}$$

Ce critère peut-être séparé en deux parties :

Bas-niveau : lien entre les observations et les états cachés, $P(h|X)$

Haut-niveau : lien entre les états cachés et la séquence d'étiquettes, $P(Y|H)$

La partie bas-niveau sera entièrement gérée par le réseau de neurones qui estimera les probabilités *a posteriori* nécessaires, liant ainsi les observations et les états cachés. La partie haut-niveau s'appuiera sur les outils de modélisation qu'offrent les MMC, notamment les modèles de langages, liant ainsi la séquence d'états cachés à la séquence d'étiquettes cibles respectant la topologie choisie (généralement gauche-droite). L'architecture complète est représentée sur la figure 2.19

Ce type de modèle a également été proposé pour la reconnaissance d'écriture manuscrite [90].

Afin d'augmenter les performances de ces architectures, les réseaux de neurones classiques peuvent être remplacés par des réseaux de neurones récurrents [91].

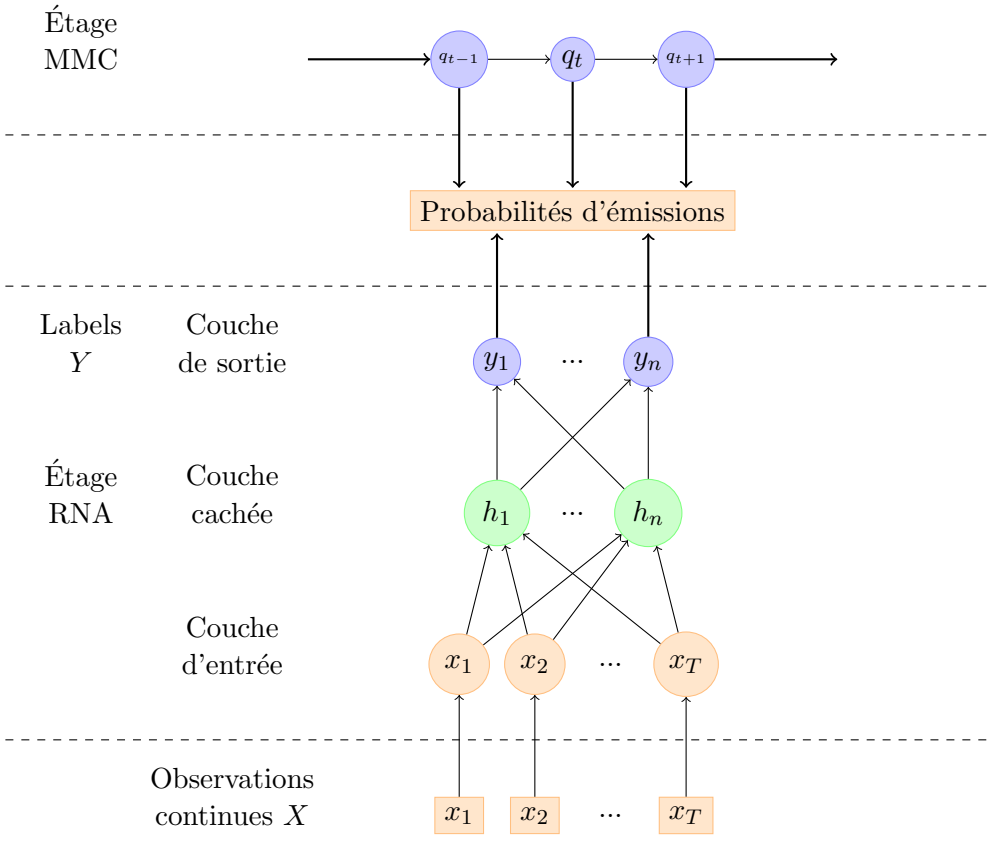


FIGURE 2.19: Représentation graphique d'un système hybride RNA-MMC

2.4.2.3 Réseaux de Neurones Récurrents

Les réseaux de neurones récurrents sont des réseaux de neurones intégrant une mémoire [91]. Une couche de neurones est appelée récurrente quand ses sorties à l'instant $t-1$ sont utilisées comme entrées à l'instant t par cette même couche. La somme pondérée des neurones est donc modifiée :

$$z_n^{h,t} = \sum_{j=1}^{J}(z_j^{h-1,t} w_{nj}) + \sum_{r=1}^{R}(z_r^{h,t-1} w_{nr}) \tag{2.51}$$

où J est le nombre de neurones de la $h-1$ et où R est le nombre de neurones de la couche h à l'instant $t-1$. Cette récurrence permet de prendre en compte la composante temporelle de la séquence d'observations à étudier, améliorant ainsi les performances. Il existe deux façons principales d'entraîner de tels réseaux :

1. La rétropropagation temporelle (Back Propagation Through Time, BPTT) [92]
2. La rétropropagation en temps réel (Real Time Back Propagation, RTBP)[93]

Comme pour les réseaux profonds, quelle que soit la méthode d'apprentissage employée, on observe le phénomène dit de *gradient vanishing* [94, 95]. On voit alors une diminution très rapide de l'erreur durant la rétropropagation, les poids appris ont tendance à prendre en compte le contexte récent et non le contexte lointain ($> 10t$).

Pour palier ce problème, Hochreiter et al [96] proposent l'utilisation de neurones particuliers appelés neurones LSTM (Long short Term Memory).

2.4.3 BLSTM/CTC

Un neurone LSTM (cf Figure 2.20[94]) est composé d'une unité de mémoire, de quatre portes (d'une entrée, une porte d'entrées, une porte d'oubli et une porte de sortie) et de trois opérateurs.

La porte d'entrée contrôle l'influence du signal entrant sur la mémoire. Un signal considéré comme non-pertinent par cette dernière générera une valeur proche de 0, et n'activera pas l'état interne de la cellule. La porte d'oubli contrôle l'influence de l'état précédent de la mémoire sur l'état courant de la cellule. En fonction des sorties de cette porte, la prise en compte du contexte passé peut être conservée ou remise à 0. La porte de sortie permet de contrôler l'influence du neurone courant sur la sortie courante et les sorties suivantes. Un neurone peut donc si nécessaire ne pas mettre à jour sa sortie.

Posons $H = \{h_1, .., h_n\}$ l'ensemble des unités LSTM dans l'architecture étudiée. Ci-dessous les formules correspondant aux 3 portes de contrôle et à la cellule centrale :

1. Porte d'entrée :

$$a_e^t = \sum_{i=1}^{l} w_{i,e} x_i^t + \sum_{h=1}^{H} w_{h,e} b_h^{t-1} + \sum_{c=1}^{C} w_{c,e} s_c^{t-1} \tag{2.52}$$

$$b_e^t = f(a_e^t) \tag{2.53}$$

2. Porte d'oubli :

$$a_f^t = \sum_{i=1}^{l} w_{i,f} x_i^t + \sum_{h=1}^{H} w_{h,f} b_h^{t-1} + \sum_{c=1}^{C} w_{c,f} s_c^{t-1} \tag{2.54}$$

$$b_f^t = f(a_f^t) \tag{2.55}$$

3. Cellule :

$$a_c^t = \sum_{i=1}^{l} w_{i,c} x_i^t + \sum_{h=1}^{H} w_{h,c} b_h^{t-1} \tag{2.56}$$

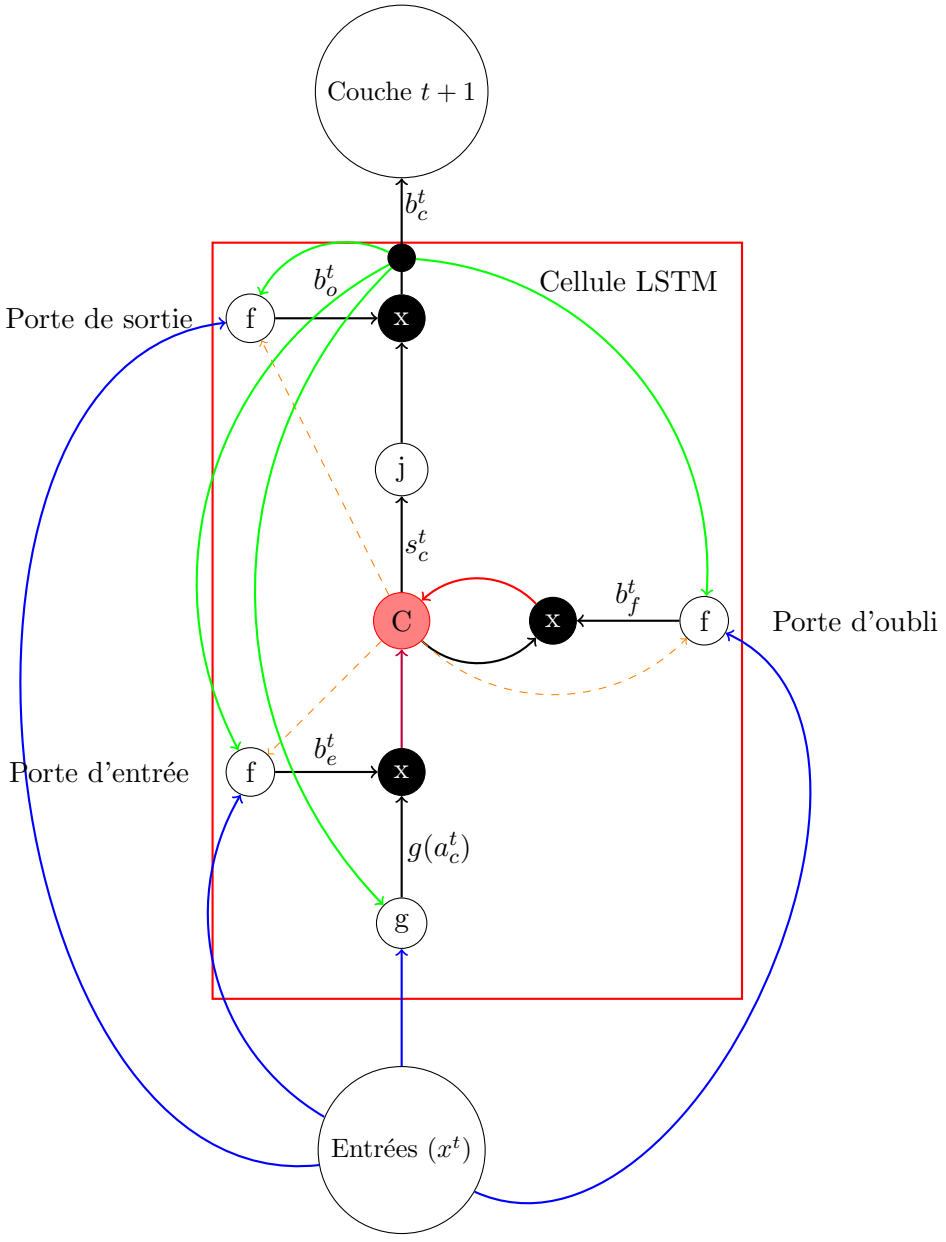


FIGURE 2.20: Représentation graphique d'un neurone LSTM[94]

$$s_c^t = b_f^t s_c^{t-1} + b_e^t g(a_c^t) \tag{2.57}$$

4. Porte de sortie :

$$a_o^t = \sum_{i=1}^{l} w_{i,o} x_i^t + \sum_{h=1}^{H} w_{h,o} b_h^{t-1} + \sum_{c=1}^{C} w_{c,o} s_c^{t-1} \tag{2.58}$$

$$b_o^t = f(a_o^t) \tag{2.59}$$

$$b_c^t = b_o^t j(s_c^t) \tag{2.60}$$

Malgré la complexité de ces neurones, l'apprentissage d'un réseau LSTM s'effectue selon une méthode BPTT comme pour les réseaux récurrents classiques.

Récemment, Graves et al. [97] proposent une amélioration des LSTM, les BLSTM (Bidirectional LSTM). Ces réseaux combinent deux réseaux de neurones LSTM. L'un parcourt le signal dans le sens chronologique des données et l'autre dans le sens antéchronologique. Les deux sorties sont combinées dans un étage CTC (cf section 2.4.3.1). En exploitant les données dans les deux sens, on dispose ainsi du contexte gauche et droite.

Cependant, ces réseaux n'ont pas la capacité d'embarquer des informations de haut niveau comme un lexique par exemple, Graves et al proposent l'ajout d'un étage de programmation dynamique le CTC (Connectionnist Temporal Classification) [46] pour remplir cette fonction.

2.4.3.1 Programmation dynamique : CTC

Cette couche CTC est une couche de neurones de type *SoftMax* ou regression logistique (comme dans un CAC). Cette fonction de tranfert particulière permet d'obtenir une sortie probabilisée. Elle peut également intégrer des informations de haut niveau comme un lexique ou un modèle de langage. Dans le cadre de la reconnaissance d'écriture, chaque neurone de cette couche représente un caractère présent dans le problème à traiter. La particularité de ce système est d'ajouter un caractère supplémentaire, le caractère *joker* ou *blanc* qui représente une absence de décision. Cela signifie que le système n'a pas assez d'informations pour décider correctement. L'ajout de ce symbole supplémentaire évite au système de prendre une décision erronée dans des situations ambiguës.

Les sorties du BLSTM sont normalisées par la fonction *SoftMax* des neurones du CTC puis un opérateur de simplification des chemins est appliqué. Ce dernier supprime les absences de décisions et supprime les lettres identiques consécutives :

```
a,a,_,_,_,_,b,b,b,_,_,c,c,_ => a,b,c
```

Finalement, un alignement de la séquence obtenue sur un lexique peut-être effectué à l'aide d'un algorithme de programmation dynamique classique. Il est même possible de développer un décodage en intégrant un modèle de phrase sous la forme d'un modèle statistiques de langage (n-gram). On a ainsi un système complet de reconnaissance de caractères extrêmement performant [1, 47]. Ces performances s'expliquent par la prise en compte du contexte bidirectionnel et par la capacité du système à rejeter des décisions peu fiables. Les décisions locales sur un caractère (non joker) sont peu fréquentes mais très fortes (généralement une probabilité supérieure à 0.90).

2.5 Conclusion

Au cours de cette section nous avons mis en avant les oppositions entre les méthodes génératives et discriminantes.

D'un coté, les méthodes génératives possèdent une grande capacité modélisante. Elles sont capables d'intégrer des informations de haut niveau comme un lexique ou un modèle de langage. Ces méthodes ont fait leurs preuves dans bien des domaines notamment la reconnaissance d'écriture et la reconnaissance de la parole. Leur caractère dynamique due à la présence d'états cachés ainsi que de sauts et d'auto-transitions leurs permettent d'aligner une séquence d'observations sur une séquence d'étiquettes de longueurs différentes. Elles permettent aussi l'utilisation d'algorithmes de type EM pour l'apprentissage qui sont moins longs que leurs homologues discriminants. Leurs limites se situent dans le cadre théorique puisque l'hypothèse d'indépendance des observations limite la prise en compte de contexte lors de la décision. De plus, le critère d'apprentissage génératif maximise une probabilité jointe ce qui fait que ces modèles ne sont pas entraînés pour faire de la classification.

Les modèles discriminants s'affranchissent de l'hypothèse d'indépendance permettant ainsi la prise en compte de contexte plus large. Leur critère d'apprentissage étant discriminant, ils modélisent directement la probabilité *a posteriori* $P(Y|X)$. Les décisions locales sont ainsi plus robustes.

Dans ces travaux, nous proposons un système hybride CAC/HMM pour traiter une tâche de reconnaissance de mots isolés manuscrits multi-scripteurs. Nous utilisons les CAC pour modéliser les informations de bas-niveau issues de l'extraction de caractéristiques de l'image à traiter. Nous profiterons ainsi de leurs capacités à modéliser des dépendances à long terme. L'étage MMC modélisant embarque des informations de haut niveau

(lexique, modèle de langage, ...) afin de corriger le premier étage. La combinaison de ces deux systèmes permet d'effectuer une tâche de reconnaissance d'écriture manuscrite à l'aide d'un CAC, architecture qui à notre connaissance n'a jamais été proposée au sein de la communauté. Les travaux récents, notamment l'architecture BLSTM/CTC [47] montrent que la solution se trouve dans la combinaison de ces deux méthodes. La majorité des travaux utilisent des réseaux de neurones au bas-niveau, nous avons voulu explorer une alternative, les CAC.

Dans le prochain chapître, nous présentons ce système hybride pour une tâche de reconnaissance de mots isolés manuscrits multi-scripteurs.

Chapitre 3

Un modèle hybride CAC/MMC pour la reconnaissance de mots manuscrits isolés

3.1 Introduction

La reconnaissance d'écriture manuscrite multi-scripteurs est un problème complexe dû à l'importante variabilité de l'écriture. La méthode probabiliste la plus couramment utilisée pour effectuer cette tâche reste encore à ce jour les MMC. Initialement entraînés selon un critère génératif basé sur la maximisation de la vraisemblance [98], ces modèles génératifs ont été adaptés avec succès à un apprentissage discriminant basé sur la maximisation de l'information mutelle (MMI) [99] entraînant des améliorations de performances. Cependant, ces modèles souffrent de l'hypothèse d'indépendance des observations et ne sont pas adaptés aux traitements de vecteurs de caractéristiques de grandes dimensions.

Depuis une dizaine d'années, les CAC [50] sont de plus en plus utilisés en modélisation séquentielle car ils ne se basent pas sur cette hypothèse et sont naturellement discriminants. Même si ces modèles graphiques ont initialement été développés pour répondre à des problématiques nécessitant des caractéristiques symboliques, comme c'est le cas en traitement automatique du langage [100], ils tendent à s'étendre aux domaines voisins comme la reconnaissance de gestes [101, 102] ou d'objets dans des scènes naturelles [103, 104]. Cette tendance est cependant freinée car les CAC ne sont pas capables de modéliser correctement des données numériques (valeurs réelles), ils ne combinent efficacement que des valeurs discrètes. Ils ne sont pas capables de modéliser des données

numériques continues comme c'est le cas par exemple avec les MMC qui intègrent un modèle de mélange de Gaussiennes pour cela.

La principale fonctionnalité d'un CAC est de réaliser un classifieur dynamique combinant des données discrètes. Les données discrètes sont aussi bien des caractéristiques d'entrée que l'état discret du modèle à l'instant précédent. Ainsi, pour dépasser cette limitation, des travaux introduisent un premier étage de classification sur des données continues, en utilisant par exemple un réseau de neurones artificiels (RNA). De cette façon, le CAC est uniquement chargé de modéliser les dépendances entre les décisions locales fournies par le premier étage. On rencontre ce genre de système en traitement automatique de la parole notamment [105, 106], mais les premiers travaux en ce sens ont été proposés en vision [107].

Pour réaliser un système de reconnaissance de mots manuscrits, le recour à un modèle discriminant semble préférable afin de prendre des décisions locales marquées, minimisant le plus possible les incertitudes. Par ailleurs, le problème de classification de séquences nécessite de modéliser les dépendances temporelles entre les classes. Un modèle de Champs Aléatoires Conditionnels répond bien à ces exigences. Il faut également être capable de prendre en compte des connaissances de haut niveau telles qu'un lexique ou un modèle de langage afin de pouvoir filtrer les solutions incohérentes qui sont proposées par l'étage de classification. Les Modèles de Markov Cachés, par leur caractère génératif, ont cette aptitude à intégrer cette seconde contrainte, en recourant notamment à des grammaires stochastiques ou à des modèles statistiques n-grammes au moment du décodage. Il faut également ajouter le fait qu'il existe plusieurs bibliothèques qui intègrent ces fonctionnalités de modélisation du langage dans le cadre du formalisme des Modèles de Markov Cachés. Il y a donc également une motivation pratique pour recourir à une modélisation de type MMC pour réaliser la fonctionnalité de décodage pilotée par des connaissances de haut niveau.

Ainsi, pour profiter des avantages des CAC et des MMC, nous proposons de les combiner dans une architecture hybride comme ce fut le cas par le passé pour les architectures Neuro-Markoviennes proposées au début des années 90. Dans ce chapitre, nous allons exposer la première contribution de cette thèse qui repose sur le développement d'un modèle hybride CAC/MMC pour la reconnaissance de mots manuscrits isolés. Nous commençons en avançant quelques arguments qui motivent notre proposition. Nous poursuivons en présentant notre modèle hybride. Enfin, nous présentons les performances de ce système sur la base de référence RIMES [1].

3.2 Proposition d'une structure hybride CAC/MMC pour reconnaissance de mots isolés

Dans leur conception initiale [50], les CAC reposent sur un modèle de régression logistique multinomiale qui s'attache à représenter la probabilité des classes possibles du problème en fonction des variables explicatives qui peuvent être continues ou discrètes, en recourant à une fonction dite Logit. La fonction Logit fait le lien entre les variables explicatives et le logarithme du rapport des probabilités des deux évènements (≪la classe considérée apparaît≫, ≪la classe considérée n'apparaît pas≫). On parvient ainsi à modéliser de nombreuses distributions statistiques comme, la distribution normale multidimensionnelle utilisée en analyse discriminante linéaire par exemple, mais également d'autres distributions, notamment celles où les variables explicatives sont booléennes. C'est à travers cette propriété que l'on peut expliquer peut être le succès des CRF dans le domaine du traitement de la langue où les variables explicatives sont les mots, donc des variables booléennes. Il semble que l'utilisation de variables explicatives réelles limite très fortement le type de distribution sous-jacente. Un autre argument en faveur des caractéristiques discrètes est lié au fait que les implémentations des CAC en Traitement Automatique du Langage intègrent un générateur automatique de variables explicatives booléennes (constitué de règles de combinaisons binaires) à partir de l'observation des séquences de mots qui leur permettent d'explorer un espace de caractéristiques de très grande dimension, et ainsi de trouver les représentations les plus pertinentes au cours de l'apprentissage. Une telle approche n'est pas immédiatement transposable lorsqu'on travaille sur des variables explicatives continues. Notre première contribution se situe dans la lignée des modèles hybrides actuellement mis en place pour traiter la reconnaissance de mots isolés (cf section 2.4). Le point fort majeur de ce genre d'approche est qu'elles permettent de profiter des avantages d'une approche discriminante et d'une approche générative. Dans le cadre de nos expériences, nous avons choisi d'appliquer le CAC sur le bas niveau afin d'exploiter sa capacité discriminante et sa prise de contexte très large pour fournir la probabilité d'apparition des caractères à un étage MMC. Utilisé comme tel, ce dernier apporte sa capacité de modélisation séquentielle et embarque les informations de haut niveau nécessaires au bon déroulement de la tâche (lexique, modèle de langage, etc.). Ce système a été évalué sur la base Rimes Mots Isolés 2009 [1] et comparé à d'autres systèmes hybrides dont le système à l'état de l'art le BLSTM-CTC [47].

Dans cette section, nous présentons notre système hybride, notamment le lien entre les deux étages et la phase d'apprentissage. Puis, nous exposons notre protocole expérimental et les résultats.

3.2.1 Vue d'ensemble du modèle hybride CAC/MMC

L'ensemble de notre système est résumé dans la Figure 3.1, il permet de bénéficier des avantages des approches génératives et discriminantes. Le premier étage est chargé d'extraire des caractéristiques de l'image. Celui-ci fournit une description continue multi-dimensionnelle de chaque trame qui sur-découpe l'image du mot analysé. Les caractéristiques sont ensuite discrétisées en recourant à une étape de classification automatique. Les descripteurs discrets obtenus constituent un dictionnaire de formes élémentaires (CodeBook) de cardinalité importante qui permet de représenter l'observation mesurée sur chaque trame. D'une manière similaire aux approches développées en Traitement Automatique du Langage, des observations de plus haut niveau peuvent être construites en analysant ensemble plusieurs trames consécutives. On dispose ainsi de plusieurs dictionnaires de CodeBook décrivant des n-grammes de trames. Cette description multi-échelle discrète alimente le modèle d'attache aux données constitué d'un Champs Aléatoire Conditionnel. Contrairement aux approches usuelles qui utilisent un champs aléatoire pour réaliser l'étiquetage de la séquence d'observations en procédant à un décodage de type Viterbi, le champs aléatoire est utilisé ici pour calculer les probabilités *a posteriori* locales de chaque trame en utilisant un décodage Forward-Backward. Comme dans la plupart des modèles hybrides, les probabilités locales sur chaque trame sont finalement inversées pour constituer les vraisemblances de chaque état du Modèle de Markov Caché dédié à la modélisation des séquences de caractères de notre problème.

