Intégration de connaissances linguistiques pour la reconnaissance de textes manuscrits en-ligne

Solen Quiniou

N° d'ordre : D 07 – 29

THÈSE

présentée devant

l'Institut National des Sciences Appliquées de Rennes

pour obtenir le grade de

DOCTEUR DE L'INSA DE RENNES
mention Informatique

par

Solen QUINIOU

Équipe d'accueil : IMADOC - IRISA

École Doctorale : MATISSE

Intégration de connaissances linguistiques pour la reconnaissance de textes manuscrits en-ligne

Soutenue le 17 décembre 2007

Composition du jury

M.	Thierry PAQUET	Professeur, Université de Rouen, PSI	Président
M.	Mohamed CHERIET	Professeur, ÉTS de Montréal, LIVIA	Rapporteurs
M.	Christian VIARD-GAUDIN	Professeur, Université de Nantes, IRCCyN	
M.	Guillaume GRAVIER	Chargé de recherche, CNRS, IRISA	Examinateurs
M.	Eric ANQUETIL	Maître de conférences, INSA de Rennes, IRISA	
M.	Guy LORETTE	Professeur, Université de Rennes 1, IRISA	

« Those leaving words hang strong from an emptiness. »
Those thieving birds, Silverchair.

« The language confuses like computers refuse
to understand how I'm feeling today. »
Map of your head, Muse.

Remerciements

Je remercie tout d'abord tous les membres de mon jury de thèse pour l'intérêt qu'ils ont porté à mon travail. Merci à Thierry Paquet d'avoir accepté de présider mon jury de thèse ainsi qu'à Mohamed Cheriet et Christian Viard-Gaudin, mes rapporteurs, pour leur relecture attentive et leurs remarques constructives qui ont permis d'améliorer ce manuscrit.

Je souhaiterais également remercier Guillaume Gravier, Pascale Sébillot et Stéphane Huet pour la collaboration scientifique menée durant ces trois années de thèse et pour toutes les idées qui en sont sorties ainsi que toutes les réponses apportées à mes questions.

Je ne pourrais jamais assez remercier Guy Lorette et Eric Anquetil, mes directeurs de thèse, pour le temps qu'ils m'ont consacré, leurs idées, leurs encouragements... Merci à Guy pour le partage de ses connaissances sur le monde de la recherche et son rôle de « grand-père scientifique ». Un grand merci à Eric pour son enthousiasme, son écoute et les idées engendrées lors de nos discussions ainsi que sa formation au monde de la recherche. Merci également à tous les deux pour vos relectures et remarques éclairées concernant mes articles et présentations.

Un grand merci à tous les membres de l'équipe IMADOC, les nouveaux membres comme les anciens, pour les bons moments passés ensemble, notamment lors des pauses café et pendant les séminaires au vert. Merci également à mes différents colocataires de bureau, Sabine, Laëtitia, Isaac et Christophe pour les agréables mois passés en leur compagnie et plus particulièrement à Harold et François, qui m'ont accompagnée dans cette aventure depuis le stage de quatrième année, ainsi qu'à Sébastien pour leur soutien dans les moments de doute mais aussi pour tous les très bons moments passés ensemble dans nos différents bureaux et en-dehors.

J'aimerais aussi remercier toutes les personnes de l'IRISA mais aussi mes proches, amis comme famille, qui ont participé aux saisies de phrases manuscrites sans lesquelles cette thèse n'aurait finalement pas pu être possible. Merci également aux doctorants de l'IRISA, et notamment à ceux avec lesquels j'ai participé à l'organisation de MajecSTIC en 2005, pour tous les bons moments passés ensemble.

Je remercie enfin ma famille et tous mes amis pour leur soutien et leurs encouragements durant ces trois années et les années précédentes et pour m'avoir permis de me changer les idées, surtout lors de la période de rédaction.

Table des matières

Table des figures

Liste des Algorithmes

Introduction générale

Avec l'apparition de périphériques mobiles tels que les assistants personnels ou encore les *Tablet PC* et les cahiers électroniques, il est nécessaire de prendre en compte les spécificités de ces terminaux afin de proposer des applications appropriées. Dans ce type de périphérique, l'interface est orientée stylo, l'interface classique clavier/souris étant à la fois plus encombrante et moins adaptée à l'utilisation de ces périphériques. En effet, ces terminaux peuvent être utilisés pour la prise de notes ou le dessin de schémas, par exemple : l'utilisation d'interfaces orientées stylo est alors plus naturelle que celle d'un clavier et d'une souris, puisqu'elle se rapproche du dessin sur une feuille de papier. De plus, ce type de périphérique permet également la prise de notes en environnement contraint, en station debout, par exemple.

Afin de pouvoir tirer partie des documents ainsi produits, il est nécessaire de proposer des méthodes de traitement adaptées, parmi lesquelles se trouve la *reconnaissance d'écriture manuscrite*. La reconnaissance d'écriture manuscrite consiste à transformer un signal numérique en sa représentation symbolique (voir figure 1).

FIG. 1 – Principe de la reconnaissance d'écriture manuscrite.

Cette thèse se place dans le cadre de la reconnaissance d'écriture manuscrite et plus particulièrement dans celui de la reconnaissance de textes manuscrits, saisis en-ligne sur des périphériques tels que des *Tablet PC*.

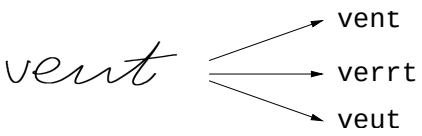
La reconnaissance d'écriture manuscrite est une tâche complexe, à-cause de la variabilité des styles d'écriture. La difficulté de cette tâche est d'autant plus importante au niveau de la reconnaissance de textes puisqu'il s'agit d'une des étapes ultimes de la reconnaissance. En effet, la reconnaissance de textes s'appuie sur de nombreux processus pour faire face à un enchaînement de problématiques allant de la segmentation de la phrase en mots à la reconnaissance de mots, en passant par la reconnaissance de lettres... Dans tous ces processus, le contexte joue un rôle fondamental pour réussir à interpréter correctement les différentes hypothèses de reconnaissance de formes.

Nous pouvons distinguer la *reconnaissance* de l'écriture manuscrite, qui s'appuie uniquement sur la reconnaissance de formes, de son *interprétation*, qui utilise également des informations sur le contexte de l'écriture. En ce qui concerne le contexte linguistique, il est utilisé aux deux derniers niveaux de la reconnaissance, à savoir celui des mots et enfin celui des phrases.

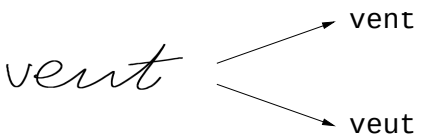
La reconnaissance de mots isolés utilise généralement des *dictionnaires* pour pouvoir lever des ambiguïtés sur la reconnaissance des caractères, en ne considérant comme réponse que les mots appartenant au dictionnaire. L'utilisation de ces informations linguistiques se situe donc au niveau lexical.

Il est aussi intéressant de tirer partie des connaissances linguistiques au niveau syntaxique. Dans ce cas, les informations concernent le contexte du mot écrit et notamment les mots qui l'entourent. Ces connaissances linguistiques sont généralement représentées grâce à des *modèles de langage* : ils permettent de définir les suites de mots possibles d'une langue donnée.

La figure 2 donne un exemple de l'apport des contextes linguistiques.



(a) Reconnaissance de formes



(b) Ajout de connaissances lexicales

(c) Ajout de connaissances syntaxiques

FIG. 2 – Importance de la prise en compte du contexte des mots.

Dans la figure 2(a), seule la forme du mot est prise en compte. Trois mots peuvent alors être proposés, comme résultat : *vent*, *verrt* et *veut*. En prenant en compte le contexte lexical (voir figure 2(b)), grâce à un dictionnaire de la langue française, le « mot » *verrt* peut être écarté puisqu'il ne s'agit pas d'un mot correct, dans la langue française. Enfin, lorsque le contexte dans lequel a été écrit le mot est pris en compte (voir figure 2(c)), comme ici la phrase entière, il est possible de désambiguïser le résultat de reconnaissance, en ne conservant que le mot *vent*.

Cadre et objectifs de la thèse

L'objectif de nos travaux est de construire un système complet de reconnaissance de phrases, en se basant sur le système de reconnaissance de mots RESIFMot, existant dans l'équipe Imadoc [Anq97, Car05]. Durant nos travaux, le système de reconnaissance de mots RESIFMot n'a pas été remis en cause mais les méthodes proposées pour la reconnaissance de phrases devront être facilement adaptables à une évolution de ce système.

Ce travail de thèse s'est focalisé sur le processus de reconnaissance de phrases, en étudiant les problématiques suivantes :

- la segmentation de la phrase en mots ;
- l'interprétation des mots en contexte de phrase, en intégrant différents types de modèles de langage.

Nous nous sommes donc intéressées à la recherche de méthodes permettant d'intégrer des connaissances linguistiques de manière efficace, dans le système de reconnaissance de phrases.

La figure 3 illustre les différentes étapes du système de reconnaissance de phrases proposé, en indiquant également le numéro du chapitre dans lequel chacune des étapes est décrite.

Nous proposons, tout d'abord, une méthode d'*extraction explicite des mots* des phrases, basée sur la caractérisation des espaces entre les lettres et les mots. À l'issue de cette première étape, un graphe de mots est construit et permet de représenter les différentes hypothèses de segmentation de la phrase en mots.

Afin de tirer partie des informations linguistiques au niveau des phrases, nous étudions différentes méthodes d'intégration des modèles de langage, dans le processus de reconnaissance.

Nous considérons également deux approches de reconnaissance de phrases :

- l'*approche MAP* (*Maximum A Posteriori*) qui cherche à maximiser la probabilité *a posteriori* de la phrase résultat ;
- l'*approche consensus*, une approche particulièrement originale qui cherche à maximiser les probabilités *a posteriori* des mots de la phrase résultat et qui a été introduite en reconnaissance automatique de la parole.

La reconnaissance de phrases s'abstrait plus ou moins de la représentation initiale des mots, puisque nous nous basons sur leur reconnaissance. Nous pouvons alors tirer partie de techniques utilisées en reconnaissance automatique de la parole, pour intégrer des connaissances linguistiques afin de désambiguïser le résultat de la reconnaissance des mots. Cependant, il faut adapter ces approches aux spécificités de la reconnaissance d'écriture manuscrite.

L'utilisation de connaissances linguistiques n'est cependant pas suffisante pour résoudre tous les problèmes de reconnaissance. Il est alors intéressant d'identifier les erreurs restantes, afin de les signaler à l'utilisateur ou encore pour pouvoir les corriger ensuite.

Nous proposons enfin une stratégie pour détecter les potentielles erreurs de reconnaissance, sur la phrase obtenue par l'approche MAP. Nous proposons ensuite de corriger ces erreurs potentielles, lorsque cette correction est possible. Quand la correction n'est pas possible, nous le signalons en rejetant le résultat de la reconnaissance.

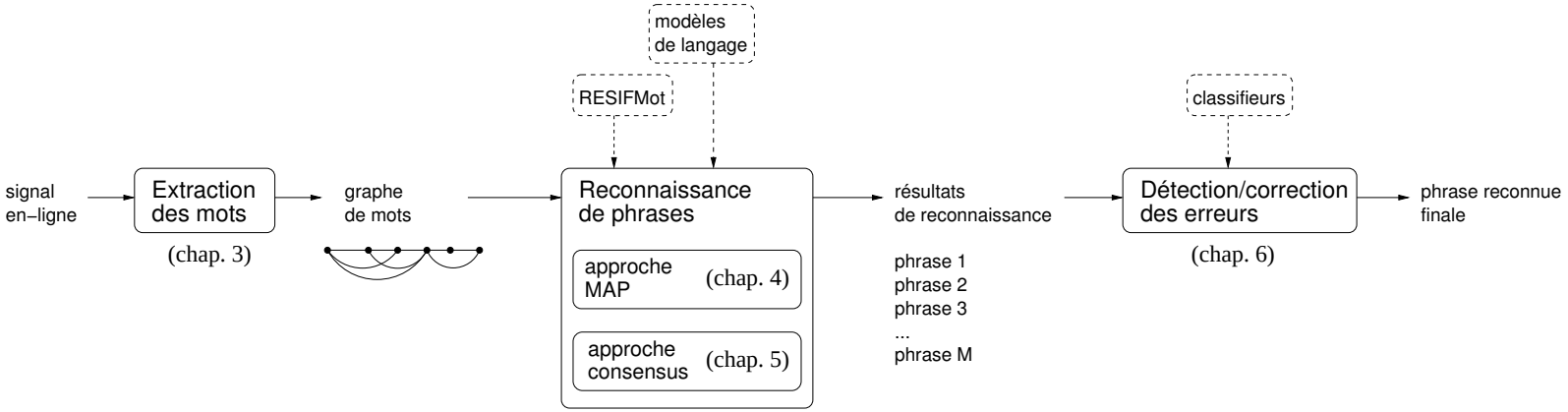


FIG. 3 – Principe du système complet de reconnaissance de phrases.

Pour réaliser cette approche complète de *détection et correction des erreurs potentielles* de reconnaissance, nous nous appuyons sur les probabilités *a posteriori* des mots, issues de l'approche consensus, ainsi que sur des classifieurs dédiés (*SVMs*, pour Séparateurs à Vaste Marge) et sur la notion de rejet.

Organisation du manuscrit

Ce manuscrit est composé d'une partie introductive puis de deux parties, basées sur des représentations différentes de l'ensemble des hypothèses de phrases. Nous considérons tout d'abord une représentation standard, sous la forme d'un graphe de mots, puis nous utilisons une représentation plus originale, basée sur des réseaux de confusion.

Les chapitres 1 et 2 présentent les états de l'art nécessaires tout au long de la thèse. Les états de l'art plus spécifiques seront présentés dans les chapitres concernés.

Dans le **chapitre 1**, nous nous focalisons plus particulièrement sur l'état de l'art lié à la reconnaissance de textes manuscrits, en introduisant tout d'abord les différences au niveau des représentations en-ligne et hors-ligne du signal. Cela permet de situer les étapes de la reconnaissance qui sont dépendantes de cette représentation et celles qui en sont plus décorrélées.

Nous présentons ensuite un état de l'art plus spécifique aux parties de la reconnaissance de textes abordées dans la thèse, afin de situer nos travaux par rapport à l'existant. Enfin, comme nous nous basons sur le système de reconnaissance de mots RESIFMot, nous le présentons, en détaillant plus particulièrement les informations qui seront utiles par la suite.

Le **chapitre 2** présente un état de l'art sur les techniques de modélisation du langage. Nous introduisons tout d'abord différents domaines d'application utilisant des modèles de langage, pour montrer l'apport de telles informations.

Nous comparons ensuite deux approches de modélisation (structurelle et statistique) pour nous focaliser finalement sur l'approche statistique, qui est celle que nous avons choisie dans nos travaux. Nous présentons alors le principe général de la modélisation statistique du langage, en nous intéressant plus particulièrement aux modèles n-grammes et n-classes, qui sont les modèles de langage que nous utilisons.

Nous présentons également d'autres modèles de langage, utilisés plus récemment et représentant des perspectives intéressantes.

La **première partie** concerne la reconnaissance de phrases, en se basant sur une représentation des hypothèses de phrases, sous la forme d'un graphe de mots [ONA97].

Dans le **chapitre 3**, nous présentons notre approche de segmentation des phrases en mots, qui se base sur la caractérisation des espaces entre les différentes parties du tracé manuscrit.

Pour pouvoir permettre la proposition d'hypothèses de segmentation additionnelles, nous ajoutons un indice de confiance sur le résultat de la caractérisation des espaces. Nous proposons alors d'étendre l'approche précédente, en utilisant cet indice de confiance pour ajouter des

hypothèses supplémentaires de segmentation, dans le graphe de mots généré. Cet ajout est réalisé en prenant en compte la combinatoire liée à la taille du graphe.

Enfin, nous présentons les expérimentations menées pour trouver la meilleure stratégie de segmentation.

Dans le **chapitre 4**, nous présentons l'intégration de modèles de langage, dans le cadre d'une reconnaissance basée sur l'approche *Maximum A Posteriori* (MAP). Dans cette approche, la phrase résultat de la reconnaissance correspond à la phrase ayant la probabilité la plus élevée, parmi les phrases représentées dans le graphe de mots. Pour identifier la phrase résultat, nous utilisons des connaissances sur la forme graphique des mots, ainsi que des informations lexicales et syntaxiques. Nous détaillons deux approches permettant d'intégrer des modèles de langage de manière efficace, pour apporter des informations syntaxiques lors de la reconnaissance de phrases.

La première approche proposée intègre les modèles de langage directement lors de l'exploration du graphe de mots, en pondérant leur importance vis-à-vis des autres informations utilisées (graphiques et lexicales) et en considérant les éventuels problèmes de segmentation.

Dans la deuxième approche proposée, des modèles de langage sont combinés et utilisés à la fois pendant l'exploration du graphe de mots et pour réordonner les hypothèses de phrases produites par cette exploration. Cela permet de tirer partie des spécificités des différents modèles (plus grande précision de la modélisation, connaissances différentes modélisées, par exemple).

Nous étudions enfin l'intégration de différents modèles de langage, pour chacune des deux approches d'intégration, afin de trouver le meilleur tant au point de vue des performances obtenues que de la place mémoire occupée.

La **seconde partie** s'appuie sur une représentation originale des hypothèses de phrases, sous la forme d'un réseau de confusion [Man00]. Cette technique est issue de la reconnaissance automatique de la parole et permet de mettre en évidence les positions de la phrase où se situent des ambiguïtés. La reconnaissance des phrases s'effectue alors selon cette approche consensus.

Le **chapitre 5** présente l'approche originale de reconnaissance basée sur les réseaux de confusion et qui s'appuie pour cela sur les probabilités *a posteriori* des mots, en cherchant à maximiser celles-ci. La probabilité *a posteriori* d'un mot correspond à la somme des probabilités de tous les chemins auxquels il appartient.

Nous introduisons tout d'abord un état de l'art sur l'utilisation des réseaux de confusion en reconnaissance automatique de parole, en expliquant le calcul des probabilités *a posteriori* des mots ainsi que la recherche de la meilleure phrase à partir de ces probabilités.

Nous proposons ensuite des modifications pour nous adapter à notre tâche de reconnaissance d'écriture, après avoir expliqué les différences liées à la reconnaissance d'écriture.

Enfin, nous présentons les expérimentations menées, en comparant notamment la nouvelle tâche de reconnaissance avec l'approche MAP, plus classique.

Dans le **chapitre 6**, nous présentons une approche originale de détection d'erreurs potentielles, au niveau des mots de la phrase résultat de la reconnaissance par approche MAP. Cette

détection s'appuie sur l'exploitation des probabilités *a posteriori* de ces mots, utilisées comme indices de confiance : cela permet de tirer partie des spécificités des deux approches de reconnaissance déjà présentées.

Nous introduisons tout d'abord un état de l'art sur les mesures de confiance utilisées en reconnaissance d'écriture manuscrite mais aussi en reconnaissance automatique de parole, notamment celles basées sur les probabilités *a posteriori* des mots.

Nous détaillons ensuite l'approche de détection proposée, en considérant deux cas d'erreurs potentielles. Nous présentons alors une approche de correction basée sur l'utilisation de classifieurs de type SVM (dédiés à chacun des deux cas d'erreurs potentielles), pour confirmer ou remettre en cause le résultat de la reconnaissance.

Nous ajoutons ensuite un mécanisme de rejet basé lui aussi sur des classifieurs dédiés, pour rejeter le résultat de la reconnaissance s'il est considéré comme non correct et qu'il ne pourra pas être corrigé par l'étape de correction proposée.

Nous présentons enfin des résultats sur les apports de cette approche, dans chacun des deux cas d'erreurs potentielles considérés, puis en considérant ces deux cas conjointement ainsi qu'en ajoutant le mécanisme de rejet. Nous comparons notamment les taux d'erreur obtenus avec ceux atteints en effectuant la reconnaissance avec l'approche MAP.

La **conclusion générale** clôt ce manuscrit, en dressant un bilan du travail effectué et en présentant également des perspectives liées aux différentes approches proposées.

Chapitre 1

État de l'art pour la reconnaissance de textes manuscrits

1.1 Introduction

L'objectif de ce chapitre est de situer la thèse par rapport au domaine de l'écriture manuscrite et plus particulièrement celui de la reconnaissance de textes en-ligne.

Dans la section 1.2, nous introduisons les deux représentations du signal manuscrit et nous montrons l'impact de cette représentation sur les différentes étapes de la reconnaissance. Nous présentons ensuite, dans la section 1.3, un état de l'art sur la reconnaissance de textes manuscrits, en nous focalisant plus particulièrement sur les problèmes de la segmentation de textes en mots ainsi que sur l'intégration efficace de connaissances linguistiques. Enfin, dans la section 1.4, nous détaillons le principe du système de reconnaissance de mots, RESIFMot, sur lequel est basé notre système de reconnaissance de phrases.

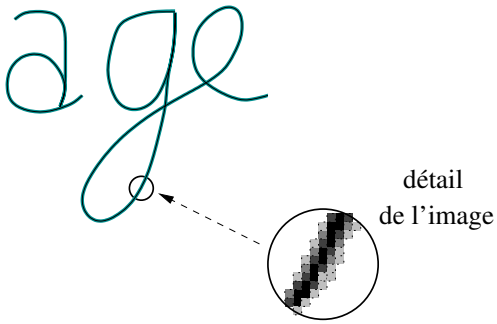
1.2 Représentation de l'écriture manuscrite

Il existe deux représentations possibles du signal manuscrit [PS00] : la représentation *hors-ligne* ou la représentation *en-ligne*.

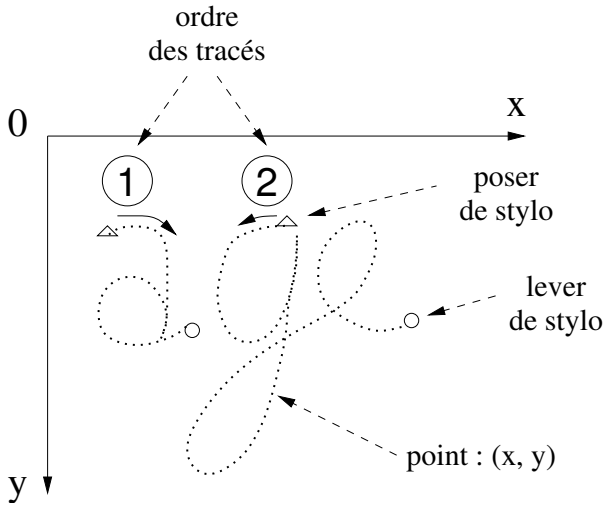
La figure 1.1 illustre ces deux types de représentation du signal manuscrit.

Dans un document *hors-ligne* (voir figure 1.1(a)), le texte est représenté par une image, obtenue par numérisation du document original. Le signal d'entrée du système de reconnaissance est alors une matrice de pixels en niveau de gris. Pour la reconnaissance, il faut au préalable identifier les parties de l'image qui correspondent aux zones d'écriture.

Un document *en-ligne* (voir figure 1.1(b)) est produit en utilisant un stylet sur un dispositif électronique (tablette, assistant personnel, par exemple). Le signal consiste en une suite de points, ordonnés temporellement. Les informations dynamiques de ce signal en font une représentation plus riche que le signal hors-ligne. Cela permet de pouvoir connaître la vitesse ou l'accélération, par exemple, mais surtout le sens et l'ordre du tracé.



(a) Signal hors-ligne



(b) Signal en-ligne

FIG. 1.1 – Représentations du signal du mot *age*.

Bien que le cadre de cette étude soit la reconnaissance d'écriture manuscrite en-ligne, cet état de l'art recouvre également des travaux sur la reconnaissance hors-ligne. En effet, plus la reconnaissance se trouve à un haut niveau (lettre puis mots puis textes), moins il existe de différences entre les approches de reconnaissance hors-ligne et en-ligne. Les domaines applicatifs sont cependant différents.

Les approches utilisées pour l'intégration de connaissances linguistiques seront alors quasiment indépendantes de la représentation du signal manuscrit. De même, nous pourrons nous inspirer des approches de segmentation hors-ligne, malgré les différences entre les approches de segmentation, car les recherches dans le domaine de la reconnaissance hors-ligne sont généralement plus avancées que celles en reconnaissance en-ligne.

1.3 État de l'art sur la reconnaissance de textes manuscrits

La reconnaissance d'écriture manuscrite est bien avancée en ce qui concerne la reconnaissance de caractères et de mots mais elle est relativement récente concernant celle des textes. Cette tâche de reconnaissance de textes est plus complexe puisqu'il s'agit de l'étape ultime d'une chaîne complète de reconnaissance. Il est notamment nécessaire de prendre en compte la segmentation du texte en mots ainsi que d'utiliser des connaissances linguistiques pour améliorer la reconnaissance des mots de ces textes.

Nous présentons tout d'abord le principe général de la reconnaissance de textes. Ensuite, nous nous focalisons plus particulièrement sur l'étape de segmentation du texte en ses mots ainsi que sur l'étape d'intégration de connaissances linguistiques, comme ce sont ces deux étapes qui nous intéressent dans la suite de cette thèse.

1.3.1 Principe général de la reconnaissance de textes

En utilisant une formalisation statistique issue de la théorie de l'information, le problème de la reconnaissance de textes manuscrits consiste à trouver la séquence de mots $\widehat{W}$ qui maximise la probabilité *a posteriori* parmi toutes les phrases W : cette approche est l'approche *MAP* (*Maximum A Posteriori*). La phrase $\widehat{W}$ est définie par :

$$\widehat{W} = \arg\max_{W} \ P(W|S) \tag{1.1}$$

avec S le signal correspondant au texte (une image, dans le cas de la reconnaissance hors-ligne ou une suite de points ordonnés temporellement, dans le cas de la reconnaissance en-ligne). En utilisant le théorème de Bayes, l'équation 1.1 peut être réécrite de la façon suivante :

$$\widehat{W} = \arg\max_{W} \ \frac{P(S|W) \ P(W)}{P(S)} \tag{1.2}$$

et, comme le signal S est constant, il n'intervient pas dans le choix de $\widehat{W}$. L'équation 1.2 se réécrit finalement en :

$$\widehat{W} = \arg\max_{W} \ P(S|W) \ P(W) \tag{1.3}$$

avec $P(S|W)$ la probabilité que le signal S ait été généré par le modèle de phrase W et $P(W)$ la probabilité *a priori* de la phrase W. Le premier terme est estimé à partir du système de reconnaissance des mots de la phrase et le deuxième terme est généralement estimé par un modèle de langage.

Nous présentons tout d'abord un état de l'art sur la segmentation de textes en mots, ce qui permet d'utiliser le système de reconnaissance de mots sur chacun des mots ainsi identifiés.

1.3.2 Extraction de mots

Le cadre de nos travaux étant la reconnaissance de textes, la tâche de segmentation consiste à retrouver les lignes puis les mots. C'est pourquoi nous ne détaillerons pas les travaux de [BSH04, BA04, JNS01, SWR$^+$03, YSR$^+$05] qui concernent plutôt la segmentation de pages de notes manuscrites pouvant contenir des dessins ou encore l'analyse de la structure des pages.

Il existe alors deux grandes approches pour la segmentation de textes en mots. Celle-ci peut être faite de manière implicite, conjointement à la reconnaissance du texte, ou de manière explicite. Dans ce dernier cas, elle est généralement réalisée lors d'une première étape afin de réduire la complexité de la tâche de reconnaissance.

Comme c'est cette approche que nous avons choisie, afin de pouvoir utiliser notre système de reconnaissance de mots RESIFMot (voir section 1.4), nous présentons plus particulièrement les travaux liés à cette tâche de segmentation. Nous nous attacherons également à regarder si les méthodes proposées permettent de s'appliquer à une segmentation à la volée, c'est-à-dire au fur et à mesure de l'écriture. Nous souhaitons en effet avoir la possibilité de faire évoluer le système vers ce type de reconnaissance.

La segmentation d'une phrase en-ligne consiste à extraire chacune des listes de traces correspondant aux différents mots de la phrase, à partir du tracé manuscrit de la phrase. C'est pourquoi nous parlerons indifféremment d'extraction de mots ou de segmentation de phrases.

La principale approche utilisée consiste à identifier les espaces entre chacune des traces, afin de savoir s'il s'agit d'un espace entre deux mots ou d'un espace à l'intérieur d'un même mot. Le problème principal est alors de trouver la méthode la plus appropriée pour calculer la distance entre deux traces : cette distance doit tenir compte des variations de style d'écriture telles que l'inclinaison de la ligne d'écriture, l'inclinaison des caractères ou encore leur taille.

Dans [Oud03], la reconnaissance est effectuée sur de l'écriture scripte et la tâche de détection d'espaces consiste à classer les espaces entre traces consécutives, à l'aide d'un réseau de neurones. Pour cela, la distance horizontale entre boîtes englobantes des traces est utilisée mais celle-ci est sujette aux variations de style d'écriture.

[LSB06] utilisent également la distance horizontale entre boîtes englobantes mais après avoir effectué une phase de pré-traitement du signal, qui le normalise par rapport aux inclinaisons de ligne d'écriture et de caractères. De plus, les lignes de bases sont également déterminées pour pouvoir normaliser la taille des caractères. Un algorithme de segmentation itératif permet ensuite d'identifier les espaces inter-mot et intra-mot. Néanmoins, cette méthode considérant la ligne de texte entière, elle n'est pas utilisable pour la reconnaissance à la volée, ce que nous souhaitons prendre en compte dans notre tâche de segmentation.