3.2.2 Apprentissage indépendant CAC/MMC

Afin d'apprendre ce type d'architecture, il est nécessaire d'apprendre l'étage CAC et les probabilités de transitions de l'étage MMC. L'apprentissage des CAC nécessite une vérité terrain niveau trame comme l'atteste l'équation 2.38. Malheureusement la majorité des bases existantes n'ont pas ce niveau d'étiquetage, nous n'avons en général accès qu'à une vérité terrain niveau mot. C'est à dire que l'on connaît la séquence de caractères présente dans l'image de chaque mot, mais on ne connaît pas la position des caractères dans l'image (la segmentation en caractères n'est pas fournie). Pour pallier ce manque d'information d'étiquetage, deux approches sont possibles. La première consiste à utiliser un premier système préalablement optimisé sur la base d'apprentissage puis à procéder à l'étiquetage de chaque trame en utilisant le premier système en mode d'alignement forcé sur la séquence de caractères de la vérité terrain de l'image du mot analysé. Naturellement le premier système doit pouvoir être entraîné sans nécessiter la vérité terrain au niveau trame. On peut choisir pour cela un système construit avec des MMC car l'algorithme Forward-Backward utilisé pour l'optimisation des modèles

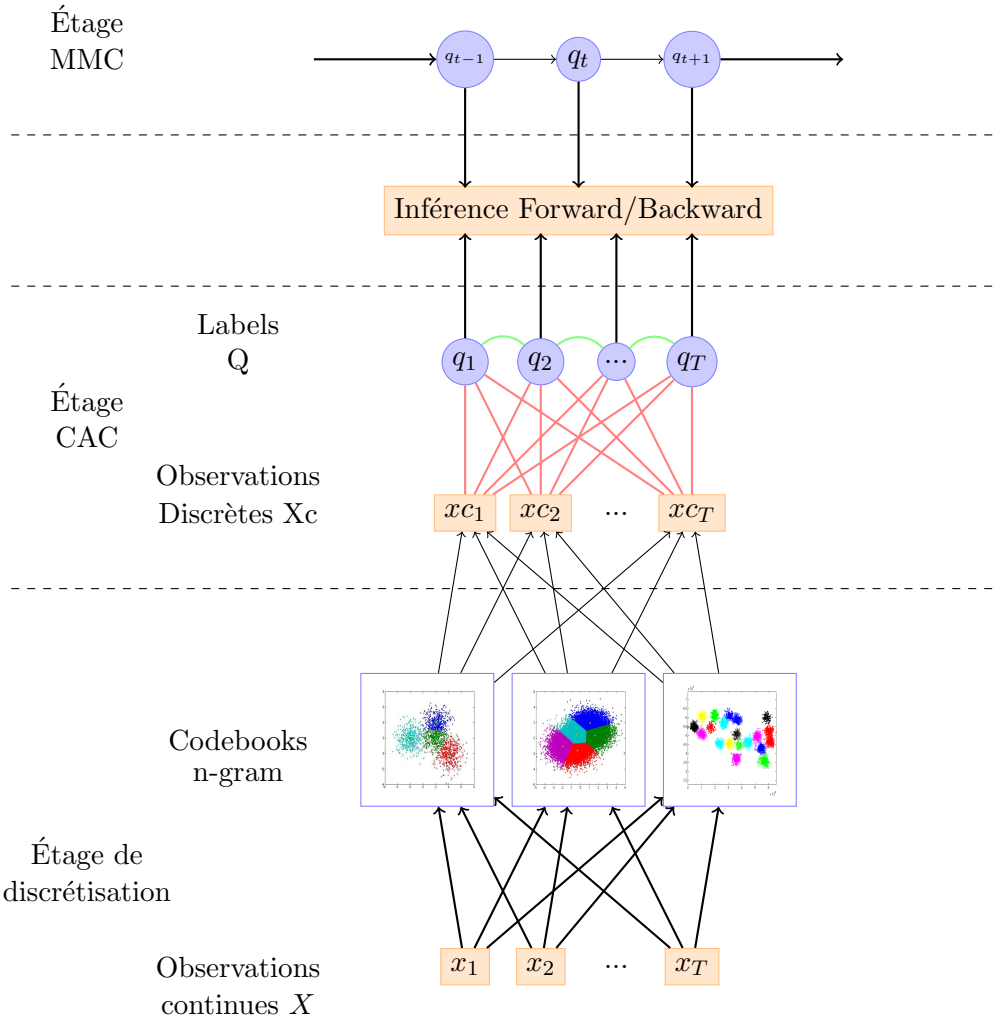


FIGURE 3.1: Description de la structure hybride CAC/MMC : Vue d'ensemble de toutes les étapes de l'extraction des caractéristiques à la reconnaissance mot

MMC palie précisément l'absence de vérité terrain au niveau trame en l'inférant selon le principe de l'algorithme EM (Expectation Maximization). La seconde approche possible pour entraîner notre modèle hybride, sans nécessiter la vérité terrain de chaque trame, consiste à développer une approche de type EM appropriée à notre architecture hybride. La première approche de la littérature pour l'apprentissage conjoint d'une structure hybride RNA/MMC a été proposée par Senior et Robinson en 1995 [108]

Dans ce travail les auteurs introduisent un critère local d'information mutuelle entre la probabilité *a posteriori* inférée grâce au Modèle HMM du mot à apprendre en utilisant l'inférence Forward Backward, et la probabilité *a posteriori* calculée par le réseau de neurones. Le réseau de neurones est mis à jour pour maximiser le critère, et donc s'adapter à la vérité terrain inférée en chaque trame depuis le niveau mot. Comme nous l'avons vu au

chapitre précédent, l'apprentissage d'un CAC est réalisé en optimisant un critère global qui est la probabilité *a posteriori* de la séquence alignée sur la vérité terrain. Néanmoins on voit bien dans l'équation du CAC 2.37 que l'étiquetage local est nécessaire dans le calcul du critère global. C'est cet étiquetage local que nous proposons d'inférer à partir de la vérité terrain du niveau mot par inférence Forward-Backward en utilisant le modèle MMC du mot. Si $y^{(n)}$ et $y'^{(n)}$ désignent des états quelconques pouvant apparaître dans le modèle du mot de l'exemple numéro n, alors nous désignons $P(q_t = y^{(n)}|X^{(n)})$ la probabilité *a posteriori* locale inférée grâce au modèle MMC. Dans l'équation 2.38 cette quantité apparaît indirectement en comptant à 1 toutes les fonctions de caractéristiques qui surviennent pour chaque état de la vérité terrain et à 0 toutes les autres. Il nous faut donc modifier ce critère pour prendre en compte l'inférence probabiliste de la vérité terrain au niveau local. C'est ce que nous proposons dans l'équation ci-dessous où l'expression au numérateur prend en compte la probabilité de l'état inférée localement, il faut alors sommer sur l'ensemble de tous les états $y^{(n)}$ et $y'^{(n)}$ de la vérité terrain.

$$L(\Lambda) = \sum_{n=1}^{N} log(\frac{\sum_{y^{(n)},y'^{(n)}} exp(\sum_{t=1}^{T}\sum_{m=1}^{M} \lambda_m P(q_t = y^{(n)}) f_j(X^{(n)}, q_{t-1} = y'^{(n)}, q_t = y^{(n)}))}{\sum_{y,y'} exp(\sum_{t=1}^{T}\sum_{m=1}^{M} \lambda_m f_m(X^{(n)}, q_{t-1} = y', q_t = y))}) \tag{3.1}$$

En dérivant ce critère par rapport aux paramètres du CAC on obtient finalement l'expression suivante où l'on voit apparaître deux termes. Le premier comptabilise les fonctions de caractéristiques lorsqu'on réalise l'alignement Forward-Backward du MMC sur la vérité terrain du mot, le second terme comptabilise les fonctions de caractéristiques lorsqu'on réalise l'inférence Forward-Backward du modèle CAC sur tous les états possibles. L'objectif est de mettre à jour les paramètres du CAC pour se rapprocher de l'alignement désiré, celui inféré par la vérité terrain.

$$\begin{aligned}\frac{\delta L(\Lambda)}{\delta \lambda_m} = &\sum_{n=1}^{N} \sum_{y^{(n)},y'^{(n)}} \sum_{t=1}^{T} [P(q_t = y^{(n)}) f_m(X^{(n)}, q_{t-1} = y'^{(n)}, q_t = y^{(n)}) \\ &- \sum_{y,y'} P(q_t = y) f_m(X^{(n)}, q_{t-1} = y', q_t = y)]\end{aligned} \tag{3.2}$$

Quelle que soit la stratégie d'apprentissage retenue pour ce modèle hybride (à l'aide d'un premier système, ou directement par inférence Forward-Backward) l'apprentissage du CAC est réalisé selon une méthode de descente de gradient stochastic (SGD), ou L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno).

3.2.3 Choix de la représentation d'entrée

A l'origine, les CAC ont été développés pour des tâches de traitement automatique de la langue utilisant des observations discrètes. Dans ce domaine, les observations sont les mots mais également des combinaisons de mots pertinentes pour la tâche que l'on souhaite réaliser. Les principales implémentations des CAC dans ce domaine proposent toutes un générateur de caractéristiques booléennes définies selon un modèle (un patron ou template) qui peut couvrir plusieurs observations consécutives (n-gramme) au prix cependant d'une combinatoire très importante. De sorte qu'il n'est pas rare d'apprendre un CAC défini sur plusieurs centaines de milliers de caractéristiques booléennes. En pratique on observe un très bon comportement de ce type de système qui travaille dans un espace d'entrée discret creux et de très grande dimension. La capacité d'un CAC à modéliser la distribution d'observations définies dans un espace multidimensionnel continu est beaucoup plus limitée en revanche. En effet, la fonction Logit, définie selon un modèle linéaire des variables explicatives, est une restriction importante du modèle CAC dans le cas de données continues. Pour dépasser ces limitations, les travaux de Gunawardana [100] ont introduit le modèle de Champs Aléatoire Conditionnel Caché (CACC). L'ajout d'une couche composée d'états cachés permet d'avoir un niveau d'abstraction supplémentaire entre les observations d'entrée et l'étage de décision comme c'est le cas pour un MMC. Dans ces travaux, les auteurs ont d'ailleurs énoncés les conditions dans lesquelles il y a une équivalence exacte entre les deux modèles à états cachés, en spécifiant notamment les fonctions de caractéristiques appropriées du CACC. Par la suite ils ont déduit une stratégie d'apprentissage efficace du modèle CACC discriminant. Ils proposent d'apprendre dans une premier temps le modèle MMC équivalent au modèle CACC, puis de poursuivre dans un second temps l'apprentissage du CACC initialisé par les paramètres du modèle MMC. L'apprentissage d'un modèle CACC initialisé aléatoirement donnant en effet de mauvais résultats.

Ces constatations nous ont conduit à privilégier une approche différente pour notre modèle hybride, qui de fait doit être capable de prendre des descriptions continues en entrée puisque nous travaillons sur des données image. Comme évoqué précédemment, nous avons choisi de discrétiser la description continue de chaque trame en recourant à une étape de clustering. La prise en compte du contexte étant primordiale en reconnaissance d'écriture, nous avons introduit différentes descriptions contextuelles de la séquence de trames, elles mêmes décrites à différentes échelles (uni-trame, bi-trames, tri-trames). Ces descriptions sont fondées sur différents codebooks (un codebook par échelle) obtenus par classification non-supervisée à l'aide d'un classifieur de type K-Means. Le fait de travailler dans un espace discret nous permet de concaténer un grand nombre

d'informations sur chacune des trames de l'image à reconnaître et ainsi de profiter pleinement des aptitudes du CAC à travailler en grandes dimensions. Cependant, ce type de méthode demande beaucoup d'optimisation en raison du nombre important d'hyper-paramètres (nombre de clusters, taille des fenêtres d'extraction, contexte à prendre en compte, etc.). Chacun des paramètres a été testé en fonction de son influence sur les performances du système complet sur une base de validation.

Dans le cadre de cette thèse, les caractéristiques utilisées sont basées sur des histogrammes orientés de gradients [109] car se sont des caractéristiques de référence en reconnaissance d'écriture de part leurs robustesses et la qualité de l'information extraite. Ces histogrammes sont extraits à partir de fenêtres de 8, 16 et 24 pixels avec décalage de 1 pixel (fenêtre glissante). On ajoute également à ces 64 caractéristiques (8 directions d'histogrammes sur des blocs de 2 par 4) 6 caractéristiques globales sur l'agencement des pixels dans les sous-blocs :

1. position du centroide vertical ;
2. position du centroide horizontal ;
3. position du pixel noir le plus haut ;
4. position du pixel noir le plus bas ;
5. la distance entre ces derniers ;
6. le nombre de pixels noirs dans la trame.

Le nombre optimal de clusters a été choisi en fonction des résultats sur la base de validation. Pour les fenêtres de 8, 16 et 24 pixels, le nombre de clusters est respectivement 1000, 2000 et 5000. En intégrant ces trois points de vues (trois échelles) dans la description de chaque trame nous disposons d'une première représentation discrète et multi-échelle de chaque trame. Il est ensuite possible d'ajouter à cette première description des informations contextuelles adaptées à chaque niveau. Ainsi pour le niveau uni-trame (fenêtre de 8 pixels), nous avons choisi d'enrichir la description de la fenêtre courante par celles des 4 fenêtres précédentes et des 4 suivantes de même niveau (uni-trame). On obtient ainsi 9 valeurs discrètes. Suivant le même principe, nous avons choisi d'enrichir la description du niveau bi-trames de la fenêtre courante par celle la fenêtre précédente et de la fenêtre suivante de même niveau (bi-trames). Nous n'avons pas ajouté d'information contextuelle pour le niveau tri-trames.

Chacune des fenêtres de contexte est choisie afin d'éviter un recouvrement avec la fenêtre courante. De cette façon nous n'ajoutons donc aucune redondance dans la description finale fournie au CAC. Afin d'évaluer l'apport de chacun des niveaux d'abstraction (uni-trame, bi-trame, tri-trame), nous avons expérimenté les configurations suivantes :

1. fenêtre de 8 pixels (I)
2. fenêtre de 8 pixels + fenêtre de 16 pixels (I+II)
3. fenêtre de 8 pixels + fenêtre de 16 pixels + fenêtre de 24 pixels (I+II+III)

3.3 Les architectures hybrides concurrentes

Comme nous avons pu le constater dans la partie bibliographique de cette étude, chapitre 2, la littérature récente est relativement riche de contributions nouvelles en matière de modélisation de séquences avec notamment des approches fondées sur les réseaux de neurones récurrents. Dans ce contexte relativement compétitif il nous est apparu opportun de ne pas limiter nos travaux à l'exploration d'une seule et unique architecture hybride. C'est la raison pour laquelle nous avons finalement choisi d'explorer un ensemble d'architectures hybrides afin d'évaluer leurs mérites et faiblesses respectives. Ce travail fut l'occasion de mettre en œuvre un certain nombre des méthodes popularisées dans la littérature. Précisons qu'elles se différencient essentiellement par leur modèle d'attache aux données car elles se fondent toutes sur un modèle MMC pour la modélisation des mots du lexique. Nous avons ainsi choisi d'explorer les architectures suivantes : le MMC standard avec pour modèle d'attache aux données le mélange de gaussiennes (GMM), une variante des MMC avec un décodage forward-backward intermédiaire estimant des probabilités *a posteriori* locales qui se substituent aux vraisemblances fournies par les Mélanges Gaussiens, un Perceptron Multi-Couche couplé avec un MMC, un réseau de neurones récurrents couplé avec un MMC et enfin un BLSTM-CTC couplé avec un MMC. Dans cette section, nous décrivons chacun de ces systèmes en donnant des informations sur leur fonctionnement et leur configuration pour cette étude.

3.3.1 GMM-MMC

La Figure 3.2 résume notre système hybride GMM-MMC. Le MMC utilisé est composé de 6 états par caractère chacun composé de 20 gaussiennes. Chacun des caractères du problème a été modélisé pour un total de 81 caractères. Cette configuration est la même que celle présentée dans [110]. La décodage s'effectue suivant la méthode classique de Viterbi (cf section 2.3.3.2). L'ensemble du modèle a été implémenté sous HTK.

3.3.2 GMM-MMC avec inférence des probabilités locales *a posteriori*

La Figure 3.3 résume notre système avec inférence des probabilités locales *a posteriori*. Il s'agit du même GMM-MMC que décrit précédemment à la différence que l'algorithme de

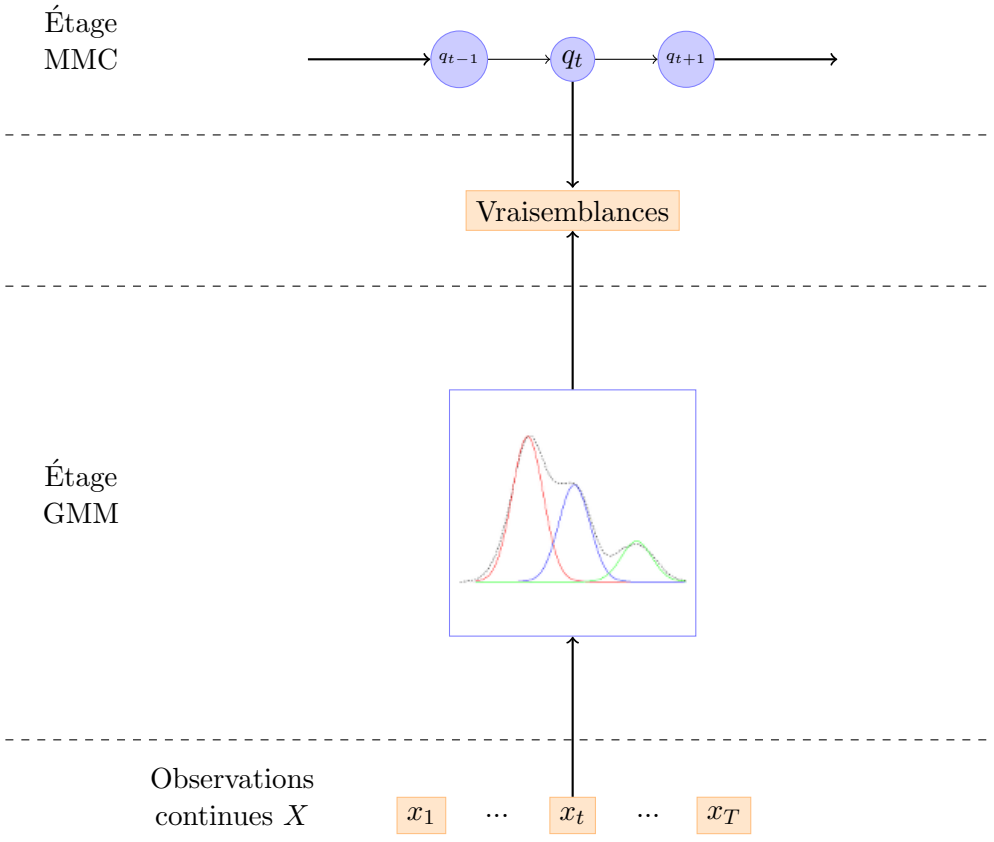


FIGURE 3.2: Représentation graphique d'un système GMM-MMC

décodage procède en deux étapes. La première réalise une inférence forward-backward pour calculer les probabilités *a posteriori* des états localement en chaque trame. La seconde réalise le décodage usuel selon l'algorithme de Viterbi dirigé par un lexique, mais en exploitant cette fois les posterior plutôt que les vraisemblances. Ce type d'approche a été proposé par S. Bengio à l'IDIAP [111, 112].

3.3.3 MLP-MMC

La Figure 3.4 résume notre système hybride MLP-MMC. L'étage MLP de cette structure est composé d'une seule couche cachée de 80 neurones utilisant une fonction tangente hyperbolique. Ce réseau a été appris à l'aide d'un algorithme de rétropropagation du gradient. Une vérité terrain niveau trame a été générée afin de fournir les éléments nécessaires au bon déroulement de l'apprentissage. Cette dernière a été obtenue à l'aide d'un alignement forcé (Viterbi) issu du système BLSTM-CTC-HMM car ce système a les plus hautes performances au niveau trames. Cet étage discriminant est combiné avec un modèle MMC composé d'un seul état caractère. Cet étage MMC est modifié en fonction

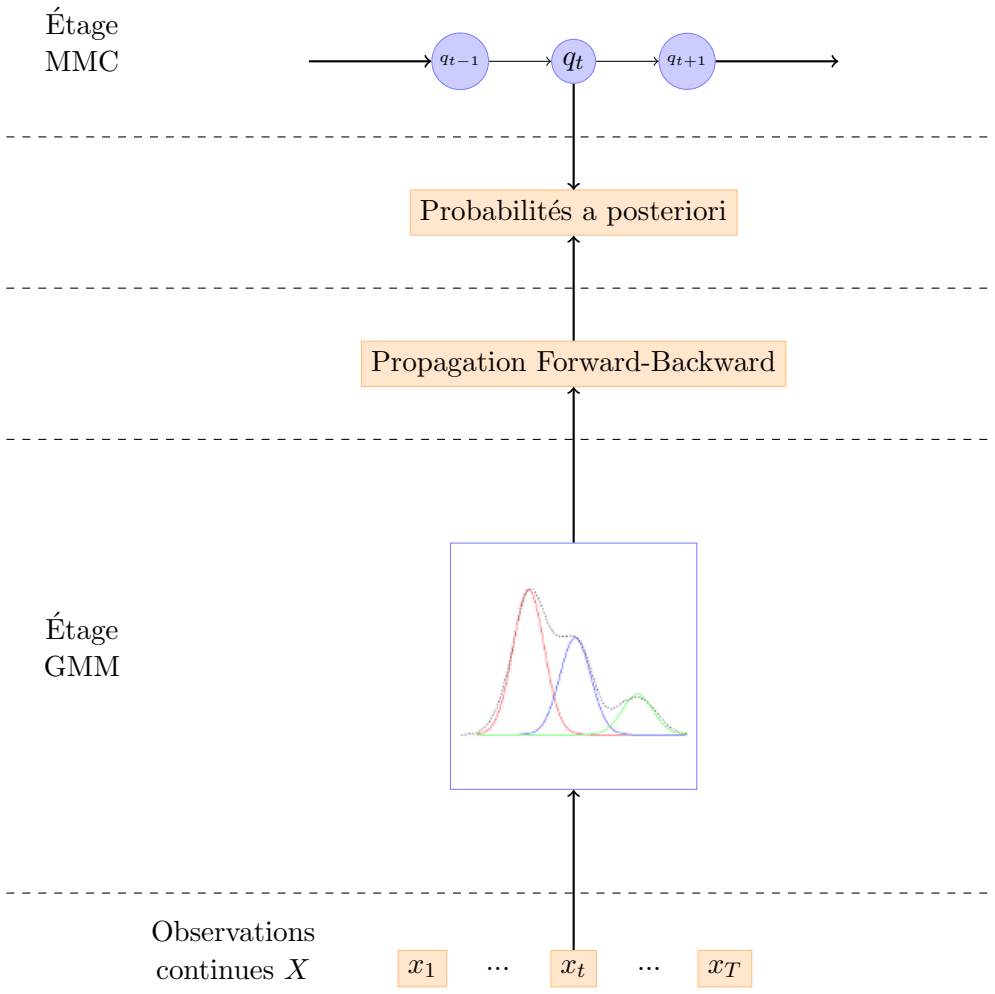


FIGURE 3.3: Représentation graphique d'un système GMM-MMC Forward-Backward

du mot du lexique à aligner. La même vérité terrain et le même étage MMC ont été utilisés pour l'ensemble des autres méthodes qui seront présentées dans la suite de ce chapitre.

3.3.4 RNN-MMC

La Figure 3.5 résume notre système hybride RNN-MMC. Ce réseau de neurones récurrents est composé d'une seule couche cachée de 80 neurones incorporant une fonction tangente hyperbolique. Ce réseau a été appris à l'aide d'un algorithme de rétropropagation du gradient.

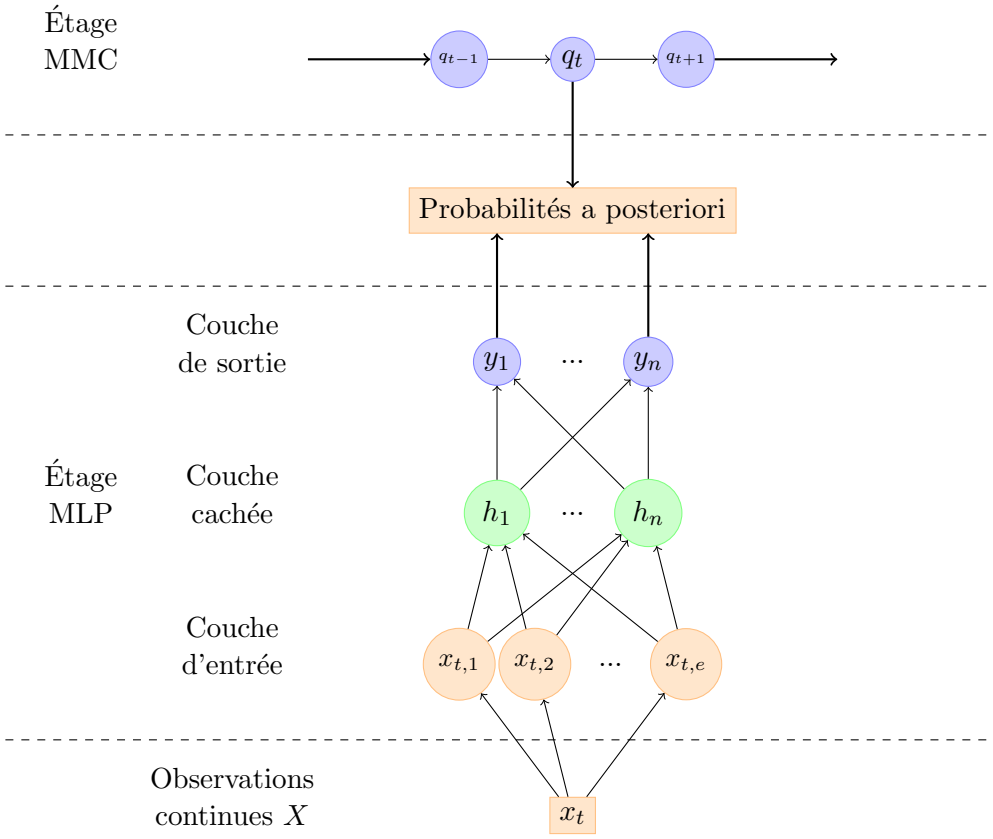


FIGURE 3.4: Représentation graphique d'un système hybride MLP-MMC

3.3.5 BLSTM-CTC-MMC

La Figure 3.6 résume notre système hybride BLSTM-CTC-HMM. Notre BLSTM est composé de deux couches cachées composées respectivement 70 et 120 neurones. Il a été appris à l'aide d'un algorithme de rétropropagation du gradient et une inférence forward-backward dirigée par le lexique au niveau du CTC. Par souci de simplification une seule des deux couches sera représentée sur la figure.

3.4 Expérimentations

Dans cette partie, nous rapportons l'ensemble des expériences effectuées sur la tâche de reconnaissance mots isolés à partir de la base manuscrite Rimes de mots isolés 2009 [1]. Ces expériences regroupent les tests visant à optimiser les hyper-paramètres du modèle hybride CAC-MMC proposé (caractéristiques, analyse du comportement local

1. http ://htk.eng.cam.ac.uk/

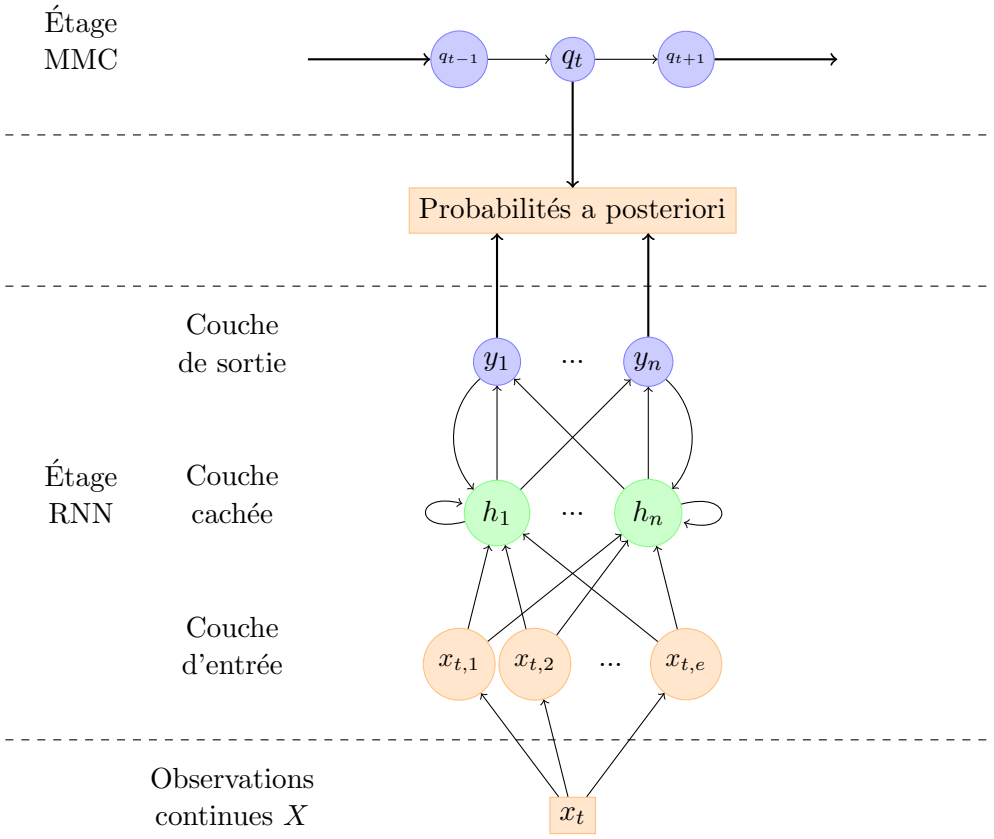


FIGURE 3.5: Représentation graphique d'un système hybride RNN-MMC

du modèle), puis à comparer les différents systèmes hybrides concurrents qui ont été présentés dans la section précédente.

3.4.1 Présentation de la base de données RIMES 2009

La base utilisée pour effectuer nos expériences est la base mots manuscrits isolés Rimes 2009 [1]. Cette base est composée de mots issus de lettres simulant une correspondance de clients avec différentes compagnies. Pour constituer cette base, chaque volontaire a reçu une identité fictive, un scénario et un thème comme par exemple "déclaration de sinistre" ou "modification de contrat". La base de mots réalisée à partir de ces correspondances fictives est constituée de 43 000 images de mots pour l'apprentissage du système, 7300 mots pour valider les paramètres et contrôler l'apprentissage et 7464 mots pour l'évaluation finale des performances. Le lexique utilisé pour tester le système contient 1600 entrées. Chaque image de mot est associée à sa transcription sous la forme d'une séquence de caractères. Cette transcription a été validée manuellement par des

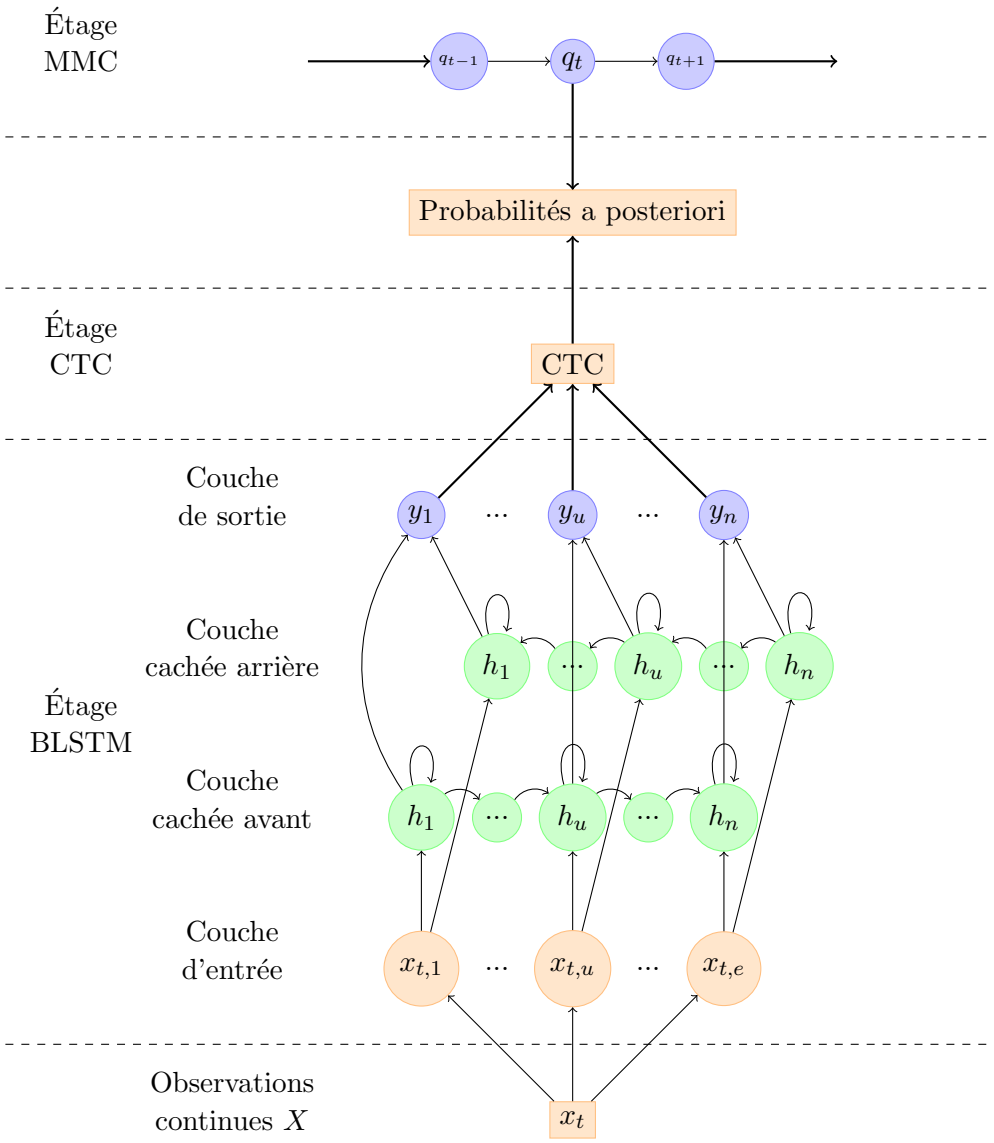
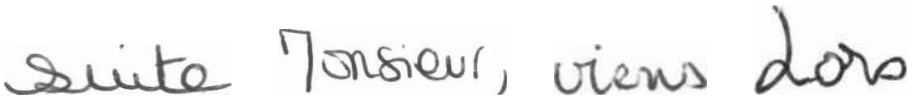


FIGURE 3.6: Représentation graphique d'un système hybride BLSTM-CTC-HMM

opérateurs afin de s'assurer de la bonne qualité de cette dernière. La Figure ci-dessous montre plusieurs exemples de mots de la base.



3.4.2 Paramètres et analyse du système CAC

3.4.2.1 Choix de l'algorithme d'apprentissage

Les deux algorithmes les plus reconnus et éprouvés dans la littérature sont L-BFGS et SGD. Nous avons étudié les avantages et les inconvénients de ces deux méthodes. L-BFGS est un algorithme quasi-Newton qui prend en compte à chaque itération l'ensemble de la base d'apprentissage. Chaque itération est donc coûteuse et le temps de convergence est long, mais on a l'assurance de converger vers une solution optimale. SGD est une méthode de gradient stochastique qui prend en compte à chaque itération un sous-ensemble de la base d'apprentissage choisi aléatoirement, de manière à avoir une meilleure capacité à généraliser. Chaque itération est donc plus rapide, et cette méthode converge rapidement vers une solution acceptable, ce qui permet d'accélérer les tests d'optimisation des hyper-paramètres et de passer à l'échelle sur des bases d'apprentissage de grande taille. Cependant, la courbe de progression des performances au cours de l'apprentissage est très irrégulière à cause du caractère aléatoire de cette méthode. Au vu des caractéristiques de la base RIMES (45 000 exemples) et du nombre d'hyper paramètres à estimer, comme la taille de fenêtre d'extraction, le pas de la fenêtre, le pas d'apprentissage, la normalisation, les pré-traitements, nous avons choisi d'utiliser l'algorithme SGD.

3.4.2.2 Réglage des hyper-paramètres pour l'apprentissage du CAC

L'agorithme SGD propose plusieurs hyper-paramètres :

1. un coefficient de régularisation (L2)
2. un critère d'arrêt en deux temps (nombre d'itérations (period) générant des scores d'erreurs successifs dont la différence ne dépasse pas une variation de valeur δ)
3. un pas d'apprentissage η

Le coefficient de régularisation L2 est également appelé *Least Squares Error (LSE)* il se définit de la façon suivante :

$$L2 = \sum_{i=1}^{n} (valeurCible_i - valeurEstime_i)^2 \tag{3.3}$$

Nous avons pris le postulat de laisser le critère d'arrêt par défaut :

— $period = 10$
— $\delta = 1e^{-5}$

Dans cette configuration l'algorithme risque en effet d'avoir convergé complètement depuis plusieurs itérations avant l'atteinte du crière d'arrêt mais nous avons au moins l'assurance de convergence complète. En cas de sur-apprentissage, le modèle final ne sera pas impacté car nous ne sauvegardons à chaque itération que le modèle maximisant le taux d'erreur trame sur la base de validation.

Le pas d'apprentissage est le paramètre le plus important, la librairie CRFsuite [113] permet une étape de calibration sur un sous-ensemble de la base d'apprentissage de façon à évaluer un ordre de grandeur du pas d'apprentissage le plus adapté. Le pas d'apprentissage proposé sur notre sous-base de 1000 exemples était relativement grand $1e^{-4}$. Nous avons choisi de le conserver car cela accélèrera la vitesse de convergence sur les premières itérations. Nous avons modifié le code initial de CRFsuite pour rajouter une division du pas d'apprentissage quand la méthode s'éloigne de la solution optimale (erreur de l'itération t est supérieur à celle de l'itération $t-1$). Nous avons opté pour une division par 2 afin de ne pas diviser le pas d'apprentissage trop vite afin que les itérations du système ne soient pas trop longues. L'ensemble de ces modifications et cette configuration nous ont permis d'optimiser un système CAC pour effectuer de la reconnaissance de mots isolés manuscrits. La section suivante détaille le comportement interne de notre système CAC.

3.4.3 Analyse du comportement interne du CAC

L'implémentation du modèle CAC a été réalisée à partir de la librairie CRFSuite [113]. La base Rimes mots isolés [1] compte 81 classes de caractères différentes. Dans la configuration du système présenté précédemment utilisant les caractéristiques discrètes multi-échelle et multi-contextuelles on compte au total 6561 fonctions de caractéristiques de transitions et 1 844 937 fonctions de caractéristiques d'observation. Les répartitions des valeurs des poids de ces fonctions sont résumées dans les deux histogrammes des figures 3.7 et 3.8. On remarque de suite, la capacité du CAC à sélectionner les caractéristiques très discriminantes. Les valeurs négatives des poids signifient que les caractéristiques s'opposent à l'occurrence de la classe considérée alors qu'à l'inverse les valeurs positives contribuent à l'apparition de la classe. Par ailleurs, les valeurs très faibles des poids associées à certaines caractéristiques traduisent le fait qu'elles ont peu d'influence dans la décision (elle ne sont pas discriminantes, et pourraient être éliminées éventuellement). La nature extrêmement piquée du premier histogramme sur 0 révèle que beaucoup de fonctions de transition ne contribuent pas à la décision. Elles n'interdisent ni ne favorisent aucun enchaînement particulier de caractères. La dissymétrie de cet histogramme qui présente plus de valeurs négatives que positives révèle par ailleurs qu'il y a sans doute beaucoup plus d'enchaînements de caractères pénalisés que d'enchaînement de caractères

favorisés sur le lexique de la base, par rapport à la combinatoire possible. Ce qui est sans doute significatif du fait que le lexique est de taille relativement faible (1600 mots). Il faut remarquer que le modèle CAC contrairement au modèle MMC, pénalise certains enchaînements de caractères mais sans jamais les interdire. Le modèle MMC quant à lui, interdit toute transition de probabilité nulle. L'analyse du second histogramme des poids révèle quant à elle un comportement assez différent sur les caractéristiques liées aux observations. En effet, l'histogramme est d'une part plus symétrique, mais en revanche il n'est pas centré sur 0 mais sur une valeur positive proche de 0.1. On déduit donc cette fois que les caractéristiques d'observations qui sont très nombreuses apportent en moyenne une information qui permet d'aider à discriminer les classes et qu'elles contribuent d'avantage à renforcer les classes qu'à les affaiblir. Etant donné le nombre très important de caractéristiques, ce constat est un peu négatif, car il montre que le CAC élimine peu de caractéristiques, elles vont en moyenne contribuer toutes un peu.

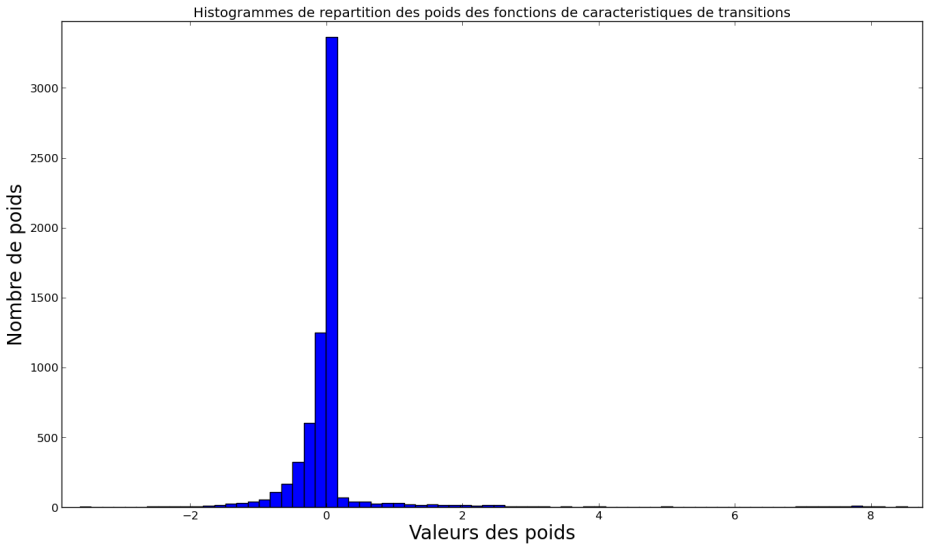


FIGURE 3.7: Répartition des valeurs des poids de transitions du CAC

On remarque de suite la capacité du CAC à sélectionner les caractéristiques très discriminantes. En effet pour les poids de transitions, 4682 poids sont inférieurs à 0.15 en valeur absolue et n'auront donc que très peu d'impact sur la décision finale. On observe que 1432 valeurs sont inférieures à -0.15 dont 660 sont comprises entre -0.5 et -1, les 5 valeurs les plus basses sont comprises entre -3 et -3.5. Les valeurs négatives signifient que l'apparition des caractéristiques associées se traduit par une probabilité faible d'apparition de certains caractères. Les 1879 valeurs restantes sont répartis de 0.15 à 12.5 avec 234 valeurs entre 0.5 et 1 et 9 valeurs entre 10 et 12.5.

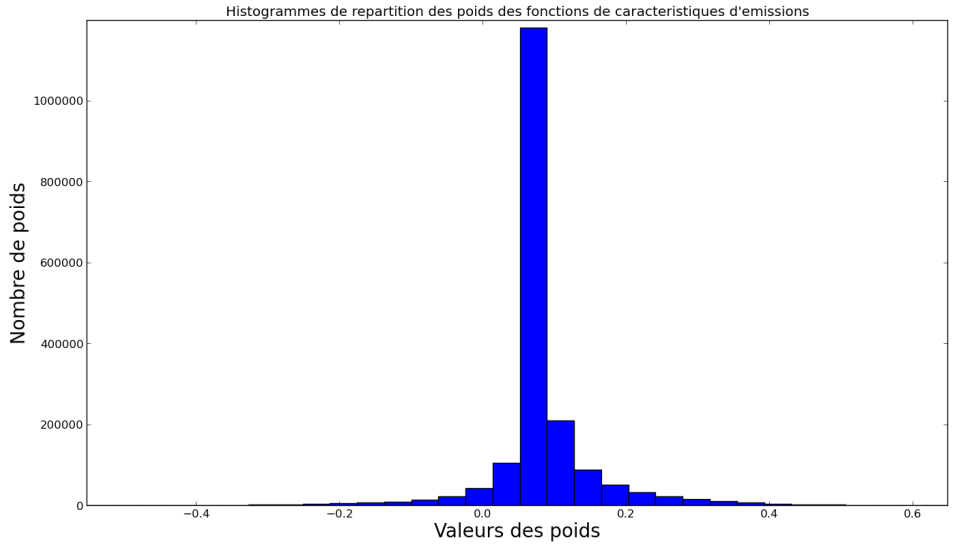


FIGURE 3.8: Répartition des valeurs des poids d'émissions du CAC

3.4.4 Apport d'une classe rejet

Quand on regarde de plus près les trames d'une image de mot manuscrit comme celui de la Figure 3.9, on peut voir que certaines trames sont très difficiles à affecter à une classe caractère particulière, même pour un être humain. Il s'agit le plus souvent des trames au début, à la fin de mot et de liaisons entre les caractères du mot.

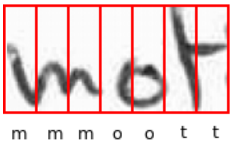


FIGURE 3.9: Image de mot manuscrit **mot** segmenté en trames étiquetées

Pour palier ce problème, nous proposons d'utiliser un caractère "joker" qui correspond à une absence de décision du système. Il sera placé au début à la fin et entre chaque caractère du mot. Ce concept présent dans les travaux d'Alex Graves [47] a fait ses preuves notamment en classification de séquences manuscrites. La segmentation classique en caractères représentée par la Figure 3.9 devient celle présente sur la Figure 3.10.

En ajoutant cette classe, nous espérons qu'au cours de l'apprentissage le système apprendra à se prononcer plus spécifiquement sur des trames contenant suffisamment d'information pour prendre une décision. Ainsi, les décisions locales devraient être beaucoup plus précises qu'avec une segmentation en caractères sans classe de rejet. Cependant, l'ajout de cette classe provoque un déséquilibre important de la base d'apprentissage (la

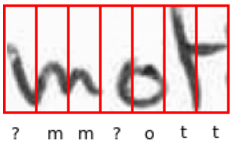


FIGURE 3.10: Image de mot manuscrit **mot** segmenté en trames et étiquetée avec des jokers

classe rejet est sur-représentée). La proportion de cette classe devra donc être contrôlée afin d'éviter un sur apprentissage, phénomène très fréquent observé pour les méthodes discriminantes.

Dans les tests réalisés, nous observons sur le tableau 3.1 ci-dessous que l'ajout de la classe rejet améliore les performances de 9,12% en Top 1, 8,15% en Top2, 9,06% en Top 3 et 12,58% en Top 5. On voit donc que cette classe apporte un gain très significatif au système discriminant, et nous retrouvons en cela l'une des propriétés du système BLSTM-CTC proposé par A. Graves.

TABLE 3.1: Influence du caractère joker sur la reconnaissance mot du système CAC-MMC avec un lexique de 1600 mots.

Sytème	ET Top 1 (Erreur Trame)	EM Top 1 (Erreur Mot)	EM Top 2 (Erreur Mot)	EM Top 3 (Erreur Mot)	EM Top 5 (Erreur Mot)
Sans Joker	63.58 %	40.42 %	37.68%	32.52 %	30.73 %
Avec Joker	51.24 %	31.30 %	29.53%	23.46 %	18.15 %

3.4.5 Analyse des performances du système CAC-MMC

Le tableau 3.2 présente les performances du système hybride CAC-MMC proposé sur la tâche de reconnaissance de mots isolés. Les performances sont présentées en fonction de la représentation multi-échelle multi-contextuelle fournie en entrée du système. On peut constater que la représentation sur 3 échelles est la plus intéressante en améliorant les performances globales du système de 1,6% au niveau trame et de 6% au niveau mot par rapport à la représentation la plus pauvre utilisant une seule échelle. On en conclut que le contexte trame est primordial pour effectuer une bonne reconnaissance mot. Plus on ajoute d'échelles plus on ajoute d'informations et plus la modélisation est précise. Par soucis de complexité du vecteur de caractéristiques, nous n'avons pas souhaité ajouter d'autres échelles. Notre meilleur système obtient une erreur mot de 31,30% en top 1, ce qui démontre le bien fondé de notre approche mettant en jeu une représentation multi-échelle et multi-contextuelle discrète de l'information. Le modèle CAC proposé

confirme donc la capacité mise en évidence dans la littérature, à savoir la capacité à travailler sur des représentations en très grande dimension. Cependant les performances obtenues restent nettement en deçà de celles des meilleurs systèmes de la littérature. On peut attribuer cela au fait que le CAC ne dispose au cours de l'apprentissage d'aucune capacité à structurer l'information d'entrée pour construire une représentation optimale pour la tâche à réaliser. Pour palier ce manque, nous avons dû fournir au système des fonctions de caractéristiques en nombre très important, et nous avons vu qu'à l'issue de l'apprentissage le système a été capable de sélectionner les représentations les plus discriminantes parmi toute celles proposées. Finalement nous constatons que le CAC est davantage un système capable de sélectionner l'information discriminante qu'un système pouvant construire une représentation de l'information qui lui est fournie. On peut penser que les Champs Aléatoires Conditionnels Cachés offriraient un meilleur comportement de ce point de vue. Hélas, par manque de temps il ne nous a pas été possible d'explorer plus avant ces modèles.

TABLE 3.2: Résultats de notre système sur la base Rimes avec un lexique de 1600 mots, pour les configurations monotrame (I), bitrame (II) et tritrame(III)

Caractéristiques	ET Top 1 (Erreur Trame)	EM Top 1 (Erreur Mot)	EM Top 2 (Erreur Mot)	EM Top3 (Erreur Mot)	EM Top5 (Erreur Mot)
CAC-MMC (I)	53.81 %	38.49 %	33.26 %	29.76 %	21.75 %
CAC-MMC (I+II)	52.55 %	34.94 %	29.82%	25.37 %	18.57 %
CAC-MMC (I+II+III)	**51.24 %**	**31.30 %**	**29.53%**	**23.46 %**	**18.15 %**

3.4.6 Étude comparative des différentes architectures hybrides

Dans cette section nous analysons les performances des différents systèmes hybrides alternatifs au modèle CAC-MMC. Pour cela nous restons sur la tâche de reconnaissance de mots isolés. De cette façon, nous pourrons évaluer les performances de notre système face aux approches concurrentes couramment utilisées en reconnaissance de l'écriture manuscrite, sur des configurations identiques. Afin de fournir la comparaison la plus juste possible, les mêmes pré-traitements et les mêmes caractéristiques (HOG) sont employées pour tous les systèmes évalués. L'ensemble des résultats est résumé dans la table 3.3

On observe que le système hybride CAC-MMC devance les MMC standards (+4.95%), les MMC avec décodage forward-backward (+7.19%) et le MLP-MMC (+1.68%). Il semble donc que la modélisation des dépendances du CAC-MMC soit supérieure à

celle de ces 3 modèles. Le CAC offre donc des décisions locales plus précises qui entraînent une meilleure reconnaissance. Mais il ne peut rivaliser avec des systèmes utilisant des neurones plus complexes : récurrents (-7.00%) et LSTM (-19.11%). Il semble en effet que le contexte apporté par la récurrence est primordial en reconnaissance. Cette étude confirme la supériorité des structures hybrides mêlant étage discriminant et génératif par rapport à des approches purement génératives. Les décisions locales du premier étage facilitent grandement la correction par l'étage supérieur. On voit ici la limite des modélisations génératives par mélanges de Gaussiennes qui n'offrent pas de décisions locales aussi tranchées. On remarque également l'écart important entre la structure BSLTM-CTC et les autres dans la classification de séquences. L'apprentissage du BLSTM-CTC lui permet de modéliser des dépendances sur le long terme, sa capacité d'oubli est également l'une de ses plus grandes forces. On peut également remarquer que le décodage forward-backward du MMC ne semble pas apporter d'amélioration. Cela est dû au fait que le passage des vraisemblances à des probabilités entraîne un changement brutal de dynamique. En effet, à l'issue du premier décodage forward backward la plupart des probabilités ont pour valeurs 0 ou 1. Comme par ailleurs, l'erreur en décision reste importante pour les GMM, on affaiblit les scores des hypothèses alternatives dans cette architecture.

TABLE 3.3: Comparaison avec d'autres systèmes sur la base Rimes avec un lexique de 1600 mots

Caractéristiques	ET Top 1 (Erreur Trame)	EM Top 1 (Erreur Mot)	EM Top 2 (Erreur Mot)	EM Top3 (Erreur Mot)	EM Top5 (Erreur Mot)
MMC	88.61 %	36.25 %	32.42 %	30.86 %	25.87 %
MMC (FB)	80.77%	38.49%	34.65 %	32.83%	26.89
MLP-MMC	58.45 %	35.21%	32.98%	29.44%	22.34%
RNN-MMC	45.81%	24.30%	21.20%	17.65 %	15.43%
CAC-MMC (I+II+III)	51.24 %	31.30 %	29.53%	23.46 %	18.15 %
BLSTM-CTC-MMC	**33.93 %**	**12.19%**	**7.38 %**	**7.10%**	**5.89 %**

3.5 Conclusion

Pour conclure, nous avons mis en place une architecture hybride CAC-MMC qui nous a permis d'effectuer la reconnaissance de mots manuscrits isolés à l'aide d'un CAC, ce qui d'après notre recherche bibliographique ne semble pas avoir encore été proposé. Nos expériences montrent que les performances du CAC dépassent celles du MMC classique, d'un MMC forward-backward et d'une structure MLP-MMC mais restent en retrait par rapport aux réseaux récurrents (RNN) et aux BLSTM-CTC. Le CAC montre

ses capacités à modéliser des dépendances sur le long terme et amène une qualité de décision locale intéressante. Même si ses performances sont moindres que celles des réseaux récurrents, sa conception est plus simple et son temps d'apprentissage est nettement plus faible. Les CAC disposent cependant d'une capacité modélisante moindre que les systèmes récurrents.

La structure BLSTM-CTC-MMC obtient les meilleurs résultats. La complexité de l'apprentissage du BLSTM-CTC lui permet de modéliser très précisement les dépendances à long terme à l'intérieur d'une séquence. Cette méthode possède actuellement les meilleures performances sur la classification de séquences.

Dans la suite de ce manuscrit, nous allons appliquer cette structure sur un problème différent : la détection de mots clés et d'expressions régulières dans des lignes d'écriture manuscrite, nécessaire pour l'extraction d'entités nommées. Cette tâche est plus difficile car les séquences à trouver seront moins contraintes par le lexique (augmentation du nombre de confusions). Dans le prochain chapitre, nous décrivons cette tâche par une partie bibliographie, puis nous détaillons les modifications de la structure précédente pour l'adapter à la détection de mots clés et d'expressions régulières. Finalement, nous testons plusieurs modules discriminants différents (MLP,RNN,CAC,BLSTM-CTC) sur différents types de requêtes.