1.3.3 Intégration de connaissances linguistiques au niveau des phrases

Il existe peu de systèmes de reconnaissance de textes ou de phrases, qu'ils soient hors-ligne ou en-ligne, mais ils s'appuient tous des connaissances linguistiques. Les *modèles de langage* utilisés permettent de représenter ces informations et viennent essentiellement du domaine de la reconnaissance automatique de la parole. L'objectif de cette section est de présenter les travaux utilisant des modèles de langage (un état de l'art sur les modèles de langage sera donné dans le chapitre 2).

Comme cette utilisation des modèles de langage est relativement indépendante de la représentation du signal manuscrit (hors-ligne ou en-ligne), nous considérerons indifféremment les travaux en reconnaissance de textes hors-ligne ou en-ligne, d'autant plus qu'il existe peu de travaux en reconnaissance en-ligne [LB07, Mar04, Per05, PVGM06].

Types de modèles de langage utilisés Les modèles de langage les plus couramment utilisés sont les *modèles bigrammes* [LB07, MB01, Mar04, Per05, VBB04, ZB04a]. Ces modèles de langage reposent sur l'estimation des probabilités d'apparition des suites de 2 mots (pour plus de détails, voir chapitre 2, section 2.4.3). L'utilisation de ces modèles de langage permet une réduction importante du taux d'erreur sur les mots, pouvant aller jusqu'à plus de 50 %.

Dans [Per05, VBB04, ZB04a], des modèles trigrammes (basées sur des suites de 3 mots) sont également considérés mais ils n'apportent pas beaucoup d'améliorations, comparé aux modèles bigrammes.

Des modèles de langage basés cette fois sur l'estimation des probabilités de suites de classes sont utilisés dans [Per05]. Les classes représentent des regroupements de mots et elles peuvent être créées en se basant sur des critères statistiques ou encore à partir de la nature grammaticale des mots, par exemple (pour plus de détails, voir chapitre 2, section 2.4.5). Dans [Per05], ces deux types de modèles n-classes sont utilisés et permettent d'obtenir des performances proches de celles obtenues avec les modèles n-grammes. Ces modèles n-classes présentent l'avantage d'être de plus petite taille que les modèles n-grammes. De plus, la combinaison des deux types de modèles n-classes permet de surpasser légèrement les performances obtenues avec le modèle n-gramme.

Intégration des modèles de langage Il existe deux façons d'intégrer le modèle de langage. La façon la plus simple consiste à l'utiliser lors d'une phase de post-traitement, pour réordonner une liste d'hypothèses de phrases obtenue lors de la reconnaissance simple [Mar04, Per05]. Cependant, la deuxième approche est plus efficace puisqu'elle intègre le modèle de langage directement durant la phase de reconnaissance, ce qui permet d'en tirer partie le plus tôt possible [LB07, MB01, PVGM06, VBB04, ZB04a].

Ces deux approches peuvent également être combinées. Ainsi, un modèle bigramme est utilisé dans [ZCB06], durant la reconnaissance, pour produire une liste d'hypothèses de phrases, alors qu'un modèle de langage basé sur une grammaire hors-contexte probabiliste (pour plus de détails, voir chapitre 2, section 2.4.7.6) permet de réordonner cette liste. Cette étape de post-traitement ne permet cependant pas d'obtenir d'amélioration supplémentaire significative.

Optimisation de l'intégration des modèles de langage Dans [BB05, LB07, ZB04b], des pondérations sont utilisées pour intégrer efficacement les modèles de langage et prendre en compte les éventuels problèmes de segmentations. L'équation 1.3 est alors modifiée :

$$\widehat{W} = \arg\max_{W} \; P(S|W) \; P(W)^{\gamma} \; \delta^{'N} \tag{1.4}$$

avec N le nombre de mots de la phrase W, γ le *poids linguistique* (GSF, pour *Grammar Scale Factor*) utilisé pour compenser l'impact du modèle de langage vis-à-vis du système de reconnaissance de mots et $\delta^{'}$ le poids pour gérer les problèmes de sur- et sous-segmentation (WIP, pour *Word Insertion Penalty*). Lorsque $\delta^{'} < 1$, les phrases courtes sont favorisées (compensation des sous-segmentations) et lorsque $\delta^{'} > 1$, ce sont les phrases longues qui sont préférées (compensation des sur-segmentations).

Comme les probabilités sont généralement faibles, il est plus fréquent d'utiliser des *log-probabilités* plutôt que les probabilités (ce qui permet aussi de diminuer la charge de calcul, les

additions étant moins coûteuses que les multiplications). L'équation 1.4 devient alors :

$$\widehat{W} = \arg\max_{W} \; log(P(S|W)) \; + \; \gamma \, log(P(W)) \; + \; \delta \, N. \tag{1.5}$$

L'optimisation des poids γ et δ permet d'obtenir une réduction plus importante du taux d'erreur sur les mots que lorsque le modèle de langage a la même importance que le système de reconnaissance de mots (c'est-à-dire quand $\gamma = 1$ et $\delta = 0$).

Dans [BB05], différentes valeurs de ces pondérations sont utilisées pour générer plusieurs hypothèses de phrases. Celles-ci sont ensuite combinées grâce à l'approche ROVER [SG00] (utilisée en reconnaissance automatique de la parole) : le choix final des mots est alors fonction du nombre d'occurrences des mots, dans l'ensemble des hypothèses de phrases. Cela permet encore d'améliorer le taux de reconnaissance (de 70,7 % à 72,3 %).

Dans nos travaux, nous considérerons l'intégration de modèles n-grammes et n-classes pour la reconnaissance, en cherchant à les utiliser le plus tôt possible et en étudiant également des combinaisons de ces modèles de langage. Nous nous attacherons également à optimiser leur intégration par rapport au système de reconnaissance de mots.

1.4 Principe du système de reconnaissance de mots : RESIFMot

RESIFMot est un système de reconnaissance de mots cursifs isolés omni-scripteur, utilisant des connaissances lexicales [Anq97, Car05] (voir figure 1.2).

Ce système est basé sur une approche analytique explicite et sur un système de reconnaissance de caractères, RESIFCar [Anq97], s'appuyant sur une modélisation des caractères par des Systèmes d'Inférence Floue (SIF). Le système peut également être qualifié d'hybride car il utilise des informations globales (corps du mot, traits proéminents). Les caractères modélisés pour notre étude sont les vingt-six lettres de l'alphabet dans leur forme minuscule et quelques symboles spéciaux.

La reconnaissance d'un mot manuscrit est décomposée en plusieurs étapes (voir figure 1.2) :

1. **Analyse physique** : détection des traits descendants fondamentaux du tracé et des signes diacritiques, estimation du corps du mot et des limites des zones ascendantes et descendantes et segmentation du mot en allographes[1] ;
2. **Reconnaissance des allographes** ;
3. **Analyse logique** : organisation des allographes dans un treillis de segmentation et parcours du treillis pour rechercher les meilleurs chemins ;
4. **Post-traitement lexical** : correction des propositions de parcours du treillis.

Nous détaillons les étapes permettant une compréhension globale du système et utiles pour les chapitres suivants.

[1] variantes de styles dans une même classe de caractère.

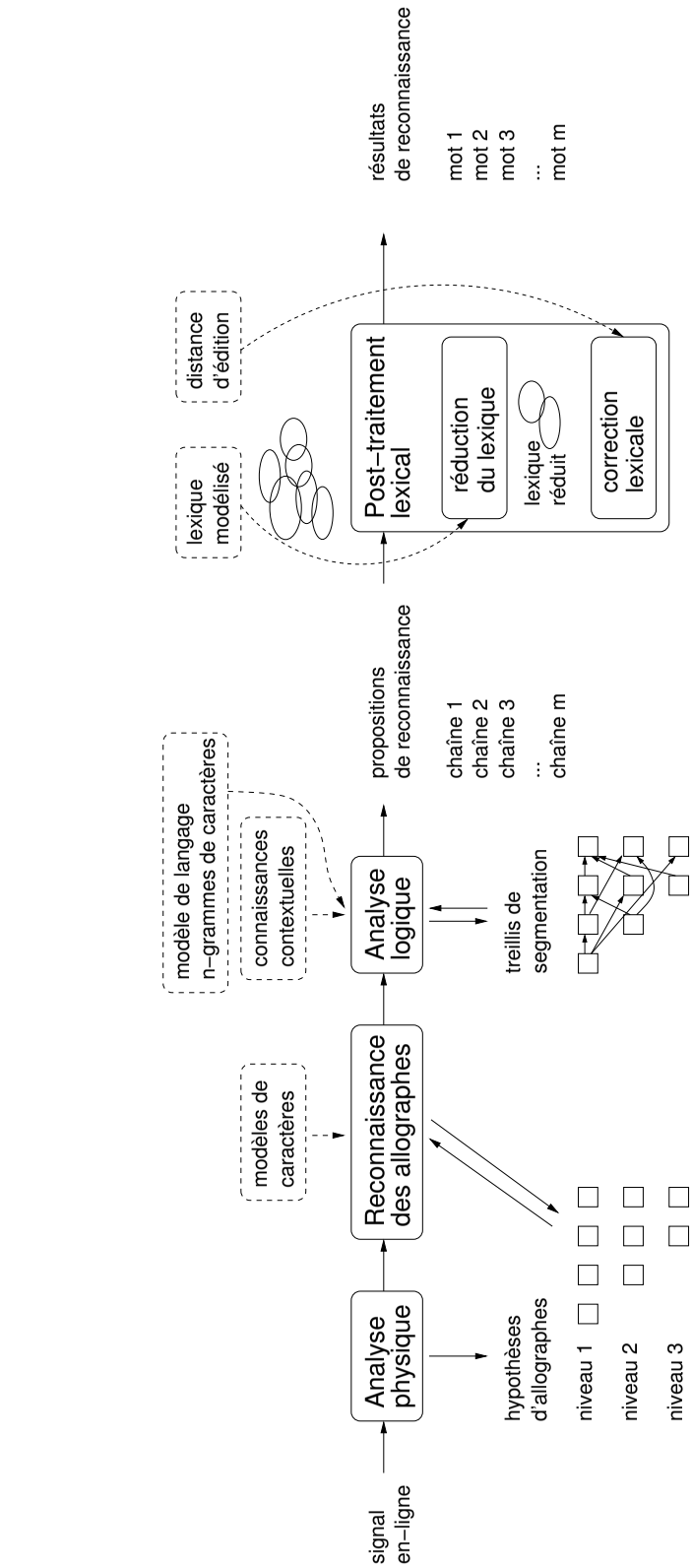


FIG. 1.2 – Principe du système de reconnaissance de mots RESIFMot.

1.4.1 Analyse physique

RESIFMot génère une sur-segmentation progressive du mot à reconnaître. Le principe de cette segmentation est basé sur des connaissances spécifiques à la structure des caractères manuscrits. La structure fondamentale correspond aux traits descendants des caractères. Chaque caractère peut ainsi contenir un, deux ou trois traits descendants fondamentaux, indépendamment du style d'écriture. Par exemple, les caractères *c*, *e* et *j* sont composés d'un seul trait descendant fondamental alors que *a*, *d* et *g* sont composés de deux traits descendants fondamentaux et que *m* est composé de trois traits descendants fondamentaux (voir figure 1.3).

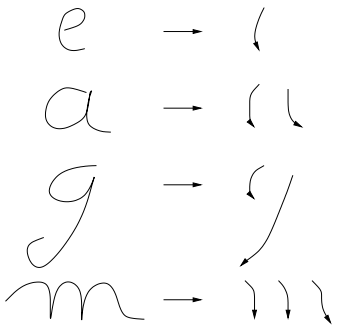


FIG. 1.3 – Traits descendants fondamentaux de certains caractères.

Le tracé du mot peut ensuite être découpé en portions contenant un, deux ou trois traits descendants. Une segmentation à trois niveaux est alors proposée, chaque niveau i contenant toutes les portions à i traits descendants du mot (voir figure 1.4). Chacune de ces portions est potentiellement un allographe.

Les traits descendants sont aussi utilisés pour déterminer le corps du mot. Il correspond en fait à une zone plus restreinte et incluse dans le corps du mot utilisé classiquement (correspondant à la zone où sont écrites les lettres *a, e, m...*) : cette zone est plus stable et moins sensible au problème d'inclinaison de la ligne d'écriture. Le corps du mot ainsi défini correspond à la zone verticale qui recouvre le plus de traits descendants fondamentaux.

1.4.2 Reconnaissance des allographes

Le système de reconnaissance de caractères RESIFCar est utilisé pour identifier chaque hypothèse d'allographe en proposant une liste de couples *(caractère, taux d'adéquation)*, où le taux d'adéquation donne la similarité entre l'hypothèse d'allographe et un modèle de caractère.

1.4.3 Analyse logique

Les résultats de la phase de reconnaissance des caractères sont organisés dans un treillis composé d'un nœud par allographe (voir figure 1.4). Les arcs orientés relient les nœuds de façon à représenter toutes les segmentations possibles du mot (chaque chemin contient le mot complet). Le but de l'analyse logique est de parcourir le treillis et d'ordonner les chemins

FIG. 1.4 – Exemple de treillis de segmentation, pour le mot *simple*.

d'après les taux d'adéquation fournis par la reconnaissance de caractères et des informations contextuelles (ligne de base, caractères adjacents, signes diacritiques).

Le parcours du treillis permet de générer la liste des meilleures chaînes de caractères, ordonnées d'après leur taux d'adéquation global. Ce taux d'adéquation global correspond à la fusion des taux d'adéquation de reconnaissance de caractères, de la mesure de la cohérence spatiale entre allographes consécutifs et des probabilités des suites de caractères constituant la chaîne. Ce taux (appelé $score_{graphique}$ dans la suite de cette thèse) est une valeur positive entière et correspond à une mesure d'adéquation permettant de qualifier la distance entre la chaîne et le signal d'entrée. Cela signifie que plus sa valeur est proche de 0 et plus la chaîne correspond à la transcription du signal d'entrée, d'après le système de reconnaissance de mots (les valeurs des taux d'adéquation sont comprises entre 1 000 et 300 000).

Nous désignons par la suite par *propositions de reconnaissance* les chaînes de caractères issues de l'analyse logique du mot manuscrit (voir figure 1.2). Elles correspondent à des hypothèses de travail pour la phase de post-traitement lexical et ne sont pas (encore) des mots.

1.4.4 Post-traitement lexical

Le rôle du post-traitement lexical est de déterminer le mot du lexique le plus proche des meilleures propositions issues du parcours du treillis de segmentation. En fait, il s'agit d'ordonner les mots du lexique par rapport à leur ressemblance aux propositions de reconnaissance. Le traitement lexical est basé sur une distance d'édition dont les opérations d'édition et leurs coûts ont été appris automatiquement, afin de corriger les erreurs spécifiques à la reconnaissance d'écriture en contexte de mot.

Le lexique utilisé pour le calcul de la distance est composé de sous-lexiques provenant d'une réduction statique du lexique global. Cette réduction correspond à une sélection d'un ou plusieurs lexiques constitués *a priori*, qui contiennent des mots visuellement proches des propositions de reconnaissance. La sélection est réalisée par comparaison des mots avec les descripteurs des sous-lexiques. La proximité visuelle introduite est basée sur une représentation des mots par une *silhouette générale* (voir figure 1.5).

La silhouette générale utilise des caractéristiques globales aux mots, détectées de manière fiable par RESIFMot et quasiment indépendantes du style d'écriture : les signes diacritiques (points sur les *i* et *j*, barres de *t*, accents), la structure physique (composée des traits descendants les plus stables) et la longueur physique (le nombre de traits descendants fondamentaux).

Pour chaque proposition de reconnaissance, le mot le plus proche du lexique est ensuite recherché, en utilisant la distance d'édition ainsi définie. Les mots résultats, appartenant au lexique réduit, sont alors triés par rapport à la valeur minimale de leur distance à une proposition de reconnaissance. Dans la suite, nous appellerons $score_{lexique}$, la distance minimale associée à un mot et calculée à partir d'une proposition de reconnaissance. La valeur de ce score est positive et, de la même façon que pour le $score_{graphique}$, plus cette valeur est proche de 0 et plus le mot est proche d'une proposition de reconnaissance (les valeurs de ce score varient entre 0 et 13).

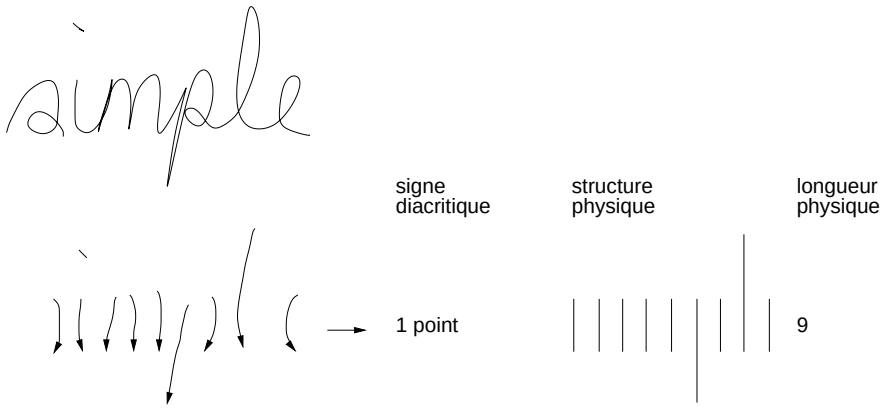


FIG. 1.5 – Exemple de silhouette générale, pour le mot *simple*.

1.5 Bilan

Dans ce chapitre, nous avons présenté un état de l'art relatif à notre tâche de reconnaissance de textes manuscrits en-ligne.

Après avoir introduit les différences au niveau de la représentation du signal manuscrit (hors-ligne ou en-ligne), nous nous sommes plus particulièrement focalisées sur deux aspects de la reconnaissance de textes, à savoir la segmentation des phrases en mots ainsi que l'utilisation de connaissance linguistiques pour guider la reconnaissance de phrases.

Comme nous nous basons sur un système de reconnaissance de mots existant, nous avons choisi d'effectuer une segmentation explicite de la phrase en ses mots. La segmentation choisie se base sur la caractérisation des espaces entre traces successives de la phrase manuscrite. Le calcul de la distance horizontale entre deux traces sera fait de telle façon qu'une extension vers une segmentation à la volée soit possible.

Nous avons vu que les systèmes de reconnaissance de textes, qu'ils soient hors-ligne ou en-ligne, tirent partie de l'utilisation de modèles de langage pour introduire des connaissances linguistiques. Dans le chapitre suivant, nous présentons un état de l'art sur les modèles de langage utilisés en reconnaissance d'écriture ainsi qu'en reconnaissance automatique de la parole.

Enfin, nous avons présenté le système de reconnaissance de mots RESIFMot, sur lequel s'appuie notre système de reconnaissance de phrases. Nous nous sommes plus particulièrement intéressées aux informations utilisées dans la suite de cette thèse.

Chapitre 2

État de l'art sur la modélisation du langage

2.1 Introduction

Les applications qui traitent des données appartenant au langage naturel, sous diverses formes (écrite ou orale, par exemple), nécessitent une représentation des caractéristiques de la langue traitée. Par exemple, dans le cas où les données sont des phrases, cette représentation donne les associations possibles de mots. Un modèle de langage correspond à une telle représentation et son utilisation permet de capter les régularités d'une langue.

Dans ce chapitre, nous présentons tout d'abord quelques domaines exploitant des modèles de langage. Nous comparons ensuite la modélisation structurelle du langage (représentation sous forme de règles de composition des phrases) et la modélisation statistique du langage (représentation sous forme de probabilités d'apparition de suites de mots), ces deux approches constituant les deux principales approches permettant de modéliser le langage naturel. Enfin, nous nous focalisons plus particulièrement sur la modélisation statistique du langage, puisqu'elle est plus adaptée à notre problématique, et nous comparons notamment différentes techniques de modélisation statistique.

2.2 Domaines d'application

Afin de montrer différentes possibilités d'exploitation des modèles de langage, nous commençons par présenter plusieurs domaines dans lesquels ils sont couramment utilisés.

Correction orthographique Les méthodes automatiques pour la correction orthographique sont principalement utilisées dans les systèmes de traitement de texte [Kuk92]. Deux types d'erreurs peuvent être distingués.

Nous trouvons tout d'abord les erreurs conduisant à des « non-mots » c'est-à-dire que le mot écrit n'est pas valide dans le langage considéré (par exemple, en français, *pra* au lieu de

par). Un lexique peut être utilisé pour détecter ces erreurs. Il peut aussi être associé avec une distance d'édition permettant de retrouver le mot qui aurait vraisemblablement dû être écrit.

Le second type d'erreur concerne les mots dont la mauvaise écriture conduit à un mot différent mais qui est valide dans le langage considéré (par exemple, en français, *lange* au lieu de *langue*). Un modèle de langage (donnant la formation des phrases à partir des mots) peut être utilisé, à la fois pour détecter ces mots en erreur et pour les corriger.

Ces deux niveaux d'informations peuvent ensuite être utilisés conjointement, de façon complémentaire. En effet, le premier type d'erreur se situe au niveau lexical, sur la formation des mots, alors que le deuxième se place au niveau syntaxique, sur la formation des phrases.

Traduction automatique La traduction automatique d'une phrase d'un langage source vers un langage cible fait intervenir deux types de modèles [BCP+90]. Un modèle de traduction permet tout d'abord d'exprimer, dans la langue cible, les idées présentées dans la phrase source. Un modèle de langage de la langue cible est aussi utilisé afin de prendre en compte les structures grammaticales correctes de cette langue, en permettant notamment la réorganisation des mots dans la phrase cible.

Étiquetage de corpus Les corpus sont de grandes collections de textes représentatifs d'une langue. Pour exploiter ces corpus dans d'autres applications, il peut être nécessaire d'associer des informations à chacun de leurs mots (catégorie grammaticale, lemme[1]...). C'est ce que réalise la tâche d'étiquetage d'un corpus.

Le plus couramment, cette tâche d'étiquetage consiste à associer sa classe grammaticale à chacun des mots du corpus [MS99]. Comme la plupart des mots peuvent avoir plusieurs classes grammaticales (par exemple, en français, *souris* peut être un verbe ou un nom), il est nécessaire d'utiliser un modèle de langage, lors de cette phase d'étiquetage, afin de résoudre les ambiguïtés en se basant sur les dépendances entre les mots de la phrase.

Recherche d'informations Avec notamment l'essor d'Internet, la recherche d'informations devient un domaine de plus en plus important. Pour répondre plus efficacement aux requêtes des utilisateurs, des modèles de langage peuvent être utilisés afin d'incorporer des connaissances linguistiques dans les moteurs de recherche. Certains de ces moteurs de recherche permettent ainsi la saisie de requête en langage naturel [Owe00], ce qui permet une meilleure interprétation des requêtes des utilisateurs.

Saisie contrainte de texte Que ce soit pour la saisie de texte sur des périphériques mobiles comme les téléphones portables ou pour la saisie de texte pour des langues ayant un alphabet de taille importante (comme le chinois, par exemple), le clavier utilisé ne permet pas d'associer directement une touche à un caractère. Des modèles de langage peuvent ainsi être utilisés afin de suggérer ou prédire les lettres ou les mots, au fur et à mesure de la saisie du texte.

Sur les téléphones mobiles, l'algorithme T9 est ainsi le plus répandu pour cette tâche [MS02]. De la même façon, des modèles de langage sont utilisés pour la saisie de textes chinois, par

[1] Forme « simple » d'un mot. Par exemple, le lemme d'un verbe conjugué sera la forme infinitive de ce verbe.

exemple. En effet, la manière la plus simple de saisir de tels textes est la saisie *pinyin* qui permet d'entrer phonétiquement les caractères. Comme plusieurs caractères se prononcent de la même façon, le modèle de langage intervient alors pour choisir celui qui a vraisemblablement été écrit [CL00].

Reconnaissance automatique de la parole La reconnaissance de la parole est un domaine qui emploie de nombreux modèles de langage [Jel00, MS99]. Ceux-ci permettent en effet d'obtenir des informations sur la langue considérée, afin de choisir la phrase la plus vraisemblable parmi celles proposées par le système de reconnaissance, à partir du signal acoustique en entrée de celui-ci. L'équation 1.3 (présentée dans le chapitre 1) s'applique également pour combiner les informations acoustiques et les informations linguistiques.

Comme l'utilisation de modèles de langage en reconnaissance de la parole est très proche de notre utilisation en reconnaissance d'écriture, de nombreuses techniques, que nous présenterons dans la suite, sont ainsi applicables dans notre domaine.

2.3 Approche structurelle et approche statistique

Le langage naturel peut être modélisé selon deux principales approches : structurelle ou statistique. Dans cette section, nous présentons ces deux approches et nous comparons leurs avantages et inconvénients, pour justifier notre choix de se tourner vers une de ces approches plus particulièrement.

Dans l'approche structurelle, un modèle de langage s'exprime le plus souvent sous forme de *règles* qui définissent une grammaire du langage. L'ensemble de telles règles définit les possibilités d'association de mots, selon leurs catégories lexicales, et permet de modéliser la structure d'une phrase donnée [Cho57]. Ces règles permettent ainsi de déterminer la validité d'une phrase : s'il existe une séquence de règles permettant de décrire la phrase alors celle-ci est acceptée, sinon elle est rejetée.

Dans l'approche statistique, la langue est représentée par les fréquences d'apparition de chacune des suites de mots pouvant être formées. Pour cela, de grands corpus, considérés comme représentatifs de la langue et de ces phénomènes, sont utilisés pour l'apprentissage des modèles de langage, par inférence statistique [MS99].

Un des avantages des approches statistiques est leur indépendance vis-à-vis de l'expertise de linguistes, comme les modèles de langage sont appris automatiquement à partir de corpus. La modélisation du langage suivant cette approche est par conséquent dépendante des données utilisées et peut également attribuer une probabilité non nulle à des phrases syntaxiquement incorrectes. Un des avantages des modèles construits sur une approche structurelle est donc de pouvoir exprimer la nature grammaticale des phrases. Néanmoins, une des principales difficultés de ces méthodes d'analyse structurelle restent la conception des grammaires, c'est-à-dire la définition des règles. En effet, celles-ci doivent être suffisamment robustes pour prendre en compte tous les phénomènes syntaxiques d'une langue.

De part la nature statistique des modèles de langage construits selon l'approche statistique, leur intégration dans un système de reconnaissance se révèle plus efficace. Les modélisations statistiques du langage permettent également une plus grande souplesse dans la réponse puisqu'ils associent une probabilité aux phrases. Cette probabilité évalue le fait que la phrase soit ou non correcte, contrairement aux approches structurelles qui associent un booléen acceptant ou rejetant la phrase.

Cependant, il existe des modèles de langage associant à la fois approche structurelle et statistique, comme les grammaires probabilistes hors-contexte (voir section 2.4.7.6), qui permettent une réponse plus souple en associant une probabilité exprimant la validité de la phrase considérée. L'utilisation de ces derniers modèles est plutôt indiquée pour des tâches dont le domaine est limité (à-cause de l'approche structurelle), comme la reconnaissance d'adresses postales manuscrites [SK97] ou la reconnaissance de la parole avec un vocabulaire spécifique et limité [JWS$^+$95], par exemple.

Dans le cas d'un domaine avec un vocabulaire plus large, ces modèles structurels du langage peuvent être utilisés lors d'une deuxième passe de reconnaissance, pour réorganiser une liste de phrases [HW01] ou en combinaison d'un modèle statistique du langage [ZCB06]. Dans ces derniers cas, un modèle statistique du langage est alors utilisé conjointement.

De part les avantages de l'approche statistique concernant l'efficacité de l'intégration de modèles de langage basés sur cette approche, dans un système de reconnaissance, et de la plus grande simplicité de l'apprentissage de ces modèles, nous avons choisi de nous tourner vers une modélisation statistique du langage, dans nos travaux. C'est pourquoi nous nous focalisons plus particulièrement sur cette approche, dans la section suivante. Nous reviendrons par la suite sur les modèles de langage intégrant également des informations de nature structurelle.

2.4 Modélisation statistique du langage

Nous présentons tout d'abord le principe général de la modélisation statistique du langage. Nous décrivons ensuite les mesures permettant d'évaluer la qualité des différents modèles de langage. Nous nous focalisons enfin sur différents modèles de langage et sur des techniques permettant de combiner des modèles de langage.

2.4.1 Principe général

La modélisation statistique du langage a pour but de capter les régularités d'une langue par inférence statistique sur de très grands corpus d'apprentissage (ensembles de textes), composés de données exprimées dans la langue étudiée [MS99].

Un *modèle de langage statistique* permet ainsi d'évaluer la probabilité *a priori* d'une suite de N mots $W = w_1 \dots w_N$. Cette probabilité est égale au produit des probabilités des mots en contexte la composant :

$$P(W) = \prod_{i=1}^{N} P(w_i|h_i) \tag{2.1}$$

avec $P(w_i|h_i)$ la probabilité d'apparition du mot w_i sachant que la suite de mots $h_i = w_1 \dots w_{i-1}$ précède directement w_i : ces probabilités $P(w_i|h_i)$ constituent les *paramètres* du modèle de langage. La suite de mots h_i est appelée *historique* du mot w_i.