Chapitre 4

Modèles hybrides pour la détection de mots clés et d'expressions régulières

4.1 Introduction

Comme présenté dans le chapitre 1 de cette thèse, la détection et la reconnaissance des entités nommées (EN) est une tâche importante pour la compréhension profonde du document car elles représentent l'information essentielle du document. Nous avons également vu dans le premier chapitre que la détection et la reconnaissance de ces EN est un problème complexe car leur définition dépend fortement de la tâche d'extraction d'information considérée. La conséquence de cette variabilité est que la plupart du temps, il n'est pas possible de considérer un dictionnaire des entités nommées recherchées. La problématique de détection et reconnaissance des entités nommées dans des images de documents ne rentre donc pas dans le cadre "classique" de la reconnaissance de l'écriture.

Afin d'aborder cette problématique, il est donc naturel de s'inspirer des méthodes issues du traitement automatique des langues pour les adapter au cadre de la reconnaissance d'écriture. Nous avons ainsi choisi de nous baser sur la détection d'indices internes et externes [21], décrite en section 1.3.1.1. Rappelons que les indices (également appelées "preuves") peuvent être des mots **amorces** qui traduisent l'apparition d'une EN dans leur voisinage proche, ou des caractéristiques internes d'une EN comme des séquences de chiffres ou des séquences de majuscules. Afin d'illustrer en détails les caractéristiques de ces preuves nous allons nous appuyer sur la Figure 4.1.

Sur cette figure, on observe que l'EN **Yannick HASSAINE** est précédée du mot **Mme** abbréviation du mot "Madame". La séquence **Mme** est donc un indice externe traduisant l'apparition d'une EN de type *nom/prénom*. Il en va de même pour le mot **Allocataire :** qui traduit l'apparition d'une entité nommée de type *numéro d'allocataire*. Concernant les indices internes on peut prendre l'exemple d'une EN de type *nom/prénom* qui est définie par une structure précise, une majuscule suivie d'une séquence de minuscules (prénom) puis une séquence de majuscules (nom de famille). Une séquence simple de majuscules peut aussi traduire l'apparition d'une EN de type *nom de compagnie* comme c'est le cas pour **CAF DE L'AIN**. Il en va de même pour une séquence de chiffres qui peut traduire l'apparition d'une entité nommée de type *numéro de téléphone* ou *numéro d'allocataire*. Comme on peut le voir il s'agit ici uniquement d'indices. Leur apparition ne traduit pas dans tous les cas l'apparition d'une EN, ils peuvent aussi générer la confusion entre deux EN proches comme *numéro de téléphone* et *numéro d'allocataire*.

Remarquons que les séquences amorces sont généralement connues et en nombre fini (contrairement aux EN), il est donc possible de les détecter et les reconnaître comme un problème traditionnel de reconnaissance de texte. La détection de ces séquences dans des images de documents est un problème bien traité dans la littérature ces dernières années, connu sous le nom de "détection de mots clefs", ou "word spotting".

En revanche, une partie des indices ne peut être compilée dans un dictionnaire. C'est le cas des séquences de lettres majuscules, des séquences de chiffres, ou encore de séquences plus contraintes pouvant définir un numéro de client (par exemple 3 lettres majuscules suivies d'une séquence de 4 chiffres, d'un 'A' puis de 3 caractères quelconques) ou une plaque d'immatriculation francaise (4 chiffres suivis de 2 lettres majuscules puis 2 chiffres). Bien que trop nombreuses pour être énumérées dans un lexique, ces séquences respectent une syntaxe précise (aussi appelé motif), qui peut être décrite par une expression régulière [1]. Bien que très utilisées sur les documents électroniques, il n'existe pas à notre connaissance de travaux concernant la détection d'expressions régulières dans des images de documents.

Afin de détecter et de reconnaître l'ensemble des séquences amorces et des indices permettant d'identifier des EN dans des images de documents, nous proposons dans ce chapitre deux contributions. La première est une adaptation de notre méthode hybride CAC/MMC à un système de word spotting pour la détection des amorces. Comme dans les expériences du chapître précédents, ce système est comparé avec de nombreux systèmes discriminants : réseaux de neurones classiques, récurrents, les CAC et bien entendu les BLSTM, afin de proposer une étude comparative la plus complète possible.

1. ou expression rationnelle

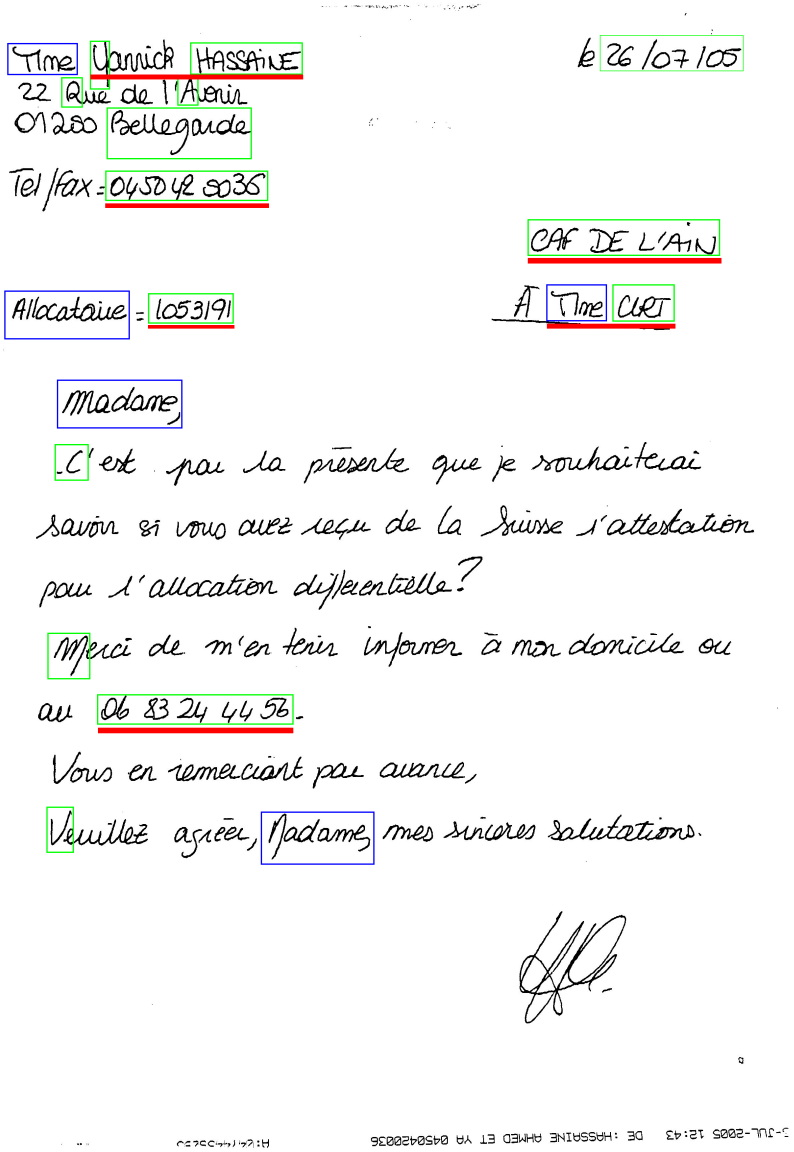


FIGURE 4.1: Exemple de documents à traiter, en rouge les entités nommées à reconnaître en plus les mots amorces en bleu et en vert les expressions régulières amorces

La deuxième contribution est une adaptation des modèles de word spotting classiques pour la détection et la reconnaissance d'expressions régulières, dans le but de détecter les indices tels que des séquences de chiffres. Nous montrons que malgré la difficulté de la tâche, des résultats très intéressants peuvent être obtenus.

Nous commençons par redéfinir la problématique de détection de mots clés et d'expressions régulières à travers une rapide bibliographie, puis nous exposons en détail le modèle MMC de détection de mots clés et d'expressions régulières dans des phrases. Puis nous comparons les performances de ces différents systèmes sur la base Rimes lignes 2011 [2]. Nous finissons par détailler nos systèmes présentés à la compétition ICDAR 2015 sur la détection de mots clés dans des documents historiques. Les performances finales n'ayant

pas encore été publiées lors de la rédaction du manuscrit, nous présentons les résultats sur la base de validation.

4.2 Détection de mots clés et d'expressions régulières

4.2.1 Détection de mots clés

La détection de mots clés dans des documents manuscrits n'est pas un problème nouveau au sein de la communauté, mais la variabilité des données à traiter et des informations à détecter rendent le problème difficile à résoudre. Cette tâche consiste à détecter une information pertinente dans un ensemble de documents. Cette information est généralement constituée d'un ensemble de mots clés. La détection de cet ensemble de mots nous informe sur le contenu du document permettant ainsi d'envisager des applications de haut niveau comme la classification [114] ou l'indexation de documents. La tâche de détection de mots clés se place donc dans la famille des méthodes de compréhension profonde du document.

Les systèmes répondant à cette problématique peuvent être classés en deux catégories distinctes suivant que l'on considère en entrée du système une chaine de caractère ASCII (approches *query by string*) ou une image du mot recherché (approches *query by image* ou *query by example*). Les approches query-by-string nécessitent donc une étape de reconnaissance des caractères, mais sont adaptées au contexte multiscripteur. En opposition, les approches de type query-by-image ne nécessitent pas de reconnaissance, mais sont sensibles aux variations d'écritures et donc généralement limitées à des contextes monoscripteurs. Ce dernier type d'approche est ainsi privilégié sur des corpus monoscripteurs où la reconnaissance est difficile, comme c'est notamment le cas sur les documents historiques. Les approches par reconnaissance sont plus indiquées sur des applications industrielles de type "traitement du courrier entrant".

D'un point de vue méthodologique, les deux types d'approche diffèrent également.

Les approches *query by image* [115–121] procèdent à la détection des mots clés à partir d'une mesure de similarité. Cette similarité peut être calculée entre l'image du mot recherché [116, 122–125] et les images de tous les mots de tous les documents, mais dans ce cas une étape préalable de segmentation du document en mots est nécessaire. Cette segmentation étant parfois délicate, des méthodes dites sans segmentation ont également été développées [115, 126, 127], qui reposent sur la détection de points d'intérêt dans les images de document similaires à ceux de la requête, afin de restreindre l'espace de recherche. Par exemple dans [115], une recherche des régions de correspondance possibles

est effectuée dans le document à l'aide d'une méhode RLSA [128]. Une fois les régions extraites, un ensemble de caractéristiques morphologiques est calculé sur ces différentes régions et de l'image du mot requête. Finalement, une étape de recherche de correspondance est effectuée entre ces différentes régions et l'image requête afin de définir les images correspondant à l'image du mot requête.

Les approches *query by string* [110, 114, 129–133] implémentent une étape de reconnaissance des caractères afin de détecter la séquence recherchée. La plupart des systèmes *query-by-string* sont plutôt basés sur une analyse des lignes de texte sans segmentation préalable en mots [110, 132, 134–137] (modèles dits *segmentation-free*). Afin d'éviter la reconnaissance intégrale des documents considérés, ces approches reposent sur des modèles génériques de ligne de texte permettant une analyse plus superficielle. Pour cela, les modèles de lignes sont constitués d'une part du modèle de la séquence recherchée, et d'autre part de modèles de remplissage (ou *filler models*) pouvant décrire des séquences quelconques. Les modèles de remplissage modélisent les séquences ne correspondant pas au mot à détecter et permettent ainsi au système de s'aligner sur des lignes de texte complètes. Ces modèles spécifiques seront détaillés dans la section 4.3.1. Le décodage des lignes se fait à l'aide d'un algorithme de recherche de meilleur chemin (Viterbi). Deux décodages sont nécessaires : un premier avec un modèle de ligne contenant le mot à détecter, et un second utilisant le modèle de remplissage uniquement. On évalue alors généralement le ratio des scores obtenus à l'issue des deux décodages pour décider si la ligne étudiée contient le mot clé recherché (cf Figure 4.2).

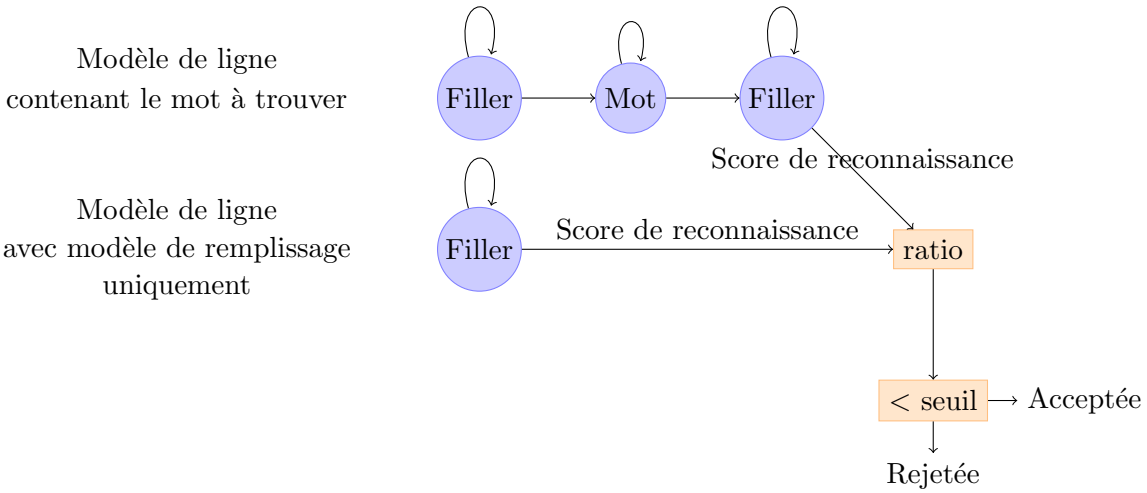


FIGURE 4.2: Fonctionnement d'une approche *query by string* avec modèle de ligne : calcul du score.

Au sein de ces approches, la méthode statistique la plus courante reste à ce jour les MMC [110, 130, 131]. Mais plus récemment, les modèles hybrides ayant montré leur supériorité par rapport aux MMC sur des problèmes de reconnaissance de mots isolés, ils ont naturellement été appliqués à la détection de mots clés [95, 133, 136].

Dans [95], les auteurs combinent le BLSTM avec une couche CTC. A l'aide d'un algorithme de *token passing*, ces probabilités sont contraintes par des connaissances lexicales afin d'obtenir la détection du lexique de mots clés recherchés. Ce système atteint de très bonnes performances sur la base IAM [2]. Dans [136], les auteurs proposent une structure BLSTM-DBN (Dynamic Bayesian Network) pour effectuer la détection de mots clés en reconnaissance de la parole. Comme il est d'usage dans les systèmes hybrides, le BLSTM modélise l'information de bas niveau issue de l'image et fournit un ensemble de caractéristiques à l'étage suivant. Les DBN peuvent être apparentés à des MMC, ce sont des modèles graphiques très utilisés en reconnaissance de la parole qui modèlisent l'évolution d'une séquence à l'aide de modèles de durée. Cet étage permet d'introduire des contraintes issues de modèles de language ou de lexiques afin de corriger les erreurs du premier étage. Dans leurs expériences, les auteurs comparent leur architecture hybride BLSTM-DBN à une approche purement DBN sur le corpus TIMIT [3]. L'ajout du BLSTM augmente les performances de 10%. De par son caractère discriminant et sa capacité à modéliser des dépendances à long terme entre les variables, le BLSTM fournit des décisions locales beaucoup plus robustes que les mélanges de gaussiennes. Cette structure semble montrer théoriquement comme expérimentalement ses capacités à segmenter et reconnaître des caractères avec une très grande précision.

Dans ce chapitre, nous proposons d'adapter le modèle CAC-MMC présenté dans le chapitre précédent pour aborder la problématique de détection de mots clés. Nous évaluons ses performances en comparaison avec les autres méthodes de la litérature, notamment les structures hybrides à base de réseaux de neurones.

La détection d'entités nommées nécessite comme présenté en introduction la détection d'amorces telles que des séquences de majuscules, ou des séquences de chiffres. Nous proposons donc également d'étendre la problématique de la détection de mots clés à la détection d'expressions régulières.

2. Average Precision of 88.15% and R-precision of 84.34%
3. https ://catalog.ldc.upenn.edu/LDC93S1

4.2.2 Détection d'expressions régulières

4.2.2.1 Définition

Une expression régulière est une chaîne de caractères permettant de décrire un ensemble de séquences respectant une syntaxe particulière. Pour cela, la syntaxe des séquences cibles est modélisée par le biais de caractères spéciaux appelés méta-symbole. Les méta-symboles décrivent des sous ensembles de caractères à partir de l'alphabet complet des caractères, comme par exemple les chiffres. Il est également possible de spécifier le nombre de caractères ou de méta-symbole que l'on peut rencontrer. Les expressions régulières peuvent également contenir des opérateurs logiques tels que le "ou" (présence du caractère 'a' *ou* 'b'). L'ensemble de séquences pouvant être modélisé par une expression régulière est donc potentiellement infini.

Les expressions régulières sont des outils très utilisés en informatique, car ils permettent d'accéder rapidement à des séquences cibles dont les valeurs exactes sont inconnues, ou partiellement connues. La détection d'expressions régulières peut être vue comme une généralisation du concept de détection de mots clés abordé précédemment.

Avant d'exposer la bibliographie sur la détection d'expressions régulières, il est nécessaire de détailler l'ensemble des notations permettant de les définir. Il existe de nombreuses manière de formaliser les expressions régulières, qui dépendent souvent du langage informatique utilisé pour en effectuer la recherche.

Dans ce travail, nous avons choisi les notations suivantes :

- # modélise la fin ou le début d'une expression régulière.
- **a** représente le caractère 'a'.
- **[a-z]** représente l'ensemble des caractère compris entre 'a' et 'z'.
- **?** représente 0 ou 1 occurence de n'importe quel caractère.
- ***** représente 1 ou plusieurs occurences de n'importe quel caractère.
- | représente le "ou" logique, c'est à dire un choix entre deux caractères

Les caractères **?** et ***** peuvent être appliqués à des caractères ou à des ensembles de caractères. On peut ainsi modéliser des motifs contenant des séquences de minuscules (#**[a-z]***#), de majuscules (#**[A-Z]***#) ou de chiffres (#**[0-9]***#). Les requêtes peuvent être plus ou moins contraintes par l'ajout de sous-séquences fixes, comme par exemple toutes les séquences commençant par *a* suivi d'au moins une lettre minuscule (#**a[a-z]***#), ou les mots terminant par *tion* (#**[a-z]*tion**#).

En complexifiant les requêtes, on peut alors extraire des informations plus complexes et plus intéressantes comme des entités nommées :

— Détection de dates : **#[0-9]{2}/[0-9]{2}/[0-9]{4}#**
— Détection de prénoms : **#[A-Z][a-z]*#**
— Détection de codes postaux et de noms de ville française : **#[0-9]{5} [A-Z]*#**.

On voit donc tout l'intérêt de la détection d'expressions régulières dans le cadre de la détection d'entités nommées. Cependant, si les expressions régulières sont très largement utilisées sur des documents numériques, leur détection dans des images de documents est un problème beaucoup moins abordé. Nous proposons donc une brève bibliographie sur ce point.

4.2.2.2 La détection d'expressions régulières dans des images de documents

La détection d'expressions régulières est une tâche appartenant au traitement automatique du langage [138–140]. Sur des documents numériques et d'un point de vue algorithmique, la détection d'expressions régulières est généralement réalisée en convertissant l'expression régulière recherchée en un automate à états finis [141, 142]. La recherche peut ensuite être efficacement résolue en visitant l'automate pour ne conserver, pour chaque étape de calcul, que l'ensemble des états atteignables. La recherche se termine lorsque la chaîne d'entrée est épuisée.

Dans le cadre du traitement de documents bruités (imprimés ou manuscrits), la problématique est plus ouverte car une étape de transcription est nécessaire pour obtenir la version ASCII du document à traiter. Par ailleurs, l'étape de reconnaissance engendre automatiquement des erreurs et/ou des incertitudes, qui pénalise la recherche exacte des séquences cibles. En effet, une substitution, une insertion ou une suppression lors de la reconnaissance provoquera une non détection de la séquence. En utilisant une stratégie de reconnaissance suivie d'une recherche d'expression régulière sur la transcription, on peut donc s'attendre à des résultats limités. Néammoins, lorsqu'un lexique est disponible, il est possible d'exploiter les différentes hypothèses locales de reconnaissance, afin de corriger les erreurs. Certains auteurs ont expérimenté ce type d'approche sur des images de documents imprimés [143–145]. Dans leurs travaux, un OCR est appliqué sur l'ensemble des documents avant de passer à l'étape de détection des expressions régulières basées sur un ensemble de règles. Afin de palier les erreurs d'OCR, des méthodes statistiques traitant les séquences sont utilisées comme par exemple les MMC. Ces derniers vont pouvoir donner localement des probabilités d'émissions des caractères (plusieurs possibilités à chaque instant) pour qu'un algortihme de recherche du meilleur chemin puisse s'aligner sur les mots voulus, malgré des erreurs locales. Par exemple, si l'on veut rechercher le mot **"madame"** mais que les probabilités les plus hautes localement donnent la séquence

medane, les erreurs pourront possiblement être corrigées, notamment si les hypothèses de reconnaissance des lettres 'a' et 'm' possèdent des probabilités significatives.

Sur les images de documents manusrits, les travaux sont beaucoup plus rares. Sans toutefois aborder le problème précis des expressions régulières, on peut mentionner les sujets proches comme la détection de dates [146] à l'aide d'un MMC, ou la détection de champs numériques [147, 148] à partir de réseaux de neurones couplés à une programmation dynamique. Bien qu'intéressants, ces travaux sont limités à des types de champs particuliers. A notre connaissance, les seuls travaux concernant un système générique pour la détection d'expressions régulières dans les documents manuscrits sont les travaux de Y.Kessentini [110]. Il s'agit d'une approche basée sur un modèle HMM de ligne contenant des méta modèles de caractères (chiffres, minuscules, majuscules) afin de détecter des expressions régulières quelconques. Cependant, les résultats obtenus sont assez limités et ne permettent pas d'envisager une application industrielle.

Dans la section suivante, nous proposons une adaptation des structures hybrides présentées dans le chapitre précédent pour détecter des expressions régulières dans des images de documents manuscrits. Nous montrons que l'utilisation des approches discriminantes permet d'obtenir un gain de performance très intéressant.

4.3 Systèmes hybrides pour la détection de mots clés et d'expressions régulières

La détection d'expressions régulières (REGEX) est globalement basée sur le même principe que la détection de mots clés dans la mesure où une séquence est recherchée dans les lignes de texte. La différence réside dans le fait que la séquence de caractères à rechercher est moins contrainte. En effet, à une expression régulière correspond un nombre important (possiblement infini) de séquences cibles. Par exemple une requête de la forme **#M[a-z]*#** (tous les mots commençant par la lettre M) décrit des mots de tailles très variables, possédant ou non des hampes et des jambages : "Me", "Mai", "Major", "Monsieur", "Magique", "Malheureusement", etc. Moins l'expression régulière est contraignante, plus le nombre et la variabilité des séquences cibles est important, et plus la tâche de détection est sensible aux erreurs de reconnaissance.

Dans le cadre de cette deuxième contribution, plusieurs méthodes discriminantes vont être testées sur les données de bas-niveau afin de fiabiliser au maximum la reconnaissance des caractères manuscrits. Les méthodes testées sont un perceptron multi-couche, un réseau de neurones récurrents, un CAC et un BLSTM-CTC. Nous conduisons les mêmes

expériences que dans le chapitre 3 appliquées à la détection de mots clés et d'expressions régulières afin de comparer les différentes méthodes utilisées.

4.3.1 Modèle MMC pour la détection de mots clés

L'étage MMC doit être capable de détecter le mot clé quelle que soit sa position au sein de la ligne. Il faut donc lui fournir la capacité de pouvoir s'aligner sur l'ensemble de la ligne. La méthode couramment utilisée est l'ajout de modèle ergodique (filler model) contenant tous les caractères possibles qui représentent l'ensemble des caractères. La distribution des probabilités de transition de ces modèles est soit uniformes soit apprise sur uncorpus représentatif du vocabulaire de l'application. Ces modèles sont placés à gauche et à droite du mot à reconnaître. En modifiant les transitions entre les modèles de remplissage et la séquence du mot clé, on permet au système de détecter la séquence voulue au début, au milieu ou à la fin de la ligne (cf Figure 4.3).

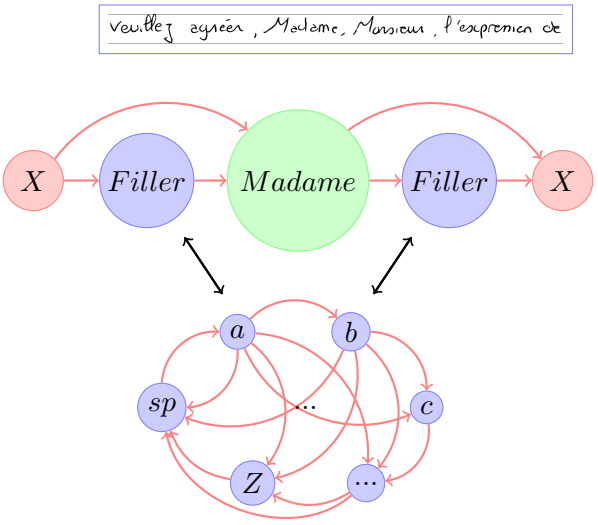


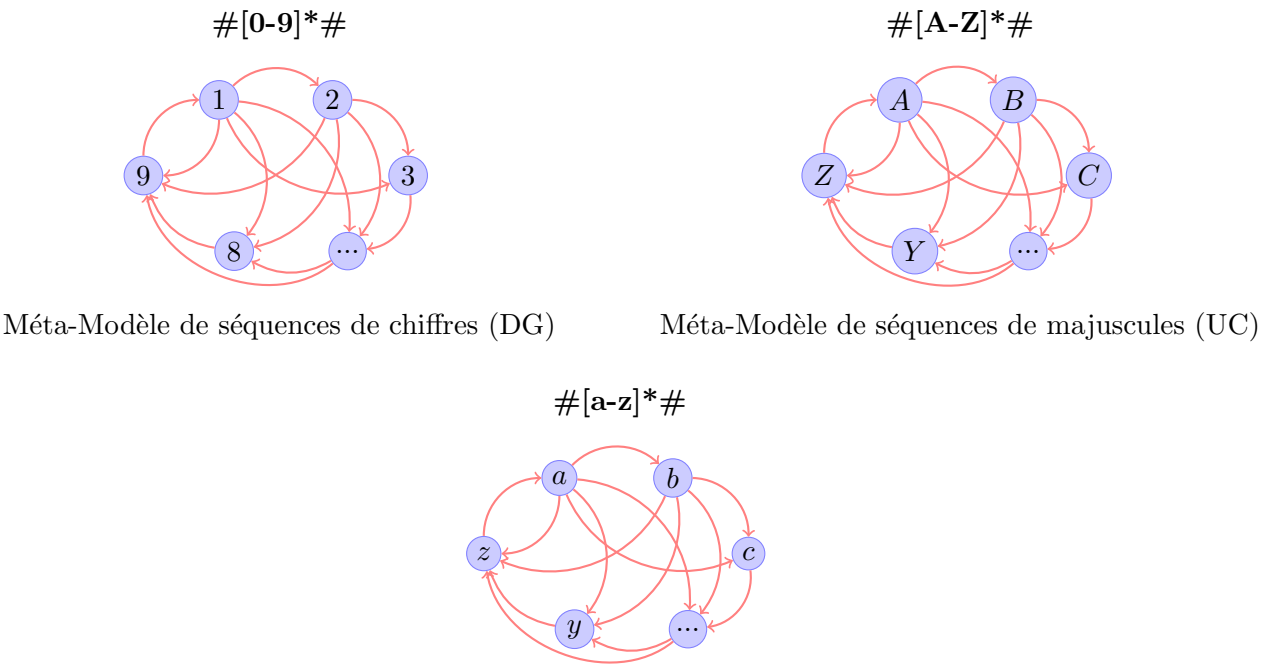
FIGURE 4.3: Modèle de ligne pour la détection de mots clés

Au sein de notre architecture hybride, les probabilités *a posteriori* sont fournies directement à l'étage de haut niveau (cf section 3.2), ainsi ce dernier ne traite que des probabilités et non des vraisemblances. De ce fait, il n'est pas nécessaire d'effectuer le double décodage avec un modèle de filler pour obtenir un ratio comme cela est fait dans les approches purement MMC, car les probabilités sont toujours comprises entre 0 et 1. La décision s'effectue en fonction de la somme des scores de chacun des caractères de la bonne séquence moyennée par le nombre de caractères de la séquence obtenue à la sortie d'un alignement de Viterbi. Ce choix est motivé par la capacité des méthodes

discriminantes à prendre des décisions locales robustes. Ainsi, chacun des étages effectue la tâche sur laquelle il est le plus performant : l'étage discriminant à la modélisation des frontières entre les caractères à partir d'informations locales, et l'étage génératif à la modélisation de séquence en embarquant des contraintes lexicales et des modèles de langage.

4.3.2 Modèle de ligne pour détection d'expressions régulières

Comme mentionné précédemment, la détection d'expressions régulières est une généralisation de la détection de mots clés. Les contraintes lexicales étant relachées, le modèle de ligne doit pouvoir s'aligner sur des séquences de taille et de contenu variables (**#re[a-z]*#**, **#[a-z]*tion#**,etc.) d'où la nécessité de mettre à en place des modèles MMC ergodiques spécifiques appelés méta-modèles (cf Figure 4.4).



Méta-Modèle de séquences de chiffres (DG)

Méta-Modèle de séquences de majuscules (UC)

Méta-Modèle de séquences de minuscules (LC)

FIGURE 4.4: Méta-modèles utilisés dans notre système hybride

Ces méta-modèles sont des modèles ergodiques de caractères contenant tous les caractères des ensembles de type **[a-b]**. Ils permettent de modéliser une partie de la variabilité de l'expression régulière à rechercher. Dans notre cas, nous utilisons majoritairement trois méta-modèles différents :

1. Minuscules : **#[a-z]*#**

2. Majuscules : #[**A-Z**]*#
3. Chiffres : #[**0-9**]*#

Précisons que nous avons implémenté ces trois métamodèles, mais qu'il est évidemment possible de réaliser tous les modèles d'ensemble, tels qu'un modèle de voyelle par exemple (**a|e|i|o|u|y**)

Chacun des méta-modèles inclut en plus le modèle d'espace afin de pouvoir modéliser la transition entre les mots. La séquence étant de taille variable, la longueur de cette dernière est controlée par l'auto-transition sur le ou les méta-modèles utilisés pour la détection. La séquence étant aussi de contenu variable, les méta-modèles nous permettent de modéliser cette variabilité de contenu. Par exemple si l'on veut détecter les mots commençant par la séquence *Ma* (#**Ma[a-z]***#), le système doit pouvoir localiser la position des mots *Madame*, *Magie*, *Manoir*, *Magistrat*, etc. Ces modèles sont très importants car ils vont nous permettre de détecter les mots *amorces* nécessaires à la détection des EN comme les dates, les prénoms, les noms de famille, les codes postaux, les numéros de référence , etc.

Les Figures 4.5, 4.6 et 4.7 présentent des exemples de modèles de ligne pour la détection de REGEX. Ces derniers permettent de détecter une expression au début de la ligne (#[**a-z**]***ion**#), au milieu de la ligne #[**A-Z**]**on**[**a-z**]*# ou à la fin de la ligne (#**agre**[**a-z**]*#). Ce modèle de ligne est également capable de détecter des expressions regulières moins contraintes comme des séquences de chiffres (#[**0-9**]*#) ou une séquence de lettres commençant par une majuscule et terminant par une séquence de minuscules de taille quelconque (#[**A-Z**][**a-z**]*#). La longueur de la séquence non contrainte (*) est définie par les auto-transitions du MMC et les probabilités *a posteriori* extraites de l'étage discriminant.

Les probabilités de transitions dans les méta-modèles du MMC étant uniformes l'alignement du Viterbi sera dirigé uniquement par les décisions discriminantes locales fournies par l'étage discriminant.

4.3.3 Chaîne de traitement

La chaîne de traitement utilisée est composée de pré-traitements classiques, d'une extraction de caractéristiques et des méthodes statistiques de reconnaissance décrites dans le chapitre précédent (cf section 3.4) pour la reconnaissance de mots isolés.

Les pré-traitements appliqués sur les images se décomposent en trois étapes. Premièrement nous appliquons une binarisation de Sauvola [149], puis nous appliquons une méthode

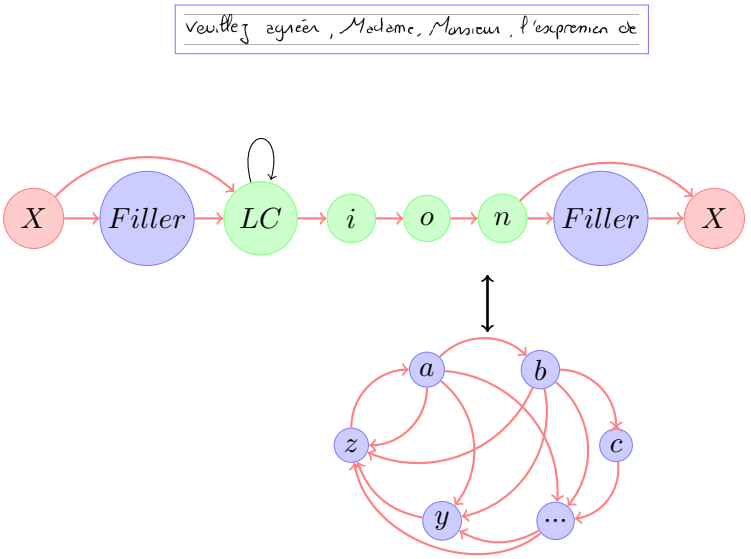


FIGURE 4.5: Étage MMC : modèle de ligne pour la détection de la séquence **#[a-z]*ion#**

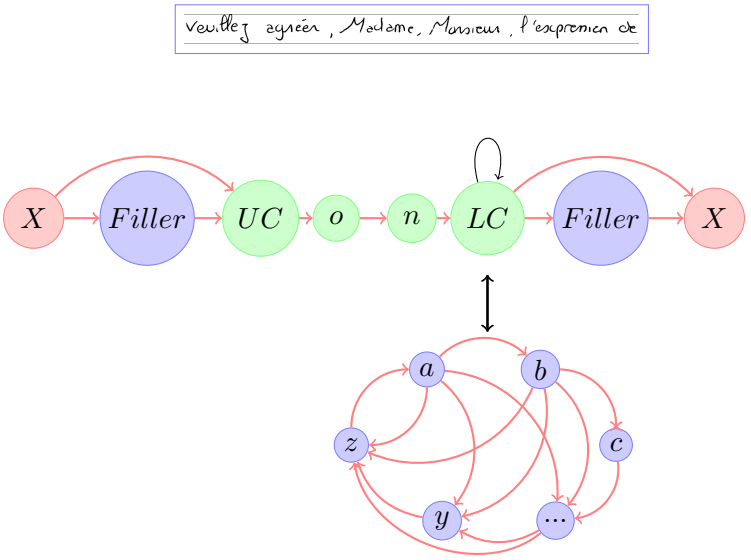


FIGURE 4.6: Étage MMC : modèle de ligne pour la détection de la séquence **#[A-Z]on[a-z]*#**

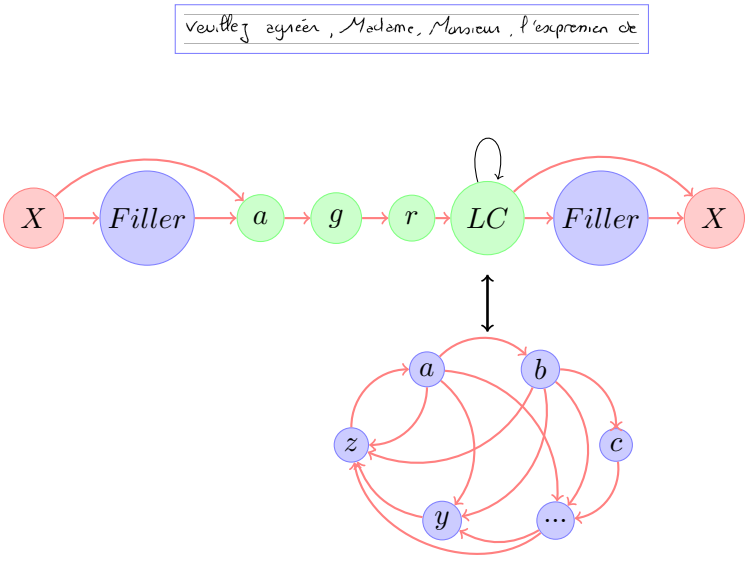


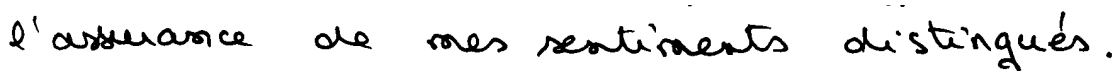
FIGURE 4.7: Étage MMC : modèle de ligne pour la détection de la séquence **#agre[a-z]*#**

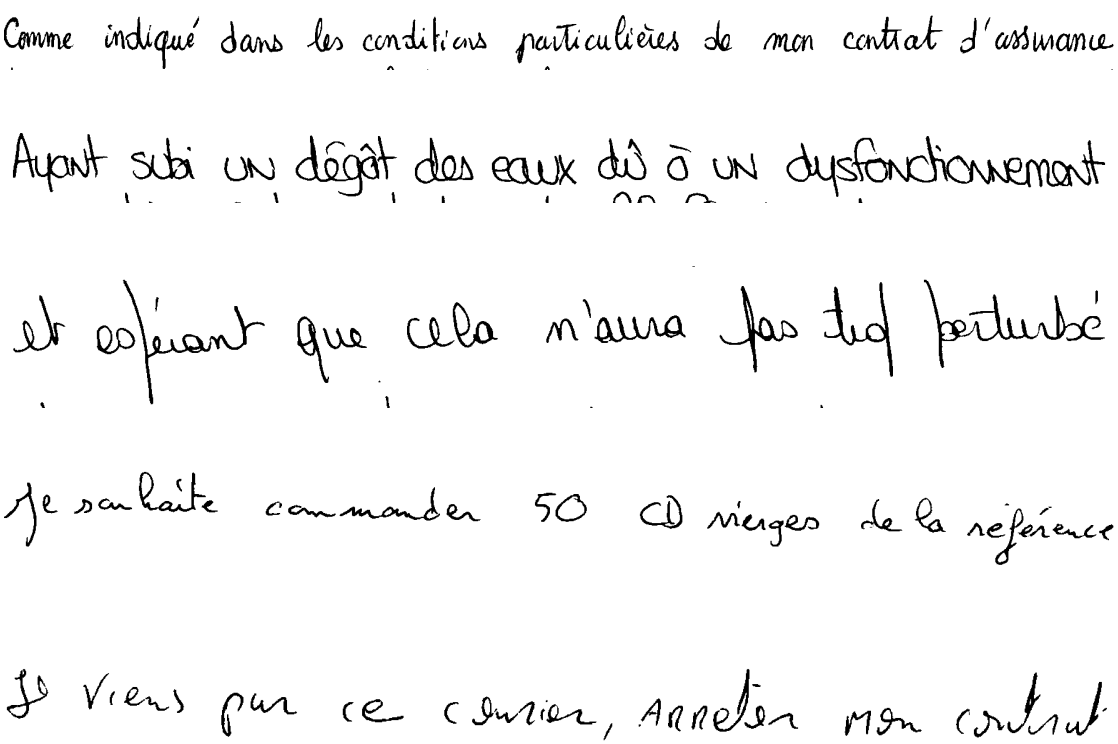
de redressement basée sur le principe suivant : l'image est pivotée suivant un angle variant de $-\Theta$ à Θ, pour chacun de ces angles un histogramme des traits verticaux H_Θ est calculé. Finalement, l'image est normalisée en hauteur à 64 pixels afin de centrer les hampes et les jambages par rapport à la ligne de base.

Pour réaliser cette expérience, nous avons choisi d'utiliser les Histogrammes de Gradients Orientés [109]. Ces derniers ont été extraits d'une fenêtre glissante de taille 8x64 pixels. Chaque fenêtre est divisée en sous-fenêtre de 4x16 pixels. Dans chacune de ces sous-fenêtres, un histogramme d'intensité du gradient est calculé suivant 8 directions différentes. Ces 16 valeurs d'histogramme sont finalement concaténées en un seul vecteur de caractéristiques de 64 dimensions.

4.4 Expériences et résultats

L'évaluation des performances de nos systèmes a été effectuée sur la base de phrases RIMES 2011 de la compétition de reconnaissance d'écriture manuscrite ICDAR [2]. La base d'apprentissage est composée de 1500 documents et les bases de validation et de test de 100 documents chacune. Quelques exemples de cette base sont montrés ci-dessous :





Il s'agit de lignes d'écriture manuscrite multi-scripteur sujettes aux inclinaisons de lignes et de mots. Certaines sont également mal segmentées, des artefacts des lignes suivantes dessous ou précédentes sont parfois présentes. Cette base nécessite donc la mise en place d'un ensemble de pré-traitements avant de passer à la phase d'apprentissage et de test.

L'évaluation se fera suivant les valeurs de Rappel (R) et de Précision (P) en comptant les vrais positifs (TP) les faux positifs (FP) et les faux négatifs (FN) sur l'ensemble des documents de la base de test. En faisant varier le seuil de rejet, nous obtenons une courbe rappel-précision à partir de l'accumulation de toutes ces valeurs :

$$R = \frac{TP}{TP + FN} \quad P = \frac{TP}{TP + FP}$$

Nous avons ainsi la possibilité de faire varier notre seuil en fonction des besoins de l'utilisateur. Ce dernier pourra s'il le souhaite favoriser la précision au détriment du rappel ou inversement. Afin de pouvoir analyser correctement les résultats obtenus il faut que les occurences dans la base de test des mots à détecter soient assez élevées. Or dans ce type de courrier, certains mots ne dépassent pas 5 ou 6 occurences dans l'ensemble de la base. Pour palier ce problème et obtenir une estimation satisfaisante, nous proposons de détecter non pas un seul mot clé mais un lexique de mots clés. Dans le cadre des expressions régulières, ce problème de manque d'occurence est moins

marqué puisqu'une requête correspond à plusieurs mots clés. Ainsi, moins la requête sera contrainte, plus le nombre d'occurence sera grand.

4.4.1 Résultats pour la détection de mots clés

Dans certains travaux, la détection de mots clés est effectuée sur une seule requête (un seul mot) à chercher dans un ensemble de documents. Dans nos expériences, nous avons décidé d'évaluer notre structure hybride sur la détection d'un ensemble de mots clés (lexique de mots clés) en même temps, afin de pouvoir évaluer la robustesse de notre structure à la confusion entre les mots [110]. Cette généralisation à un lexique de mots clés permet également d'avoir une idée des performances de notre système sur la catégorisation de documents [114]. Nous avons évalué cinq systèmes différents sur des lexiques de taille 25, 50 et 100 mots clés, contenant les mêmes mots que dans les travaux de [110] afin de pouvoir comparer les systèmes entre eux.

Les Figures 4.8, 4.9 et 4.10 montrent les courbes rappel-précision obtenues pour l'ensemble des cinq systèmes sur chacun des lexiques (25, 50 et 100 mots clés). On observe sans surprise que les résultats globaux des cinq méthodes diminuent en fonction de la taille du lexique utilisé. On remarque aussi que la structure BLSTM-CTC-MMC obtient les meilleurs résultats avec un *break-even point* au alentour de 85 % (point où rappel=précision). Elle est suivie dans l'ordre par le système RNN-MMC, le CAC-MMC, le MLP-MMC et finalement le MMC classique. Ces résultats nous permettent de voir l'apport des différentes méthodes discriminantes lors de la décision finale (détection du mot clé) par rapport à une approche purement générative. Ces résultats nous permettent également de comparer les différentes méthodes discriminantes entre elles. Comme cela a déjà été démontré dans d'autres travaux, l'architecture BLSTM-CTC-MMC est de loin la meilleure. On voit également que le CAC offre de meilleures décisions qu'un MLP standard (5%). On peut aussi observer qu'une structure récurrente classique (RNN-MMC) dépasse les performances du CAC d'environ 5%. Il semble donc que le contexte temporel apporté par la récurrence soit un atout important dans la reconnaissance de données séquentielles. Remarquons qu'assez logiquement, le classement des systèmes est le même que dans le chapitre précédent.

Pendant la phase d'apprentissage, il est important de contrôler la convergence des systèmes afin d'éviter le sur-apprentissage. Afin de vérifier que l'erreur niveau trame est un bon critère de contrôle de convergence, nous avons évalué la relation entre cette dernière (une fois que la méthode a convergé) et le break-even point de chaque méthode (résultat final). Dans cette expérience, nous calculons la moyenne des break-even point

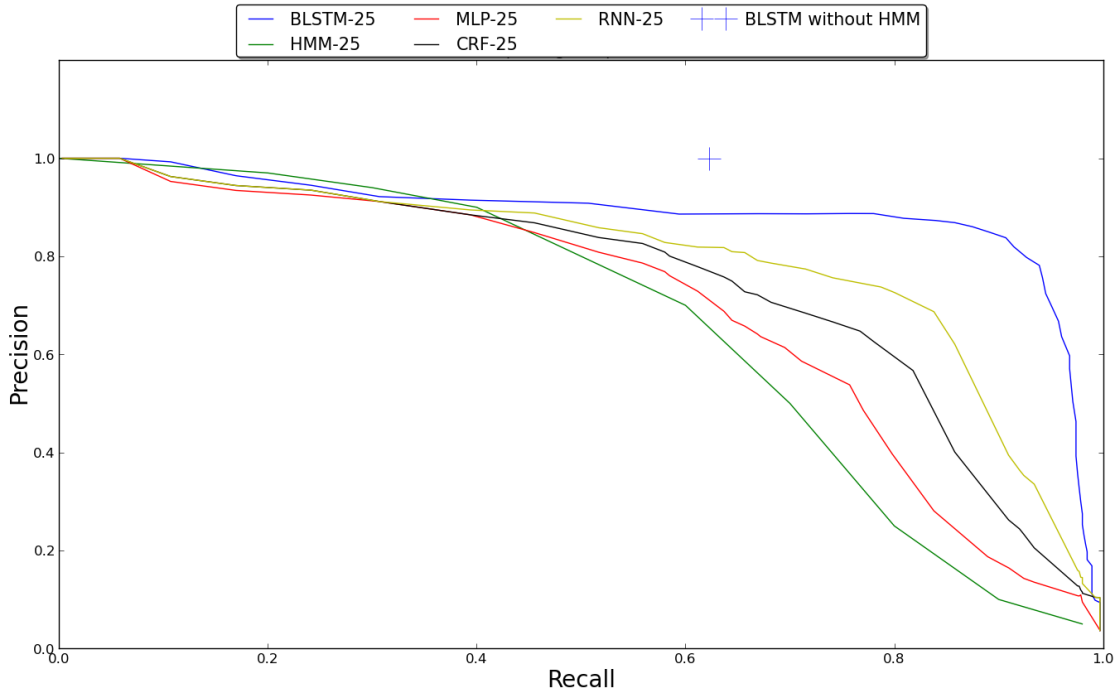


FIGURE 4.8: Performances de l'ensemble des systèmes sur un lexique à 25 entrées

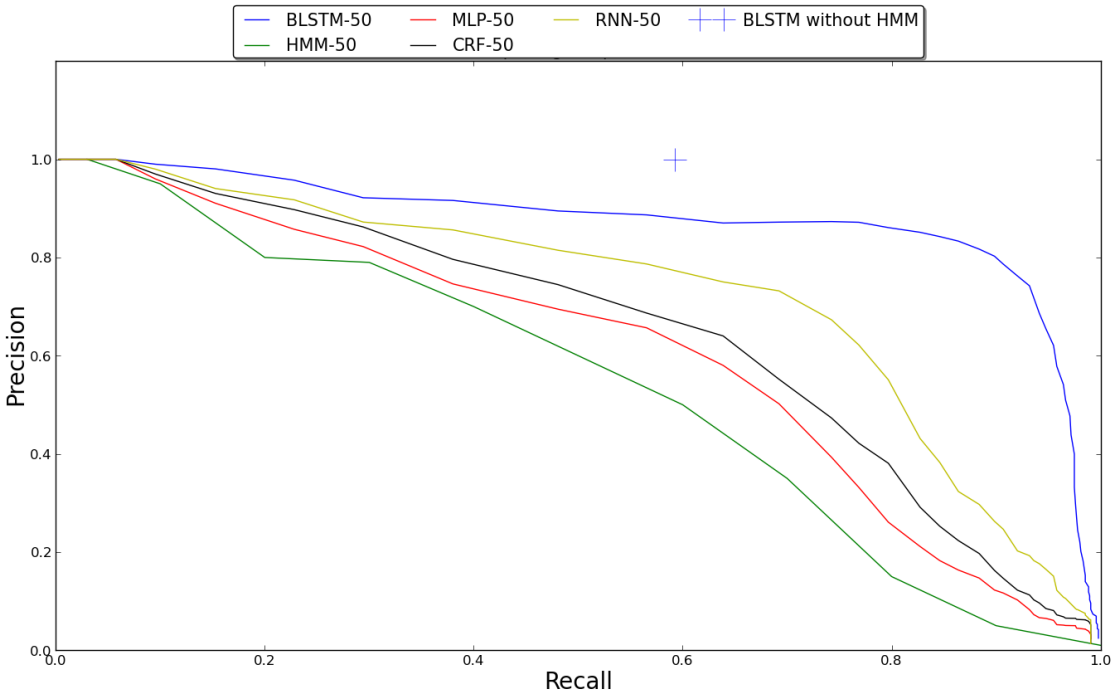


FIGURE 4.9: Performances de l'ensemble des systèmes sur un lexique à 50 entrées

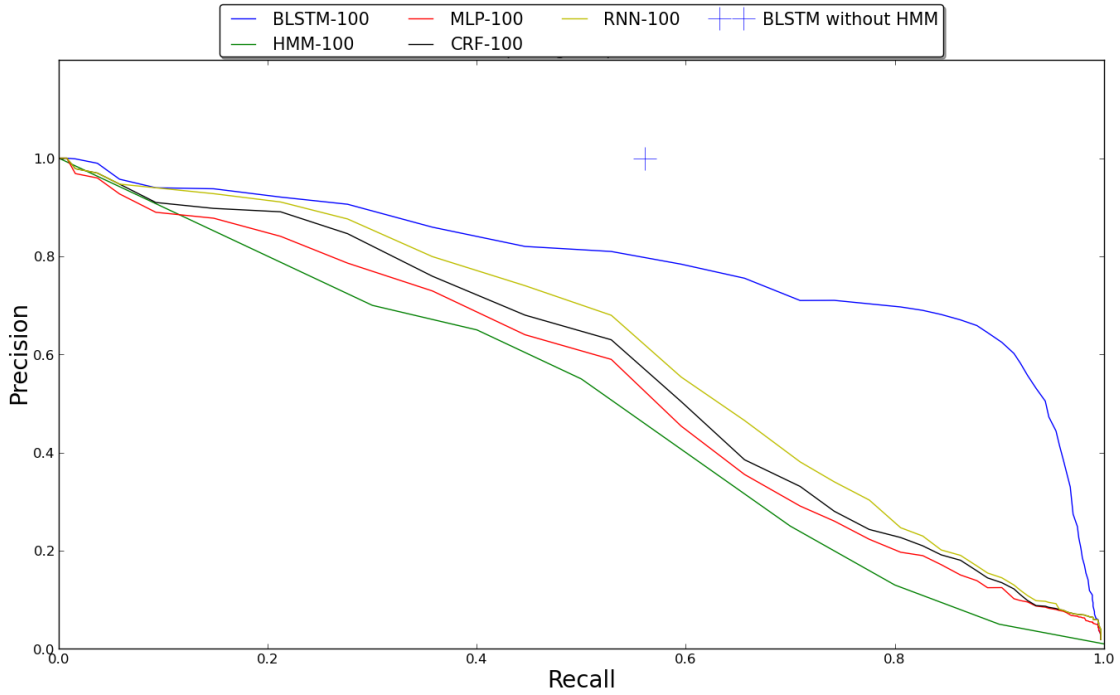


FIGURE 4.10: Performances de l'ensemble des systèmes sur un lexique à 100 entrées

des cinq méthodes étudiées sur l'ensemble des 3 lexiques (25,50,100). L'ensemble des résultats est illustré dans la Figure 4.11.