Afin que $P(w_i|h_i)$ puisse être calculée pour $i = 1$, nous ajoutons un marqueur de début de phrase <*s*> ; nous avons ainsi $h_1 =$ <*s*>. De plus, pour que la somme des probabilités de toutes les phrases W soit égale à 1, un marqueur de fin de phrase </*s*> est ajouté et inclus au produit des probabilités des mots, l'équation 2.1 devenant ainsi :

$$P(W) = \prod_{i=1}^{N+1} P(w_i|h_i) \tag{2.2}$$

avec $w_{N+1} =$ </*s*>.

2.4.2 Évaluation de la qualité des modèles de langage

Le rôle d'un modèle de langage est d'estimer la probabilité *a priori* d'une séquence de mots. Ainsi, plus une séquence de mots W sera vraisemblablement conforme au modèle de langage, plus sa probabilité sera élevée. La mesure la plus simple pour évaluer la qualité d'un modèle de langage est donc de mesurer la probabilité donnée par ce modèle de langage, sur un ensemble de test T le plus représentatif de la langue considérée et composé des phrases $(W_1 \dots W_t)$. À partir des probabilités $P(w_i|h_i)$ données par le modèle de langage ML, la probabilité des phrases peut être calculée en utilisant l'équation 2.2 et la probabilité de l'ensemble de test T est alors donnée par :

$$P(T) = \prod_{k=1}^{t} P(W_k). \tag{2.3}$$

Ainsi, le modèle de langage attribuant la probabilité la plus élevée à l'ensemble T sera considéré comme le meilleur modèle.

De part sa meilleure expressivité et son indépendance vis-à-vis de la taille de l'ensemble de test, la mesure d'*entropie croisée* est la plus couramment utilisée. Elle vient de la relation entre prédiction et compression, issue de la théorie de l'information [Sha48] : étant donné un modèle de langage ML qui assigne une probabilité $P(T)$ à un ensemble de test T, un algorithme de compression peut être créé pour encoder le texte sur $-\log_2(P(T))$ bits. L'entropie croisée H_{ML} d'un modèle de langage ML, sur un ensemble de test T, est ainsi définie par :

$$H_{ML}(T) = -\frac{1}{|T|_w} \log_2(P(T)) \tag{2.4}$$

avec $|T|_w$ la longueur du texte T, mesurée en nombre de mots. Cette valeur peut être interprétée comme le nombre moyen de bits nécessaires pour encoder chacun des $|T|_w$ mots de l'ensemble de test, en utilisant l'algorithme de compression associé au modèle ML. Le meilleur modèle de langage sera alors celui dont l'entropie croisée est la plus faible.

La *perplexité*, mesure dérivée de l'entropie croisée, est en fait la plus utilisée pour évaluer les performances d'un modèle de langage. La perplexité PPL_{ML} d'un modèle de langage

ML, sur un ensemble de test T, correspond à la réciproque de la probabilité moyenne donnée à chacun des mots de l'ensemble de test T. Elle est reliée à l'entropie croisée par l'équation :

$$PPL_{ML}(T) = 2^{H_{ML}(T)}. \tag{2.5}$$

Cette quantité rend compte du pouvoir prédictif du modèle ML. Ainsi, $PPL_{ML}(T) = k$ signifie que le modèle de langage ML hésite en moyenne entre k mots. Ainsi, à l'instar de l'entropie croisée, plus la perplexité est faible et meilleur est le modèle de langage associé.

Dans le cadre de la reconnaissance d'écriture, il peut aussi être intéressant de mesurer les taux de reconnaissance obtenus sur un ensemble de test, en utilisant différents modèles de langage dans un système de reconnaissance. En réalité, nous évaluerons plutôt le *taux d'erreur* sur les mots (voir section 3.4.1.2), obtenu sur l'ensemble de test, cet indicateur étant bien corrélé avec la perplexité [KP02] : plus il est faible et meilleures sont les performances du système de reconnaissance.

2.4.3 Modèles n-grammes

Le principal problème de l'équation 2.2 est le nombre élevé d'historiques différents existant, ce qui conduit à un nombre très important de probabilités à estimer. De plus, la plupart de ces historiques apparaissent trop peu de fois dans le corpus pour que leur probabilité soit estimée de manière suffisamment fiable. Une solution pour résoudre ce problème est de regrouper les historiques dans des classes d'équivalence :

$$P(w_i|h_i) \approx P(w_i|\Phi(h_i)) \tag{2.6}$$

avec $\Phi(h_i)$ qui associe sa classe d'équivalence à chaque historique h_i.

Les modèles de langage de type n-gramme sont les plus utilisés [MS99]. Ils se basent sur des suites de n mots, la valeur n étant appelée *ordre* du modèle n-gramme : les historiques qui se terminent par les mêmes $n-1$ mots sont ainsi considérés comme équivalents. L'appellation des modèles n-grammes vient du domaine des modèles de Markov dont ces modèles sont une instance. Un modèle n-gramme peut ainsi être vu comme un modèle de Markov d'ordre $n-1$.

À partir de l'équation 2.6, nous obtenons alors :

$$P(w_i|\Phi(h_i)) = P(w_i|w_{i-n+1}^{i-1}). \tag{2.7}$$

avec $w_{i-n+1}^{i-1} = w_{i-n+1} \dots w_{i-1}$ l'historique réduit aux $n-1$ mots précédant w_i.

Les probabilités $P(w_i|w_{i-n+1}^{i-1})$ de l'équation 2.7 sont généralement estimées au *maximum de vraisemblance* (MLE) sur l'ensemble d'apprentissage :

$$P_{\text{MLE}}(w_i|w_{i-n+1}^{i-1}) = \frac{N(w_{i-n+1}^{i-1}w_i)}{\sum_{w_j \in V} N(w_{i-n+1}^{i-1}w_j)} \tag{2.8}$$

avec $N(.)$ le nombre d'occurrences de la suite de mots en paramètre, dans le corpus d'apprentissage, et V le vocabulaire.

Considérons un exemple simple[2] de modèle bigramme ($n = 2$). Le corpus d'apprentissage est composé de trois phrases : *« Antoine écoute Thom »*, *« Denis écoute une autre chanson »* et *« Elle écoute une chanson de Lionel »*. Calculons maintenant $P(\textit{Antoine écoute une chanson})$:

$$P(\textit{Antoine}|\text{<s>}) = \frac{N(\text{<s> }\textit{Antoine})}{\sum_{w \in V} N(\text{<s> } w)} = \frac{1}{3}. \tag{2.9}$$

$$P(\textit{écoute}|\textit{Antoine}) = \frac{N(\textit{Antoine écoute})}{\sum_{w \in V} N(\textit{Antoine } w)} = \frac{1}{1}. \tag{2.10}$$

$$P(\textit{une}|\textit{écoute}) = \frac{N(\textit{écoute une})}{\sum_{w \in V} N(\textit{écoute } w)} = \frac{2}{3}. \tag{2.11}$$

$$P(\textit{chanson}|\textit{une}) = \frac{N(\textit{une chanson})}{\sum_{w \in V} N(\textit{une } w)} = \frac{1}{2}. \tag{2.12}$$

$$P(\text{</s>}|\textit{chanson}) = \frac{N(\textit{chanson }\text{</s>})}{\sum_{w \in V} N(\textit{chanson } w)} = \frac{1}{2}. \tag{2.13}$$

Avec les équations 2.9 à 2.13, nous obtenons :

$$\begin{aligned} P(\textit{Antoin écoute une chanson}) &= P(\textit{Antoine}|\text{<s>}) \times P(\textit{écoute}|\textit{Antoine}) \times P(\textit{une}|\textit{écoute}) \\ &\quad \times P(\textit{chanson}|\textit{une}) \times P(\text{</s>}|\textit{chanson}) \\ &\approx 0,06. \end{aligned} \tag{2.14}$$

En pratique, l'ordre n des modèles n-grammes utilisés est généralement égal à 2 (modèle *bigramme*) ou 3 (modèle *trigramme*) voire 4 ou 5 mais rarement au-delà puisque les performances du modèle de langage ne sont pas ou peu améliorées [Goo01] alors qu'il y a beaucoup de paramètres à estimer. En effet, pour un modèle trigramme avec un vocabulaire de 20 000 mots, le nombre de probabilités à estimer est 8.10^{12}. Cette limitation de l'ordre n rend ainsi difficile la prise en compte des dépendances longues dans les phrases.

Une autre limite de cette approche est qu'une probabilité non nulle peut être assignée à des n-grammes impossibles d'un point de vue linguistique. Réciproquement, des n-grammes possibles d'un point de vue linguistique peuvent avoir une probabilité nulle s'ils n'apparaissent pas dans le corpus d'apprentissage. En reprenant le corpus d'apprentissage de l'exemple précédent et en considérant la phrase *« Lionel écoute une chanson »*, nous obtenons ainsi :

$$P(\textit{écoute}|\textit{Lionel}) = \frac{N(\textit{Lionel écoute})}{\sum_{w \in V} N(\textit{Lionel } w)} = \frac{0}{1}. \tag{2.15}$$

L'équation 2.15 nous donne $P(\textit{Lionel écoute une chanson}) = 0$, le bigramme *« Lionel écoute »* n'étant pas présent dans l'ensemble d'apprentissage. Ceci est une mauvaise estimation car il est possible d'obtenir cette phrase, celle-ci étant correcte en français.

[2] Exemple adapté de [CG98].

2.4.4 Problème du manque de données

Malgré la réduction du nombre de probabilités à estimer, grâce au regroupement des historiques dans des classes d'équivalence (modèles n-grammes), certaines suites de mots possibles peuvent ne pas apparaître dans le corpus d'apprentissage : leur probabilité d'apparition sera estimée comme nulle alors qu'elle ne l'est pas. Pour traiter ce problème, une technique consiste à assigner une probabilité non nulle aux suites n'apparaissant pas dans le corpus d'apprentissage, en utilisant un *modèle de lissage*.

Pour ce faire, une solution générale, repose sur l'estimation de la probabilité conditionnelle $\widehat{P}(w_i|w_{i-n+1}^{i-1})$ en combinant deux composantes : un *modèle de décompte* et un *modèle de redistribution* (en anglais, ces deux composantes sont regroupées sous le terme *smoothing*). Les probabilités d'apparition des suites de mots présentes dans l'ensemble d'apprentissage sont alors réduites, puis la probabilité résultante est ensuite redistribuée.

Nous présentons tout d'abord les modèles de décompte et de redistribution existants puis trois exemples de techniques de lissage, combinant chacune un modèle de décompte et un modèle de lissage.

2.4.4.1 Décompte des probabilités

Le modèle de décompte est utilisé pour prélever une partie des probabilités allouées aux suites de mots présentes dans l'ensemble d'apprentissage. Deux techniques permettent d'extraire cette masse de probabilités.

Le *décompte linéaire* (en anglais, *Linear Discounting*) consiste à prélever une partie de la probabilité d'un n-gramme, *relativement* à son nombre d'occurrences dans l'ensemble d'apprentissage. La probabilité réduite, $\widehat{P}_{\mathrm{DecLin}}$, du n-gramme w_{i-n+1}^{i} s'exprime alors par :

$$\widehat{P}_{\mathrm{DecLin}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{N(w_{i-n+1}^{i}) \times \mathbf{r}}{N(w_{i-n+1}^{i-1})} & \text{si } N(w_{i-n+1}^{i}) > 0 \\ 0 & \text{sinon} \end{cases} \tag{2.16}$$

et la masse de probabilités ainsi mise de côté est alors donnée par :

$$\sum_{N(w_{i-n+1}^{i})>0} P_{\mathrm{MLE}}(w_i|w_{i-n+1}^{i-1}) \; - \sum_{N(w_{i-n+1}^{i})>0} \widehat{P}_{\mathrm{DecLin}}(w_i|w_{i-n+1}^{i-1}) = 1 \; - \; \mathbf{r} \tag{2.17}$$

avec $N(w_{i-n+1}^{i})$ le nombre d'occurrences du n-gramme w_{i-n+1}^{i}, dans l'ensemble d'apprentissage, et $\mathbf{r}$ le facteur de décompte relatif au nombre d'occurrences du n-gramme ($0 < \mathbf{r} < 1$).

Le *décompte absolu* (en anglais, *Absolute Discounting*) consiste à extraire une valeur *fixe* de la probabilité d'un n-gramme, indépendante de son nombre d'occurrences. La probabilité réduite $\widehat{P}_{\mathrm{DecAbs}}$ se note alors :

$$\widehat{P}_{\mathrm{DecAbs}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{N(w_{i-n+1}^{i}) - \mathbf{D}}{N(w_{i-n+1}^{i-1})} & \text{si } N(w_{i-n+1}^{i}) > 0 \\ 0 & \text{sinon} \end{cases} \tag{2.18}$$

et la masse de probabilités ainsi obtenue est alors :

$$\sum_{N(w_{i-n+1}^{i})>0} P_{\mathrm{MLE}}(w_i|w_{i-n+1}^{i-1}) - \sum_{N(w_{i-n+1}^{i})>0} \widehat{P}_{\mathrm{DecAbs}}(w_i|w_{i-n+1}^{i-1}) = \frac{\left|\left\{w_{i-n+1}^{i-1} : N(w_{i-n+1}^{i}) > 0\right\}\right| \times \mathbf{D}}{N(w_{i-n+1}^{i-1})} \tag{2.19}$$

avec $\left|\left\{w_{i-n+1}^{i-1} : N(w_{i-n+1}^{i}) > 0\right\}\right|$ le nombre d'historiques distincts w_{i-n+1}^{i-1}, précédant le mot w_i et **D** le facteur de décompte fixe sur les n-grammes.

2.4.4.2 Redistribution des probabilités

Une fois que la masse de probabilités a été extraite des n-grammes présents, il faut ensuite la redistribuer. Cette redistribution se fait généralement en considérant la probabilité du n-gramme à l'ordre $n-1$ et éventuellement récursivement aux ordres inférieurs. Ainsi, l'estimation de la probabilité du trigramme inconnu *« le miroir casse »* est réalisée à partir de celle du bigramme correspondant *« miroir casse »*. Une fois encore, deux techniques existent pour la redistribution des probabilités.

Le *repli* (en anglais, *back-off*) redistribue la masse précédemment extraite, aux *seuls n-grammes non-présents*. La probabilité du n-gramme w_{i-n+1}^{i} s'exprime alors par :

$$\widehat{P}_{\mathrm{Repli}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \widehat{P}_{\mathrm{Dec}}(w_i|w_{i-n+1}^{i-1}) & \text{si } N(w_{i-n+1}^{i}) > 0 \\ \lambda(w_{i-n+1}^{i-1}) \times \widehat{P}_{\mathrm{Repli}}(w_i|w_{i-n+2}^{i-1}) & \text{sinon} \end{cases} \tag{2.20}$$

avec $\widehat{P}_{\mathrm{Dec}}$ correspondant soit à $\widehat{P}_{\mathrm{DecLin}}$ soit à $\widehat{P}_{\mathrm{DecAbs}}$, $\lambda(w_{i-n+1}^{i-1})$ un poids dit de *repli* (dépendant de l'historique w_{i-n+1}^{i-1}) et w_{i-n+2}^{i} le n-gramme d'ordre inférieur correspondant au n-gramme w_{i-n+1}^{i}. Lors de l'utilisation d'un modèle de langage basé sur une redistribution par repli, si la probabilité d'un trigramme, par exemple, n'est pas estimée dans ce modèle, c'est celle du bigramme correspondant qui sera utilisée (ou celle de l'unigramme, si la probabilité du bigramme n'est pas estimée).

L'*interpolation*, quant à elle, redistribue les probabilités sur l'*ensemble des n-grammes*, qu'ils aient ou non été rencontrés. La probabilité du n-gramme w_{i-n+1}^{i} est alors définie par :

$$\widehat{P}_{\mathrm{Interpol}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \widehat{P}_{\mathrm{Dec}}(w_i|w_{i-n+1}^{i-1}) & \text{si } N(w_{i-n+1}^{i}) > 0 \\ \quad + \lambda(w_{i-n+1}^{i-1}) \times \widehat{P}_{\mathrm{Interpol}}(w_i|w_{i-n+2}^{i-1}) & \\ \lambda(w_{i-n+1}^{i-1}) \times \widehat{P}_{\mathrm{Interpol}}(w_i|w_{i-n+2}^{i-1}) & \text{sinon.} \end{cases} \tag{2.21}$$

Avec les modèles à redistribution par interpolation, et contrairement aux modèles avec redistribution par repli, la probabilité d'un trigramme, par exemple, est combinée avec celles du bigramme et de l'unigramme correspondants, que la probabilité du trigramme soit ou non estimée dans le modèle de langage.

2.4.4.3 Trois exemples de techniques de lissage

Nous présentons maintenant 3 techniques de *lissage* [CG98], combinant chacune un modèle de décompte et un modèle de redistribution.

Repli de Katz Ce modèle [Kat87] (en anglais, *Katz backing-off*) s'appuie sur l'estimation de Good-Turing (modèle de décompte linéaire). Celui-ci repose sur le fait qu'un n-gramme apparaissant r fois apparaît en réalité r^* fois, où r^* est défini par :

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{2.22}$$

avec r le nombre d'occurrences du n-gramme w_{i-n+1}^{i} et n_r le nombre de n-grammes apparaissant exactement r fois dans l'ensemble d'apprentissage. Le modèle de redistribution utilisé est de type repli.

La probabilité d'un n-gramme, en utilisant cette technique, est donnée par :

$$\widehat{P}_{\text{Katz}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{d_r \times r}{\sum_{w_i \in V} N(w_{i-n+1}^{i-1} w_i)} & \text{si } N(w_{i-n+1}^{i}) > 0 \\ \gamma(w_i|w_{i-n+1}^{i-1}) \times \widehat{P}_{\text{Katz}}(w_i|w_{i-n+2}^{i-1}) & \text{sinon} \end{cases} \tag{2.23}$$

avec

$$\gamma(w_i|w_{i-n+1}^{i-1}) = \frac{1 - \sum_{w_i \in V : N(w_{i-n+1}^{i-1} w_i) > 0} \widehat{P}_{\text{Katz}}(w_i|w_{i-n+1}^{i-1})}{\sum_{w_i \in V : N(w_{i-n+1}^{i-1} w_i) = 0} \widehat{P}_{\text{Katz}}(w_i|w_{i-n+2}^{i-1})} \tag{2.24}$$

le poids de repli et d_r un ratio de décompte, utilisé pour diminuer le nombre d'occurrences des n-grammes apparaissant dans l'ensemble d'apprentissage. En réalité, les n-grammes apparaissant plus de k fois sont jugés fiables et leur nombre d'occurrences n'est pas modifié. Ce ratio est donc calculé de la façon suivante :

$$d_r = \begin{cases} \frac{r^*}{r} & \text{si } r \leq k \\ 1 & \text{sinon.} \end{cases} \tag{2.25}$$

Katz suggère de fixer la valeur de k à 5.

Repli avec décompte absolu Ce modèle (en anglais, *Absolute Discounting backing-off*) utilise un modèle de décompte absolu et une redistribution de type repli [NEK94]. La probabilité d'un n-gramme est calculée par :

$$\widehat{P}_{\text{Abs}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{\max\{N(w_{i-n+1}^{i}) - D,\, 0\}}{\sum_{w_i \in V} N(w_{i-n+1}^{i-1} w_i))} & \text{si } N(w_{i-n+1}^{i}) > 0 \\ \gamma(w_i|w_{i-n+1}^{i-1}) \times \widehat{P}_{\text{Abs}}(w_i|w_{i-n+2}^{i-1}) & \text{sinon} \end{cases} \tag{2.26}$$

avec

$$\gamma(w_i|w_{i-n+1}^{i-1}) = \frac{D}{\sum_{w_i \in V} N(w_{i-n+1}^{i-1} w_i))} N_{1+}(w_{i-n+1}^{i-1} \bullet) \tag{2.27}$$

le poids de repli, $N_{1+}(w_{i-n+1}^{i-1}\bullet) = \left|\left\{w_i : N(w_{i-n+1}^{i}) > 0\right\}\right|$ le nombre de mots distincts qui suivent l'historique w_{i-n+1}^{i-1} et qui apparaissent une fois ou plus et D un décompte fixe dont la valeur est estimée par :

$$D = \frac{n_1}{n_1 \ + \ 2n_2} \tag{2.28}$$

avec n_1 (respectivement n_2) le nombre total de n-grammes apparaissant exactement une fois (respectivement deux fois), dans l'ensemble d'apprentissage.

Interpolation de Kneser-Ney modifiée Cette dernière technique, proposée par Chen et Goodman [CG98] (en anglais, *Kneser-Ney modified interpolated*), est une modification du lissage de Kneser-Ney [KN95]. Nous expliquons tout d'abord cette technique se basant sur un décompte absolu et une redistribution par interpolation, avant d'exposer les modifications apportées par Chen et Goodman.

Prenons l'exemple d'un modèle bigramme dans lequel le mot *Elysées* est très fréquent mais n'apparaît qu'après un seul des mots du corpus d'apprentissage, le mot *Champs*. La probabilité de l'unigramme *Elysées* est élevée et le décompte absolu attribuera alors une forte probabilité au mot *Elysées* apparaissant après un nouveau mot (différent de *Champs*). Il serait plus raisonnable de ne pas lui donner une probabilité aussi élevée puisqu'à chaque fois qu'il est présent, il suit *Champs*, auquel cas sa probabilité est bien modélisée par le modèle bigramme. Ainsi, pour Kneser et Ney, la probabilité unigramme ne devrait pas être proportionnelle au nombre d'occurrences d'un mot mais au nombre de mots différents qu'il suit ; le nombre de comptes attribué à chacun des unigrammes sera alors simplement le nombre de mots qu'il suit. La probabilité d'un mot est alors donnée par :

$$\begin{aligned} \widehat{P}_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) \ &= \ \gamma(w_{i-n+1}^{i-1}) \ \times \ P_{\mathrm{MLE}}(w_i|w_{i-n+1}^{i-1}) \\ &\quad + (1 \ - \ \gamma(w_{i-n+1}^{i-1})) \ \times \ \frac{N_{1+}(\bullet \ w_{i-n+2}^{i})}{\sum_{w_i \in V} \ N_{1+}(\bullet \ w_{i-n+2}^{i-1} w_i)} \end{aligned} \tag{2.29}$$

avec

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D}{\sum_{w_i \in V} \ N(w_{i-n+1}^{i-1} w_i)} \ N_{1+}(w_{i-n+1}^{i-1}\bullet) \tag{2.30}$$

le poids de repli, $P_{\mathrm{MLE}}(w_{i-n+1}^{i-1})$ la probabilité initiale de w_{i-n+1}^{i} et $N_{1+}(\bullet \ w_{i-n+2}^{i-1} w_i)$ le nombre de mots w_{i-n+1} tel que w_{i-n+1}^{i} apparaît une fois ou plus.

La modification apportée consiste à utiliser trois paramètres de décompte, D_1, D_2 et D_{3+}, appliqués aux n-grammes apparaissant respectivement une, deux et trois fois ou plus. Ainsi, le paramètre γ de l'équation 2.30 se réécrit en :

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1 \ \times \ N_1(w_{i-n+1}^{i-1}\bullet) \ + \ D_2 \ \times \ N_2(w_{i-n+1}^{i-1}\bullet) \ + \ D_{3+} \ \times \ N_{3+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i \in V} \ N(w_{i-n+1}^{i-1} w_i)} \tag{2.31}$$

avec $N_2(w_{i-n+1}^{i-1}\bullet)$ et $N_{3+}(w_{i-n+1}^{i-1}\bullet)$ définis de la même façon que $N_1(w_{i-n+1}^{i-1}\bullet)$, pour des fréquences d'apparition égales soit à deux fois, soit à trois fois ou plus, respectivement.

Les valeurs optimales des paramètres D_1, D_2 et D_{3+} sont définies par les relations :

$$\begin{cases} D_1 = 1 \ - \ 2Y \ \times \ \frac{n_2}{n_1} \\ D_2 = 2 \ - \ 3Y \ \times \ \frac{n_3}{n_2} \\ D_{3+} = 3 \ - \ 4Y \ \times \ \frac{n_4}{n_3} \end{cases} \tag{2.32}$$

avec $Y = \frac{n_1}{n_1 + 2n_2}$ et n_1, n_2, n_3 et n_4 le nombre total de n-grammes apparaissant, respectivement, exactement une fois, deux fois, trois fois et quatre fois, dans l'ensemble d'apprentissage.

2.4.5 Modèles n-classes

Le principe des *modèles n-classes* est de considérer des séquences de n classes et non plus de n mots, comme c'est le cas pour les modèles n-grammes. Les mots sont ainsi regroupés en classes, de telle façon que des mots appartenant à une même classe ont un comportement similaire. Ces modèles comportent alors moins de probabilités à estimer que les modèles n-grammes. De plus, le regroupement des mots en classes permet aussi de pouvoir estimer la probabilité d'une suite de mots qui n'est pas présente dans le corpus d'apprentissage si la séquence de classes correspondantes est, elle, présente dans le corpus.

Il existe différentes variantes des modèles n-classes, dépendant de la façon de calculer la probabilité $P(w_i|\Phi(h_i))$ ainsi que du choix de la classification des mots. Nous détaillons différentes approches dans les sous-sections suivantes.

2.4.5.1 Calcul des probabilités à partir des classes

La probabilité $P(w_i|\Phi(h_i))$ est calculée à partir de la classe du mot w_i et de celles des mots de son historique h_i. Il existe alors plusieurs façons de prendre en compte les classes des mots [Goo01], la plus utilisée étant l'approche *ibm* [BPdSL92]. Dans ce cas, un mot ne peut appartenir qu'à une seule classe : cette classification est alors dite *déterministe*. La probabilité $P(w_i|\Phi(h_i))$ est alors donnée par l'équation suivante :

$$P_{\text{ClassDeterm}}(w_i|w_{i-n+1}^{i-1}) = P(w_i|c_i) \times P(c_i|c_{i-n+1}^{i-1}) \tag{2.33}$$

avec c_i la classe à laquelle appartient le mot w_i et c_{i-n+1}^{i-1} la séquence de classes correspondant à l'historique w_{i-n+1}^{i-1}. Les probabilités $P(w_i|c_i)$ et $P(c_i|c_{i-n+1}^{i-1})$ peuvent être estimées au maximum de vraisemblance par :

$$P_{\text{MLE}}(w_i|c_i) = \frac{N(w_i)}{N(c_i)} \tag{2.34}$$

et

$$P_{\text{MLE}}(c_i|c_{i-n+1}^{i-1}) = \frac{N(c_{i-n+1}^{i-1}c_i)}{\sum_{c_j \in C} N(c_{i-n+1}^{i-1}c_j)} \tag{2.35}$$

avec C l'ensemble des classes.

Comme dans le cas des modèles n-grammes, un modèle de lissage peut être utilisé pour calculer ces probabilités (voir section 2.4.4).

Dans le cas de la *classification stochastique* [Nie97], un mot peut appartenir à plusieurs classes. Il faut alors tenir compte de toutes les séquences de classes pouvant correspondre à la séquence de mots w_{i-n+1}^{i}. L'équation 2.33 devient alors :

$$P_{\text{ClassStoch}}(w_i|w_{i-n+1}^{i-1}) = \sum_{\substack{c_i \in C(w_i) \\ c_{i-n+1}^{i-1} \in C(w_{i-n+1}) \times \cdots \times C(w_{i-1})}} P(w_i|c_i) \times P(c_i|c_{i-n+1}^{i-1}) \tag{2.36}$$

avec $C(w_i)$ l'ensemble des classes auxquelles appartient le mot w_i et $C(w_{i-n+1}) \times \ldots \times C(w_{i-1})$ l'ensemble de toutes les séquences de classes correspondant à la séquence de mots w_{i-n+1}^{i-1}.

Les *modèles multiclasses* [YIS03] se situent entre les deux modèles précédents. En effet, chaque mot possède deux classes : l'une appelée *classe prédictive*, quand il est en position w_i dans le calcul de la probabilité $P(w_i|\Phi(h_i))$, et l'autre appelée *classe conditionnelle*, lorsqu'il fait partie de l'historique h_i. L'équation 2.33 devient alors :

$$P_{\text{Multiclasse}}(w_i|w_{i-n+1}^{i-1}) = P(w_i|c_p(w_i)) \times P(c_i|c_c(w_{i-n+1}^{i-1})) \tag{2.37}$$

avec $c_p(w_i)$ la classe prédictive du mot w_i et $c_c(w_{i-n+1}^{i-1})$ la séquence de classes conditionnelles correspondant à la séquence de mots w_{i-n+1}^{i-1}.

Cette classification permet de faire la distinction entre les voisinages gauche et droit d'un mot, comme ces voisinages peuvent être différents. Par exemple, les articles français *le* et *l'* ont des voisinages gauches similaires et appartiennent donc à la même classe prédictive. En revanche, leur voisinage droit est différent puisque *le* précède un mot commençant par une consonne alors que *l'* précède un mot commençant par une voyelle : leurs classes conditionnelles seront donc différentes.