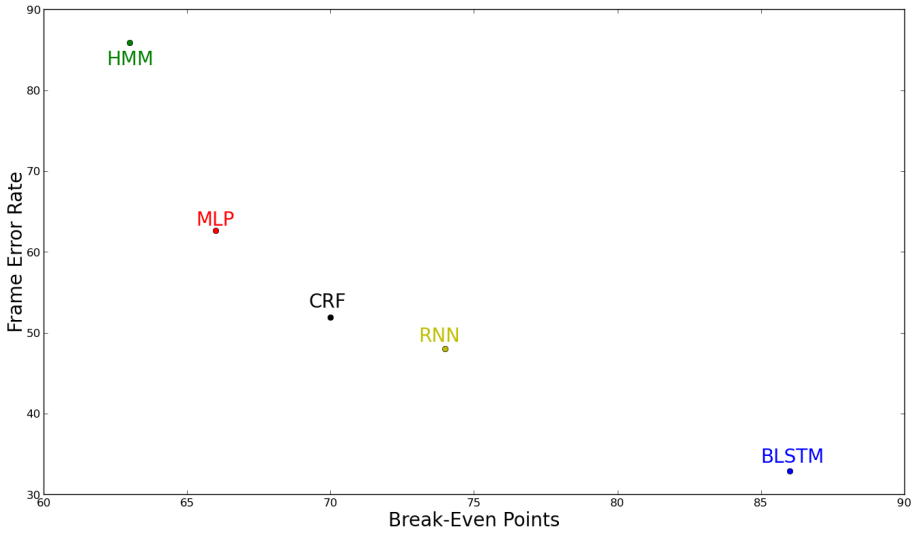


FIGURE 4.11: Erreur moyenne niveau trame en fonction du break even point moyen

On peut voir que l'erreur niveau trame est directement liée aux performances globales des systèmes. Plus l'erreur niveau trame est basse, plus le break-even point sera grand.

On peut en conclure que l'erreur niveau trame sur cette base de données est un bon critère de contrôle de qualité de l'apprentissage.

4.4.2 Détection de mots clefs par matching exact

Comme discuté dans l'introduction de ce chapitre, les systèmes de détection de mots clefs (et d'expressions régulières) appliqués sur des images de documents cherchent à exploiter plusieurs hypothèses de reconnaissance. De cette manière, certaines erreurs de reconnaissance de caractères peuvent être rattrappées en alignant les hypothèses sur les mots du lexique. Cependant, cette correction s'effectue au détriment d'une précision moindre. En effet, si certaines séquences sont corrigées à bon escient, des fausses détections sont susceptibles d'être générées par des alignements de mauvaises séquences sur des mots du lexique proche. Devant les hautes performances affichées par le BLSTM, nous avons essayé d'appliquer une stratégie sans décodage avec lexique, puis d'appliquer une recherche exacte du mot dans la transcription brute.

L'intérêt de cette méthode est qu'elle maximise la précision au détriment du rappel. En effet, l'absence de correction interdit les bonnes corrections (limitation du rappel), mais aussi les mauvaises (maximisation de la précision). Dans la pratique, on observe des précisions égales à 100%. En effet, les fausses alarmes sont très peu probables car elle ne peuvent être générées que par une erreur de reconnaissance qui coïnciderait avec la requête effectuée. Finalement, seul le rappel varie, quelle que soit la méthode utilisée (détection ou non).

L'ensemble des résultats des méthodes MLP-MMC, CAC-MMC, RNN-MMC et BLSTM-CTC-MMC est résumé dans les Tableaux 4.1, 4.2, 4.3 pour des lexiques à 25, 50 et 100 mots clefs. Ces résultats sont également reportés sur les Figures 4.8, 4.9 et 4.10, par des croix bleues.

TABLE 4.1: Résultats de la détection de mots clés sans correction sur un lexique à 25 mots clés.

Méthode	Taille du lexique	Rappel	Précision
MLP-MMC	25	5.2%	100%
CAC-MMC	25	8.5%	100%
RNN-MMC	25	10.2%	100%
BLSTM-CTC-MMC	25	**62.3 %**	100 %

La première remarque est que les système basés sur les MLP, CAC et RNN ne sont pas viables, proposant des rappels en dessous des attentes industrielles (R<10%). En revanche, les performances obtenues par le BLSTM-CTC sont assez surprenantes, car

TABLE 4.2: Résultats de la détection de mots clés sans correction sur un lexique à 50 mots clés.

Méthode	Taille du lexique	Rappel	Précision
MLP-MMC	50	3.7%	100%
CAC-MMC	50	7.3%	100%
RNN-MMC	50	8.9%	100%
BLSTM-CTC-MMC	50	**59.3 %**	100 %

TABLE 4.3: Résultats de la détection de mots clés sans correction sur un lexique à 100 mots clés.

Méthode	Taille du lexique	Rappel	Précision
MLP-MMC	100	2.2%	100%
CAC-MMC	100	4.8%	100%
RNN-MMC	100	6.2%	100%
BLSTM-CTC-MMC	100	**56.1 %**	100 %

des rappels corrects sont obtenus (entre 56% et 62% selon la taille du lexique), pour des précisions de 100%. Rappelons que pour une tâche de catégorisation de documents, il a été montré qu'il vallait mieux détecter peu de mots de manière sûre, plutôt que de nombreux mots avec des erreurs [114]. Une précision parfaite devrait donc profiter largement à ce type d'applications.

Une autre remarque concerne le temps de calcul. En effet, le temps de traitement est beaucoup plus rapide puisqu'aucun décodage n'est effectué après la phase de reconnaissance par le BLSTM.

Dans la plupart des applications, les performances globales d'un système peuvent être grandement améliorées par l'ajout d'un modèle de langage, d'un lexique ou toutes autres informations de haut niveau. Ici nous avons fait le choix de ne permettre aucune correction en appliquant simplement un méthode de matching exacte. Contre toute attente, les résultats obtenus nous situent largement au dessus des courbes rappel-précision des systèmes avec décodage (voir Figures 4.8, 4.9 et 4.10).

4.4.3 Résultats pour la détection d'expressions régulières

Cette section présente les résultats de nos système hybrides pour la détection d'expressions régulières. Afin de comparer nos résultats à une méthode de référence, nous avons effectué les mêmes expériences que dans [110], qui sont à notre connaissance les seuls travaux concernant la détection de REGEX dans des documents manuscrits. Dans

cette étude, les auteurs se sont intéressés à la détection de quatre expressions régulières différentes, les sous-chaînes : *effe* (**#effe[a-z]*#**), *pa* (**#pa[a-z]*#**), *com* (**#com[a-z]*#**)) et *cha* (**#cha[a-z]*#**). Les systèmes ont été testé, ces comparaisons sont illustrées dans les Figures 4.12, 4.13, 4.14 et 4.15.

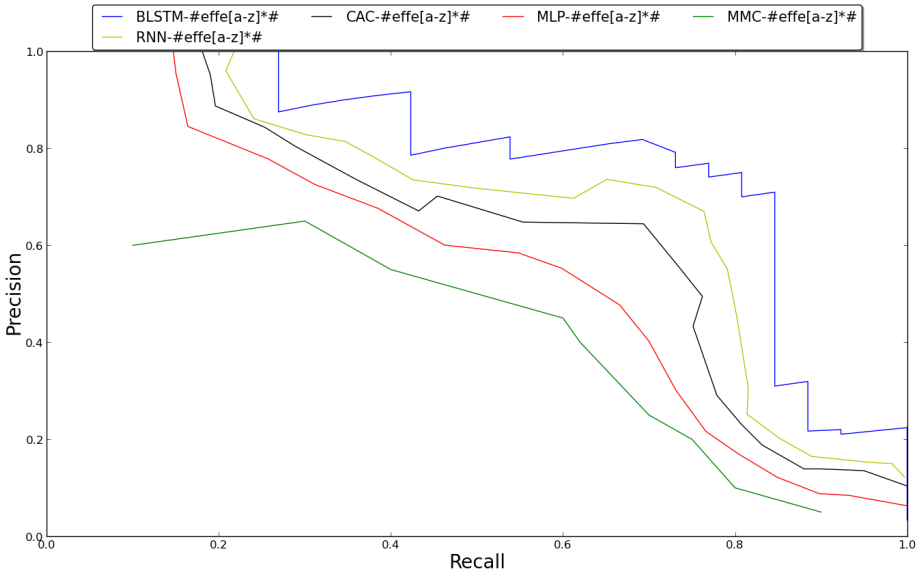


FIGURE 4.12: Performances de la détection des expressions régulières : **#effe[a-z]*#**

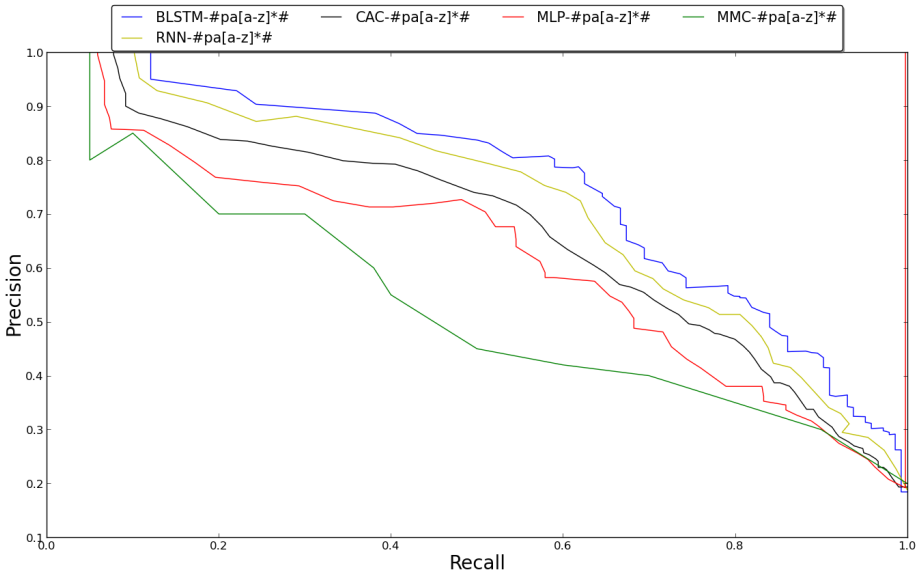


FIGURE 4.13: Performances de la détection des expressions régulières : **#pa[a-z]*#**

On observe que le système BSLTM-CTC-MMC obtient de très bonnes performances. En effet, on observe des précisions moyennes autour de 75% alors que les requêtes demandées sont très peu contraintes par rapport à une tâche de détection de mots clés. En

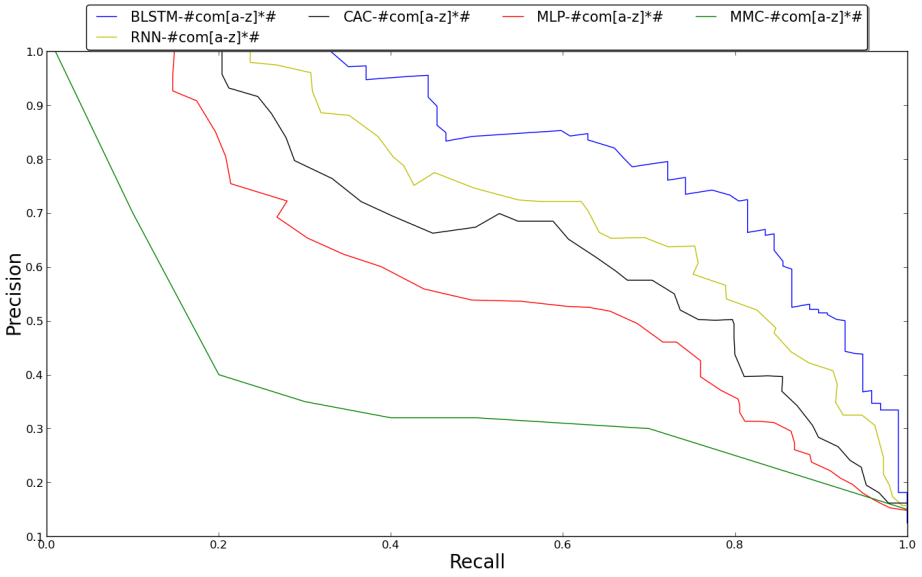


FIGURE 4.14: Performances de la détection des expressions régulières : **#com[a-z]*#**

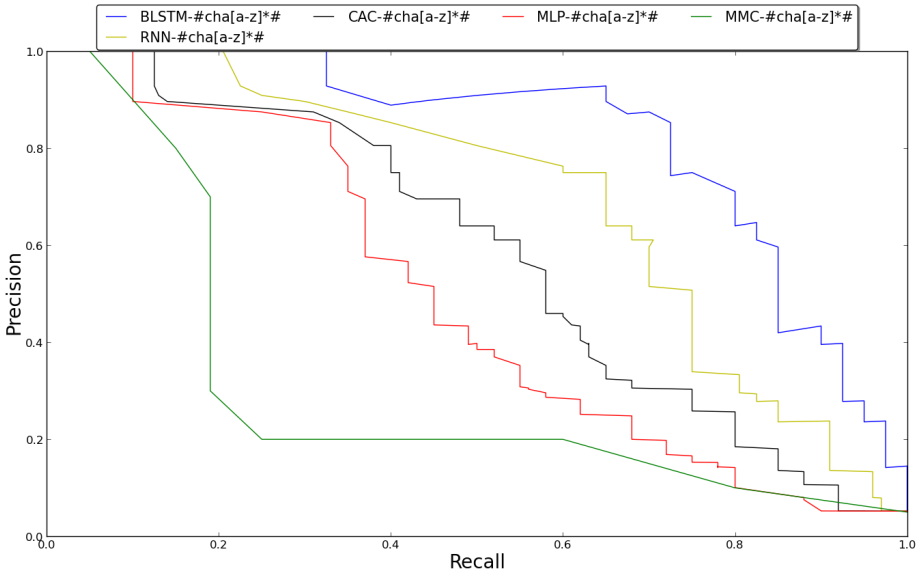


FIGURE 4.15: Performances de la détection des expressions régulières : **#cha[a-z]*#**

effet, l'expressions régulières **#cha[a-z]*#** correspond à la détection d'un lexique vaste contenant des mots comme : chat, chaque, chaussettes, changer, changement, chamallow, etc.

La tendance globale des quatre courbes montrent le classement suivant dans l'ordre décroissant de performances : BLSTM-CTC, RNN, CAC, MLP et MMC. On observe des écarts de 20 à 40% entre la méthode la plus performante le BLSTM/CTC et la moins performante le MMC. Le CAC est toujours meilleur que le MMC et le MLP avec des écarts de performances de 10 à 20% avec le MMC et entre 5 et 10% avec le MLP. Mais comme pour la détection de mots clés, les performances des systèmes récurrents sont supérieures.

Nous avons également effectué d'autres expériences sur des requêtes différentes comme **#[a-z]*er#**,**#[a-z]*tion#**, **#[a-z]*tt[a-z]*#** et **#[a-z]*mm[a-z]*#**. Les résultats sur les requêtes **#[a-z]*mm[a-z]*#** et **#[a-z]*tion#** (cf Figure 4.17 et 4.18) restent bons, on observe le même classement au niveau performance et les mêmes écarts entre les systèmes. Cependant, l'ensemble des systèmes semble avoir des difficultés à traiter les deux autres requêtes. Ceci est certainement dû au fait que deux lettres identiques consécutives comme le *t* sont très difficiles à détecter avec précision. En effet, l'extraction se faisant par fenêtre glissante, deux trames de deux *m* ou deux *t* consécutifs sont très difficile à différencier temporellement, elles sont souvent fusionnées pour ne constituer qu'une seule occurence de la lettre. Les difficultés sur l'autre requête sont quant à elles certainement dû au fait que la lettre *r* peut-être confondue avec un grand nombre de caractères (*n*,*u*,...). L'ensemble des résultats obtenus sont résumés dans les Figures 4.16, 4.17, 4.18 et 4.19.

Nous avons également testé des requêtes très peu contraintes comme la détection d'une séquence aléatoire de majuscules (**#[A-Z]*#**) et de chiffres (**#[0-9]#**). Le traitement de ce type de requête est bien plus complexe que les précédentes car dans ce cas on dispose de très peu de contraintes de taille et de contenu. En effet, la requête (**#[0-9]#**) devra détecter aussi bien *1* que *0123456789*. L'ensemble des résultat est présenté dans les Figures 4.20 et 4.21.

On demande donc au système de reconnaître des entités complexes car les chiffres sont très peu présents dans la base d'apprentissage et on ne lui donne aucune contrainte de taille ni de contenu. L'ensemble des chiffres doit donc être reconnu correctement si l'on veut que la détection soit considérée acceptable. La problématique est la même pour les séquences de majuscules elles aussi très peu présentes dans la base d'apprentissage. On peut observer que la détection de séquences de majuscules peut atteindre un bon niveau de précision (cf Figure 4.20) alors que la séquences de chiffres a quant à elle atteint un niveau intéressant de rappel (cf Figure 4.21).

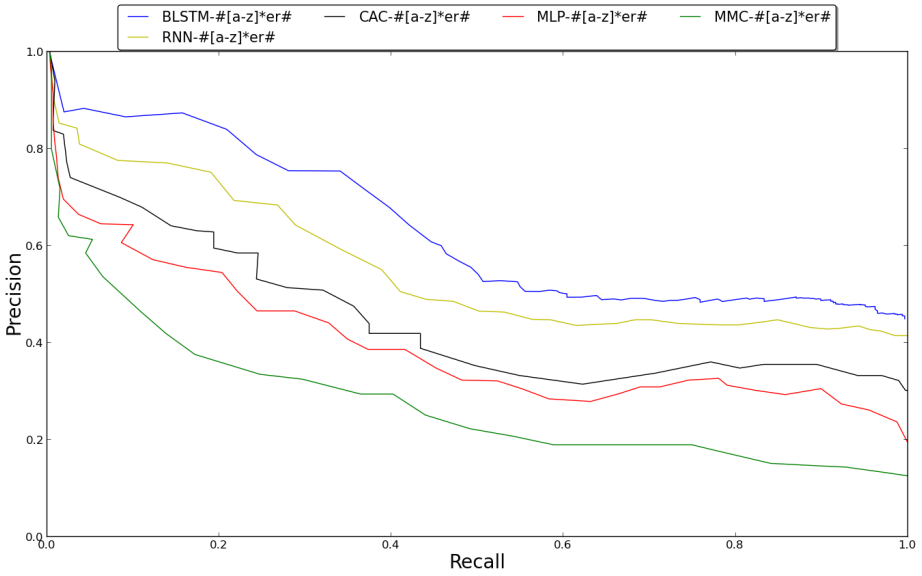


FIGURE 4.16: Performances de la détection des expressions régulières : **#[a-z]*er#**

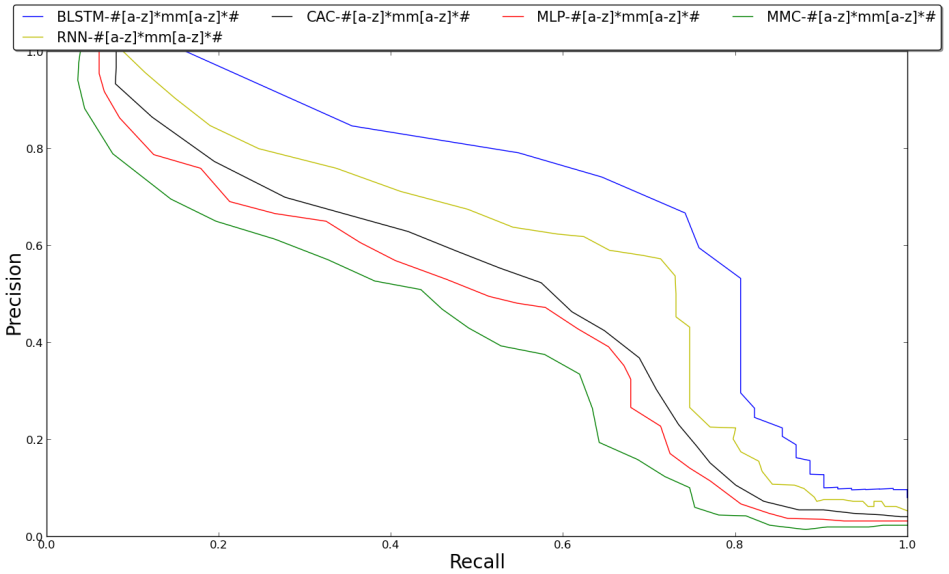


FIGURE 4.17: Performances de la détection des expressions régulières : **#[a-z]*mm[a-z]*#**

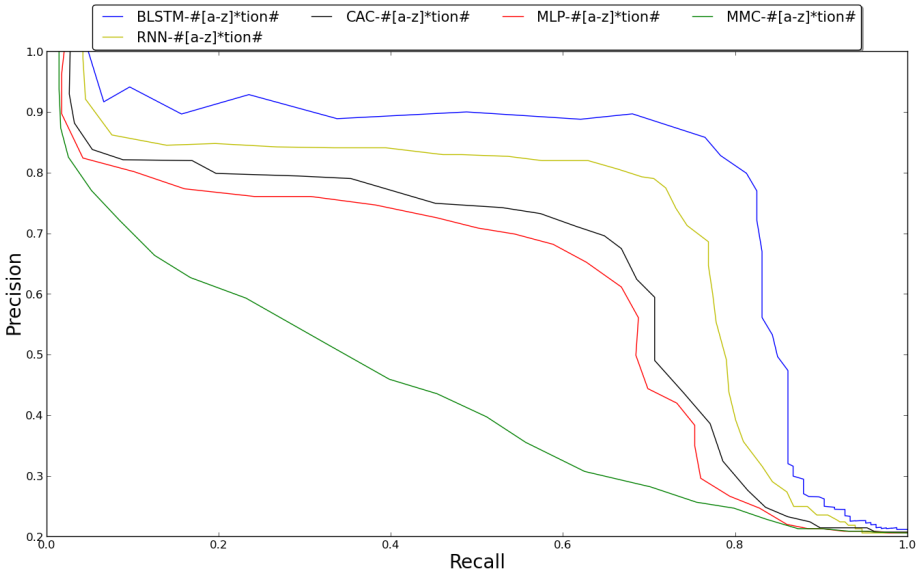


FIGURE 4.18: Performances de la détection des expressions régulières : **#[a-z]*tion#**

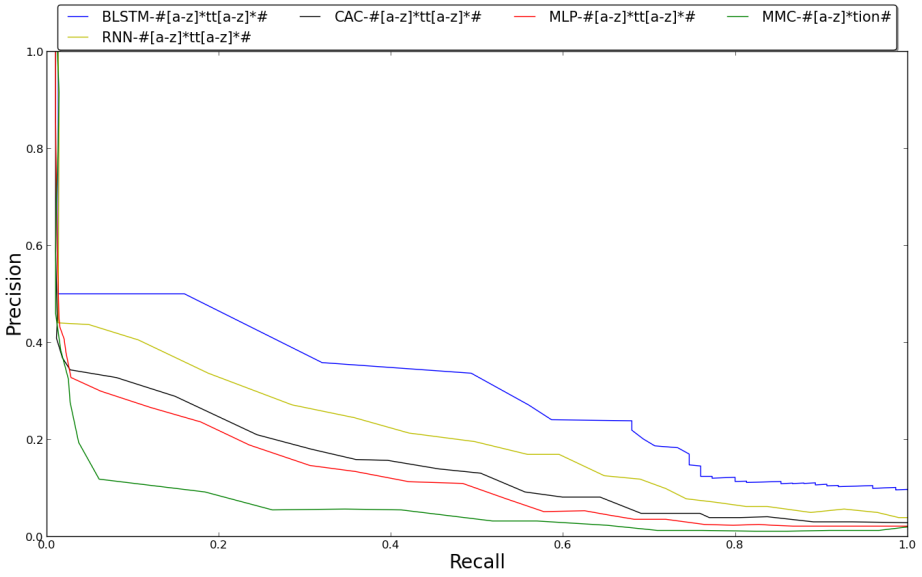


FIGURE 4.19: Performances de la détection des expressions régulières : **#[a-z]*tt[a-z]*#**

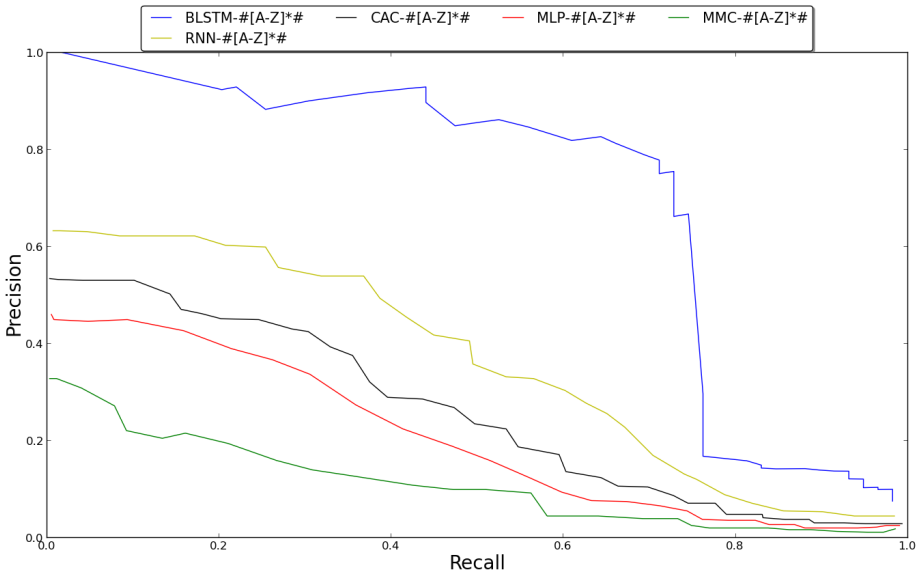


FIGURE 4.20: Performances de la détection des expressions régulières sur une séquence aléatoire de majuscule (#[**A-Z**]*#)

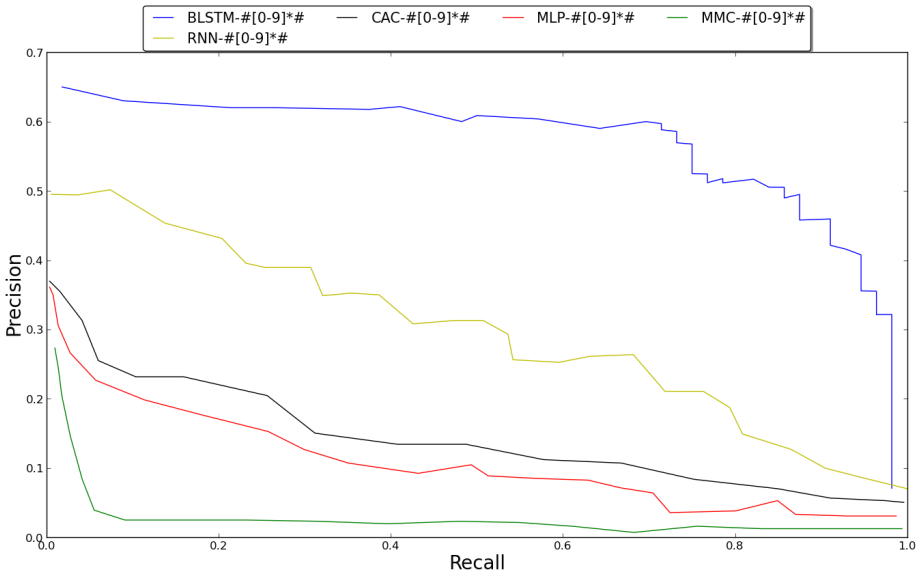


FIGURE 4.21: Performances de la détection des expressions régulières sur une séquence aléatoire de chiffres (#[**0-9**]#)

Dans les deux cas, c'est une nouvelle fois le BLSTM-CTC qui domine nettement les autres approches. Ces résultats sont très encourageants car la base d'apprentissage comporte peu de chiffres et de majuscules. On peut donc penser que l'architecture BLSTM-CTC pourrait obtenir des résultats encore supérieurs en fournissant davantage d'exemples.