2.4.5.2 Choix des classes

Classes prédéfinies Les classes peuvent tout d'abord être construites manuellement. Ainsi, les couleurs ou encore les jours de la semaine peuvent constituer des classes. Afin de pouvoir apprendre les probabilités du modèle de langage, le corpus doit être étiqueté, c'est-à-dire que chacun des mots du corpus doit être associé avec sa classe. Cette approche n'est cependant possible que si le nombre de classes est réduit. Ainsi, dans [Ros94], des classes ont été construites pour les compagnies d'aviation ou encore les villes.

Le modèle de langage le plus utilisé parmi ceux à base de classes prédéfinies est le *modèle POS (Part Of Speech)* [MS99, Nie97]. Dans ce modèle de langage, les classes utilisées correspondent aux *catégories morpho-syntaxiques* des mots, c'est-à-dire leurs catégories grammaticales telles que nom, verbe.... Dans ce cas, un mot peut appartenir à plusieurs classes, ce qui correspond à une classification stochastique. Par exemple, le mot *souris* peut être soit un nom, soit un verbe.

Bien qu'étant basé sur une approche statistique, ce type de modèle exprime des connaissances syntaxiques. Néanmoins, les modèles à base de classes construites automatiquement se révèlent généralement plus performants, ce que nous présentons dans le paragraphe suivant.

Construction automatique des classes Lors de la construction automatique des classes, le regroupement se fait à l'aide d'un *critère statistique* à maximiser. Ces méthodes procèdent de manière itérative : à chaque étape, la meilleure classification est recherchée, en partant de la classification obtenue à l'itération précédente et en la modifiant. La classification finale obtenue est déterministe puisque chacun des mots n'appartient qu'à une seule classe. Il reste cependant à choisir le nombre optimal de classes qui permet d'obtenir le meilleur compromis entre les performances du modèle de langage et la taille de celui-ci. Il existe deux principaux algorithmes pour construire cette classification, que nous détaillons par la suite.

Dans l'algorithme d'échange [NEK94, MLN98], donné par l'algorithme 2.1, les mots du vocabulaire sont tout d'abord classés selon leur fréquence d'apparition décroissante, dans le corpus d'apprentissage. Ensuite, une classe est créée pour chacun des $C-1$ premiers mots et les mots restants sont regroupés dans une même classe. À chaque itération, l'algorithme tente de déplacer chacun des mots de sa classe vers une des autres et le déplacement choisi pour chaque mot est celui qui permet d'obtenir la classification qui minimise la perplexité sur le corpus d'apprentissage. La complexité de cet algorithme, en terme de nombre de calculs de perplexité, est alors $O(VCK)$, avec V le nombre de mots du vocabulaire, C le nombre de classes et K le nombre maximal d'itérations.

Dans l'algorithme de Brown [BPdSL92], présenté par l'algorithme 2.2, les mots sont aussi ordonnés selon leur fréquence d'apparition décroissante et une classe est créée, cette fois, pour chacun des C premiers mots. Chacune des itérations correspond à la création d'une classe pour un des mots restants, puis à la recherche de la fusion de deux classes qui permette de minimiser également la perplexité sur le corpus d'apprentissage. La complexité de cet algorithme, toujours en terme de nombre de calculs de perplexité, est alors $O(VC^2)$.

Dans l'algorithme de Brown, la classification finale de tous les mots du vocabulaire ne sera réalisée qu'à la fin de l'algorithme alors que dans l'algorithme d'échange chacune des classifications intermédiaires correspond à une classification de tous les mots du vocabulaire.

2.4.5.3 Bilan

Un des avantages des modèles n-classes est leur compacité. En effet, comme il y a moins d'historiques différents à considérer, ces modèles de langage comportent moins de paramètres. Cela permet aussi de considérer des historiques plus longs, pour un nombre de paramètres comparable à celui d'un modèle n-gramme traitant d'historiques plus courts. Les modèles n-classes permettent aussi de parer au problème du manque de données. En effet, ils permettent d'estimer la probabilité d'une séquence de mots non présente dans le corpus d'apprentissage si la séquence de classes correspondante était présente dans ce corpus d'apprentissage. L'utilisation de ces modèles de langage est ainsi bien indiquée lorsque les données d'apprentissage sont en quantité limitée.

Toutefois, ces modèles de langage sont moins précis que les modèles n-grammes puisque les mots sont regroupés en classes et que des informations sont ainsi perdues.

Algorithme 2.1 : Algorithme d'échange, pour la création de classes [NEK94].

Données :
V mots correspondant au vocabulaire;

Résultat :
C classes qui représentent le partitionnement des mots;

début
- ordonner les mots selon leur fréquence décroissante;
- créer une classe pour chacun des $(C-1)$ premiers mots;
- créer une classe regroupant les $(V-C+1)$ mots restants;
- **tant que** *critère de fin non satisfait* **faire**
  - **pour** *i* = 1 **à** V **faire**
    - **pour** *j* = 1 **à** C **faire**
      - calculer la perplexité obtenue sur le corpus d'apprentissage lorsque le mot w_i est déplacé de sa classe c_i vers la classe c_j;
    - changer le mot w_i de sa classe c_i vers la classe c_j qui permette d'obtenir la perplexité minimale sur le corpus d'apprentissage;

fin

Algorithme 2.2 : Algorithme de Brown, pour la création de classes [BPdSL92].

Données :
V mots correspondant au vocabulaire;

Résultat :
C classes qui représentent le partitionnement des mots;

début
- ordonner les mots selon leurs fréquences décroissantes;
- créer une classe pour chacun des C premiers mots;
- **pour** *i* = $C+1$ **à** V **faire**
  - créer une classe pour le i^{e} mot (classe C_{C+1});
  - **pour** *j* = 1 **à** $C+1$ **faire**
    - **pour** *k* = 1 **à** $C+1$ **faire**
      - calculer la perplexité obtenue sur le corpus d'apprentissage lorsque les classes c_j et c_k sont fusionnées;
  - regrouper les deux classes c_j et c_k qui permettent d'obtenir la perplexité minimale sur le corpus d'apprentissage;

fin

2.4.6 Combinaison de modèles de langage

L'intérêt de combiner des modèles de langage est de tirer partie des spécificités de chacun d'eux et ainsi de surpasser le meilleur d'entre eux [Goo01]. En effet, les informations représentées par différents modèles de langage peuvent être de nature différente et complémentaire et la combinaison de ces modèles permet alors de disposer de plus d'informations. Par exemple, les modèles à base de classes morpho-syntaxiques regroupent les mots selon leur nature grammaticale alors que les modèles à base de classes statistiques les regroupent selon des critères de similarité statistique : ces deux types d'informations peuvent se révéler complémentaires.

La méthode la plus simple pour combiner des modèles de langage est l'*interpolation linéaire*. Le modèle de langage correspondant à l'interpolation linéaire de K modèles de langage est ainsi obtenu en pondérant chacun de ces modèles, de la façon suivante :

$$\widetilde{P}_{\mathrm{InterpolLin}}(w_i|\Phi(h_i)) = \sum_{k=1}^{K} \lambda_k \, P_k(w_i|\Phi(h_i)) \tag{2.38}$$

avec λ_k le poids du modèle de langage ML_k, tel que $0 \leq \lambda_k \leq 1$. Les valeurs de ces poids sont réglées de façon à optimiser la qualité du modèle de langage $ML_{\mathrm{InterpolLin}}$ (voir section 2.4.2 pour une présentation des différentes mesures permettant d'évaluer la qualité d'un modèle de langage) et tel que :

$$\sum_{k=1}^{K} \lambda_k = 1. \tag{2.39}$$

L'*interpolation log-linéaire* [Kla98] permet d'obtenir de meilleurs résultats que la simple interpolation linéaire. En effet, cette méthode permet une meilleure synthèse des modèles de langage, surtout lorsqu'ils ont des ordres de grandeur différents [BM05], c'est-à-dire que les probabilités estimées dans chacun de ces modèles peuvent varier dans des plages de valeurs différentes. La combinaison des modèles de langage, donnée par l'équation 2.38, devient alors :

$$\widetilde{P}_{\mathrm{InterpolLogLin}}(w_i|\Phi(h_i)) = \frac{1}{Z(\Phi(h_i))} \prod_{k=1}^{K} P_k(w_i|\Phi(h_i))^{\lambda_k} \tag{2.40}$$

avec $Z(\Phi(h_i))$ une constante de normalisation utilisée pour assurer :

$$\sum_{w_i \in V} \widetilde{P}_{\mathrm{InterpolLogLin}}(w_i|\Phi(h_i)) = 1. \tag{2.41}$$

En utilisant les log-probabilités des modèles de langage et en omettant la constante de normalisation (ce qui est souvent le cas en reconnaissance de la parole, à-cause de son faible effet [BM05]), l'équation 2.40 peut se réécrire en une somme pondérée des log-probabilités des différents modèles de langage :

$$\log\left(\widetilde{P}_{\mathrm{InterpolLogLin}}(w_i|\Phi(h_i))\right) \approx \sum_{k=1}^{K} \lambda_k \, \log\left(P_k(w_i|\Phi(h_i))\right). \tag{2.42}$$

2.4.7 Autres modèles de langage

Les modèles de langage n-grammes ou à base de classes, qu'elles soient de nature statistique ou morpho-syntaxique essentiellement, sont parmi les plus couramment utilisés. Il existe néanmoins d'autres modèles de langage qui ont suscité de l'intérêt dans des travaux plus récents. Nous présentons, dans cette section, d'autres techniques de modélisation statistique du langage et plus particulièrement les avantages et inconvénients de chacune d'elles.

2.4.7.1 Modèles à historique variable

Comme présenté dans la section 2.4.3, l'ordre n des modèles n-grammes est généralement inférieur à 4 ou 5, ce qui ne permet pas de prendre en compte des dépendances longues dans les phrases. Lorsque l'ordre d'un modèle n-gramme augmente, la probabilité de rencontrer exactement chacun des historiques existants diminue. En revanche, la possibilité d'observer des contextes similaires, c'est-à-dire ayant en commun la plupart des mots, augmente.

Les *modèles n-grammes distants* [HAH+93] (*skipping*, en anglais) s'appuient sur cette observation, en ignorant certains mots de l'historique, dans un modèle n-gramme. Ces modèles sont généralement utilisés en combinaison d'un modèle n-gramme standard, afin de tenir compte à la fois d'un contexte proche et d'un contexte plus lointain. Un modèle n-gramme peut ainsi être combiné avec des modèles n-grammes distants du même ordre ou d'ordre supérieur. Ainsi, l'interpolation linéaire d'un modèle 5-grammes avec des modèles 5-grammes distants peut donner le modèle suivant :

$$\widetilde{P}(w_i|\Phi(h_i)) = \lambda_1\, P(w_i|w_{i-4}^{i-1}) + \lambda_2\, P_{\text{Dist}}(w_i|w_{i-4}w_{i-3}w_{i-1}) + \lambda_3\, P_{\text{Dist}}(w_i|w_{i-4}w_{i-2}w_{i-1}) \tag{2.43}$$

alors que l'interpolation linéaire d'un modèle trigramme avec des modèles 5-grammes distants peut être de la forme suivante :

$$\widetilde{P}(w_i|\Phi(h_i)) = \lambda_1\, P(w_i|w_{i-2}^{i-1}) + \lambda_2\, P_{\text{Dist}}(w_i|w_{i-4}w_{i-3}w_{i-1}) + \lambda_3\, P_{\text{Dist}}(w_i|w_{i-4}w_{i-2}w_{i-1}) \tag{2.44}$$

avec λ_i le poids de chaque modèle de langage. Un des avantages de ces modèles de langage est donc la prise en compte d'historiques similaires mais non strictement identiques au contexte courant. Néanmoins, l'ajout de modèles n-grammes distants s'accompagne d'une augmentation du nombre de paramètres du modèle de langage global, ce qui peut être un inconvénient.

Une autre extension des modèles n-grammes repose sur la variation de la longueur de l'historique, de façon à ne garder un historique long seulement si l'information qu'il apporte est pertinente : ces modèles sont appelés *modèles varigrammes* [SO00]. Leur principe est ainsi d'assurer un compromis entre la pertinence du modèle de langage (en gardant les historiques les plus informatifs) et son nombre de paramètres. Les n-grammes de ce modèle sont ainsi représentés sous la forme d'un arbre dont la profondeur varie selon ses branches (voir figure 2.1).

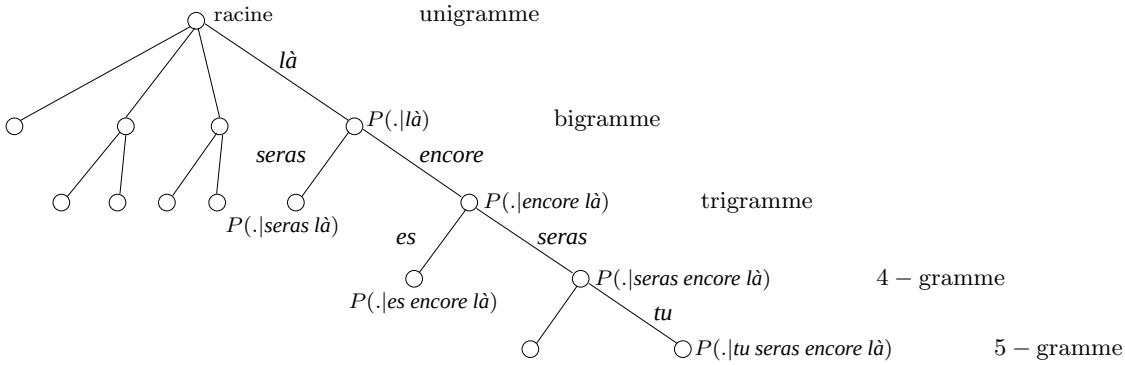


FIG. 2.1 – Exemple de représentation d'un modèle varigramme, sous forme d'arbre.

Comme chaque nœud représente un contexte, des nœuds proches et associés à des probabilités conditionnelles similaires sont ainsi fusionnés. Dans le cas de modèles n-grammes classiques, l'arbre aurait une profondeur fixe égale à $n-1$. L'avantage des modèles varigrammes est ainsi la diminution du nombre de paramètres, tout en gardant des performances similaires.

Il existe d'autres modèles de langage dans lesquels l'historique est modifié afin que sa longueur puisse être augmentée. Nous pouvons citer les *modèles permugrammes* [STHKN95] dans lesquels les historiques considérés correspondent à une permutation des mots du contexte courant ou encore les *modèles multigrammes* [DB95] dans lesquels l'historique est considéré comme une suite de groupes de mots. La plupart de ces approches n'ont cependant permis d'obtenir que peu d'améliorations par rapport aux modèles n-grammes. De plus, ils augmentent la taille du modèle de langage résultant puisqu'ils sont souvent combinés avec un modèle n-gramme.

2.4.7.2 Modèles à base de cache

Un scripteur qui écrit un mot est susceptible de l'écrire de nouveau, dans un futur proche. Cette observation est la base des *modèles à base de cache* [KM90]. Ces modèles prennent en compte un historique allant de plusieurs dizaines à plusieurs centaines de mots. Les probabilités $P(w_i|\Phi(h_i))$ sont alors évaluées à partir des fréquences d'apparition des mots, non plus dans le corpus d'apprentissage mais dans le cache. Toutefois, les modèles à base de cache sont généralement combinés avec des modèles n-grammes, dont les probabilités sont calculées à partir d'un corpus d'apprentissage :

$$\widetilde{P}(w_i|\Phi(h_i)) = \lambda\, P(w_i|w_{i-n+1}^{i-1}) \;+\; (1-\lambda)\, P_{\text{Cache}}(w_i|w_{i-K+1}^{i-1}) \tag{2.45}$$

avec K la longueur du cache, telle que $K \gg n$. Il existe plusieurs stratégies pour traiter les mots du cache afin de calculer $P_{\text{Cache}}(w_i|w_{i-K+1}^{i-1})$.

Dans un modèle à base de *cache régulier* [KM90], la probabilité d'un mot est proportionnelle à son nombre d'occurrences dans le cache de longueur fixe K et est définie par l'équation :

$$P_{\text{CacheReg}}(w_i|w_{i-K+1}^{i-1}) = \frac{1}{K} \sum_{j=i-K+1}^{i-1} \delta(w_j, w_i) \tag{2.46}$$

avec $\delta(w_j, w_i) = 1$ si $w_j = w_i$, 0 sinon.

Dans un modèle à base de *cache à oubli exponentiel* [Cla99] (en anglais, *history decay*), la position du mot dans le cache est prise en compte : plus un mot est loin dans le cache, moins il est susceptible d'être écrit de nouveau. Ainsi, la contribution d'un mot w_i décroît de manière exponentielle avec l'augmentation de la distance entre ce mot w_j et le mot courant w_i, l'équation 2.46 devenant :

$$P_{\text{CacheExp}}(w_i|w_{i-K+1}^{i-1}) = \beta \frac{1}{K} \sum_{j=i-K+1}^{i-1} \delta(w_j, w_i) \times \exp(-\alpha(i-j)) \tag{2.47}$$

avec α le poids d'oubli et β une constante de normalisation définie de façon à obtenir :

$$\sum_{w_i \in V} P_{\text{CacheExp}}(w_i|w_{i-K+1}^{i-1}) = 1. \tag{2.48}$$

Bien que les modèles à base de cache permettent de diminuer considérablement la perplexité, le taux d'erreur des systèmes de reconnaissance de parole utilisés dans [KM90, Cla99] ne diminue que très peu. Ce problème vient essentiellement de la présence d'erreurs de reconnaissance parmi les mots du cache. En effet, le système de reconnaissance à tendance à répéter les erreurs présentes dans le cache. Par exemple, si le scripteur a écrit *« efface tous les souvenirs »* et que le système a reconnu *« efface tous les sourires »* alors, si le scripteur écrit *« mes souvenirs sont loin »*, le système reconnaîtra plutôt *« mes sourires sont loin »*, comme la probabilité de *sourires* est plus élevée que celle de *souvenirs*. L'utilisation d'un modèle à base de cache, dans un système de reconnaissance, doit ainsi s'accompagner d'un mécanisme de correction des erreurs par l'utilisateur, afin que les performances du système puissent être améliorées [Goo01].

2.4.7.3 Modèles à entropie maximale

Les *modèles à entropie maximale* [DD72] ont récemment été utilisés pour modéliser le langage [Ros94, BPP96]. Ces modèles permettent d'incorporer différentes sources d'informations (n-grammes, n-classes, n-grammes distants...), en exprimant celles-ci sous forme de contraintes. Ces contraintes sont représentées par des fonctions caractéristiques qui sont appliquées à un mot w_i et à son historique w_1^{i-1}. Une fonction de caractéristique f_c retourne 1 si la contrainte c qu'elle exprime est vérifiée par le mot w_i et son historique w_1^{i-1}, 0 sinon. Par exemple, si la fonction f_c correspond au trigramme *« le miroir casse »*, cela s'exprime par :

$$f_{\textit{le miroir casse}}(w_1^i) = \begin{cases} 1 & \text{si } w_{i-2} = \textit{le} \text{ et } w_{i-1} = \textit{miroir} \text{ et } w_i = \textit{casse} \\ 0 & \text{sinon.} \end{cases} \tag{2.49}$$

La probabilité d'un mot w_i est alors définie à partir des fonctions caractéristiques f_c par l'équation suivante :

$$P_{\text{MaxEnt}}(w_i|w_1^{i-1}) = \frac{\exp(\sum_c \lambda_c f_c(w_1^i))}{z(w_1^i)} \tag{2.50}$$

avec λ_c des paramètres réels obtenus par un algorithme d'apprentissage comme l'algorithme *GIS (Generalized Iterative Scaling)* [DD72, BPP96] ; ces paramètres fixent le poids de chaque fonction caractéristique. $z(w_1^i)$ est une constante de normalisation, calculée afin d'obtenir :

$$\sum_{w_i} P_{\text{MaxEnt}}(w_i|w_1^{i-1}) = 1. \tag{2.51}$$

Ces modèles permettent ainsi de combiner différents types d'informations de manière élégante. En effet, le modèle combiné est appris directement, au lieu d'être obtenu par combinaison de modèles appris séparément. Néanmoins, l'inconvénient majeur de ce type de modèle est le temps de calcul élevé, aussi bien lors de son apprentissage que lors de son utilisation, le nombre de fonctions caractéristiques pouvant atteindre plusieurs millions. De plus, ces modèles n'ont pas encore permis d'obtenir de réduction significative de la perplexité [Goo01].

2.4.7.4 Modèles à base de réseaux de neurones

Il peut être intéressant de pouvoir exprimer une similarité entre mots. En effet, si la phrase *« les fantômes dansent entre les arbres »* est présente dans le corpus d'apprentissage, il devrait être possible de généraliser la phrase *« des lutins courent dans la forêt »* en lui donnant une probabilité voisine, en s'appuyant sur le fait que *fantômes* et *lutins* ont des rôles grammaticaux et sémantiques similaires (il en est de même pour *arbres* et *forêt* ainsi que pour *dansent* et *courent* ou encore *les* et *des*). L'approche basée sur des *réseaux de neurones* [BDVJ03] permet d'exprimer une similarité entre mots. Dans cette approche, chacun des mots du vocabulaire est représenté par un vecteur de m caractéristiques, avec $m \ll |V|$ ($m = 30$, dans [BDVJ03]). Deux mots ayant des rôles syntaxiques et sémantiques similaires auront ainsi des vecteurs de caractéristiques proches.

La probabilité $P(w_i|w_{i-n+1}^{i-1})$ est représentée par une fonction $f(w_i, \ldots, w_{i-n+1})$. Cette fonction se décompose en deux parties et est illustrée par la figure 2.2.

Une matrice C de dimension $|V| \times m$ permet tout d'abord d'obtenir le vecteur de caractéristiques de taille m, pour chacun des $|V|$ mots du vocabulaire. Ensuite, le réseau de neurones prend en entrée $m \times (n-1)$ valeurs, correspondant aux vecteurs de caractéristiques $(C(w_{i-n+1}), \ldots, C(w_{i-1}))$ des mots de l'historique w_{i-n+1}^{i-1}. La couche de sortie comporte $|V|$ neurones, soit un pour chaque mot du vocabulaire. Ainsi, pour un historique w_{i-n+1}^{i-1} donné, la j^{e} sortie correspond à la probabilité $P(w_{i_j}|w_{i-n+1}^{i-1})$, w_{i_j} étant le j^{e} mot du vocabulaire. Le réseau de neurones comporte également une ou plusieurs couches cachées ainsi que d'éventuelles connexions directes des neurones de la couche d'entrée vers ceux de la couche de sortie.

La projection des mots dans le nouvel espace de représentation, c'est-à-dire leurs vecteurs de caractéristiques (représentés par la matrice C), ainsi que les poids du réseau de neurones sont appris de manière simultanée.

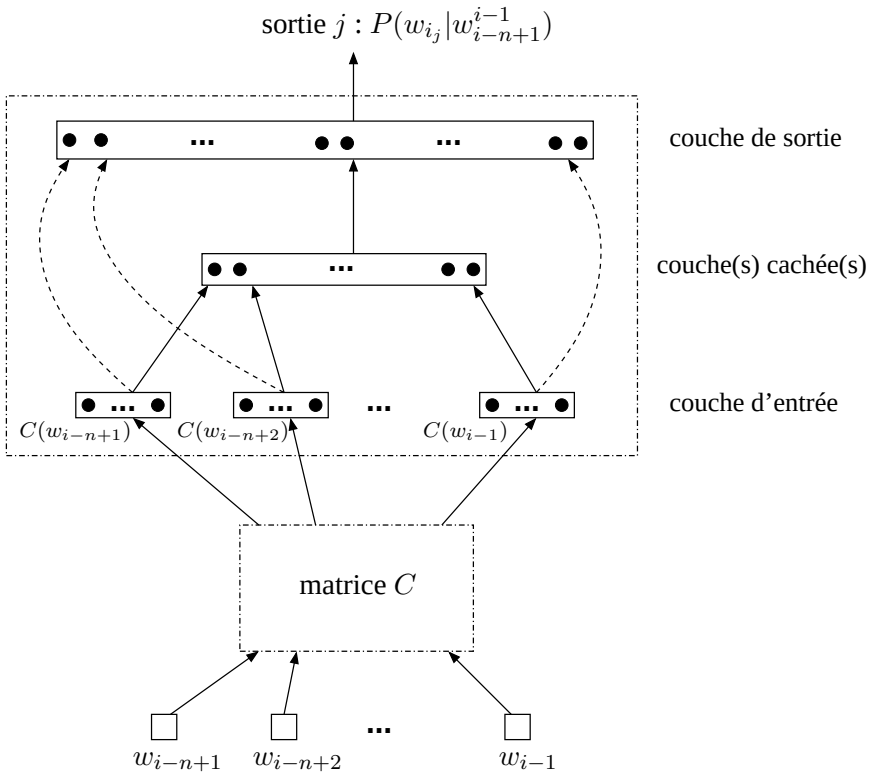


FIG. 2.2 – Architecture du modèle de langage à base de réseau de neurones.

Parmi les premiers résultats obtenus avec ces modèles de langage à base de réseaux de neurones, [SG04] ont obtenu une baisse du taux d'erreur de 22,6 % à 21,8 %, le premier taux étant obtenu avec un modèle 4-gramme. Ces modèles de langage permettent aussi de prendre en compte un historique plus long, le nombre de neurones en entrée du réseau étant proportionnel à la longueur de l'historique. Malgré les améliorations apportées par [MB05] pour diminuer le temps de calcul lors de l'apprentissage du modèle de langage ainsi que lors de son utilisation, ces temps de calcul restent trop élevés. Ces modèles de langage constituent néanmoins une approche prometteuse.

2.4.7.5 Modèles à base de mélange de phrases

Il existe généralement différents types de phrases, dans un corpus. Il peut alors être intéressant de regrouper les phrases en fonction de leur thème, de leur style... et de modéliser séparément chacun d'entre eux, sur une sous-partie du corpus d'apprentissage [IOR94, IO99].

La probabilité d'une phrase est alors calculée en utilisant chacun des sous-modèles de langage correspondant aux différents types, de la façon suivante :

$$P_{\mathrm{MelPhr}}(w_i|\Phi(h_i)) = \sum_{t=1}^{T} \sigma_t \, P_t(w_i|\Phi(h_i)) \tag{2.52}$$

avec $P_t(w_i|\Phi(h_i))$ la probabilité donnée par le modèle de langage correspondant au type de phrase s_t et σ_t les paramètres d'interpolation des phrases, correspondant aux probabilités *a priori* de chacun des types de phrases, et calculés afin d'obtenir :

$$\sum_{t=1}^{T} \sigma_t = 1. \tag{2.53}$$

Ce modèle de langage peut aussi être interpolé avec un modèle de langage appris sur le corpus d'apprentissage global, l'équation 2.52 devenant ainsi :

$$\widetilde{P}_{\mathrm{MelPhr}}(w_i|\Phi(h_i)) = \sum_{t=1}^{T} \sigma_t \ [\lambda_t \, P_t(w_i|\Phi(h_i)) + (1-\lambda_t) \, P(w_i|\Phi(h_i))] \tag{2.54}$$

avec λ_t les poids d'interpolation.

L'identification des types de phrases présents dans le corpus d'apprentissage se fait généralement de manière automatique [Goo01, IO99], en minimisant la perplexité des phrases. Lors de cette classification, les probabilités *a priori* des types s_t sont calculées simultanément.

La séparation des phrases selon leur type permet une estimation plus fine des probabilités et permet d'obtenir de meilleurs résultats qu'avec un modèle n-gramme global [Cla99]. Toutefois, le modèle de langage obtenu est de taille plus importante, proportionnelle au nombre de types de phrases qui peut aller de plusieurs dizaines à plusieurs centaines. De plus, cette approche est surtout bénéfique lorsque la taille des corpus d'apprentissage est importante [Goo01].

Les deux derniers types de modèles de langage que nous présentons maintenant sont basés sur une approche statistique mais s'appuient plus particulièrement sur des connaissances structurelles. Ils offrent ainsi un compromis entre les deux approches structurelle et statistique.

2.4.7.6 Grammaires probabilistes hors-contexte

Comme présenté dans la section 2.3, une grammaire peut représenter un langage par des règles, l'enchaînement de celles-ci permettant de décrire la structure des phrases. Les grammaires hors-contexte permettent de décrire des langages réguliers. Ces langages sont relativement simples et non-ambigüs. Les langages naturels, quant à eux, sont d'une nature plus complexe et peuvent présenter de nombreuses ambiguïtés. Ainsi, une phrase peut être décrite par plusieurs suites de règles. L'approche probabiliste des grammaires hors-contexte [MS99, JWS$^+$95] permet de prendre en compte les différentes dérivations possibles d'une phrase.

Une *grammaire probabiliste hors-contexte GPHC* (en anglais, *Probabilist Context-Free Grammar*) est définie par :

- un ensemble de terminaux, correspondant aux mots du langage : $\{w_k\}$ avec $w_k \in V$;
- un ensemble de non-terminaux, correspondant aux objets grammaticaux : $\{N_i\}$;
- un symbole de départ : N_1 ;

– un ensemble de règles de dérivation : $\{N_i \rightarrow \xi_j\}$ avec ξ_j une séquence de terminaux et de non-terminaux ;
– un ensemble de probabilités (évaluées automatiquement sur des corpus étiquetés) associées aux règles et tel que :

$$\forall i, \quad \sum_j P(N_i \rightarrow \xi_j) = 1. \tag{2.55}$$

Une probabilité est associée à chaque dérivation possible d'une phrase et correspond au produit des probabilités de toutes les règles qui composent la dérivation d. La probabilité d'une phrase W est alors calculée à partir des probabilités de l'ensemble des dérivations D_W qui lui sont associées, de la façon suivante :

$$P_{\mathrm{GPHC}}(W) = \sum_{d \in D_W} P(d). \tag{2.56}$$

La table 2.1 présente un exemple de grammaire probabiliste hors-contexte dans laquelle S, GV, GP, GN, V et P sont des non-terminaux et *regarde*, *les gens*, *avec*, *dans* et *des fleurs* sont des terminaux. Dans cette table, les règles de la grammaire sont données avec leur probabilité.

S	→	GV GP	0,7	V	→	*regarde*	1
S	→	V GN	0,3	GN	→	*les gens*	0,3
GV	→	V GN	1,0	P	→	*avec*	0,6
GP	→	P GN	1,0	GN	→	*des fleurs*	0,2
GN	→	GN GP	0,5	P	→	*dans*	0,4

TAB. 2.1 – Exemple de règles d'une grammaire probabiliste hors-contexte.

En utilisant cette grammaire, la phrase *« regarde les gens avec des fleurs »* peut être associée à deux dérivations, qui sont représentées par les arbres syntaxiques d_1 et d_2 (voir figure 2.3). La probabilité associée à l'arbre d_1 est alors :

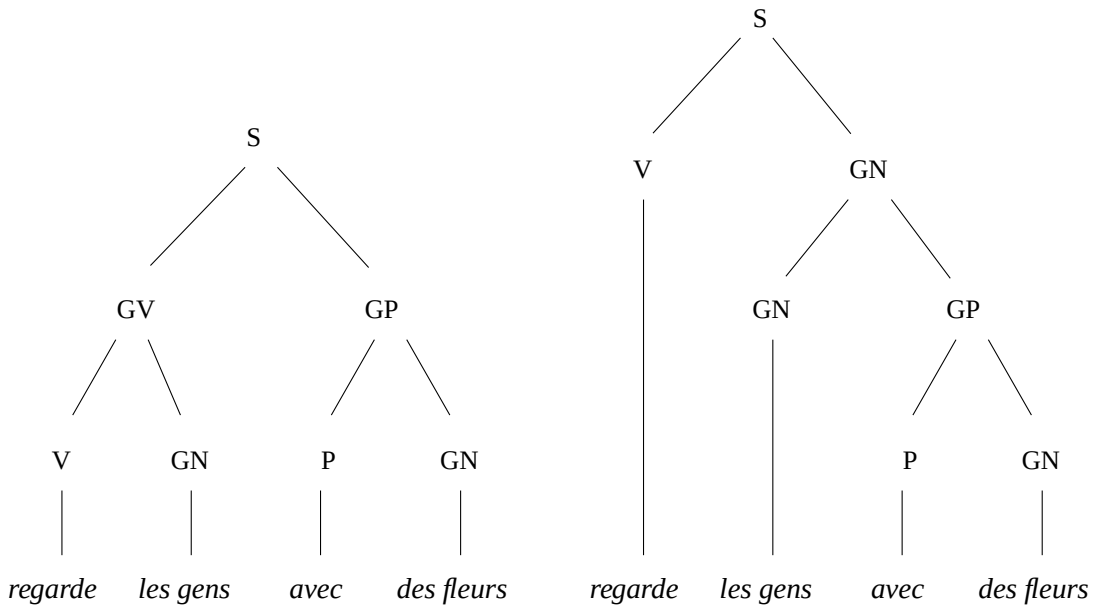
$$\begin{aligned} P(d_1) &= P(\mathrm{S} \rightarrow \mathrm{GV\ GP}) \\ &\quad \times P(\mathrm{GV} \rightarrow \mathrm{V\ GN}) \times P(\mathrm{V} \rightarrow \textit{regarde}) \times P(\mathrm{GN} \rightarrow \textit{les gens}) \\ &\quad \times P(\mathrm{GP} \rightarrow \mathrm{P\ GN}) \times P(\mathrm{P} \rightarrow \textit{avec}) \times P(\mathrm{GN} \rightarrow \textit{des fleurs}) \\ &= 0,0252. \end{aligned} \tag{2.57}$$

La probabilité associée à l'arbre de dérivation d_2 est calculée de la même façon :

$$P(d_2) = 0,0054. \tag{2.58}$$

La probabilité finale de la phrase est alors :

$$P(\textit{regarde les gens avec des fleurs}) = P(d_1) + P(d_2) = 0,0306. \tag{2.59}$$



(a) Arbre de dérivation d_1. (b) Arbre de dérivation d_2.

FIG. 2.3 – Arbres de dérivation correspondant aux successions possibles de règles appliquées, pour la phrase *« regarde les gens avec des fleurs »*.

Ces modèles de langage à base de grammaires probabilistes hors-contexte permettent de prendre en compte des dépendances plus longues ainsi que des informations sur la structure syntaxique des phrases. Cependant, cette modélisation est relativement coûteuse en termes de calculs. De plus, il est difficile de déterminer l'ensemble des règles décrivant le langage et permettant d'obtenir une grammaire suffisamment robuste. C'est pourquoi, ils sont plutôt adaptés à des tâches dont la grammaire associée est relativement simple. Par exemple, en reconnaissance de parole, pour un système d'interrogation de noms de restaurants, [JWS+95] diminuent le taux d'erreur sur les mots de 34,6% (en utilisant un modèle bigramme) à 29,6 % en utilisant une grammaire probabiliste hors-contexte. En revanche, pour une tâche plus complexe de reconnaissance de phrases manuscrites, [ZCB06] n'obtiennent qu'une faible réduction du taux d'erreur sur les mots (de 20,7 % à 20,6 %), en combinant une grammaire probabiliste hors-contexte avec un modèle bigramme.

2.4.7.7 Modèles de langage structurés

Les *modèles de langage structurés* [CJ00, Cha01] sont basés sur des grammaires lexicalisées. Dans ce type de grammaire, un mot joue le rôle de *tête*, pour chaque constituant détecté.

L'exemple illustré par la figure 2.4 donne l'arbre de la dérivation partielle de la phrase *« des enfants construisent un bonhomme de neige dans le jardin pendant que la radio passe ta chanson préférée »*.

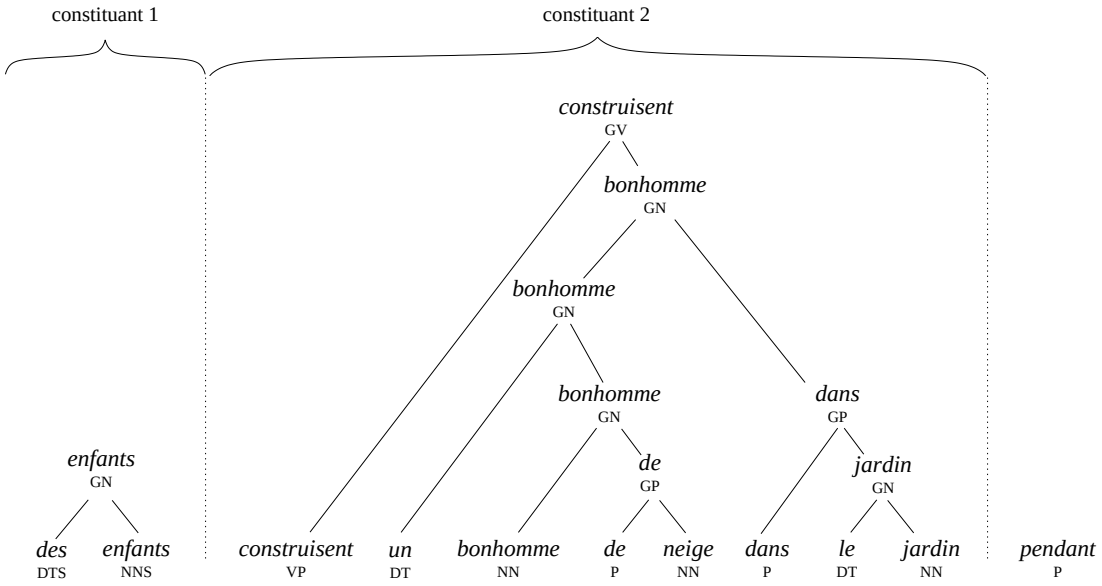


FIG. 2.4 – Arbre de dérivation partielle, pour la phrase *« des enfants construisent un bonhomme de neige dans le jardin pendant que la radio passe ta chanson préférée »*.

Dans cette dérivation, les mots associés aux feuilles sont accompagnés de leur étiquette *POS* et les nœuds sont annotés avec la tête et le type (groupe nominal, groupe verbal…) du constituant auquel ils sont associés. Par exemple, pour le premier constituant, *des* est un déterminant pluriel, *enfants* est un nom pluriel et la tête de ce constituant est *enfants*, ce constituant étant un groupe nominal (noté *GN* dans le type associé à la tête *enfants*).

Le mot *pendant* est alors prédit à partir des têtes des deux constituants qui le précèdent, c'est-à-dire *enfants* et *construisent*, et non à partir des mots *le* et *jardin*, comme ce serait le cas si un modèle trigramme était utilisé.

Le calcul des probabilités, dans ce type de modèle, est ainsi conditionné par les têtes des constituants. La probabilité $P(w_i|\Phi(h_i))$ est ainsi donnée par

$$P_{\mathrm{MLS}}(w_i|w_1^{i-1}) = \frac{\sum_{d\in w_1^i} P(w_i, h_{-1,d}, h_{0,d}) \times P(d)}{\sum_{d\in w_1^{i-1}} P(d)} \tag{2.60}$$

avec d la dérivation associée à w_1^i et $h_{-1,d}$ et $h_{0,d}$ les têtes des deux derniers constituants de la dérivation d.

Les modèles de langages structurés permettent de prendre en compte des dépendances à longue distance. Néanmoins, ces modèles de langage sont aujourd'hui encore trop coûteux en termes de temps de calcul pour qu'ils puissent être utilisés de manière efficace dans un système de reconnaissance.

2.5 Bilan

Nous avons présenté quelques unes des techniques principales de modélisation du langage parmi les nombreuses techniques existantes. Cependant, comparativement aux modèles n-grammes qui sont les modèles de langage standards de la littérature, la plupart des autres techniques ne permettent d'obtenir que peu d'amélioration des performances. De plus, cette relative amélioration s'accompagne généralement d'une augmentation considérable de la complexité du modèle de langage ou encore du temps de calcul nécessaire pour son utilisation.

Dans notre système de reconnaissance d'écriture manuscrite, pour l'informatique nomade, nous souhaitons utiliser des modèles de langage peu coûteux aussi bien en termes de temps de calcul qu'au niveau de leur complexité (qui peut s'exprimer par leur nombre de paramètres, notamment). C'est pourquoi, dans le cadre de cette thèse, nous avons choisi de nous focaliser sur les modèles n-grammes ainsi que sur les modèles n-classes, ceux-ci permettant d'obtenir un compromis intéressant entre les performances obtenues et la taille du modèle de langage.

Dans la suite, nous étudierons la manière de coupler ces modèles de langage le plus efficacement possible avec le système de reconnaissance, en considérant également des combinaisons de ces modèles de langage afin de tirer partie de leurs spécificités, tout en tenant compte de l'intégration la plus pertinente avec le système de reconnaissance. Nous les utiliserons également en association avec d'autres informations, dans le cadre de premiers travaux sur la vérification de la validité des résultats de reconnaissance et la correction éventuelle de ceux-ci, en se basant sur une nouvelle représentation des hypothèses de reconnaissance, sous la forme de réseaux de confusion.

Première partie

Utilisation de graphes de mots pour la reconnaissance de phrases

Introduction

Dans cette partie, nous nous intéressons au développement d'un système de reconnaissance de phrases manuscrites. Cette reconnaissance se décompose en deux étapes : la segmentation de la phrase en mots puis la reconnaissance, à partir des segmentations proposées et en utilisant des connaissances linguistiques pour prendre en compte le contexte de la phrase. Ces connaissances linguistiques sont représentées par des modèles de langage.

Pour pouvoir, d'une part, représenter l'ensemble des segmentations possibles et, d'autre part, exploiter au mieux ces différentes alternatives, nous utilisons des *graphes de mots* [ONA97]. Ces graphes permettent de représenter, de façon compacte, les différentes hypothèses de segmentation qui correspondent également aux différentes hypothèses de mots. La reconnaissance de phrase consistera alors à trouver le meilleur chemin, dans ce graphe (chacun des chemins correspond à une phrase possible). Cette approche est appelée *Maximum A Posteriori* (MAP) car elle cherche à maximiser la probabilité de la phrase résultat.

Pour pouvoir exploiter au mieux cette représentation, il faut veiller à ce que le graphe conserve une taille raisonnable. En effet, des graphes de mots de trop grande taille occuperont une place trop importante en mémoire et, de plus, leur exploitation sera également plus coûteuse en terme de temps de calcul. Nous nous attacherons à prendre en compte la combinatoire liée à la taille des graphes de mots, dans les approches que nous proposerons.

Cette partie est composée de deux chapitres.

Nous présentons tout d'abord notre approche de segmentation des phrases en mots, dans le chapitre 3. Elle se base sur la caractérisation des espaces entre les différentes parties du tracé manuscrit. Pour pouvoir permettre la proposition d'hypothèses de segmentation alternatives, nous ajoutons également un indice de confiance sur le résultat de la caractérisation des espaces. Nous proposons alors d'étendre l'approche précédente, en utilisant cet indice de confiance pour ajouter des hypothèses supplémentaires de segmentation, dans le graphe de mots généré. Cet ajout est réalisé en prenant en compte la combinatoire liée à la taille du graphe.

Dans le chapitre 4, nous présentons l'intégration de modèles de langage, dans le cadre d'une reconnaissance basée sur l'approche *Maximum A Posteriori* (MAP). Pour cela, nous utilisons des connaissances sur la forme graphique des mots, ainsi que des informations lexicales et syntaxiques. Nous détaillons deux approches permettant d'intégrer, de manière efficace, des modèles de langage, pour apporter des informations syntaxiques lors de la reconnaissance de phrases. La première approche proposée intègre les modèles de langage directement lors de l'exploration du graphe de mots, en prenant en compte son importance vis-à-vis des autres informations utilisées (graphiques et lexicales) et en considérant les éventuels problèmes de

segmentation. Dans la deuxième approche proposée, des modèles de langage sont combinés et utilisés à la fois pendant l'exploration du graphe de mots et pour réordonner les hypothèses de phrases produites par cette exploration. Cela permet de tirer partie des spécificités des différents modèles (plus grande précision de la modélisation, connaissances différentes modélisées, par exemple). Nous étudions enfin l'intégration de différents modèles de langage, pour chacune des deux approches d'intégration, afin de trouver le meilleur tant au point de vue des performances obtenues que de la place mémoire occupée.

Chapitre 3

Extraction automatique des mots

3.1 Introduction

La reconnaissance de phrases manuscrites se déroule généralement en deux étapes. La première étape consiste en l'extraction de ses mots (ou segmentation de la phrase en ses mots) de manière automatique : celle-ci peut être effectuée implicitement ou explicitement.

L'objectif de l'*extraction explicite des mots*, proposée dans ce chapitre, est d'extraire les mots manuscrits, du signal correspondant à la phrase manuscrite à reconnaître. Nous souhaitons également proposer plusieurs hypothèses de segmentation de la phrase, pour s'assurer de la présence de la bonne segmentation. Cet ajout devra se faire en contrôlant la combinatoire liée au nombre d'hypothèses de segmentation. De plus, notre stratégie d'extraction de mots est définie de manière à pouvoir être facilement étendue pour une utilisation « à la volée ».

La figure 3.1 illustre ces deux étapes. Le graphe de mots permet de représenter les différentes hypothèses de segmentation proposées.

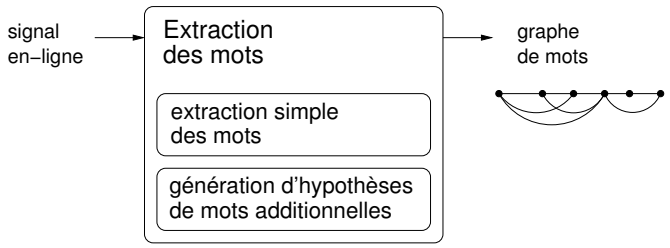


FIG. 3.1 – Principe de l'extraction de mots proposée.

Les travaux, présentés dans ce chapitre, ont été réalisés en collaboration avec François Bouteruche, de l'équipe IMADOC [QBA07].

Dans ce chapitre, nous présentons notre méthode d'extraction explicite des mots. Dans la section 3.2, nous détaillons son principe, qui se base sur la caractérisation des espaces inter-traces. Nous présentons ensuite la construction du graphe de mots permettant de représenter

les différentes hypothèses de mots ainsi extraites. Dans la section 3.3, nous étendons notre méthode pour pouvoir traiter d'éventuelles sur- et sous-segmentations, en remettant en cause la classification de certains espaces inter-traces, tout en contrôlant la complexité du graphe de mots. Enfin, dans la section 3.4, nous présentons les résultats des expérimentations menées.

3.2 Extraction simple des mots

L'objectif de l'extraction explicite des mots consiste à isoler chacun des sous-ensembles de traces correspondant aux mots de la phrase manuscrite considérée.

Notre tâche d'extraction de mots s'appuie sur la *caractérisation des espaces inter-traces* du signal en-ligne représentant la phrase manuscrite à traiter. Nous considérons trois types d'espaces inter-traces :

- *espace intra-mot* : espace entre deux traces, à l'intérieur d'un même mot ;
- *espace inter-mot* : espace entre deux traces de deux mots consécutifs ;
- *espace inter-ligne* : espace entre deux traces de deux mots consécutifs et situés sur deux lignes consécutives.

La figure 3.2 donne un exemple de phrase manuscrite à traiter. Des exemples d'espaces inter-ligne, inter-mot et intra-mot sont aussi illustrés. Nous constatons ainsi que la distance entre deux mots peut être proche de celle entre deux traces d'un même mot.

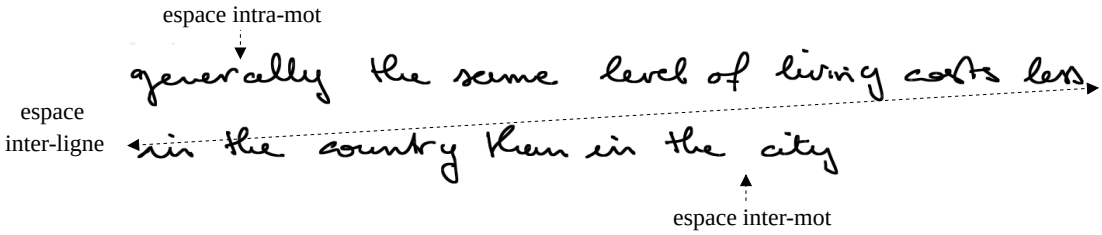


FIG. 3.2 – Exemple de phrase manuscrite.

À partir du résultat de la classification des espaces inter-traces, un graphe de mots est ensuite construit, pour représenter la segmentation obtenue.

Nous détaillons ces deux étapes, dans les sections suivantes.

3.2.1 Caractérisation des espaces inter-traces

L'algorithme de caractérisation des espaces inter-traces, donné par l'algorithme 3.1, analyse les traces au fur et à mesure de leur écriture : cela permet une extension aisée à une reconnaissance d'écriture « à la volée », ce qui constituait un de nos objectifs.

Pour chaque trace T_{nouv} nouvellement écrite, l'espace entre cette trace et la trace précédemment écrite T_{ref} (aussi appelée *trace de référence*) est caractérisé en l'un des trois types d'espaces. Cette caractérisation est basée sur le calcul de la distance Δx_{ref}^{nouv} entre deux traces. En fait, le calcul de cette distance ne s'appuie pas seulement sur la trace précédemment écrite

Algorithme 3.1 : Algorithme de caractérisation des espaces inter-traces.

Données :
les traces manuscrites de la phrase à reconnaître;

Résultat :
les espaces inter-traces classés en espaces intra-mot, inter-mot ou inter-ligne;

début
  Initialisation
  GRB initialisé avec les premières traces écrites, ordonnées temporellement;
  tant que *nouvelle trace* T_{nouv} *écrite* **faire**
    détection des lignes de base de l'écriture;
    calcul de la distance Δx_{ref}^{nouv} entre les traces T_{ref} et T_{nouv};
    classification de l'espace E_{ref}^{nouv} entre T_{ref} et T_{nouv};
    mise à jour de GRB;
fin

T_{ref} mais également sur un groupe de traces précédemment écrites, appelé *Groupe Référence de Base* (GRB) : ce groupe de traces représente un contexte spatial relatif, par rapport auquel la distance Δx_{ref}^{nouv} peut être comparée. La figure 3.3 illustre le calcul de la distance Δx_{ref}^{nouv}, ainsi que le GRB correspondant utilisé. Un classifieur est enfin utilisé pour caractériser l'espace inter-trace, en se basant sur la distance Δx_{ref}^{nouv} ainsi que sur d'autres d'informations.

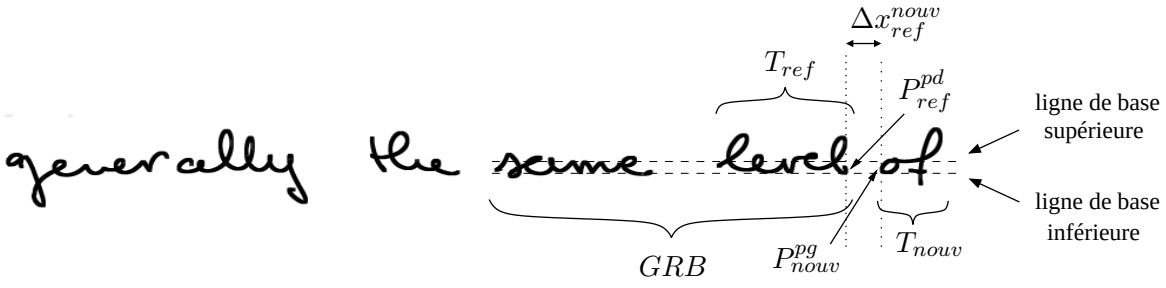


FIG. 3.3 – Exemple de calcul de distance Δx_{ref}^{nouv} entre deux traces.

Dans les sous-sections suivantes, nous détaillons les différentes étapes de l'algorithme de caractérisation des espaces inter-traces.

3.2.1.1 Initialisation

Au début de la tâche d'extraction de mots, le Groupe Référence de Base GRB est initialisé avec les premières traces écrites de la phrase manuscrite (voir sous-section suivante).

3.2.1.2 Détection des lignes de base inférieure et supérieure

L'algorithme utilisé pour détecter ces deux lignes de base est le même que celui utilisé pour la détection du corps des mots, dans notre système de reconnaissance de mots (voir chapitre 1, section 1.4). Notre algorithme utilise, pour cela, un groupe de traces représentant au moins trois ou quatre caractères : ce groupe de traces correspond au GRB précédemment présenté.

Afin de déterminer le nombre de traces à conserver dans le GRB, nous nous appuyons sur les *traits descendants fondamentaux* (voir chapitre 1, section 1.4). Comme un caractère est composé de un à trois traits descendants fondamentaux, nous conservons suffisamment de traces pour que le GRB courant contienne au moins dix traits descendants fondamentaux.

Le processus de détection des lignes de base, et ainsi celui de caractérisation des espaces inter-traces, peut débuter dès que le nombre de traces écrites correspond au moins à dix traits descendants fondamentaux : cela permet d'initialiser GRB. Ce premier GRB est aussi celui utilisé pour la caractérisation des premiers espaces inter-traces, bien que les premières traces appartiennent à celui-ci.

3.2.1.3 Calcul de la distance Δx_{ref}^{nouv}

Comme nous travaillons sur de l'écriture latine, la première trace d'un nouveau mot est le plus vraisemblablement située à la droite de la dernière trace du mot précédent. Pour calculer la distance Δx_{ref}^{nouv} (voir figure 3.3) entre la dernière trace écrite T_{ref} (appelée aussi *trace de référence*) et la trace qui vient d'être écrite T_{nouv}, nous proposons d'utiliser la distance en x entre le point le plus à droite de la trace T_{ref} et le point le plus à gauche de la trace T_{nouv} :

$$\Delta x_{ref}^{nouv} = x_{P_{nouv}^{pg}} - x_{P_{ref}^{pd}} \tag{3.1}$$

avec P_{nouv}^{pg} le point le plus à gauche de T_{nouv} et P_{ref}^{pd} le point le plus à droite de T_{ref}.

Comme l'algorithme de caractérisation des espaces inter-traces est conçu de manière à pouvoir être étendu à la reconnaissance « à la volée », nous voulons éviter des étapes de prétraitements qui sont généralement effectuées sur le signal afin de corriger des problèmes d'inclinaison des caractères ou de la ligne d'écriture ou encore de taille des caractères. Notre distance Δx_{ref}^{nouv} doit ainsi être insensible à ces variations d'écriture.

Afin de traiter le problème de l'inclinaison des caractères (en anglais, *slant*), nous ne considérons que les points situés entre les lignes de base inférieure et supérieure. Ainsi, les points P_{nouv}^{pg} et P_{ref}^{pd} seront choisis parmi les points contenus entre ces lignes de base. Dans l'exemple donné par la figure 3.4, si le point le plus à droite du mot *said* était choisi, un espace intra-mot serait détecté comme ce point est à droite du point P_{nouv}^{pg}.

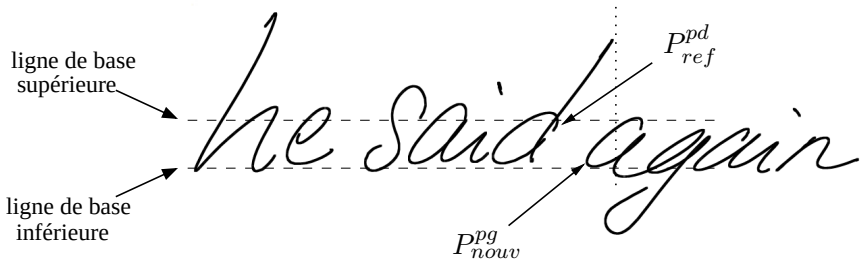


FIG. 3.4 – Choix des points P^{pg}_{nouv} et P^{pd}_{ref} pour traiter le problème d'inclinaison des caractères.

Afin de pallier au problème de l'inclinaison de la ligne d'écriture (en anglais, *skew*), seules les dernières traces écrites avant T_{nouv} sont conservées dans le GRB, qui est ensuite utilisé pour effectuer la détection des lignes de base.

3.2.1.4 Classification de l'espace E^{nouv}_{ref}

Une fois que la distance Δx^{nouv}_{ref} a été calculée, nous utilisons un *réseau de neurones à fonction à base radiale* (RBFN) pour classer l'espace inter-trace. Les entrées de ce réseau sont les suivantes :

- la longueur de l'espace inter-traces courant, relativement aux traces précédentes : cela correspond à la distance Δx^{nouv}_{ref} ;
- des informations sur les tailles relatives des espaces inter-traces précédents : cela correspond aux distances Δx^{nouv}_{ref} maximale et médiane, dans le groupe de référence courant GRB ;
- des informations pour l'identification des espaces inter-lignes : cette information est donnée par la distance entre le haut de la boîte englobante de la trace T_{nouv} et la ligne de base inférieure.

Les sorties du RBFN sont les scores associés à chacune des trois classes, c'est-à-dire l'espace intra-mot, l'espace inter-mot et l'espace inter-ligne. Le type de l'espace inter-trace considéré est ainsi celui associé à la classe ayant obtenu le score le plus élevé.

3.2.1.5 Mise à jour du GRB

Une fois l'espace E^{nouv}_{ref} classé, la trace T_{nouv} est ajoutée au groupe GRB. La trace la plus ancienne temporellement, dans le GRB, peut ensuite être supprimée s'il reste au moins dix traits descendants fondamentaux après sa suppression du GRB.

La gestion du GRB peut être vue comme une fenêtre glissante, qui se déplace au fur et à mesure de l'écriture et du traitement des traces. Cette gestion est alors particulièrement adaptée pour une extraction des mots « à la volée ».

3.2.2 Construction du graphe de mots

À partir des résultats de la caractérisation des espaces inter-traces, nous construisons un graphe de mots. À chaque fois qu'un espace inter-ligne ou inter-mot est détecté, un nœud est construit : chaque nœud représente ainsi une frontière de segmentation entre deux mots. Les arcs, quant à eux, regroupent les traces telles que les espaces inter-traces entre chacune des traces d'un arc sont des espaces intra-mots. Ils représentent ainsi les mots manuscrits détectés.

La figure 3.5 donne un exemple de graphe de mots construit à partir de la caractérisation des espaces inter-traces de la phrase *« generally the same level of living costs less in the country than in the city »*. Nous constatons que les mots *living* et *country* sont sur-segmentés alors que les mots *« than in »* sont sous-segmentés.