4.5 Application : ICDAR 2015 Handwriting Word Spotting Competition

Afin d'évaluer les capacités de l'architecture BLSTM-CTC-MMC sur d'autres corpus, nous avons choisi de participer à la compétition ICDAR 2015 sur la base mono-scripteur manuscrite **Bentham collection**.

La base fournie pour cette compétition est composée de notes et de brouillons de publications ou de travaux en cours, de lettres adressées à ou rédigées par Jeremy Bentham, etc. Elle contient 60 000 documents de taille diverses écrits à différentes périodes de la vie de l'auteur. Les images sont très bruitées, on trouve beaucoup d'artefacts ou des lignes mal segmentées avec des hampes ou des jambages des lignes voisines, ainsi que des mots soulignés ou raturés.

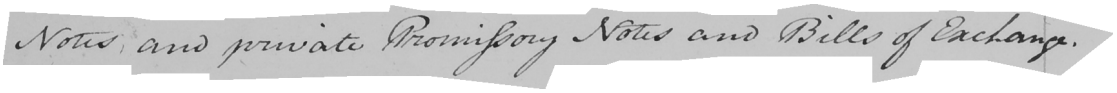


FIGURE 4.22: Exemple d'image du corpus de la compétition ICDAR 2015

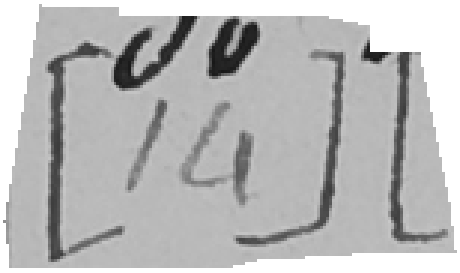


FIGURE 4.23: Exemple d'image du corpus de la compétition ICDAR 2015

Au sein de cette compétition nous avons choisi de participer à deux évalutations, la première sur la détection de mots clés à partir d'une requête ASCII (query by string) et la deuxième sur la détection de mots clés à partir d'images de mots (query by example).

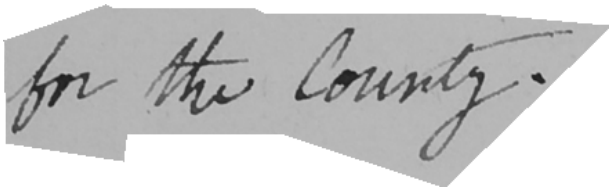


FIGURE 4.24: Exemple d'image du corpus de la compétition ICDAR 2015

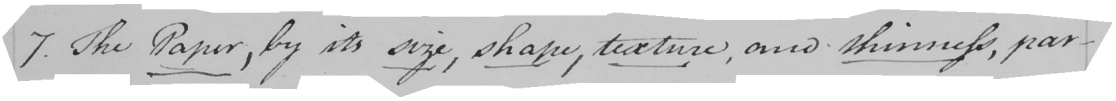


FIGURE 4.25: Exemple d'image du corpus de la compétition ICDAR 2015

4.5.1 Systèmes proposés

La base fournie étant manuscrite multi-scripteurs, nous avons mis en œuvre un certain nombre de pré-traitements afin de réduire la variabilité de l'écriture et faciliter ainsi la reconnaissance. Dans le cadre de cette compétition, nous avons choisi d'appliquer une binarisation adaptative gaussienne, un algorithme de correction d'inclinaison de lignes et de caractères comme proposé dans l'article [150].

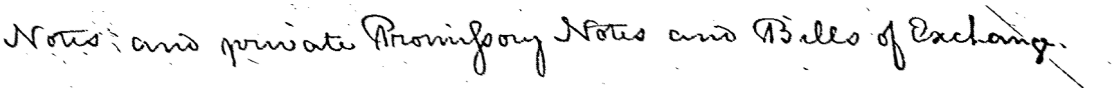


FIGURE 4.26: Exemple d'image du corpus de la compétition ICDAR 2015 après pré-traitements

FIGURE 4.27: Exemple d'image du corpus de la compétition ICDAR 2015 après pré-traitements

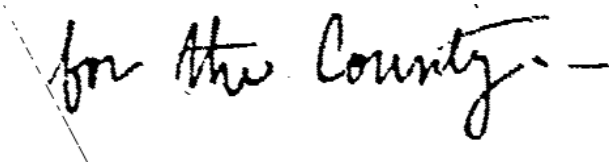


FIGURE 4.28: Exemple d'image du corpus de la compétition ICDAR 2015 après prétraitements

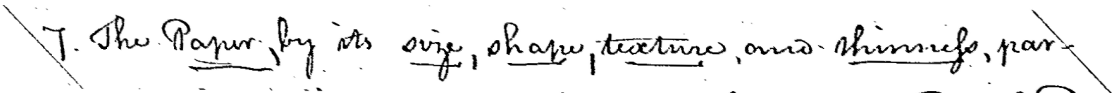


FIGURE 4.29: Exemple d'image du corpus de la compétition ICDAR 2015 après prétraitements

4.5.2 Caractéristiques utilisées

Caractéristiques des pixels (SPF) Les Caractéristiques Des Pixels (SPF) est le vecteur de caractéristiques proposé dans [47]. Il s'agit d'une compression de l'information contenue dans les colonnes de pixels, ce qui en fait donc une caractéristique de bas-niveau.

Une trame de longueur $P = 1$ pixel est décrite par un ensemble de caractéristiques extrait via des opérateurs mathématiques :

- la moyenne des niveaux de gris ;
- la position du centre de gravité ;
- le moment vertical du second ordre des pixels noirs ;
- la position du pixel noir le plus haut et celle du pixel noir le plus bas ;
- la variation de cette position par rapport aux trames adjacentes ;
- le nombre de transitions noir/blanc entre le pixel noir le plus haut et le pixel noir le plus bas ;
- la proportion de pixels noirs entre le pixel noir le plus haut et le pixel noir le plus bas.

Soit neuf valeurs décrivant une trame d'un pixel de long.

Histogramme de Gradients Orientés

Pour réaliser cette expérience, nous avons choisi d'utiliser les Histogrammes de Gradients Orientés [109]. Nous utilisons des trames de longueur $P = 8$ pixels, nous découpons l'image en 8 sous-fenêtres sans recouvrement de dimension 8×8, le gradient est quantifié en 8 directions. Ces valeurs sont concaténées pour obtenir un vecteur de caractéristiques de taille 64.

ORB

Les caractéristiques ORB (Oriented FAST and Rotated BRIEF) [151] sont dérivées des caractéristiques BRIEF (Binary Robust Independant Elementary Features) [152].

BRIEF est un vecteur de caractéristiques où chaque caractéristique représente un ensemble de comparaisons d'intensités lumineuses (des tests) entre des pairs de pixels provenant d'une image floutée. Un test τ est défini par :

$$\tau(p; a, b) = \begin{cases} 0 & \text{si } p(a) \geq p(b) \\ 1 & \text{si } p(a) < p(b) \end{cases} \tag{4.1}$$

avec p l'image floutée, $p(a)$ est l'intensité lumineuse d'un pixel aux coordonnées $a_1 = (a_x, a_y)$ et $p(b)$ est l'intensité lumineuse d'un pixel aux coordonnées $b = (b_x, b_y)$ Le choix des pixels est aléatoire et suit une loi normale. Le vecteur binaire, construit via ces comparaisons à différentes échelles, a l'avantage d'être très rapide à construire et à utiliser. En effet la distance de Hamming est utilisée pour obtenir la distance entre deux vecteurs BRIEF.

Une caractéristique ORB est définie comme une combinaison de caractéristiques BRIEF :

$$f_n(p) := \sum_{i \leq 1 \leq n}^{I} 2^{i-1} \tau(p; a_i, b_i) \tag{4.2}$$

On passe donc d'une représentation binaire avec les tests à une représentation réelle avec les caractéristiques. Comme pour les HOG cette extraction de caractéristiques s'effectue dans plusieurs sous-fenêtres.

Afin d'être invariant au redimensionnement, cette comparaison s'effectue à différentes échelles sur l'image. Dans notre application nous avons choisi de manière empirique 32 sous fenêtres de dimension 4×4 sans recouvrement, chaque sous fenêtre est tournée de 72 degrés (soit 5 nouvelles images par sous fenêtre) et chaque sous fenêtre est redimensionnée 2 fois de suite avec un ratio de 1.5, soit 3 échelles au total. Cela constitue donc un vecteur de caractéristique de dimension $32 \times 5 \times 3 = 480$.

4.5.3 Tâche 1 (query by string)

Le système proposé est le même que celui proposé dans la section 4.3 : l'architecture hybride BLSTM-CTC-MMC.

4.5.4 Tâche 2 Système 1 (query by example)

Ce système est l'adaptation de notre système 1 basé sur l'architecture BLSTM-CTC-MMC. Afin de remplacer la requête ASCII par une requête image, nous avons ajouté une étape de décodage BLSTM-CTC sans correction MMC à partir de l'image requête. On obtient ainsi une requête ASCII qui sera fournit au système 1 afin d'avoir l'emplacement du mot recherché. L'ensemble du système est résumé par la Figure 4.30.

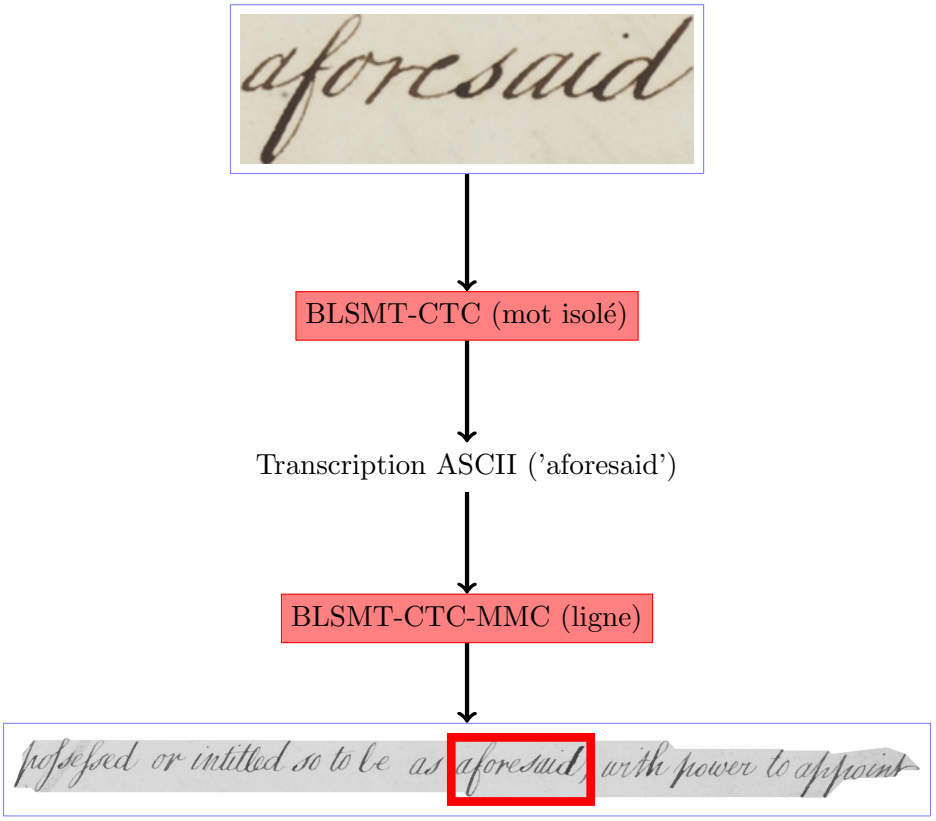


FIGURE 4.30: Résumé du système 2

4.5.5 Tâche 2 Système 2 (query by example)

Le problème du système 2 est le nombre d'erreurs locales dans la requête issu du décodage BLSTM-CTC sans étage correcteur MMC. Nous proposons donc de palier ce problème en ajoutant un étage correcteur contenant le lexique de tous les mots présents dans la base complète (9557 entrées). Cette étage est le même que présenté dans la section 3.2, un système hybride BLSTM-CTC-MMC pour la reconnaissance de mot isolés. L'ensemble du système est résumé par la Figure 4.31.

4.5.6 Résultats

A l'heure où nous écrivons ces lignes nous ne disposons pas encore des résultats des autres systèmes sur cette base, ni de la base de test. Dans cette partie nous présentons donc uniquement, les performances de nos systèmes sur une tâche de *query by string* pour le système 1 et query by example sur le système 2 et 3. Nous disposons pour cela de 2213 images exclues des 11144 images de la base d'apprentissage afin d'évaluer les performances de nos systèmes. L'évaluation se fera suivant les valeurs de Rappel (R) et

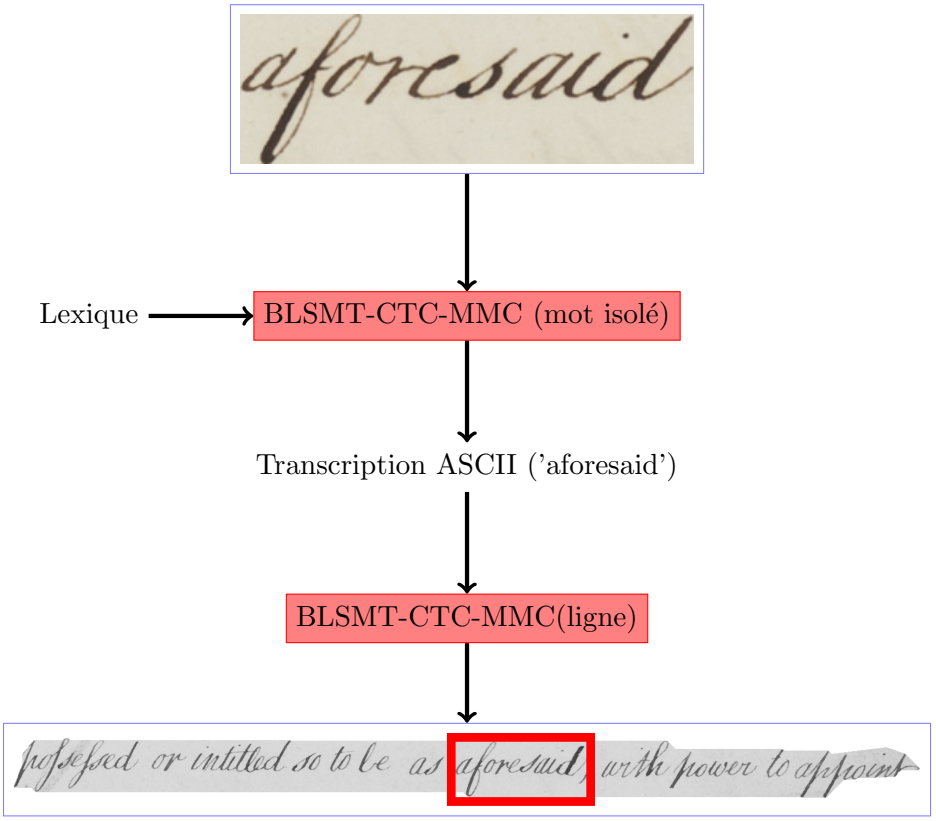


FIGURE 4.31: Résumé du système 3

de Précision (P) en comptant les vrais positifs (TP) les faux positifs (FP) et les faux négatifs (FN) sur l'ensemble des documents de base de test. En faisant varier le seuil de rejet, nous obtenons des courbes de rappel-précision à partir de l'accumulation de toutes ces valeurs :

$$R = \frac{TP}{TP + FN} \quad P = \frac{TP}{TP + FP}$$

Nous avons effectué deux expériences, une première comparant les performances des vecteurs de caractéristiques utilisées résumée sur la Figure 4.32 et les performances des systèmes 1,2 et 3 sur leurs tâches respectives résumée sur la Figure 4.33.

On peut voir que la combinaison des trois vecteurs HOG, ORB et SPF est la meilleure configuration possible avec un break even point à 84 %. On observe également que le vecteur de caractéristiques HOG semble être le vecteur de caractéristique unique le plus discriminant avec un break even point à 81 %, suivi par les caractéristique SPF (break even point à 76%) et finalement ORB (break even point à 76%). Les performances du système combinant les caractéristiques HOG et SPF montrent que parfois la combinaison

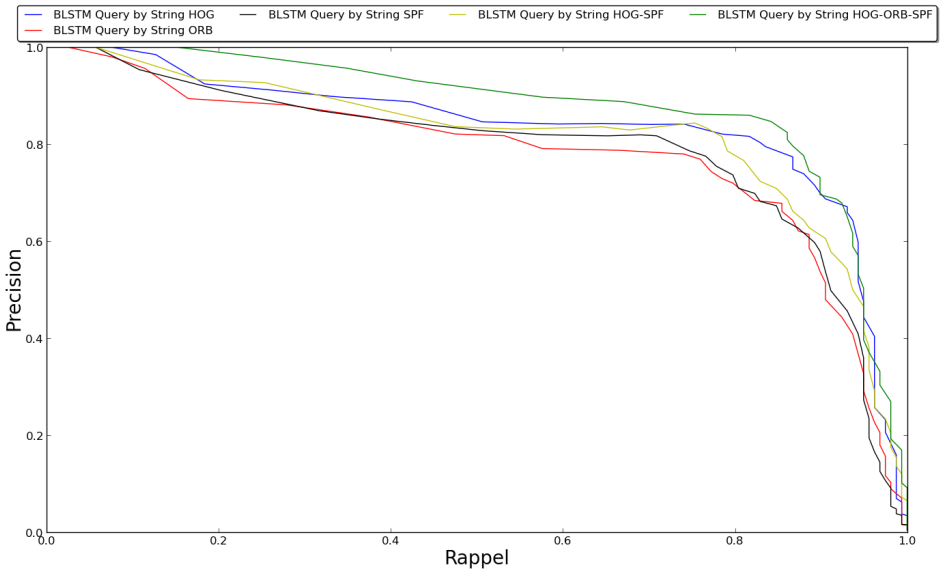


FIGURE 4.32: Comparaison des différentes caractéristiques sur une tâche de *query by string*

de caractéristiques peut amener à une diminution des performances globales du système (break even point à 79%).

Ces résultats prouvent que les caractéristiques doivent toujours être validées car la combinaison n'apporte pas forcément de meilleurs performances. De plus, nous n'avons aucune assurance que le classement de performances est le même si la base d'apprentissage avait été différente.

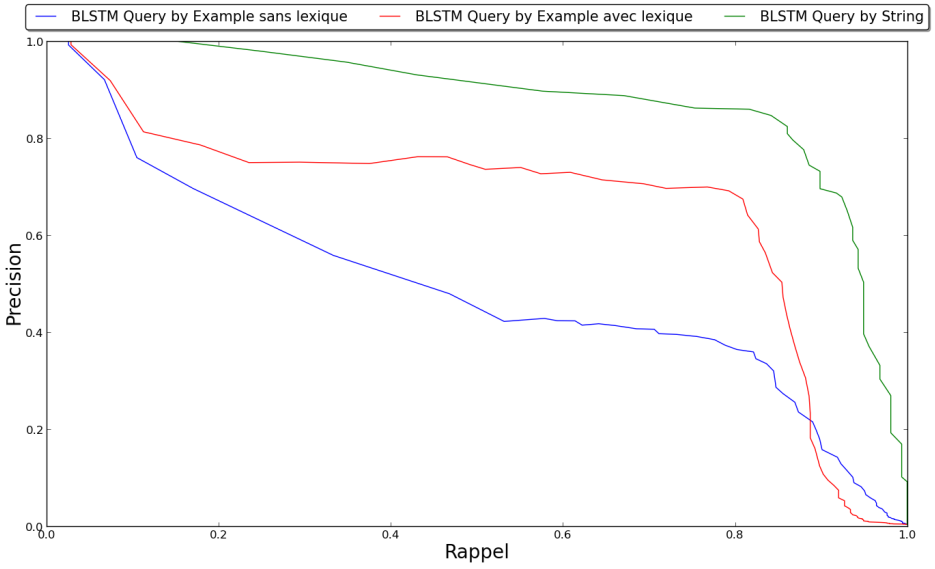


FIGURE 4.33: Performances des systèmes 1, 2 et 3

En ce qui concerne les performances des trois modes de traitement de la requête, on observe logiquement sur la Figure 4.33 que plus la requête initiale est bruitée moins les performances du système sont bonnes. On obtient respectivement des break even point de 46, 70 et 84 % pour les système 3, 2 et 1. La correction lexicale de la requête initiale augmente de 24% les performances même si le lexique est de taille importante (enviro 10000 entrées). Cependant, le lexique ne corrige pas toutes les confusions c'est pourquoi l'on observe un écart de 14 % entre le système 1 et 2.

On peut ajouter que même si toutes les erreurs des requêtes du système 2 et 3 sont corrigées, leurs performances ne pourraient pas dépasser celles du système 1. Dans le meilleur des cas elles ne pourront que les égaler.

On peut également soulever un problème récurrent en détection de mots clés par une méthode de *query by example*. La question est de savoir si l'on doit reconnaître la transcription ASCII sur l'image fournie ou réellement trouver une correspondance exacte de l'image fournie. Si l'on prend par exemple les images 4.34, 4.35 et 4.36, doit-on détecter tous les mots *Bentham* ou *"Bentham*, ? tous les mots *dJeremy* ou *Jeremy* ? tous les mots *parliament* ou uniquement les occurences du mot *parliament* soulignées ? Au sein de la

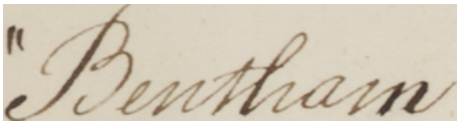


FIGURE 4.34: Exemple de requête ambigüe

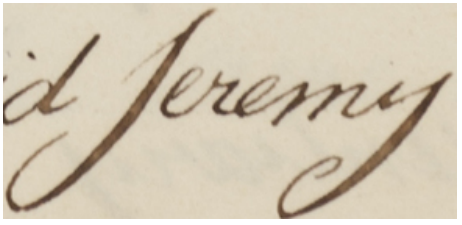


FIGURE 4.35: Exemple de requête ambigüe

FIGURE 4.36: Exemple de requête ambigüe

littérature la cible semble dans la plupart des cas être la transcription ASCII sauf applications particulières comme la détection d'écriture pour un scripteur particuliers. Ce problème est donc par définition mal posé et mal borné, pourquoi vouloir reconnaître

une image précise alors que l'objectif final est une transcription ASCII ? On ne fait qu'ajouter de la confusion et du bruit à la requête initiale.

4.6 Conclusion

Dans ce chapitre, nous avons proposé un système hybride générique permettant la détection de mots clés et d'expressions régulières dans des lignes de documents. Cela nous a permis dans un premier temps de proposer un benchmark de différentes méthodes discriminantes : MLP,RNN,CAC et BLSTM-CTC, ainsi qu'une approche purement MMC. Nous avons prouvé que les réseaux de neurones récurrents peuvent traiter avec plus de précision les tâches de reconnaissance d'écriture manuscrite même dans des conditions difficiles comme la détection d'expressions régulières. Ce benchmark nous a également permis de montrer la supériorité de l'architecture BLSTM-CTC-MMC (*break even point* à 80% sur le lexique à 100 requêtes). L'autre observation intéressante est que le système CAC-MMC montre des performances supérieures aux MMC standards et à la structure MLP-MMC mais ne peut rivaliser avec des réseaux de neurones récurrents.

Le BLSTM-CTC-MMC a montré des performances très intéressantes sur des requêtes moins contraintes comme les expressions régulières. On peut observer des écarts de plus de 40% entre le BLSTM-CTC-MMC et une approche purement MMC. De plus, ce système semble pouvoir gérer des requêtes très peu contraintes comme les séquences de majuscules et de chiffres avec des scores de rappel-précision intéressants. Ce système est également capable de donner de bonnes performances sans modèle de langage ni lexique pour corriger les erreurs de transcription, ce qui montre une capacité à détecter des informations sans lexique comme les EN par exemple.

Conclusion Générale

Ces travaux de thèse ont permis de mettre en place un système hybride de reconnaissance de mots isolés manuscrits s'appuyant sur les CAC, ainsi qu'un système générique de détection de mots clés et d'expressions régulières dans des documents manuscrits non contraints. Au cours de ces expériences différents systèmes ont été testés. Nous avons ainsi pu comparer l'apport local de différentes méthodes discriminantes (MLP, RNN, CAC, BLSTM-CTC). Tous ces systèmes ont été mis en place afin d'évaluer la capacité des systèmes actuels à effectuer la détection et la reconnaissance d'entités nommées dans des documents manuscrits non contraints.

Afin de donner des éléments de réponse à cette question, nous avons commencé par définir les entités nommées et étudier les systèmes déjà existants pour détecter ces entités dans différents domaines et différents type de documents. Nous avons également étudié les différents indices permettant de localiser les entités nommées dans les documents afin de faciliter la tâche de détection et reconnaissance.

Ainsi le deuxième chapitre a passé en revue l'ensemble des méthodes statistiques couramment utilisés en reconnaissance de l'écriture manuscrite notamment les modèles graphiques probabilistes comme les MMC, les CAC. Nous avons complèté cette bibliographie par une description des architectures hybrides notamment la méthode la plus performante actuellement en classification de séquences le BLSTM-CTC.

Notre première contribution est une architecture hybride CAC-MMC effectuant une reconnaissance mots isolés manuscrits, cette architecture a été comparée à 4 autres systèmes implémentant un MLP-MMC, un RNN-MMC, un MMC seul et un BLSTM-CTC-MMC. Afin d'utiliser les CAC pour traiter des valeurs continues, il nous a fallu travailler sur la représentation des données et discrétiser les données à l'aide d'une approche par classification non-supervisée. Nous avons également travaillé sur le contexte et les échelles de représentation afin de fournir un vecteur de caractéristiques le plus complet et le plus informatif possible. L'inférence entre les deux étages se fait en substituant des vraisemblances du MMC par les probabilités *a posteriori* issues de l'étage discriminant. Les performances du CAC-MMC se placent entre celles du MLP-MMC et

celles du MMC seul mais en-dessous de celles des deux approches incoporant des réseaux récurrents. D'après notre bibliographie, aucun système implémentant des CAC pour la reconnaissance d'écriture manuscrite n'a été proposé jusqu'à présent.