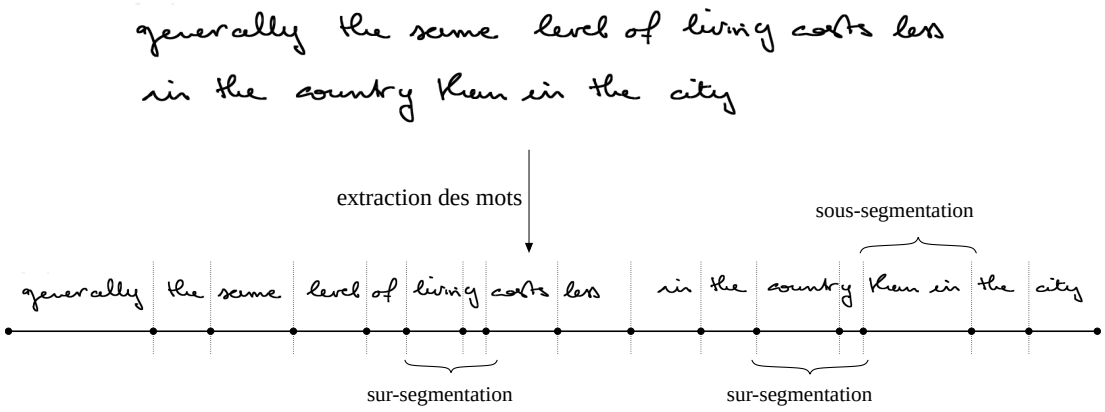


FIG. 3.5 – Exemple de graphe de mots, construit à partir de l'extraction simple des mots d'une phrase manuscrite.

Pour pouvoir gérer les éventuelles sur- et sous-segmentations en ajoutant des hypothèses alternatives d'extraction de mots, tout en contrôlant la taille du graphe, nous étendons notre algorithme de caractérisation des espaces inter-traces.

3.3 Prise en compte de plusieurs hypothèses d'extraction

Pour s'assurer de la présence des extractions correctes des mots, une solution pourrait être de considérer toutes les hypothèses d'extraction possibles des traces d'une phrase. La complexité du graphe serait alors en $O(T^2)$, en termes de nombre d'arcs, avec T le nombre de nœuds (qui correspond aussi au nombre de traces composant la phrase).

Pour des problèmes de stockage du graphe en mémoire ainsi que de temps de calcul lors de l'exploitation du graphe, cette solution n'est pas raisonnable. De plus, la prise en compte d'un trop grand nombre d'hypothèses de segmentation ajoute surtout du bruit, c'est-à-dire des segmentations incorrectes. Cela risque d'apporter des erreurs, lors de la phase de reconnaissance de phrases sur le graphe de mots. Les expérimentations présentées dans la section 3.4 confirment cette idée.

Il est donc important de choisir les hypothèses d'extraction de mots ajoutées dans le graphe, afin qu'elles soient les plus pertinentes possibles. Pour cela, nous nous appuyons sur un indice évaluant la confiance en le résultat de la caractérisation des espaces inter-traces.

Nous détaillons, dans les sous-sections suivantes, le calcul de cet indice de confiance ainsi que son utilisation pour créer de nouvelles hypothèses d'extraction de mots visant à traiter les éventuels problèmes de sur- et sous-segmentation.

3.3.1 Ajout d'un indice de confiance sur la caractérisation des espaces

Afin d'évaluer la fiabilité de la première réponse de la classification des espaces inter-traces, nous associons un *indice de confiance* au résultat du RBFN. Si cet indice de confiance est trop faible, la seconde réponse du classifieur est aussi considérée.

L'indice de confiance utilisé correspond à la différence relative $diff_{top2}$ entre les scores des deux meilleures classes. Nous apprenons un seuil $\sigma_{reconsider}$ sur cet indice de confiance, en utilisant un rejet d'ambiguïté [MA06]. Ce type de rejet permet d'apprendre un seuil en fixant le pourcentage d'éléments qui doivent être rejetés c'est-à-dire, ici, le pourcentage d'espaces inter-traces qui doit être reconsidéré. Si l'indice de confiance $diff_{top2}$ est inférieur à ce seuil appris, la première réponse du RBFN peut être reconsidérée et des hypothèses additionnelles d'extraction de mots peuvent être générées.

3.3.2 Extension du graphe de mots

Les indices de confiance associés à chacun des espaces inter-traces permettent maintenant d'ajouter des hypothèses additionnelles d'extraction de mots (représentées par des arcs, dans le graphe de mots).

Afin de contrôler la taille du graphe de mots, lors de ces ajouts d'arcs, nous nous imposons des limitations. Tout d'abord, les espaces inter-lignes sont considérés comme étant correctement détectés et ne seront pas remis en cause. Les autres restrictions concernent les éventuelles sur- et sous-segmentations, dont nous détaillons le traitement dans les paragraphes suivants. Ces limitations ne sont pas trop restrictives puisqu'elles permettent d'identifier correctement 99,54 % des segmentations correctes, dans l'ensemble de test.

3.3.2.1 Traitement des éventuelles sous-segmentations

Pour limiter la création d'arcs supplémentaires, nous considérons que le nombre de mots potentiellement sous-segmentés (représentés par un même arc) ne peut pas être supérieur à trois. Un arc du graphe de mots initial sera ainsi séparé en au plus trois parties. Cette approximation est raisonnable puisque, dans la base de test, seules 4,8 % des sous-segmentations ne pourront pas être corrigées.

Afin de choisir les potentiels points de séparation d'un arc donné, ses espaces intra-mot qui ont une distance Δx_{ref}^{nouv} positive et un indice de confiance $diff_{top2}$ inférieur au seuil de reconsidération $\sigma_{reconsider}$ sont ordonnés selon leur indice de confiance croissant. Ensuite, seuls au plus les deux premiers espaces intra-mot ainsi classés sont considérés comme de potentiels

espaces inter-mot. Des arcs et des nœuds additionnels sont alors créés, à partir des traces de l'arc courant considéré, comme illustré en pointillé sur la figure 3.6.

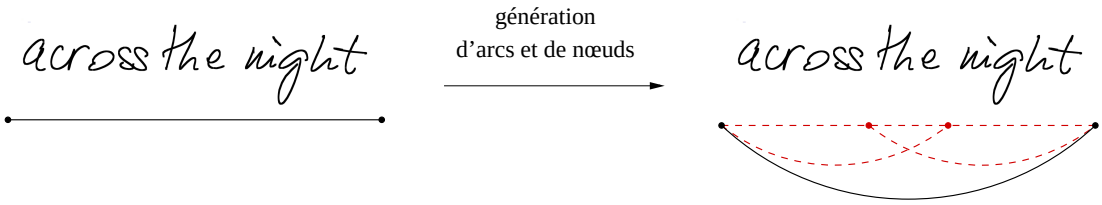


FIG. 3.6 – Exemple de création d'arcs et de nœuds, pour traiter une sous-segmentation.

3.3.2.2 Traitement des éventuelles sur-segmentations

De la même façon que pour le traitement des éventuelles sur-segmentations, nous considérons qu'un mot ne peut pas être sur-segmenté en plus de trois parties. Les arcs ajoutés regrouperont au plus trois mots. Avec cette limitation, nous ne pourrons pas remettre en cause 2,4 % des sur-segmentations, présentes dans la base de test.

Afin de traiter ces éventuelles sur-segmentations, nous considérons chaque groupe de trois arcs consécutifs. Si un ou deux espaces inter-mot (représentés par les nœuds du graphe) ont leur indice de confiance $diff_{top2}$ inférieur au seuil de reconsidération $\sigma_{reconsider}$, des arcs additionnels sont créés, comme illustré en pointillé sur la figure 3.7.

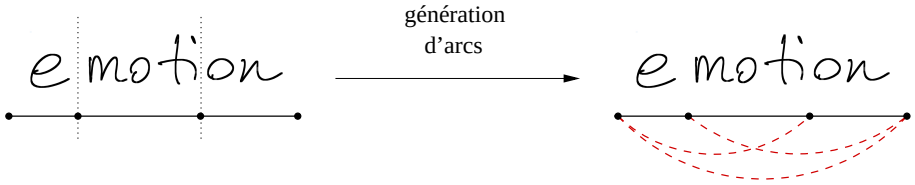


FIG. 3.7 – Exemple de création d'arcs, pour traiter une sur-segmentation.

De plus, ces arcs ne peuvent être créés que si le nombre de traits descendants fondamentaux qu'ils contiennent est inférieur à 25. Cette limite correspond au nombre de traits descendants fondamentaux du mot le plus long de notre vocabulaire.

3.3.2.3 Création du graphe de mots final

Le graphe final est obtenu après avoir traité chacune des sur- et sous-segmentations.

La figure 3.8 donne le graphe de mots complet résultat de l'extraction des mots, pour la phrase manuscrite déjà présentée par la figure 3.5. Les arcs et nœuds représentés en pointillé rouge correspondent aux ajouts effectués pour traiter les sur- et sous-segmentations initiales.

Nous remarquons ainsi qu'il existe maintenant un chemin correspondant à la bonne segmentation de la phrase en ses mots.

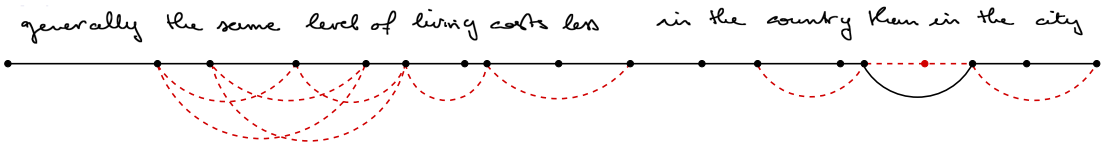


FIG. 3.8 – Exemple de graphe de mots complet, construit en reconsidérant certaines classifications d'espaces inter-traces (en pointillé, arcs et nœuds ajoutés).

3.4 Expérimentations

Dans cette section, nous présentons les expérimentations menées sur l'extraction automatique des mots. L'approche proposée s'appuie sur la génération d'hypothèses additionnelles en se basant sur un seuil $\sigma_{reconsider}$. L'objectif de ces expérimentations est de trouver le seuil qui permet d'assurer la présence de la segmentation correcte de la phrase dans le graphe de mots généré, tout en contrôlant la taille de ce graphe.

Nous introduisons tout d'abord les données manuscrites utilisées, ainsi que les différents indicateurs qui nous permettent de comparer les différentes stratégies étudiées.

3.4.1 Cadre expérimental

3.4.1.1 Bases manuscrites

Les phrases manuscrites ont été saisies sur un *TabletPC* à l'aide d'une interface de saisie développée pour cette campagne de saisie. Le protocole de saisie n'impose aucune contrainte sur le style d'écriture. Cependant, les phrases ne comportent que des lettres minuscules.

Chaque saisie comporte au maximum 20 phrases, choisies aléatoirement parmi celles du corpus $Brown_{saisie}$ (voir section 4.5.1.1).

La base complète est constituée de 942 phrases, saisies par 42 scripteurs. Cette base est séparée en une base d'apprentissage et une base de test. Les caractéristiques de ces deux bases sont données dans la table 3.1.

La base d'apprentissage $Saisie_{app}$ est utilisée pour optimiser les paramètres liés à l'extraction des mots, c'est-à-dire pour apprendre le RBFN utilisé pour la classification des espaces inter-traces ainsi que pour calculer les seuils $\sigma_{reconsider}$ correspondant au pourcentage de classifications d'espaces inter-traces à remettre en cause. La base de test $Saisie_{test}$ est utilisée pour l'évaluation des stratégies proposées.

TAB. 3.1 – Caractéristiques des bases de phrases manuscrites d'apprentissage et de test.

Statistiques	Base $Saisie_{app}$	Base $Saisie_{test}$
Nombre de scripteurs	25	17
Nombre de phrases	517	425
Nombre de mots	8 047	6 362
Nombre de traces	32 288	24 987

3.4.1.2 Mesures de performance

Pour évaluer l'extraction automatique des mots (ainsi que la reconnaissance des phrases, dans les chapitres suivants), nous présentons des indicateurs sur les graphes de mots construits.

Densité du graphe (DG) Elle permet d'évaluer la taille du graphe de mots, en nombre d'arcs, relativement au nombre de mots composant les phrases :

$$DG = \frac{nb_{arcs}}{nb_{mots}} \times 100 \tag{3.2}$$

avec nb_{arcs} le nombre d'arcs présents dans le graphe et nb_{mots} le nombre de mots à reconnaître.

Taux de présence des arcs (TPA) Il évalue le pourcentage des arcs du graphe qui correspondent à des mots à reconnaître :

$$TPA = \frac{nb_{arcs}^{présents}}{nb_{arcs}} \times 100 \tag{3.3}$$

avec $nb_{arcs}^{présents}$ le nombre d'arcs du graphe correspondant à des mots à reconnaître.

Taux de présence des mots (TPM) Ce taux permet de mesurer le taux de reconnaissance maximal pouvant être atteint :

$$TPM = \frac{nb_{mots}^{présents_m}}{nb_{mots}} \times 100 \tag{3.4}$$

avec $nb_{mots}^{présents_m}$ le nombre de mots à reconnaître présents parmi les m mots des listes de mots candidats, données par le système de reconnaissance de mots RESIFMot et associées aux arcs du graphe de mots.

Taux de reconnaissance des mots (TRM) Pour évaluer la reconnaissance de phrases (présentée dans les chapitres suivants), nous n'utilisons pas le taux de reconnaissance sur les phrases. En effet, il suffit qu'un mot ne soit pas correctement reconnu pour que la phrase ne

le soit pas, elle aussi ; les phrases longues se trouvent alors particulièrement pénalisées. Nous utilisons alors le *taux de reconnaissance sur les mots*, défini par :

$$TRM = \frac{nb_{mots} - nb_{mots}^{subst} - nb_{mots}^{suppr}}{nb_{mots}} \times 100 \tag{3.5}$$

avec nb_{mots}^{subst} le nombre de substitutions de mots entre la phrase à reconnaître et la phrase résultat et nb_{mots}^{suppr} le nombre de suppressions de mots de la phrase à reconnaître. Pour calculer le nombre minimal de substitutions et de suppressions pour passer de la phrase à reconnaître à celle en résultat, nous utilisons une distance d'édition (distance de Damerau-Levenshtein [Dam64]). Le coût de chacune de ces opérations est fixé à 1. Classiquement, l'opération d'insertion d'un mot dans la phrase résultat n'intervient pas dans ce calcul mais il intervient dans celui de la *précision* des mots, indicateur que nous n'utilisons pas ici.

Taux d'erreur sur les mots (TEM) Ce taux donné par rapport au taux de reconnaissance des mots et correspond à :

$$TEM = 100 - TRM. \tag{3.6}$$

3.4.2 Choix de la meilleure stratégie d'extraction des mots

Notre tâche d'extraction de mots, présentée dans ce chapitre, comporte une étape qui permet l'ajout d'hypothèses de segmentation supplémentaires, par rapport à notre extraction simple. L'ajout de ces hypothèses est contrôlé par le seuil $\sigma_{reconsider}$, appris automatiquement à partir du pourcentage d'hypothèses de segmentation initiales à remettre en cause.

L'objectif des expérimentations présentées est de trouver le seuil $\sigma_{reconsider}$ qui correspond à la stratégie permettant d'obtenir le meilleur compromis entre l'augmentation de la taille du graphe et la présence de la bonne segmentation dans ce graphe. Pour cela, nous comparons différentes stratégies de segmentation, qui ne diffèrent que par leur seuil $\sigma_{reconsider}$.

La figure 3.9 illustre la comparaison de 5 stratégies de segmentation, entre elles et par rapport à la segmentation manuelle, qui correspond à la vérité terrain. Nous identifions ces stratégies par le pourcentage de classifications d'espaces inter-traces remises en cause. La comparaison est effectuée sur la base $Saisie_{app}$. Cette comparaison est effectuée en termes de DG, TPA ainsi que de TPM et TRM, pour évaluer l'impact des stratégies de segmentation sur la reconnaissance des phrases, qui constitue le but final. Pour le taux de reconnaissance des mots, les résultats ont été obtenus en utilisant un modèle de langage bigramme (pour les détails concernant l'intégration des modèles de langage ainsi que les résultats des expérimentations correspondantes, voir le chapitre 4).

Nous observons une augmentation significative des trois taux présentés en passant d'une extraction automatique simple des mots à une segmentation ajoutant des hypothèses alternatives, à partir de 10 % de résultats de classification inter-traces remis en cause. Cette augmentation des taux se stabilise ensuite assez rapidement et le taux de reconnaissance des mots décroît même légèrement. Cette petite baisse peut s'expliquer par le fait que l'ajout d'hypothèses alternatives d'extraction de mots ne rajoute vraisemblablement que du bruit, les hypothèses ajoutées représentant essentiellement des mauvaises segmentations. Lors de l'ajout d'hypothèses

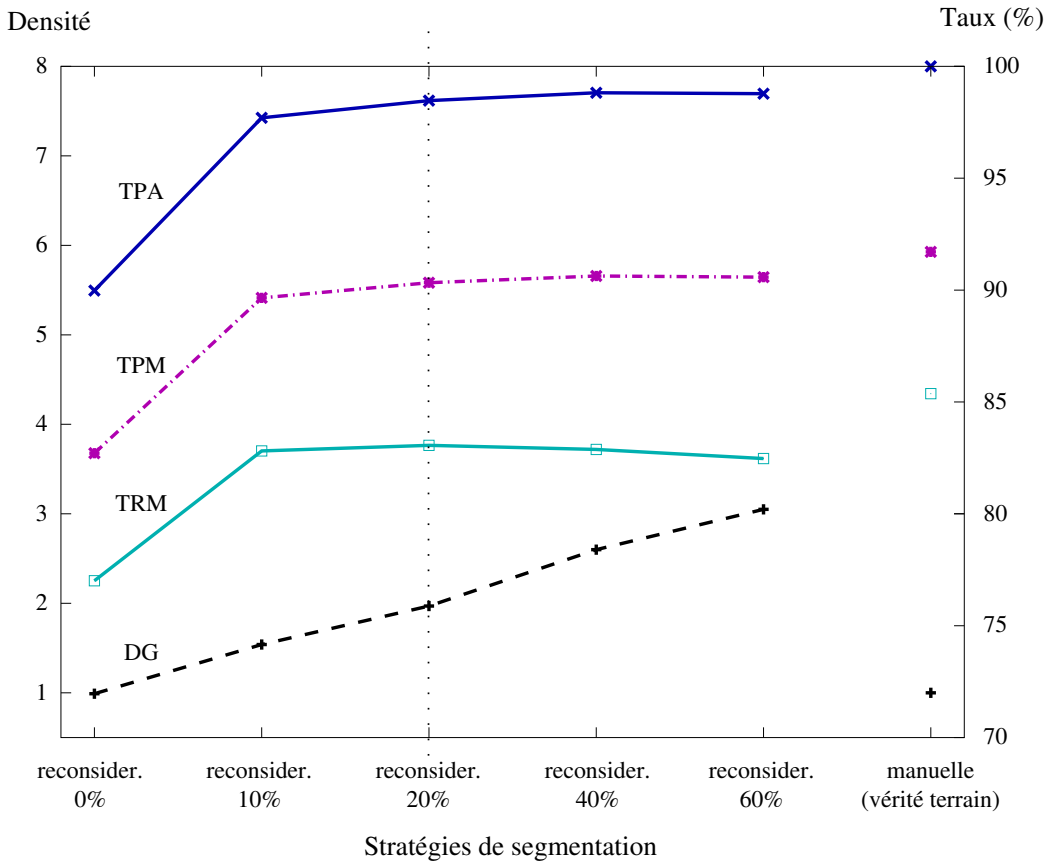


FIG. 3.9 – Comparaison des différentes stratégies d'extraction de mots.

alternatives de segmentation, nous constatons également que la densité du graphe augmente quasiment linéairement, sans stabilisation.

La meilleure stratégie d'extraction automatique des mots correspond donc à celle qui approche le plus les résultats obtenus avec une extraction manuelle, maximisant ainsi les trois taux présentés et minimisant la densité du graphe de mots, pour contrôler la taille de celui-ci. Nous considérons alors que la meilleure stratégie d'extraction automatique de mots correspond à celle *reconsidérant 20 % des classifications* des espaces inter-traces (correspondant à un seuil $\sigma_{reconsider}$ = 0,84) : cette stratégie de segmentation sera utilisée dans les différentes expérimentations sur la reconnaissance des phrases, présentées dans les chapitres suivants. Nous aurions pu choisir la stratégie reconsidérant 10 % des classifications des espaces inter-traces mais, comme la densité du graphe n'augmente pas beaucoup en passant à 20 % de reconsidération des classifications, le choix de cette stratégie basée sur la reconsidération de 20 % des classifications pourrait permettre de plus grandes améliorations des performances, lors de l'étape de reconnaissance des phrases.

Nous comparons maintenant les mêmes stratégies que précédemment mais sur la base de test, pour montrer le pouvoir de généralisation de la base d'apprentissage. La base d'apprentissage est utilisée pour choisir la meilleure stratégie d'extraction des mots et nous voulons vérifier que c'est aussi la meilleure, sur la base de test. La table 3.2 regroupe ainsi les résultats obtenus en comparant les mêmes stratégies que précédemment, en utilisant les mêmes indicateurs.

TAB. 3.2 – Comparaison des différentes stratégies d'extraction de mots, sur la base de test.

Stratégie d'extraction des mots		DG	TPA	TPM	TRM
Manuelle		1,00	100,00 %	93,70 %	88,48 %
Automatique avec reconsidération	0 %	0,99	89,86 %	84,38 %	79,77 %
	10 %	1,58	97,94 %	91,92 %	86,20 %
	20 %	**2,04**	**98,85 %**	**92,72 %**	**86,52 %**
	40 %	2,71	99,21 %	93,05 %	86,53 %
	60 %	3,19	99,20 %	93,05 %	86,44 %

Nous obtenons les mêmes conclusions que sur la base d'apprentissage, en remarquant que l'ajout d'hypothèses de segmentation permet d'augmenter de 7 à 9 % les différents taux (TPA, TPM et TRM), par rapport à une extraction automatique simple des mots. Pour la stratégie avec 20 % de reconsidération, le taux de reconnaissance des mots passe de 79,77 % (avec une extraction simple des mots) à 86,52% : cela correspond à une réduction relative de 33,4 % du taux d'erreur sur les mots. De plus, ces trois taux sont relativement proches de ceux obtenus avec l'extraction manuelle qui constitue la vérité terrain (taux en moyenne 1 à 2 % inférieurs). Enfin, la densité moyenne des graphes de mots reste raisonnable.

Ces trois indicateurs permettent aussi d'identifier différents types d'erreurs qui pourront ensuite être corrigées en optimisant différentes parties du système de reconnaissance : le TPA permet de détecter les erreurs issues de la génération des hypothèses de segmentation alors que le TPM permet de mettre en évidence les erreurs du système de reconnaissance de mots et que le TRM permet la détection d'erreurs provenant du système de reconnaissance de phrases.

3.5 Bilan

Dans ce chapitre, nous avons présenté une approche pour l'extraction automatique des mots des phrases manuscrites, de manière explicite. L'approche que nous avons proposée est basée sur la classification des espaces inter-traces de la phrase, à l'aide d'un RBFN. Cela permet d'identifier les traces de chacun des mots de celle-ci.

Afin de pouvoir éventuellement remettre en cause les résultats de cette classification, nous avons introduit une stratégie de reconsidération des espaces inter-traces ainsi caractérisés. Cette remise en cause est basée sur des seuils appris automatiquement. Cette étape permet d'ajouter des hypothèses alternatives d'extraction de mots, dans le graphe de mots, tout en contrôlant sa taille. L'ajout de ces hypothèses permet d'augmenter significativement le taux d'arcs correspondant à de bonnes extractions de mots.

Chapitre 4

Intégration de connaissances linguistiques pour la reconnaissance de phrases

4.1 Introduction

La reconnaissance de phrases manuscrites permet de reconnaître chacun des mots d'une phrase, segmentée au préalable (la segmentation peut aussi être effectuée conjointement à la tâche de reconnaissance). Pour ce faire, des connaissances linguistiques sont associées aux informations de nature plutôt graphique, issues du système de reconnaissance de mots.

Nous nous intéressons ici aux informations au niveau syntaxique. L'utilisation de ces connaissances permet de tirer partie du contexte des mots pour lever des ambiguïtés sur leur reconnaissance isolée. Pour intégrer ces informations, nous nous appuyons sur une représentation des hypothèses de segmentation sous forme de graphe de mots, comme illustré par la figure 4.1.

Le graphe de mots est obtenu à l'issue de l'extraction des mots (voir chapitre 3). Le module de reconnaissance de phrases recherche ensuite le meilleur chemin dans le graphe de mots, en utilisant notre système de reconnaissance de mots RESIFMot, pour reconnaître chacun des mots manuscrits associés aux arcs du graphe, et en intégrant également des connaissances linguistiques grâce aux *modèles de langage*. Le résultat de cette étape est une liste de M-meilleures phrases correspondant à la phrase manuscrite. Une étape éventuelle de post-traitement syntaxique, utilisant de nouveaux modèles de langage, permet de réordonner cette liste d'hypothèses de phrases. Le résultat final de la reconnaissance est alors une liste de M phrases, la première phrase de cette liste étant la phrase résultat.

Dans ce chapitre, nous exposons notre approche pour la reconnaissance de phrases manuscrites en-ligne. Dans la section 4.2, nous introduisons le principe général de la reconnaissance de phrases s'appuyant sur des modèles de langage. Nous détaillons ensuite les deux approches basées, d'une part, sur l'intégration des modèles de langage lors de l'exploration du graphe de mots (voir section 4.3) et, d'autre part, sur la combinaison de modèles de langage lors d'une phase de post-traitement syntaxique (voir section 4.4). Enfin, dans la section 4.5, nous

présentons nos résultats issus de l'étude de modèles n-grammes et n-classes (statistiques ou morpho-syntaxiques) pour la reconnaissance de phrases selon les deux approches proposées.

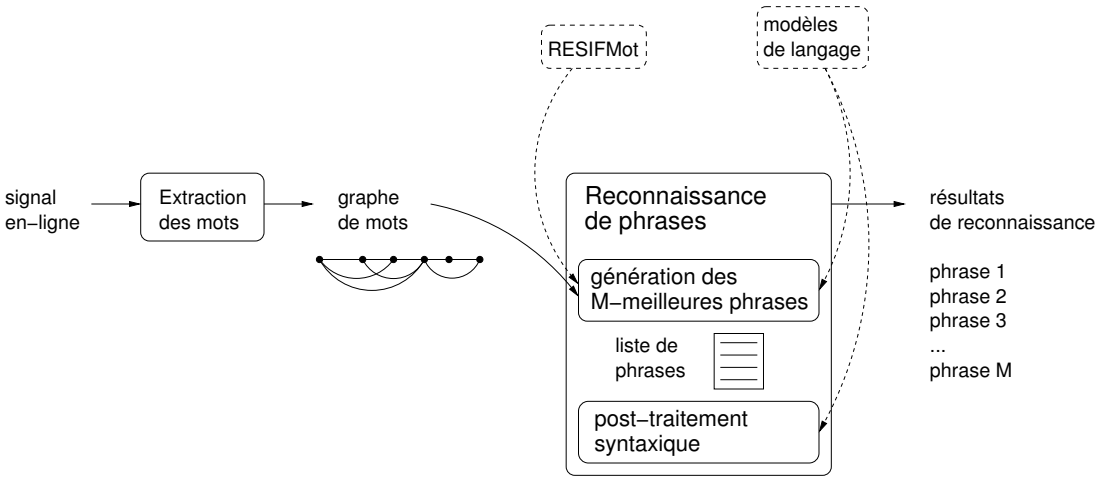


FIG. 4.1 – Principe du système de reconnaissance de phrases manuscrites.

4.2 Principe général

Comme nous l'avons vu dans la section 1.3 du chapitre 1 (exprimé par l'équation 1.5), le but de la reconnaissance de phrases manuscrites est de trouver la phrase $\widehat{W}$ qui correspond la mieux au signal manuscrit S, en utilisant un système de reconnaissance ainsi que des connaissances linguistiques sur les phrases. Comme nous effectuons une extraction explicite des mots de la phrase manuscrite, l'équation 1.5 peut s'exprimer de la façon suivante :

$$\widehat{W} = \arg\max_{W} \sum_{i=1}^{N} log(P(s_i|w_i)) \; + \; \gamma \, log(P(w_i|\phi(h_i))) \; + \; \delta \tag{4.1}$$

avec $log(P(s_i|w_i))$ le score donné par le système de reconnaissance de mots qui prend en entrée la partie s_i du signal S correspondant à l'hypothèse d'extraction du mot w_i et $log(P(w_i|\phi(h_i)))$ le score donné par le modèle de langage au mot w_i, étant donné son contexte $\phi(h_i)$ (voir chapitre 2, sur la modélisation du langage). Pour rappel, les paramètres γ et δ sont utilisés, respectivement, pour pondérer l'importance du modèle de langage vis-à-vis du système de reconnaissance et pour compenser les éventuelles sur- et sous-segmentations. En fait, comme notre système de reconnaissance de mots RESIFMot n'est pas probabiliste (voir chapitre 1), nous approximons l'équation 4.1 par :

$$\widehat{W} = \arg\max_{W} \sum_{i=1}^{N} score(s_i|w_i) \; + \; \gamma \, log(P(w_i|\phi(h_i))) \; + \; \delta \tag{4.2}$$

avec $score(s_i|w_i)$ un score associé à chacun des mots résultats w_i et calculé à partir d'informations données par notre système de reconnaissance RESIFMot (voir le chapitre 1 pour une explication plus détaillée de ces informations). L'information principalement utilisée est le score donné à l'issue du post-traitement lexical, $score_{lexique}$. Dans le cas où plusieurs mots de la liste de mots candidats $l_i = liste_{mots}(s_i)$ ont des scores lexicaux égaux, nous utilisons en plus leur score graphique, $score_{graphique}$. Le score $score(s_i|w_i)$ est ainsi donné par :

$$score(s_i|w_i) = \begin{cases} score_{lexique}(w_i) & \text{si } score_{lexique}(w_i) \text{ unique dans } l_i \\ score_{lexique}(w_i) + \alpha(w_i) & \text{sinon} \end{cases} \quad (4.3)$$

avec $\alpha(w_i)$ un cœfficient utilisant le score graphique du mot w_i et défini par :

$$\alpha(w_i) = \frac{score_{graphique}(w_i) - score^{min}_{graphique}(l_i)}{score^{max}_{graphique}(l_i) - score^{min}_{graphique}(l_i)} \left(score^{suiv}_{lexique}(w_i) - score_{lexique}(w_i)\right) \quad (4.4)$$

avec $score^{min}_{graphique}(l_i)$ (respectivement $score^{max}_{graphique}(l_i)$) le score graphique minimal (respectivement maximal) parmi ceux associés aux mots de la liste l_i et $score^{suiv}_{lexique}(w_i)$ le premier score lexical différent de celui de w_i, parmi les mots de la liste l_i qui suivent le mot w_i. L'ajout de ce cœfficient permet d'associer un score unique à chacun des mots de la liste tout en conservant leur ordonnancement initial dans cette liste, dans le cas où plusieurs mots candidats ont le même score lexical.

4.3 Intégration de modèles de langage pendant l'exploration du graphe de mots

Afin que l'intégration du modèle de langage soit la plus efficace, en participant le plus tôt possible à la reconnaissance, celui-ci est utilisé au moment de l'exploration du graphe de mots, en conjonction du score de chacun des mots candidats du graphe.

4.3.1 Modèles de langage de type modèle bigramme

La reconnaissance de phrases est réalisée par la recherche du meilleur chemin dans le graphe de mots qui représente les différentes hypothèses d'extraction de mots. L'algorithme de reconnaissance, donné par l'algorithme 4.1, s'appuie sur l'algorithme de Viterbi [For73] (basé sur la programmation dynamique) pour effectuer efficacement cette recherche qui correspond au calcul de l'équation 4.2.

Les informations utilisées par cet algorithme sont celles présentes dans le graphe de mots et illustrées par la figure 4.2. Le système de reconnaissance de mots est utilisé pour donner la liste de mots candidats correspondant au signal (c'est-à-dire l'ensemble des traces) associé à chacun des arcs du graphe. Un score $score(s_i|w_i)$ (voir équation 4.3) est donné à chacun des mots de cette liste ordonnée. Le modèle de langage donne enfin une probabilité à chacun des couples de mots candidats qui appartiennent à des listes associées à des arcs consécutifs. La figure 4.2 illustre, en pointillé, les probabilités associées aux couples de mots formées du mot *keen* et de chacun des mots appartenant à la liste d'un des arcs précédents.

Algorithme 4.1 : Algorithme de reconnaissance de phrases.

Données :
G : le graphe de mots composé de T nœuds n_t et des arcs a_t^{t+x} (ou a_{t-y}^{t}) définis par leur nœud de début n_t (ou n_{t-y} avec $y \in \{1, 2, 3\}$) et leur nœud de fin n_{t+x} avec $x \in \{1, 2, 3\}$ (ou n_t);
$score_{max}(w_i)$: le score du meilleur chemin allant jusqu'au mot w_i;
$pred_{max}(w_i)$: le mot meilleur prédécesseur sur le meilleur chemin allant jusqu'à w_i;
$score(s_i|w_i)$: le score du mot w_i (donné par le système de reconnaissance de mots);
$log(P(w_i|w_j))$: la probabilité du bigramme $w_j w_i$ (donné par le modèle de langage);

Résultat :
$\widehat{W}$: la phrase résultat, qui a le score maximal dans le graphe de mots G;

début
- **pour chaque** *nœud* n_t *du graphe* G*, dans un ordre topologique* **faire**
  - **pour chaque** *arc* a_t^{t+x} *successeur du nœud* n_t **faire**
    - **pour chaque** *mot candidat* w_i *de l'arc* a_t^{t+x} **faire**
      - **pour chaque** *arc* a_{t-y}^{t} *prédécesseur du nœud* n_t **faire**
        - **pour chaque** *mot candidat* w_j *de l'arc* a_{t-y}^{t} **faire**
          - $score = score_{max}(w_j) + (score(s_i|w_i) + \gamma\, log(P(w_i|w_j)) + \delta)$;
          - **si** $score > score_{max}(w_i)$ **alors**
            - $score_{max}(w_i) = score$;
            - $pred_{max}(w_i) = w_j$;
  - $\widehat{W} = \widehat{w}_N$;
  - $w_i = \widehat{w}_N$;
  - **tant que** $w_i \neq \widehat{w}_1$ **faire**
    - $w_i = pred_{max}(w_i)$;
    - $\widehat{W} = w_i \,.\, \widehat{W}$;

fin

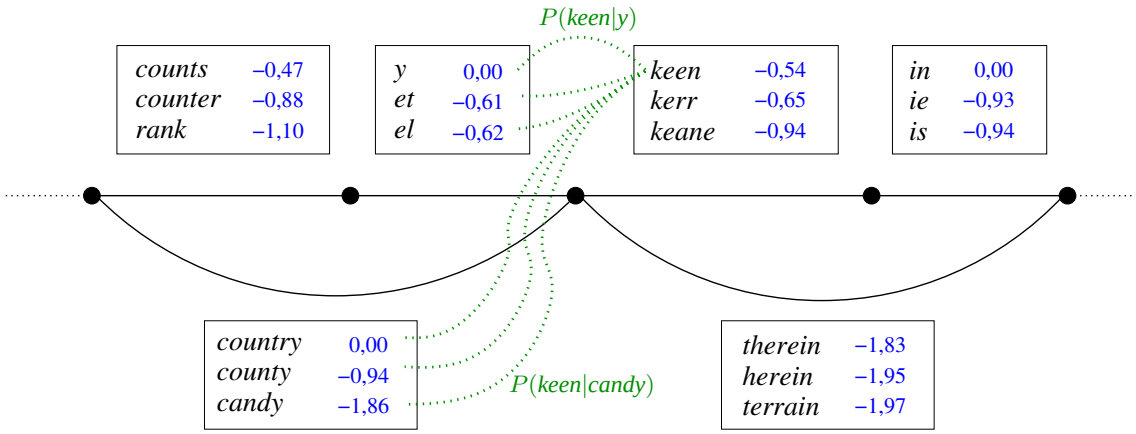


FIG. 4.2 – Exemple de sous-graphe de mots avec les probabilités donnés par le modèle de langage bigramme, pour le mot *keen* (en pointillé), ainsi que les mots candidats et leur score $score(s_i|w_i)$ associé, pour chaque arc.

Ainsi, pour chacun des mots candidats de ces listes, le meilleur chemin allant jusqu'à chacun des mots considérés est mémorisé ainsi que le score associé à ce chemin et dont le calcul est donné dans l'algorithme 4.1. Lorsque le dernier nœud du graphe est atteint et que le dernier mot du meilleur chemin est identifié, la séquence de mots qui mène jusqu'à ce mot peut-être reconstituée. Cette séquence correspond alors à la phrase résultat.

4.3.2 Extension aux modèles n-grammes d'ordre supérieur

Afin d'étendre l'utilisation des modèles de langage à des modèles d'ordre supérieur à 2, lors de la reconnaissance, il faut prendre en compte le fait que la probabilité d'un mot n'est plus conditionnée par le seul mot qui le précède mais par une séquence de $n-1$ mots (dans le cas des modèles n-grammes et n-classes). L'incorporation d'un tel modèle de langage nécessite donc une modification de l'espace de recherche, représenté par le graphe de mots [ONA97]. En effet, les mots qui apparaissent dans différents contextes doivent être dupliqués et associés avec leur contexte (composé des $n-2$ mots les précédant).

La figure 4.3 illustre la modification du graphe de mots, dans le cas où un modèle trigramme est utilisé. Chacun des mots candidats est dupliqué autant de fois qu'il a de contextes différents. Ainsi, dans cet exemple, le mot *keen* est dupliqué 6 fois et associé à chaque fois avec un de ses contextes. Le calcul du score du meilleur chemin menant à chacun des mots du graphe peut alors être effectué en utilisant l'algorithme de reconnaissance, en restreignant maintenant la sélection des mots prédécesseurs à ceux possédant un contexte commun avec le mot considéré. L'algorithme 4.1 est ainsi étendu par l'algorithme 4.2.

En reprenant l'exemple de la figure 4.3, pour calculer le meilleur chemin menant au mot *keen* d'historique associé *candy*, seuls les mots des arcs prédécesseurs dont le bigramme (constitué d'un mot et de son historique) se termine par le mot *candy* seront considérés dans ce calcul.

Algorithme 4.2 : Extension de l'algorithme 4.1, pour tous les modèles de langage.

Données :
G : le graphe de mots;
$score_{max}(w_i)$: le score du meilleur chemin allant jusqu'au mot w_i;
$pred_{max}(w_i)$: le mot meilleur prédécesseur sur le meilleur chemin allant jusqu'à w_i;
$score(s_i|w_i)$: le score du mot w_i;
$log(P(w_i|h_j w_j))$: la probabilité du n-gramme $h_j w_j w_i$;

Résultat :
$\widehat{W}$: la phrase résultat, qui a le score maximal dans le graphe de mots G;

début
  pour chaque *nœud* n_t *du graphe* G*, dans un ordre topologique* **faire**
    pour chaque *arc* a_t^{t+x} *successeur du nœud* n_t **faire**
      pour chaque *mot candidat* w_i *(avec son historique* h_i*) de l'arc* a_t^{t+x} **faire**
        pour chaque *arc* a_{t-y}^{t} *prédécesseur du nœud* n_t **faire**
          pour chaque *mot candidat* w_j *(avec son historique* h_j*) de l'arc* a_{t-y}^{t} **faire**
            si $h_i \subset h_j w_j$ **alors**
              $score = score_{max}(w_j) \; + \; (score(s_i|w_i) \; + \; \gamma \, log(P(w_i|h_j w_j)) \; + \; \delta)$;
              si $score > score_{max}(w_i)$ **alors**
                $score_{max}(w_i) = score$;
                $pred_{max}(w_i) = w_j$;
  $\widehat{W} = \widehat{w}_N$;
  $w_i = \widehat{w}_N$;
  tant que $w_i \neq \widehat{w}_1$ **faire**
    $w_i = pred_{max}(w_i)$;
    $\widehat{W} = w_i \,.\, \widehat{W}$;
fin

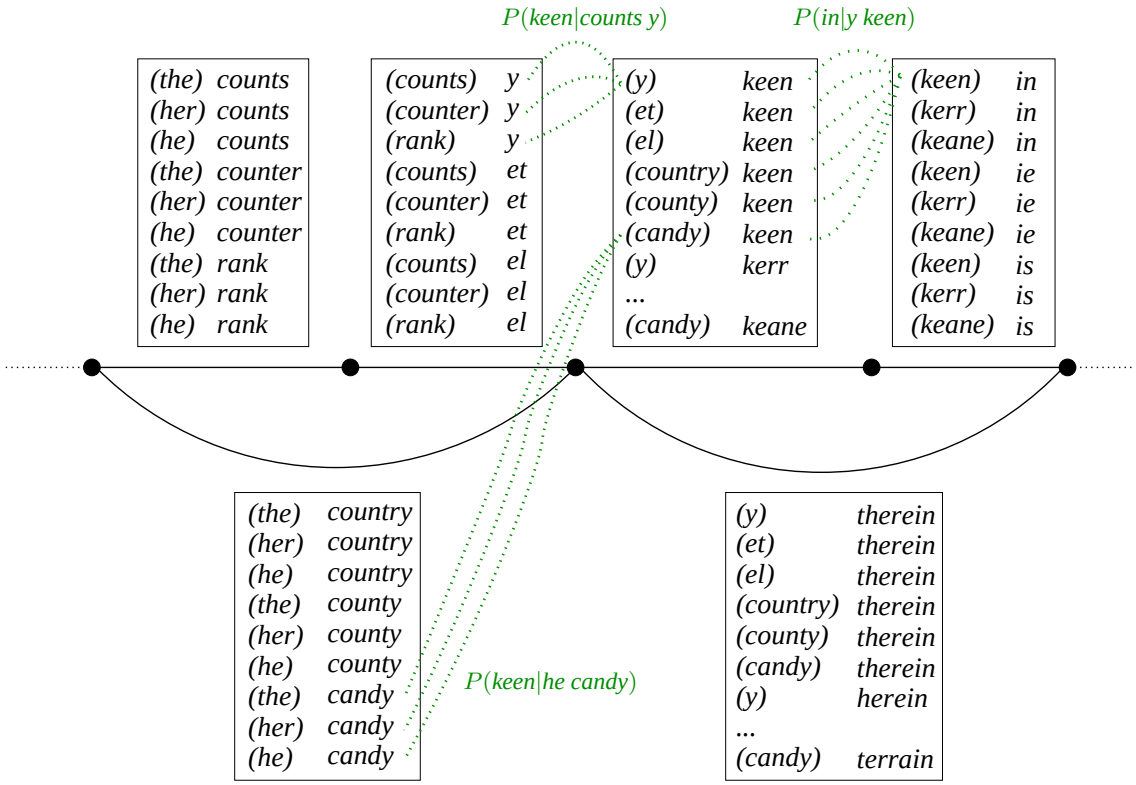


FIG. 4.3 – Extension du sous-graphe de mots de la figure 4.2, pour intégrer un modèle de langage trigramme.

L'extension donné par l'algorithme 4.2 permet aussi de considérer des modèles de langage d'ordre supérieur à 3.

Cette modification dans le graphe de mots entraîne une augmentation de sa taille puisque la taille des listes de mots candidats est fonction de l'ordre du modèle de langage. En effet, la taille d'une liste de mots l_i, en nombre de mots (avec leur historique associé), est alors $|l_i|_w = m^{n-1}$, avec m le nombre de mots candidats de la liste de mots initiale donnée par le système de reconnaissance de mots et n l'ordre du modèle de langage. Cette augmentation de la taille de l'espace de recherche entraîne également une augmentation du temps de calcul pour trouver le meilleur chemin dans le graphe, c'est-à-dire la phrase résultat.

La complexité de l'algorithme de reconnaissance de phrases, en terme de nombre de calculs de scores de chemins, est en $O(m^n T)$, avec m le nombre de mots candidats de la liste de mots initiale donné par le système de reconnaissance de mots, n l'ordre du modèle de langage et T le nombre de nœuds du graphe.

En pratique, les modèles de langage intégrés directement lors de la reconnaissance des phrases peuvent difficilement être d'ordre supérieur à 3.

4.4 Intégration et combinaison de modèles de langage en post-traitement

Comme nous l'avons vu dans la section précédente, il est trop coûteux en terme de temps de calcul (dû à la taille du graphe de mots) d'utiliser des modèles de langage d'ordre supérieur à 3, en les intégrant directement lors du parcours du graphe de mots. Afin de pouvoir utiliser ce type de modèle de langage, nous les intégrons lors d'une phase de post-traitement, comme illustré par la figure 4.4.

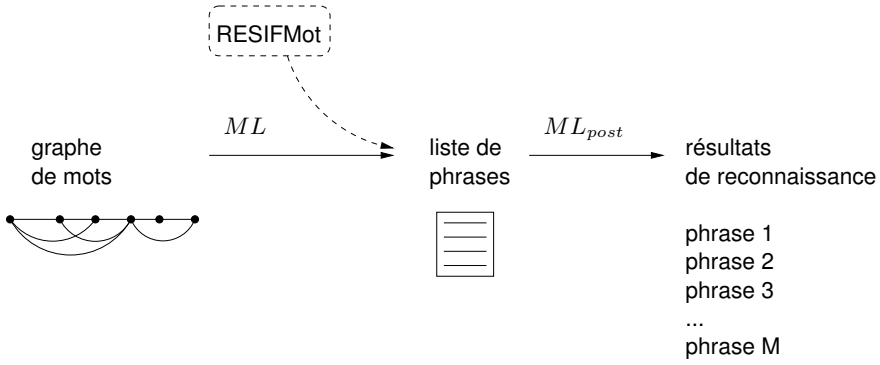


FIG. 4.4 – Principe de la combinaison des modèles de langage.

Un premier modèle de langage ML est utilisé pour générer une liste de M-meilleures hypothèses de phrases lors d'une reconnaissance, comme présenté dans la section 4.3.2. Le modèle de langage ML_{post} (d'ordre 4 ou 5 voire plus) permet ensuite de réordonner cette liste de phrases, en attribuant un nouveau score à chacune des phrases.

4.4.1 Génération de listes de M-meilleures phrases

Afin de générer une liste de M-meilleures hypothèses de phrases, au lieu d'une seule phrase résultat, l'algorithme 4.2 est modifié. Ce nouvel algorithme est donné par l'algorithme 4.3.

Pour limiter la taille de l'espace de recherche, la liste de phrases conservée pour chacun des mots candidats est réduite aux M hypothèses de phrases les plus vraisemblables. Seules ces hypothèses pourront ensuite être étendues pour continuer l'exploration du graphe de mots : cela explique l'étape d'élagage des listes. Cet algorithme s'appuie sur l'algorithme de recherche en faisceau (en anglais, *beam search*), qui est une approximation de l'algorithme de Viterbi.

Algorithme 4.3 : Extension de l'algorithme 4.2, pour générer une liste de M meilleures phrases résultats.

Données :
G : le graphe de mots;
$liste_{w_i}$: la liste des M meilleures hypothèses de phrases se terminant par w_i;
$màj(liste_{w_i}, w_i, liste_{w_j}, score)$: la fonction qui met à jour la liste $liste_{w_i}$ en y ajoutant les phrases de la liste $liste_{w_j}$, étendues avec le mot w_i et le score $score$;
$élague(liste_{w_i})$: la fonction qui élague la liste $liste_{w_i}$, pour ne conserver que les M meilleures hypothèses de phrases;

Résultat :
$\left\{\widehat{W}\right\}_M$: la liste des M meilleures phrases résultats, ayant les scores maximaux dans le graphe de mots G;

début
  pour chaque *nœud* n_t *du graphe* G*, dans un ordre topologique* **faire**
    pour chaque *arc* a_t^{t+x} *successeur du nœud* n_t **faire**
      pour chaque *mot candidat* w_i *(avec son historique* h_i*) de l'arc* a_t^{t+x} **faire**
        pour chaque *arc* a_{t-y}^{t} *prédécesseur du nœud* n_t **faire**
          pour chaque *mot candidat* w_j *(avec son historique* h_j*) de l'arc* a_{t-y}^{t} **faire**
            si $h_i \subset h_j w_j$ **alors**
              $score = (score(s_i|w_i) \ + \ \gamma \, log(P(w_i|h_j w_j)) \ + \ \delta)$;
              $màj(liste_{w_i}, w_i, liste_{w_j}, score)$;
        $élague(liste_{w_i})$;
  $\left\{\widehat{W}\right\}_M = liste_{\widehat{w}_N}$;
fin

4.4.2 Combinaison avec un modèle de langage de type n-gramme

Un nouveau score est associé à chacune des hypothèses de phrases $\widehat{W}$ de la liste $\left\{\widehat{W}\right\}_M$, en combinant les probabilités des modèles de langage ML et ML_{post}, par :

$$\widehat{W} = \arg\max_{W} \sum_{i=1}^{N} score(s_i|w_i) + \gamma\, log(P(w_i|\phi(h_i))) + \gamma_{post}\, log(P_{post}(w_i|\phi_{post}(h_i))) + \delta \tag{4.5}$$

avec $log(P(w_i|\phi(h_i)))$ la log-probabilité donnée par le modèle de langage ML au mot w_i associé à l'historique $\phi(h_i)$ (de poids correspondant γ) et $log(P_{post}(w_i|\phi_{post}(h_i)))$ la log-probabilité donnée par le modèle de langage ML_{post} au mot w_i associé à l'historique $\phi_{post}(h_i)$ (de poids correspondant γ_{post}). Les historiques $\phi(h_i)$ et $\phi_{post}(h_i)$ peuvent être différents et, en général, nous avons :

$$|\phi_{post}(h_i)|_w > |\phi(h_i)|_w \tag{4.6}$$

avec $|\phi(h_i)|_w$ (respectivement $|\phi_{post}(h_i)|_w$) la longueur de l'historique $\phi(h_i)$ (respectivement $\phi_{post}(h_i)$), en nombre de mots.

La liste des M-meilleures phrases est ensuite réordonnée en fonction de ce nouveau score et la phrase résultat correspond à celle qui se trouve en tête de la liste.

L'utilisation du premier modèle de langage ML, pour générer la liste des M-meilleures hypothèses de phrases, permet de sélectionner des phrases qui bénéficient déjà de l'apport d'un modèle de langage. La qualité de ces phrases est importante puisque les performances du post-traitement utilisant le modèle de langage ML_{post} dépendent des phrases à réordonner.

Pour réordonner les phrases, nous pourrions simplement utiliser la probabilité donnée par le modèle de langage ML_{post}. En combinant la probabilité du modèle ML_{post} à celle du modèle ML (ainsi qu'au score graphique de la phrase), cela permet aussi de tirer partie des spécificités des deux modèles de langage. En effet, le modèle de langage ML_{post} peut ne pas simplement être du même type que le modèle ML et avoir un ordre plus grand : il peut également s'appuyer sur d'autres informations. Par exemple, un modèle bigramme peut être utilisé pour générer la liste des M-meilleures phrases et cette liste sera ensuite réordonnée en utilisant un modèle n-classe à base de classes morpho-syntaxiques. Dans ce cas, des modifications doivent être apportées, ce que nous présentons dans la sous-section suivante.

4.4.3 Combinaison avec un modèle à base de classes morpho-syntaxiques

Les modèles n-classes à base de classes morpho-syntaxiques (voir chapitre 2, section 2.4.5) se basent sur les séquences de classes morpho-syntaxiques correspondant à une phrase pour estimer la probabilité de celle-ci (voir section 2.4.5.2). Cependant, plusieurs séquences de classes peuvent correspondre à une phrase donnée. Ainsi, pour calculer la probabilité de la phrase, il faut soit prendre en compte toutes les séquences de classes possibles, soit trouver la séquence de classes la plus probable.

C'est la deuxième approche que nous avons choisie ici.

L'approche utilisée pour trouver la meilleure séquence de classes morpho-syntaxiques a été proposée par [HGS07], avec qui nous avons travaillé en collaboration durant cette thèse.

Dans cette approche, un étiqueteur est utilisé pour désambiguïser la phrase et ainsi obtenir la séquence de classes morpho-syntaxiques la plus probable. La probabilité donnée par le modèle de langage à base de classes morpho-syntaxiques est ensuite combinée à celle du modèle de langage initial, en utilisant de nouveau l'équation 4.5. La liste des M-meilleures phrases est alors réordonnée en se basant sur le nouveau score de chacune des phrases.

Après avoir exposé deux approches permettant de tirer partie des informations apportées par des modèles de langage lors de la reconnaissance de phrases, nous présentons les résultats des expérimentations mises en œuvre pour étudier et comparer l'impact de modèles n-grammes et de modèles n-classes statistiques et morpho-syntaxiques.

4.5 Expérimentations

L'objectif des expérimentations que nous présentons est de trouver le ou les modèles de langage permettant d'obtenir les meilleures performances lors de la reconnaissance de phrases. Pour cela, nous étudions et comparons l'impact des modèles n-grammes et n-classes (à base de classes statistiques ou de classes morpho-syntaxiques), en termes de performances au niveau de la reconnaissance des phrases mais aussi en fonction de leur nombre de paramètres. Nous prenons également en compte l'influence de la segmentation des mots sur les résultats obtenus.

Nous étudions et comparons tout d'abord l'impact de modèles n-grammes et de modèles n-classes intégrés directement lors de l'exploration du graphe de mots. Nous présentons ensuite des résultats sur la combinaison de modèles de langage, en les utilisant pour réordonner des listes de phrases. Nous introduisons tout d'abord les corpus que nous utilisons pour construire les modèles de langage étudiés ainsi que des indicateurs permettant de comparer ces modèles, indépendamment de leur utilisation pour la reconnaissance.

4.5.1 Cadre expérimental

4.5.1.1 Bases linguistiques

Corpus Nous utilisons deux corpus de langue anglaise, pour construire les modèles de langage étudiés : les corpus $Susanne$ [Sam95] et $Brown$ [FK79]. Le corpus $Susanne$ correspond en fait à un sous-ensemble de phrases du corpus $Brown$. L'utilisation de ces deux corpus permet d'évaluer l'impact de la taille du corpus sur l'apprentissage des modèles de langage.

La première étape, pour utiliser un corpus, est une étape de nettoyage. Durant cette étape, les abréviations sont remplacées par les mots complets qu'elles abrègent. De plus, différentes balises d'indication de fonte ou d'autres informations ainsi que les signes de ponctuation sont supprimés. Enfin, chacune des phrases présentes est entourée des marqueurs de début et fin de phrase, <*s*> et </*s*>.

Nous découpons ensuite chacun de ces corpus en un corpus d'apprentissage (pour créer les modèles de langage) et un corpus de test, dont un sous-ensemble correspond aux phrases

utilisées pour la saisie des données manuscrites. La table 4.1 présente les caractéristiques de ces différents sous-corpus.

TAB. 4.1 – Caractéristiques des corpus d'apprentissage et de test, créés à partir des corpus $Susanne$ et $Brown$.

	Corpus $Susanne$		Corpus $Brown$		
Statistiques	$Susanne_{app}$	$Susanne_{test}$	$Brown_{app}$	$Brown_{test}$	$Brown_{saisie}$
Nombre de phrases	5 645	1 250	46 836	6 118	2 598
Nombre de mots	113 025	16 435	900 109	102 567	34 563

Nous avons de choisi d'utiliser un lexique fermé, c'est-à-dire que tous les mots du corpus de test sont présents dans le lexique utilisé (d'autres mots complètent également le lexique). Il alors faut que tous les mots des phrases saisies (appartenant à $Brown_{saisie}$) aient leurs probabilités estimées dans les modèles de langage construits et donc qu'ils soient présents dans les corpus d'apprentissage. Les phrases des corpus de test sont ainsi sélectionnées afin que chacun des mots de ces phrases apparaissent au moins une fois dans le corpus d'apprentissage correspondant. Les corpus $Brown_{app}$ et $Brown_{test}$ sont alors créés afin d'avoir :

$$Susanne_{app} \subset Brown_{app} \quad \text{et} \quad Susanne_{test} \subset Brown_{test}. \tag{4.7}$$

Lexique Le lexique utilisé est l'intersection des vocabulaires des corpus $Susanne$ et $Brown$: il correspond donc au vocabulaire du corpus $Susanne$, restreint aux mots composés uniquement des 26 lettres de l'alphabet (les nombres, par exemple, en sont donc exclus). Il comporte ainsi 13 748 mots.

Construction des modèles de langage Nous utilisons l'outil SRILM (*SRI Language Modeling Toolkit*) [Sto02] pour construire les modèles de langage étudiés. Cet outil permet de construire des modèles n-grammes utilisant différentes techniques de lissage des probabilités (voir chapitre 2, sur les différents modèles de langage) ainsi que des modèles n-classes à base de classes statistiques, à l'aide d'un outil permettant de construire automatiquement les classes en spécifiant le nombre de classes souhaitées.

Nous utilisons l'interpolation de Kneser-Ney modifiée (voir section 2.4.4.3) comme technique de lissage, pour chacun des modèles de langage créés ; cette technique a été présentée dans [CG98] comme étant la meilleure technique de lissage.

4.5.1.2 Mesures de performance

Perplexité (PPL_{ML}) Pour évaluer la qualité des modèles de langage, le premier indicateur utilisé est la *perplexité* (voir section 2.4.2).

Couverture des phrases saisies ($Couv_n^{ML}$) Pour évaluer les informations pouvant être apportées par les modèles de langage, nous mesurons la *couverture* de l'ensemble des phrases saisies, à un ordre donné. Dans le cas d'un modèle trigramme, par exemple, la couverture à l'ordre 3 représente le taux de trigrammes du corpus $Brown_{saisie}$ qui sont effectivement estimés par le modèle de langage à cet ordre (et donc sans utiliser de lissage vers l'ordre 2 ou 1). Cela permet, par exemple, de mesurer l'apport pouvant être obtenu en passant d'un modèle d'ordre 2 à un modèle d'ordre 3. Cette couverture est donnée par :

$$Couv_n^{ML} = \frac{\sum_{w_{i-n+1}^i \in V^n} \delta_{SAISIE \cap ML}(w_{i-n+1}^i)}{\sum_{w_{i-n+1}^i \in V^n} \delta_{SAISIE}(w_{i-n+1}^i)} \times 100 \tag{4.8}$$

avec V le vocabulaire utilisé, $SAISIE$ le corpus $Brown_{saisie}$, ML le modèle de langage d'ordre n considéré et δ la fonction de Kronecker définie par :

$$\delta_E(w_{i-n+1}^i) = \begin{cases} 1 & \text{si le } n\text{-gramme } w_{i-n+1}^i \text{ est présent dans l'ensemble } E \\ 0 & \text{sinon} \end{cases} \tag{4.9}$$

avec E l'ensemble contenant les n-grammes ou n-classes d'ordre n considéré : $SAISIE \cap ML$ correspond alors aux n-grammes des phrases de la base de test qui sont estimés par le modèle de langage et $SAISIE$ correspond aux n-grammes présents dans la base de test ($Brown_{saisie}$).