Enfin la principale contribution de cette thèse est un système générique de détection de mots clés et d'expressions régulières s'appuyant sur une architecture hybride. Ce système permet de traiter des lignes complètes à l'aide d'un modèle MMC modélisant les lignes de textes. De toutes les architectures testées, le BLSTM-CTC-MMC a montré les meilleurs performances avec et sans étage correctif MMC. Le niveau de performance atteint par ce système est également très intéressant même sur des requêtes très peu contraintes comme une séquence quelconque de majuscules ou de chiffres.

Au vu de ces résultats, il semble que cette architecture peut permettre la reconnaissance d'informations sans corrections lexicales comme c'est le cas pour les entités nommées.

Les perspectives pour faire évoluer ces travaux sont nombreuses et peuvent être déclinées selon plusieurs axes : espace de représentation des caractéristiques, apprentissage embarqué EM pour CAC, reconnaissance de mots isolés, détection et reconnaissance de mots clés et d'expressions régulières.

Premièrement, l'espace de représentation des caractéristiques. Nous avons vu la nécessité de discrétiser les caractéristiques afin de pouvoir obtenir une modélisation correcte avec un CAC. Il faudrait étudier d'autres combinaisons d'échelles, ajouter des niveaux d'abstraction afin d'enrichir le vecteur de caractéristiques proposé. Cependant, afin de palier l'augmentation importante de taille de ce dernier, il faudra surement mettre en place des méthodes de parallélisation afin que les temps de calculs ne soient pas trop importants.

Deuxièmement, l'apprentissage embarqué EM pour CAC. Nous avons posé la théorie de l'apprentissage embarqué, cependant, nous n'avons pas eu le temps de développer un apprentissage CAC-MMC joint comme c'est le cas dans la structure BLSTM-CTC. Cette amélioration augmentera surement les performances locales du CAC car elle permettra d'entrainer le CAC suivant un critère mot et non un critère trame comme c'est le cas actuellement.

Concernant la reconnaissance de mots isolés, nous n'avons pas testé l'approche de type CACC. L'ajout des états cachés au sein d'un modèle CAC amène la flexibilité dont manque les CAC. Même si les méthodes hybrides semblent proposer de meilleurs résultats, l'étude de ce système permettrait d'enrichir notre étude comparative.

Finalement, il serait intéressant d'intégrer notre système générique de détection de mots clés et d'expressions régulières dans un système de plus haut niveau permettant de détecter les entités nommées dans un document. Nous pourrions rajouter par exemple

un étage CAC permettant d'étiqueter les EN comme cela est fait en TAL [153, 154]. Cet étage prendrait comme caractéristiques les séquences reconnues par l'étage de reconnaissance après extraction. Nous aurions ainsi répondu complètement à la problématique proposée par ITESOFT et réduit un peu plus la frontière séparant le traitement des documents ASCII et des documents manuscrits. Quoi qu'il en soit ces travaux montrent que l'objectif est désormais atteignable avec les technologies de réseaux récurrents.

Publications liées

Le travail présenté dans cette thèse a fait l'objet des publications suivantes.

Gautier Bideault, Luc Mioulet, Clément Chatelainy and Thierry Paquet, **Benchmarking discriminative approaches for word spotting in handwritten documents**, *ICDAR 2015*, pp. 201-205, Nancy, 2015.

G. Bideault, L. Mioulet, C. Chatelain, T. Paquet, **A Hybrid BLSTM-HMM for spotting Regular Expressions**, *ICPRAM*, pp. 5-12, 2015.

Mioulet L., Bideault G., Chatelain C., Paquet T. and Brunessaux S., **Exploring multiple feature combination strategies with a recurrent neural network architecture for off-line handwriting recognition**, *SPIE - Document Recognition and Retrieval*, 94020F, 2015.

G. Bideault, L. Mioulet, C. Chatelain, T. Paquet, **Spotting handwritten words and REGEX using a two stage BLSTM-HMM architecture**, *SPIE - Document Recognition and Retrieval*, 94020G, 2015.

Mioulet, L., Bideault G., Chatelain C., Paquet T., Brunessaux S., **BLSTM-CTC Combination Strategies for Off-line Handwriting Recognition**, *ICPRAM*, pp. 173-180, 2015.

Bideault, L. Mioulet C. Chatelain and T. Paquet, **A hybrid CRF/HMM approach for handwriting recognition**, in *ICIAR*, pp. 403-410, Portugal, 2014.

Bibliographie

[1] E. Grosicki and H. El Abed, "Icdar 2009 handwriting recognition competition," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pp. 1398–1402, IEEE, 2009.

[2] E. Grosicki and H. El-Abed, "Icdar 2011-french handwriting recognition competition," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 1459–1463, IEEE, 2011.

[3] M. Ehrmann, *Les entités nommées, de la linguistique au TAL : statut théorique et méthodes de désambiguïsation.* PhD thesis, Paris 7, 2008.

[4] G. Koch, *Catégorisation automatique de documents manuscrits : application aux courriers entrants.* PhD thesis, Rouen, 2006.

[5] D. Miller, S. Boisen, R. Schwartz, R. Stone, and R. Weischedel, "Named entity extraction from noisy input : speech and ocr," in *Proceedings of the sixth conference on Applied natural language processing*, pp. 316–324, Association for Computational Linguistics, 2000.

[6] S. Rosset, C. Grouin, and P. Zweigenbaum, *Entités nommées structurées : guide d'annotation Quaero.* LIMSI-Centre national de la recherche scientifique, 2011.

[7] N. Chinchor, D. D. Lewis, and L. Hirschman, "Evaluating message understanding systems : an analysis of the third message understanding conference (muc-3)," *Computational linguistics*, vol. 19, no. 3, pp. 409–449, 1993.

[8] D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, and M. Tyson, "Fastus : A finite-state processor for information extraction from real-world text," in *IJCAI*, vol. 93, pp. 1172–1178, 1993.

[9] R. Grishman and B. Sundheim, "Message understanding conference-6 : A brief history.," in *COLING*, vol. 96, pp. 466–471, 1996.

[10] R. Grishman and B. Sundheim, "Design of the muc-6 evaluation," in *Proceedings of a workshop on held at Vienna, Virginia : May 6-8, 1996*, pp. 413–422, Association for Computational Linguistics, 1996.

[11] B. M. Sundheim, "Overview of results of the muc-6 evaluation," in *Proceedings of a workshop on held at Vienna, Virginia : May 6-8, 1996*, pp. 423–442, Association for Computational Linguistics, 1996.

[12] R. Merchant, M. E. Okurowski, and N. Chinchor, "The multilingual entity task (met) overview," in *Proceedings of a workshop on held at Vienna, Virginia : May 6-8, 1996*, pp. 445–447, Association for Computational Linguistics, 1996.

[13] S. Sekine and H. Isahara, "Irex project overview," in *Proceedings of the IREX Workshop*, pp. 7–12, Citeseer, 1999.

[14] S. Sekine and Y. Eriguchi, "Japanese named entity extraction evaluation : analysis of results," in *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pp. 1106–1110, Association for Computational Linguistics, 2000.

[15] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task : Language-independent named entity recognition," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 142–147, Association for Computational Linguistics, 2003.

[16] D. Santos, N. Seco, N. Cardoso, and R. Vilela, "Harem : An advanced ner evaluation contest for portuguese," in *Proceedings of LREC*, pp. 1986–1991, 2006.

[17] B. Daille and E. Morin, "Reconnaissance automatique des noms propres de la langue écrite : les récentes réalisations," *TAL. Traitement automatique des langues*, vol. 41, no. 3, pp. 601–621, 2000.

[18] T. Poibeau, "Deconstructing harry" : une évaluation des systemes de repérage d'entités nommées," *Revue de la société d'électronique, d'électricité et de traitement de l'information*, vol. 5, pp. 25–33, 2001.

[19] T. Poibeau, "Extraction automatique d'information(du texte brut au web sémantique)," 2003.

[20] N. Friburger, *Reconnaissance automatique des noms propres : application à la classification automatique de textes journalistiques*. PhD thesis, Tours, 2002.

[21] D. McDonald, "Internal and external evidence in the identification and semantic categorization of proper names," *Corpus processing for lexical acquisition*, pp. 21–39, 1996.

[22] A. Nazarenko, B. Habert, A. Salem, *et al.*, "Les linguistiques de corpus," 1997.

[23] Y. Shinyama and S. Sekine, "Named entity discovery using comparable news articles," in *Proceedings of the 20th international conference on Computational Linguistics*, p. 848, Association for Computational Linguistics, 2004.

[24] R. Gaizauskas, K. Humphreys, H. Cunningham, and Y. Wilks, "University of sheffield : description of the lasie system as used for muc-6," in *Proceedings of the 6th conference on Message understanding*, pp. 207–220, Association for Computational Linguistics, 1995.

[25] E. Brill, "Transformation-based error-driven learning and natural language processing : A case study in part-of-speech tagging," *Computational linguistics*, vol. 21, no. 4, pp. 543–565, 1995.

[26] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks, "University of sheffield : Description of the lasie-ii system as used for muc-7," in *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*, Citeseer, 1998.

[27] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman, "Nyu : Description of the mene named entity system as used in muc-7," in *In Proceedings of the Seventh Message Understanding Conference (MUC-7*, Citeseer, 1998.

[28] G. R. Krupka and K. Hausman, "Isoquest inc. : Description of the netowl (tm) extractor system as used for muc-7," in *Proceedings of MUC*, vol. 7, 1998.

[29] D. Lin, "Using collocation statistics in information extraction," in *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, vol. 66, Citeseer, 1998.

[30] R. Grishman, "The nyu system for muc-6 or where's the syntax ?," in *Proceedings of the 6th conference on Message understanding*, pp. 167–175, Association for Computational Linguistics, 1995.

[31] M. A. Przybocki, J. G. Fiscus, J. S. Garofolo, and D. S. Pallett, "1998 hub-4 information extraction evaluation," in *Proc. DARPA Broadcast News Workshop,(Herndon, Va, USA)*, pp. 13–18, 1999.

[32] J. Makhoul, R. Schwartz, C. Lapre, and I. Bazzi, "A script-independent methodology for optical character recognition," *Pattern Recognition*, vol. 31, no. 9, pp. 1285–1294, 1998.

[33] L. M. Lorigo and V. Govindaraju, "Offline arabic handwriting recognition : a survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 712–724, 2006.

[34] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition : a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 63–84, 2000.

[35] U.-V. Marti and H. Bunke, "Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system," *International journal of Pattern Recognition and Artificial intelligence*, vol. 15, no. 01, pp. 65–90, 2001.

[36] T. Artières and P. Gallinari, "Stroke level hmms for on-line handwriting recognition," in *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pp. 227–232, IEEE, 2002.

[37] P. Morasso, L. Barberis, S. Pagliano, and D. Vergano, "Recognition experiments of cursive dynamic handwriting with self-organizing networks," *Pattern Recognition*, vol. 26, no. 3, pp. 451–460, 1993.

[38] V. Margner and H. El Abed, "Arabic handwriting recognition competition," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, pp. 1274–1278, IEEE, 2007.

[39] C. Chatelain, L. Heutte, T. Paquet, *et al.*, "A two-stage outlier rejection strategy for numerical field extraction in handwritten documents," in *ICPR*, vol. 3, pp. 224–227, 2006.

[40] C. Chatelain, L. Heutte, and T. Paquet, "Segmentation-driven recognition applied to numerical field extraction from handwritten incoming mail documents," in *Document Analysis Systems VII*, pp. 564–575, Springer, 2006.

[41] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 3, pp. 418–435, 1992.

[42] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[43] A. L. Koerich, Y. Leydier, R. Sabourin, and C. Y. Suen, "A hybrid large vocabulary handwritten word recognition system using neural networks with hidden markov models," in *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pp. 99–104, IEEE, 2002.

[44] M. Gilloux, B. Lemarié, and M. Leroux, "A hybrid radial basis function network/-hidden markov model handwritten word recognition system," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 394–397, IEEE, 1995.

[45] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 4, pp. 767–779, 2011.

[46] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification : labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, ACM, 2006.

[47] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 855–868, 2009.

[48] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[49] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields : Probabilistic models for segmenting and labeling sequence data," 2001.

[50] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," *Introduction to statistical relational learning*, pp. 93–128, 2006.

[51] D. Hébert, *Champs aléatoires conditionnels pour l'extraction de structures dans les images de documents*. PhD thesis, Rouen, 2013.

[52] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 10, p. 1848, 2007.

[53] K. M. Sayre, "Machine recognition of handwritten words : A project report," *Pattern recognition*, vol. 5, no. 3, pp. 213–228, 1973.

[54] D. Koller and N. Friedman, *Probabilistic graphical models : principles and techniques*. MIT press, 2009.

[55] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 1. springer New York, 2006.

[56] K. P. Murphy, *Machine learning : a probabilistic perspective*. MIT press, 2012.

[57] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, pp. 1554–1563, 1966.

[58] L. E. Baum, "An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[59] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[60] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," *Bell System Technical Journal, The*, vol. 62, no. 4, pp. 1035–1074, 1983.

[61] A. V. Lukashin and M. Borodovsky, "Genemark. hmm : new solutions for gene finding," *Nucleic acids research*, vol. 26, no. 4, pp. 1107–1115, 1998.

[62] K. Tokuda, H. Zen, and A. W. Black, "An hmm-based speech synthesis system applied to english," in *Speech Synthesis, 2002. Proceedings of 2002 IEEE Workshop on*, pp. 227–230, IEEE, 2002.

[63] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260–269, 1967.

[64] G. Antoniol, F. Brugnara, M. Cettolo, and M. Federico, "Language model representations for beam-search decoding," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, pp. 588–591, IEEE, 1995.

[65] S. Ortmanns, A. Eiden, H. Ney, and N. Coenen, "Look-ahead techniques for fast beam search," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 3, pp. 1783–1786, IEEE, 1997.

[66] L. R. Welch, "Hidden markov models and the baum-welch algorithm," *IEEE Information Theory Society Newsletter*, vol. 53, no. 4, pp. 10–13, 2003.

[67] H. Wallach, *Efficient training of conditional random fields*. PhD thesis, Citeseer, 2002.

[68] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 134–141, Association for Computational Linguistics, 2003.

[69] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy, "Accelerated training of conditional random fields with stochastic gradient methods," in *Proceedings of the 23rd international conference on Machine learning*, pp. 969–976, ACM, 2006.

[70] S. F. Chen and R. Rosenfeld, "A gaussian prior for smoothing maximum entropy models," tech. rep., DTIC Document, 1999.

[71] A. McCallum and B. Wellner, "Toward conditional models of identity uncertainty with application to proper noun coreference," 2003.

[72] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 188–191, Association for Computational Linguistics, 2003.

[73] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum, "Interactive information extraction with constrained conditional random fields," in *AAAI*, vol. 4, pp. 412–418, 2004.

[74] A.-R. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," *InterSpeech*, 2010.

[75] G. Zweig and P. Nguyen, "A segmental crf approach to large vocabulary continuous speech recognition," *Automatic Speech Recognition & Understanding*, pp. 152–157, December 2009.

[76] Y.-H. Sung and D. Jurafsky, "Hidden conditional random fields for phone recognition," in *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pp. 107–112, IEEE, 2009.

[77] T. A. Stephenson, H. Bourlard, S. Bengio, and A. C. Morris, "Automatic speech recognition using dynamic bayesian networks with both acoustic and articulatory variables," *ICSLP*, vol. 2, pp. 951–954, October 2000.

[78] T. A. Stephenson, H. Bourlard, S. Bengio, and A. C. Morris, "Automatic speech recognition using dynamic bayesian networks with both acoustic and articulatory variables," in *IN 6TH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING : ICSLP 2000 (INTERSPEECH 2000*, Citeseer, 2000.

[79] E. Trentin and M. Gori, "A survey of hybrid ann/hmm models for automatic speech recognition," *Neurocomputing*, vol. 37, no. 1, pp. 91–126, 2001.

[80] H. A. Bourlard and N. Morgan, *Connectionist speech recognition : a hybrid approach*, vol. 247. Springer, 1994.

[81] H. Bourlard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *Neural Networks, IEEE Transactions on*, vol. 4, no. 6, pp. 893–909, 1993.

[82] Y. Bengio, Y. LeCun, and Y. LeRec, "Ann/hmm hybrid for on-line handwriting recognition," *Neural Computation*, vol. 7, pp. 1289–1303, November 1995.

[83] A. W. Senior and A. J. Robinson, "An off-line cursive handwriting recognition system," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 3, pp. 309–321, 1998.

[84] A. W. Senior, "Off-line cursive handwriting recognition using recurrent neural networks," 1994.

[85] A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid svm/hmm architectures for speech recognition.," in *INTERSPEECH*, pp. 504–507, Citeseer, 2000.

[86] M. Liu, Y. Xie, Z. Yao, and B. Dai, "A new hybrid gmm/svm for speaker verification," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 4, pp. 314–317, IEEE, 2006.

[87] C. M. Bishop *et al.*, "Neural networks for pattern recognition," 1995.

[88] F. Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[89] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[90] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "Lerec : A nn/hmm hybrid for on-line handwriting recognition," *Neural Computation*, vol. 7, no. 6, pp. 1289–1303, 1995.

[91] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," *Back-propagation : Theory, architectures and applications*, pp. 433–486, 1995.

[92] P. J. Werbos, "Backpropagation through time : what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[93] R. J. Williams and D. Zipser, "Experimental analysis of the real-time recurrent learning algorithm," *Connection Science*, vol. 1, no. 1, pp. 87–111, 1989.

[94] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[95] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 2, pp. 211–224, 2012.

[96] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[97] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *Artificial Neural Networks : Formal Models and Their Applications–ICANN 2005*, pp. 799–804, Springer, 2005.

[98] A. V. Nefian and M. H. Hayes III, "Maximum likelihood training of the embedded hmm for face detection and recognition," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1, pp. 33–36, IEEE, 2000.

[99] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *Speech and audio processing, ieee transactions on*, vol. 2, no. 2, pp. 291–298, 1994.

[100] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification.," in *INTERSPEECH*, pp. 1117–1120, 2005.

[101] H. Chung and H.-D. Yang, "Conditional random field-based gesture recognition with depth information," *Optical Engineering*, vol. 52, no. 1, pp. 017201–017201, 2013.

[102] H.-D. Yang, S. Sclaroff, and S.-W. Lee, "Sign language spotting with a threshold model based on conditional random fields," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 7, pp. 1264–1277, 2009.

[103] Y.-F. Pan, X. Hou, and C.-L. Liu, "Text localization in natural scene images based on conditional random field," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pp. 6–10, IEEE, 2009.

[104] Y. Wang and S. Gong, "Conditional random field for natural scene categorization.," in *BMVC*, pp. 1–10, Citeseer, 2007.

[105] A.-r. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition.," in *INTERSPEECH*, pp. 2846–2849, 2010.

[106] G. Zweig and P. Nguyen, "A segmental crf approach to large vocabulary continuous speech recognition," in *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pp. 152–157, IEEE, 2009.

[107] S. Kumar and M. Hebert, "Discriminative random fields : A discriminative framework for contextual interaction in classification," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1150–1157, IEEE, 2003.

[108] A. Senior and T. Robinson, "Forward-backward retraining of recurrent neural networks.," *Advances in Neural Information Processing Systems*, pp. 743–749, 1996.

[109] J. A. Rodrıguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *Int. Conf. on Frontiers in Handwriting Recognition*, 2008.

[110] Y. Kessentini, C. Chatelain, and T. Paquet, "Word spotting and regular expression detection in handwritten documents," in *ICDAR*, 2013.

[111] H. Bourlard, S. Bengio, Q. Zhu, B. Mesot, N. Morgan, *et al.*, "Towards using hierarchical posteriors for flexible automatic speech recognition systems," tech. rep., IDIAP, 2004.

[112] H. Ketabdar, J. Vepa, S. Bengio, and H. Bourlard, "Posterior based keyword spotting with a priori thresholds," in *International Conference on Spoken Language Processing (ICSLP)*, no. LIDIAP-CONF-2006-017, 2006.

[113] N. Okazaki, "Crfsuite : a fast implementation of conditional random fields (crfs)," *URL http ://www. chokkan. org/software/crfsuite*, 2007.

[114] T. Paquet, L. Heutte, G. Koch, and C. Chatelain, "A categorization system for handwritten documents," *International Journal on Document Analysis and Recognition*, vol. 15, no. 4, pp. 315–330, 2012.

[115] B. Gatos and I. Pratikakis, "Segmentation-free word spotting in historical printed documents," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pp. 271–275, IEEE, 2009.

[116] A. Kumar, C. Jawahar, and R. Manmatha, "Efficient search in document image collections," in *Computer Vision–ACCV 2007*, pp. 586–595, Springer, 2007.

[117] T. M. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pp. 218–222, IEEE, 2003.

[118] H. Cao and V. Govindaraju, "Template-free word spotting in low-quality manuscripts," in *Proceedings of the 6th International Conference on Advances in Pattern Recognition*, pp. 135–139, 2007.

[119] T. Adamek, N. E. O'Connor, and A. F. Smeaton, "Word matching using single closed contours for indexing handwritten historical documents," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 153–165, 2007.

[120] M. Rusinol, D. Aldavert, R. Toledo, and J. Lladós, "Browsing heterogeneous document collections by a segmentation-free word spotting method," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 63–67, IEEE, 2011.

[121] J. A. Rodríguez-Serrano, F. Perronnin, J. Lladós, and G. Sánchez, "A similarity measure between vector sequences with application to handwritten word image retrieval," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1722–1729, IEEE, 2009.

[122] Y. Liang, M. C. Fairhurst, and R. M. Guest, "A synthesised word approach to word retrieval in handwritten documents," *Pattern Recognition*, vol. 45, no. 12, pp. 4225–4236, 2012.

[123] T. M. Rath, V. Lavrenko, and R. Manmatha, "A statistical approach to retrieving historical manuscript images without recognition," tech. rep., DTIC Document, 2003.

[124] T. M. Rath, R. Manmatha, and V. Lavrenko, "A search engine for historical manuscript images," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 369–376, ACM, 2004.

[125] T. Van der Zant, L. Schomaker, and K. Haak, "Handwritten-word spotting using biologically inspired features," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 1945–1957, 2008.

[126] M. Rusinol, D. Aldavert, R. Toledo, and J. Lladós, "Efficient segmentation-free keyword spotting in historical document collections," *Pattern Recognition*, vol. 48, no. 2, pp. 545–555, 2015.

[127] P. P. Roy, J.-Y. Ramel, and N. Ragot, "Word retrieval in historical document using character-primitives," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pp. 678–682, IEEE, 2011.

[128] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer graphics and image processing*, vol. 20, no. 4, pp. 375–390, 1982.

[129] C. Choisy, "Dynamic handwritten keyword spotting based on the nshp-hmm," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 1, pp. 242–246, IEEE, 2007.

[130] J. A. Rodríguez-Serrano and F. Perronnin, "Handwritten word-spotting using hidden markov models and universal vocabularies," *Pattern Recognition*, vol. 42, no. 9, pp. 2106–2116, 2009.

[131] S. Thomas, C. Chatelain, L. Heutte, and T. Paquet, "An information extraction model for unconstrained handwritten documents," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 3412–3415, IEEE, 2010.

[132] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934–942, 2012.

[133] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, and Y. Kessentini, "A deep hmm model for multiple keywords spotting in handwritten documents," *To appear in Pattern Analysis and Applications*, 2015.

[134] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Hmm-based word spotting in handwritten documents using subword models," in *Pattern recognition (icpr), 2010 20th international conference on*, pp. 3416–3419, IEEE, 2010.

[135] S. Wshah, G. Kumar, and V. Govindaraju, "Script independent word spotting in offline handwritten documents based on hidden markov models," in *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pp. 14–19, IEEE, 2012.

[136] M. Wöllmer, F. Eyben, A. Graves, B. Schuller, and G. Rigoll, "A tandem blstm-dbn architecture for keyword spotting with enhanced context modeling," in *Proc. of NOLISP*, 2009.

[137] A. H. Toselli and E. Vidal, "Fast hmm-filler approach for key word spotting in handwritten documents," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pp. 501–505, IEEE, 2013.

[138] W. Holzinger, B. Krüpl, and M. Herzog, *Using ontologies for extracting product features from web pages*. Springer, 2006.

[139] R. Sidhu and V. K. Prasanna, "Fast regular expression matching using fpgas," in *Field-Programmable Custom Computing Machines, 2001. FCCM'01. The 9th Annual IEEE Symposium on*, pp. 227–238, IEEE, 2001.

[140] M. N. Garofalakis, R. Rastogi, and K. Shim, "Spirit : Sequential pattern mining with regular expression constraints," in *VLDB*, vol. 99, pp. 7–10, 1999.

[141] T. L. Booth, *Sequential machines and automata theory*, vol. 3. Wiley New York, 1967.

[142] J. E. Hopcroft, *Introduction to automata theory, languages, and computation*. Pearson Education India, 1979.

[143] A. L. Spitz, "Using character shape codes for word spotting in document images," in *Shape, Structure and Pattern Recognition*, pp. 382–389, 1995.

[144] A. L. Spitz, "Determination of the script and language content of document images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 3, pp. 235–245, 1997.

[145] A. R. Dengel and B. Klein, "smartfix : A requirements-driven system for document analysis and understanding," in *Document Analysis Systems V*, pp. 433–444, Springer, 2002.

[146] M. E. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Segmentation and recognition of handwritten dates : an hmm-mlp hybrid approach.," pp. 248–262, 2003.

[147] C. Chatelain, L. Heutte, and T. Paquet, "Recognition-based vs syntax-directed models for numerical field extraction in handwritten documents," in *ICFHR, Montreal, Canada*, p. 6p, 2008.

[148] C. Chatelain, L. Heutte, and T. Paquet, "A two-stage outlier rejection strategy for numerical field extraction in handwritten documents," in *ICPR, Hong Kong, China*, vol. 3, pp. 224–227, 2006.

[149] J. Sauvola, T. Seppanen, S. Haapakoski, and M. Pietikainen, "Adaptive document binarization," in *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, vol. 1, pp. 147–152, IEEE, 1997.

[150] A. Vinciarelli and J. Luettin, "A new normalization technique for cursive handwritten words," *Pattern recognition letters*, vol. 22, no. 9, pp. 1043–1050, 2001.

[151] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb : an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.

[152] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief : Binary robust independent elementary features," in *Computer Vision–ECCV 2010*, pp. 778–792, Springer, 2010.

[153] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies-Volume 1*, pp. 359–367, Association for Computational Linguistics, 2011.

[154] D. Downey, M. Broadhead, and O. Etzioni, "Locating complex named entities in web text.," in *IJCAI*, vol. 7, pp. 2733–2739, 2007.