Nombre de paramètres (Nb_{param}^{ML}) Pour mesurer la taille des modèles de langage, nous calculons leur *nombre de paramètres*, qui correspond en fait aux probabilités estimées par le modèle de langage considéré, et qui est donné par l'équation 4.10, pour les modèles n-grammes :

$$Nb_{param}^{n-gramme} = \sum_{k=1}^{n} \sum_{w_{i-k+1}^i \in V^k} \delta_{ML}(w_{i-k+1}^i) \tag{4.10}$$

avec $\delta_{ML}(w_{i-k+1}^i)$ définie comme précédemment.

Dans le cas des modèles n-classes, des paramètres supplémentaires sont ajoutés, correspondant aux probabilités des mots dans leurs classes respectives et le nombre total de paramètres est alors donné par :

$$Nb_{param}^{n-classe} = \sum_{k=1}^{n} \sum_{c_{i-k+1}^i \in C^k} \delta_{ML}(c_{i-k+1}^i) + \sum_{w_i \in V} \sum_{c_j \in C} \delta_{c_j}(w_i) \tag{4.11}$$

avec $\delta_{ML}(c_{i-k+1}^i)$ définie par :

$$\delta_{ML}(c_{i-k+1}^i) = \begin{cases} 1 & \text{si le } n\text{-classe } c_{i-k+1}^i \text{ est estimé par le modèle de langage} \\ 0 & \text{sinon} \end{cases} \tag{4.12}$$

et $\delta_{c_j}(w_i)$ définie par :

$$\delta_{c_j}(w_i) = \begin{cases} 1 & \text{si } c_j \text{ est une classe de } w_i \\ 0 & \text{sinon.} \end{cases} \tag{4.13}$$

4.5.2 Intégration d'un modèle de langage pendant l'exploration du graphe

Dans cette section, nous présentons les résultats obtenus en étudiant l'intégration de modèles n-grammes et de modèles n-classes statistiques durant l'exploration du graphe de mots, pour la reconnaissance des phrases.

L'optimisation des différents paramètres (poids γ et δ et nombre de classes statistiques) est réalisée sur la base d'apprentissage $Saisie_{app}$ (voir section 3.4.1.1) et les résultats donnés sont obtenus sur la base de test $Saisie_{test}$, en utilisant ces paramètres optimaux.

4.5.2.1 Étude des modèles n-grammes

Nous étudions tout d'abord l'intégration de modèles unigrammes, bigrammes et trigrammes, en comparant les apports du passage de l'ordre 1 (modèles unigrammes) à l'ordre 2 (modèles bigrammes) puis de l'ordre 2 à l'ordre 3 (modèles trigrammes) ainsi que l'impact de la taille du corpus sur la qualité des modèles de langage. Nous revenons ensuite sur l'importance des poids γ et δ, utilisés pour pondérer, respectivement, l'impact du modèle de langage et des sur- et sous-segmentations.

Influence de la taille du corpus Dans la table 4.2, nous présentons des statistiques sur les modèles unigrammes, bigrammes et trigrammes, construits sur les corpus $Brown_{app}$ et $Susanne_{app}$, indépendamment de leur utilisation pour la reconnaissance de phrases.

TAB. 4.2 – Influence de la taille du corpus sur les modèles n-grammes.

Modèle de langage	Corpus $Susanne$			Corpus $Brown$		
	Nb_{param}^{ML}	$Couv_n^{ML}$	PPL_{ML}	Nb_{param}^{ML}	$Couv_n^{ML}$	PPL_{ML}
Unigramme	13 750	100,00 %	757	13 750	100,00 %	760
Bigramme	81 893	52,23 %	259	274 625	66,11 %	268
Trigramme	87 491	6,19 %	247	333 185	18,88 %	243

Le calcul de la couverture à l'ordre n ($Couv_n^{ML}$) permet essentiellement d'avoir une idée du bénéfice apporté par le passage des modèles bigrammes aux modèles trigrammes, en donnant le pourcentage de n-grammes de l'ordre n qui apparaissent dans le corpus $Brown_{saisie}$ et qui sont aussi estimés dans le modèle de langage. La couverture des modèles unigrammes est de 100 % puisque le corpus $Brown_{saisie}$ a été construit de façon à ce que tous les mots des phrases appartiennent au lexique, c'est-à-dire qu'ils ont tous une probabilité unigramme.

En passant du modèle bigramme au modèle trigramme (estimés sur le corpus $Brown$), l'augmentation du taux de reconnaissance sur les mots risque d'être limitée car seuls 18,88 % des trigrammes sont estimés dans le modèle trigramme correspondant et pourront donc tirer partie de ce passage à l'ordre supérieur. La couverture des trigrammes est encore plus faible pour le modèle trigramme construit sur le corpus $Susanne$.

La couverture des bigrammes est relativement bonne et son augmentation absolue, en passant du corpus $Susanne$ au corpus $Brown$, est à peu près la même que dans le cas des tri-

grammes, soit une augmentation d'environ 13 %. L'augmentation de la couverture (pour les bigrammes et les trigrammes), en passant du corpus $Susanne$ au corpus $Brown$, aurait pu être meilleure puisque le nombre de paramètres (Nb^{ML}_{param}) des modèles créés sur le corpus $Brown$ est environ 3 fois supérieur à celui des modèles créés sur le corpus $Susanne$.

Le calcul de la perplexité (PPL_{ML}) et de la couverture aux ordres supérieurs est réalisé en prenant, comme corpus de test, le corpus $Brown_{saisie}$ (les phrases manuscrites des bases d'apprentissage et de test sont saisies à partir de ces phrases). Les perplexités des modèles n-grammes construits sur les corpus $Susanne$ et $Brown$ sont relativement proches les unes des autres, surtout en ce qui concerne les modèles unigrammes et les modèles trigrammes.

Nous considérons maintenant les taux de reconnaissance obtenus avec les modèles bigramme et trigramme, présentés dans la table 4.2.

TAB. 4.3 – Taux de reconnaissance obtenus en intégrant des modèles n-grammes.

Extraction des mots	Modèle de langage	Corpus $Susanne$	Corpus $Brown$
Manuelle	Aucun	79,27 %	
	Unigramme	84,08 %	84,17 %
	Bigramme	87,66 %	88,48 %
	Trigramme	88,10 %	88,87 %
Automatique	Aucun	71,44 %	
	Unigramme	81,25 %	81,42 %
	Bigramme	85,66 %	86,51 %
	Trigramme	86,06 %	86,86 %

Les résultats obtenus, lorsque la segmentation est manuelle, permettent de mesurer l'impact des modèles de langage, indépendamment du biais pouvant être dû à de mauvaises extractions de mots. Nous constatons tout d'abord l'amélioration apportée par un modèle de langage puisque l'utilisation ne serait-ce que d'un modèle unigramme (construit sur le corpus $Brown$) permet de réduire le taux d'erreur sur les mots de 23,64 %, relativement à une reconnaissance de phrases sans utilisation de modèle de langage. L'utilisation de modèles de langage d'ordre supérieur permet de diminuer encore le taux d'erreur sur les mots. En effet, l'intégration d'un modèle de langage bigramme (construit sur le corpus $Brown$) permet de réduire le taux d'erreur sur les mots de 44,43 %, relativement à une reconnaissance de phrases sans utilisation de modèle de langage, alors que l'intégration d'un modèle trigramme (construit aussi sur le corpus $Brown$) permet de diminuer ce taux d'erreur de 46,31 %.

Nous observons également l'importance de la taille du corpus utilisé pour l'apprentissage des modèles de langage puisque les taux de reconnaissance obtenus avec les modèles n-grammes construits sur le corpus $Brown$ sont environ 0,8 % au-dessus de ceux obtenus avec les modèles de langage construits sur le corpus $Susanne$, excepté pour les modèles unigrammes où les taux de reconnaissance obtenus sont très proches.

En considérant maintenant les résultats obtenus lorsque les mots des phrases sont extraits automatiquement, les mêmes conclusions peuvent être tirées même si l'augmentation du taux de reconnaissance est encore plus forte, par rapport à une reconnaissance sans modèle de langage. Cela est dû au fait que l'ajout de connaissances linguistiques aide dans le choix du chemin passant par la segmentation correcte alors que, sans modèle de langage, le chemin choisi ne peut pas passer par une des hypothèses alternatives de segmentation qui ont été générées.

La réduction du taux d'erreur, relativement à une reconnaissance sans modèle de langage, est alors de 34,94 % en utilisant le modèle unigramme, de 52,77 % avec le modèle bigramme et de 53,99 % avec le modèle trigramme (ces modèles sont construits sur le corpus $Brown$).

Influence des poids linguistique et de segmentation (γ et δ) Dans l'équation 4.2, correspondant à la tâche de reconnaissance de phrases, deux paramètres sont utilisés, pour pondérer l'importance du modèle de langage vis-à-vis du système de reconnaissance et pour gérer les problèmes de sur- et de sous-segmentation. Nous revenons maintenant sur leur importance.

La figure 4.5 illustre tout d'abord l'évolution du taux de reconnaissance, sur la base $Saisie_{app}$, en fonction de la variation du poids linguistique γ, et en utilisant un modèle bigramme, dans le cas de l'extraction manuelle des mots.

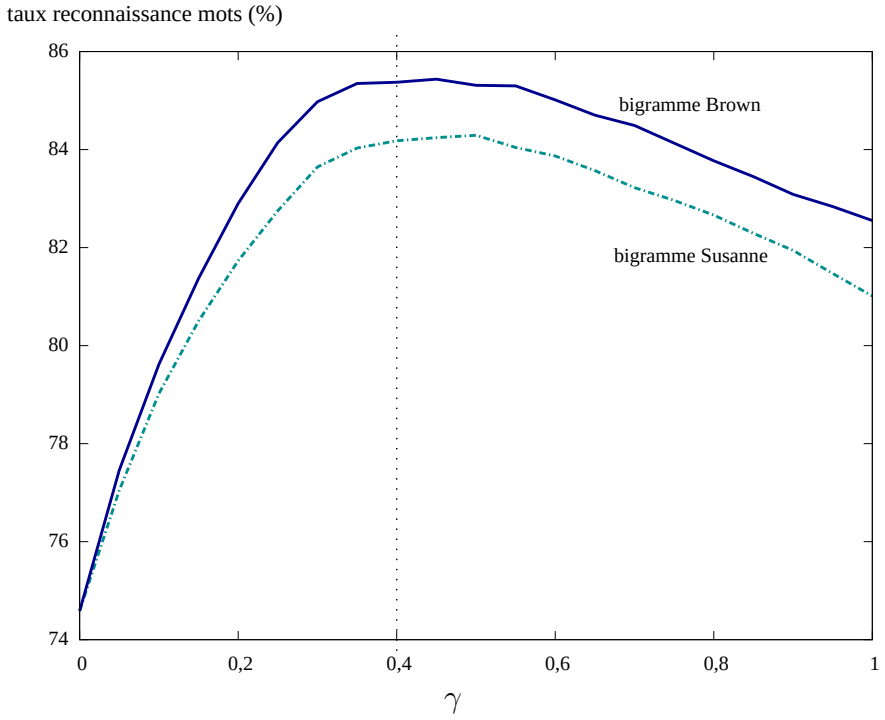


FIG. 4.5 – Évolution du taux de reconnaissance sur les mots extraits manuellement, pour les modèles bigrammes, en fonction du poids γ.

La valeur optimale de ce facteur γ est choisie en le faisant varier entre 0 et 1, par pas de 0,1 et en choisissant la valeur permettant de maximiser le taux de reconnaissance sur les mots de la base $Saisie_{app}$. Nous remarquons l'importance de ce facteur linguistique puisque, quand les informations fournies par le modèle de langage et par le système de reconnaissance ont la même importance ($\gamma = 1$), le taux de reconnaissance obtenu avec chacun des modèles bigrammes est environ 3 % en-dessous du taux obtenu avec une valeur optimale pour γ.

La figure 4.6 illustre, quant à elle, l'évolution du taux de reconnaissance sur les mots lorsque l'on fait varier les deux facteurs γ et δ, en utilisant le modèle bigramme construit sur le corpus $Brown$ et lorsque les mots sont extraits automatiquement.

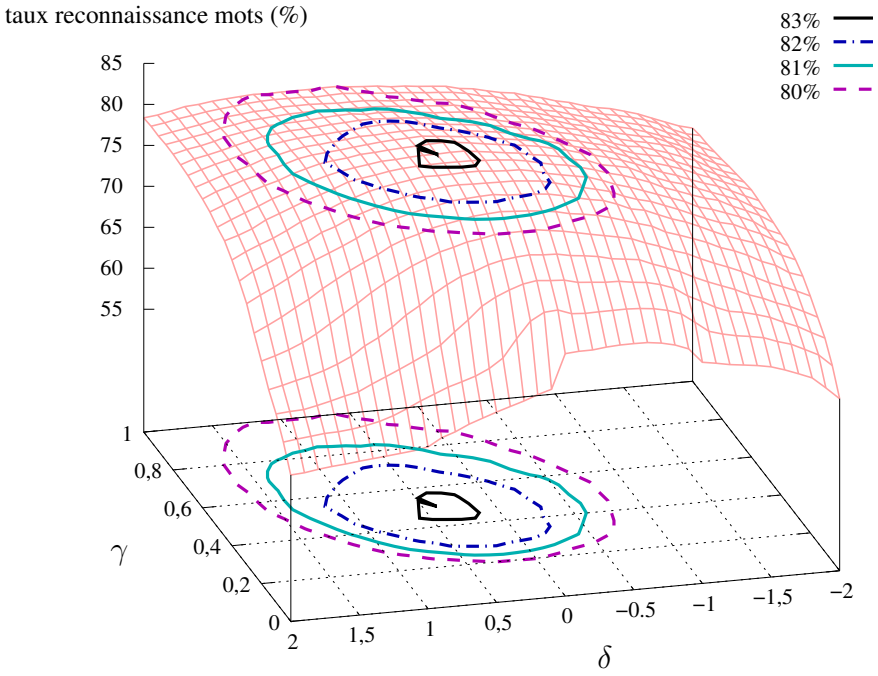


FIG. 4.6 – Évolution du taux de reconnaissance sur les mots extraits automatiquement, pour le modèle bigramme du corpus $Brown$, en fonction des poids γ et δ.

Ces deux facteurs sont optimisés conjointement, en faisant varier leur valeur par pas de 0,1, entre 0 et 1, pour γ et entre -2 et 2, pour δ. La valeur optimale de δ étant positive, les phrases longues seront favorisées et, comme présenté dans la sous-section 1.3.3, cela signifie que notre approche de segmentation automatique a tendance à sur-segmenter les mots.

Comme dans le cas de la segmentation manuelle, l'optimisation de ces deux poids est importante. En effet, la différence entre les taux de reconnaissance des mots, obtenus avec et sans optimisation, est de 4 % (sans optimisation, $\gamma = 1$ et $\delta = 0$).

4.5.2.2 Étude des modèles n-classes statistiques

Nous nous intéressons maintenant aux modèles n-classes à base de classes statistiques. Ces modèles de langage sont intéressants car ils sont plus compacts, en terme de place mémoire utilisée. De plus, ils permettent une meilleure généralisation des séquences de mots non rencontrées à l'apprentissage. Ils sont néanmoins moins précis que les modèles n-grammes.

Nous étudions ainsi le compromis entre la taille des modèles n-classes (en terme de nombre de paramètres) et la précision de ceux-ci (mesurée par la perplexité ou encore le taux de reconnaissance sur les mots).

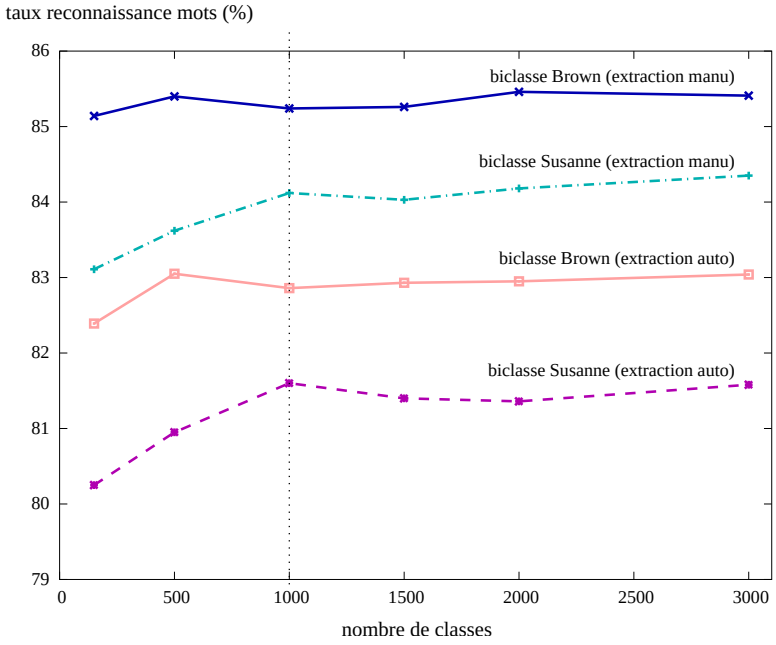
Choix du nombre de classes La première étape, pour construire un tel modèle de langage, est de créer les classes, en utilisant un des algorithmes présentés dans la section 2.4.5.2. L'algorithme disponible dans l'outil SRILM est l'algorithme de Brown (voir algorithme 2.2) : c'est celui que nous utilisons. Pour créer les classes, il faut fixer *a priori* le nombre de classes que l'on souhaite obtenir à l'issue de l'algorithme. Nous comparons l'impact de ce nombre de classes, en termes de taux de reconnaissance sur les mots et de perplexité, ce qui est illustré par la figure 4.7.

Dans la sous-figure 4.7(a), nous comparons les taux de reconnaissance obtenus sur les mots de la base $Saisie_{app}$ avec des modèles biclasses construits sur les corpus $Susanne$ et $Brown$, lorsque l'on fait varier le nombre de classes de ces modèles de langage.

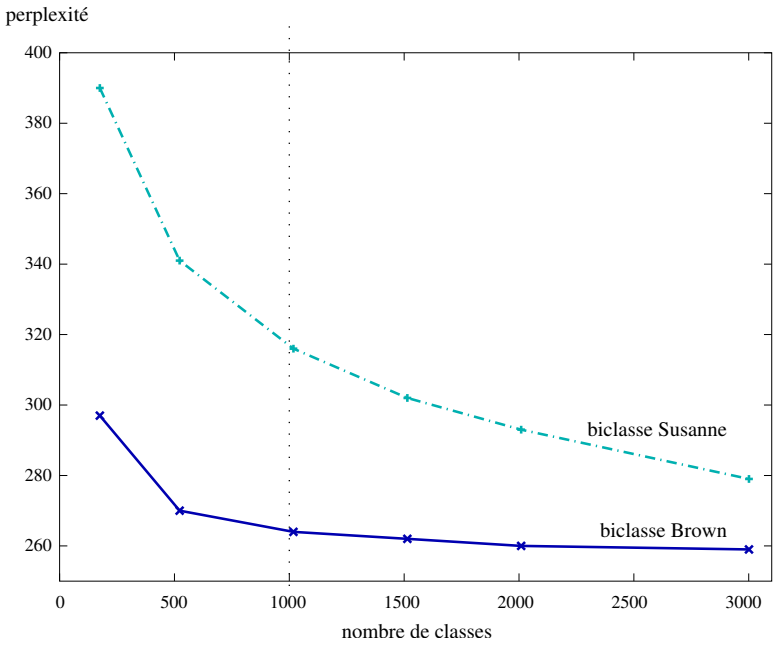
L'augmentation du taux de reconnaissance est plus forte pour le modèle biclasse créé sur le corpus $Susanne$ que pour celui créé sur le corpus $Brown$, lorsque l'on passe de 150 à 500 classes, quelle que soit la stratégie d'extraction des mots. Cela peut être dû à la plus petite taille du corpus $Susanne$: ainsi, en augmentant le nombre de classes, la précision du modèle augmente, ce qui est plus bénéfique pour les corpus de petite taille. À partir de 1 000 classes (voire même de 500, pour le modèle biclasse estimé sur le corpus $Brown$), le taux de reconnaissance se stabilise, aussi bien dans le cas de l'extraction manuelle des mots que dans celui de leur extraction automatique.

Dans la sous-figure 4.7(b), nous comparons aussi les modèles biclasses construits sur chacun des deux corpus, mais en terme de perplexité. La perplexité permet en effet d'estimer la qualité d'un modèle de langage, indépendamment de son utilisation dans une tâche de reconnaissance.

Nous constatons que la perplexité se stabilise aussi à partir de 1 000 classes, pour le modèle biclasse estimé sur le corpus $Brown$, alors qu'elle ne cesse de décroître pour le modèle estimé sur le corpus $Susanne$. Cela montre effectivement une corrélation entre la perplexité et le taux de reconnaissance, bien que le taux de reconnaissance permette de mieux choisir le modèle de langage le plus adéquat pour la reconnaissance. C'est pour cette raison que nous optimisons les taux de reconnaissance des mots sur la base $Saisie_{app}$, pour choisir le meilleur modèle de langage, au lieu d'utiliser leur perplexité. Néanmoins, le calcul du taux de reconnaissance est plus coûteux en termes de données (il faut disposer d'une base de données manuscrites, en plus de celle de test) et de coût de calcul (il faut effectuer la reconnaissance, en optimisant à chaque fois les poids présentés précédemment).



(a) Taux de reconnaissance sur les mots



(b) Perplexité

FIG. 4.7 – Évolution du taux de reconnaissance et de la perplexité, pour les modèles biclasses statistiques, en fonction du nombre de classes.

Influence de la taille du corpus et du nombre de classes Pour étudier l'influence de la taille du corpus ainsi que du nombre de classes considéré, nous nous intéressons aux modèles à base de 150 classes ainsi qu'à ceux à base de 1 000 classes. Au vu des résultats précédents, nous considérons que le modèle à base de 1 000 classes statistiques est le meilleur modèle n-classe.

Dans la table 4.4, nous présentons les statistiques sur les modèles de langage, indépendamment de leur utilisation pour la reconnaissance des phrases.

TAB. 4.4 – Influence de la taille du corpus sur les modèles n-classes.

Modèle de langage	Corpus $Susanne$			Corpus $Brown$		
	Nb_{param}^{ML}	$Couv_n^{ML}$	PPL_{ML}	Nb_{param}^{ML}	$Couv_n^{ML}$	PPL_{ML}
Biclasse$_{1000}$	38 701	54,45 %	316	144 198	77,05 %	264
Triclasse$_{1000}$	49 251	11,01 %	294	218 653	26,14 %	240
Biclasse$_{150}$	12 056	74,75 %	390	18 195	97,36 %	297
Triclasse$_{150}$	27 113	23,42 %	364	100 115	57,88 %	271

Le *nombre de paramètres* permet tout d'abord de connaître le nombre de probabilités estimées dans un modèle de langage et ainsi la place mémoire qu'il occupera.

Les modèles n-classes$_{150}$ comportent en moyenne 2 à 3 fois moins de paramètres que les modèles n-classes$_{1000}$. Cette différence est encore plus importante lorsque l'on compare les modèles biclasse$_{150}$ et biclasse$_{1000}$, construits tous les deux sur le corpus $Brown$. Cela est dû au fait qu'il y a plus de biclasses à estimer pour le modèle biclasse$_{1000}$. Ainsi, le passage du corpus $Susanne$ au corpus $Brown$ permet de rencontrer des biclasses qui n'étaient pas présents dans le corpus $Susanne$.

En passant des modèles biclasses aux modèles triclasses correspondants, le nombre de paramètres augmente assez significativement. La couverture à l'ordre 3 est aussi assez importante et meilleure que dans le cas des modèles trigrammes (nous revenons sur la comparaison des modèles n-grammes et n-classes, dans la sous-section 4.5.2.3). De plus, le passage du corpus $Susanne$ au corpus $Brown$ permet d'augmenter la couverture, ce qui profite surtout aux modèles triclasses.

La différence entre les perplexités obtenues par les modèles n-classes$_{150}$ et celles obtenues par les modèles n-classes$_{1000}$ est plus importante sur le corpus $Susanne$ (70 points, en moyenne) que sur le corpus $Brown$ (en moyenne, 30 points). Cela peut s'expliquer par le fait que les modèles n-classes$_{150}$ sont moins précis que les modèles n-classes$_{1000}$. Ainsi, le passage à un corpus plus grand permet d'affiner les probabilités de ces modèles, ce qui a un plus gros impact sur leur qualité. Cependant, lors du passage des modèles biclasses aux modèles triclasses, la taille du corpus a peu d'influence sur l'amélioration. En effet, l'amélioration est en moyenne de 20 points, quel que soit le corpus.

Nous comparons maintenant les taux de reconnaissance obtenus, pour voir si les mêmes conclusions peuvent être tirées : ces résultats sont présentés dans la table 4.5.

TAB. 4.5 – Taux de reconnaissance obtenus en utilisant des modèles n-classes.

Extraction des mots	Modèle de langage	Corpus $Susanne$	Corpus $Brown$
Manuelle	Aucun	79,27 %	
	Biclasse$_{1000}$	87,60 %	88,37 %
	Triclasse$_{1000}$	87,80 %	88,42 %
	Biclasse$_{150}$	86,70 %	88,26 %
	Triclasse$_{150}$	86,67 %	88,67 %
Automatique	Aucun	71,44 %	
	Biclasse$_{1000}$	85,68 %	86,47 %
	Triclasse$_{1000}$	85,87 %	86,50 %
	Biclasse$_{150}$	84,53 %	86,26 %
	Triclasse$_{150}$	84,66 %	86,47 %

Le passage du corpus $Susanne$ au corpus $Brown$ est bénéfique pour tous les modèles de langage puisque le taux de reconnaissance augmente en moyenne de 1,5 à 2 %, en utilisant un modèle n-classe estimé sur le corpus $Brown$ au lieu du modèle correspondant estimé sur le corpus $Susanne$. Cette amélioration absolue est indépendante de la stratégie de segmentation utilisée (manuelle ou automatique). En revanche, le passage des modèles biclasses aux modèles triclasses correspondants n'apporte pas d'amélioration significative.

En ce qui concerne le passage des modèles n-classes$_{150}$ aux modèles n-classes$_{1000}$, cette amélioration de la précision des modèles n-classes est plus importante sur le corpus $Susanne$. En effet, comme vu précédemment, le passage au corpus $Brown$ est plus bénéfique pour les modèles n-classes$_{150}$. Cela entraîne une différence moins importante entre les taux de reconnaissance obtenus par les deux types de modèles n-classes, construits sur le corpus $Brown$.

4.5.2.3 Bilan : comparaison des modèles n-grammes et n-classes

Nous revenons maintenant sur les résultats obtenus par les modèles n-grammes et par les modèles n-classes, en les comparant afin de trouver le modèle de langage qui assure le meilleur compromis entre ses performances et sa taille.

Les taux de reconnaissance obtenus avec les modèles biclasses$_{1000}$ sont très proches de ceux obtenus avec les modèles bigrammes, quel que soit le corpus utilisé. Cependant, les modèles trigrammes sont meilleurs que les modèles triclasses$_{1000}$ et cette différence est plus importante sur le corpus $Susanne$.

Ces résultats sont indépendants de la *stratégie de segmentation* utilisée. Néanmoins, le bénéfice des modèles de langage est plus important lorsque l'extraction des mots est automatique. En effet, bien que les taux de reconnaissance soient en moyenne 2 % inférieur à ceux obtenus avec une segmentation manuelle, la réduction relative moyenne du taux d'erreur est de 53 % (contre 44 % dans le cas de la segmentation manuelle), avec les modèles bigramme et

trigramme construits sur le corpus $Brown$. En fait, avec la segmentation automatique, le taux de reconnaissance initial est 8 % inférieur à celui obtenu avec la segmentation manuelle : les modèles de langage participent donc au choix de la meilleure segmentation.

En revanche, le nombre de paramètres des modèles n-classes$_{1000}$ est *plus de 2 fois inférieur* à celui des modèles n-grammes.

Les performances obtenues avec le modèle biclasse à base de 1 000 classes statistiques sont proches de celles réalisées avec le modèle bigramme, en terme de taux de reconnaissance, et le modèle biclasse contient moins de paramètres. C'est pour ces raisons que nous choisissons d'utiliser ce modèle *biclasse*$_{1000}$ (construit sur le corpus $Brown$), dans la suite des expérimentations sur la combinaison de modèles de langage.

4.5.3 Combinaison de modèles de langage en post-traitement

Nous étudions maintenant les résultats obtenus en combinant des modèles n-grammes ou des modèles n-classes (à base de classes statistiques ou morpho-syntaxiques) avec le modèle biclasse$_{1000}$, pour réordonner des listes d'hypothèses de phrases.

4.5.3.1 Choix du nombre de phrases

Le post-traitement est réalisé à partir de la liste des M-meilleures phrases, issue d'une première étape de reconnaissance utilisant le modèle de langage biclasse$_{1000}$. Le nombre de phrases ainsi considérées lors du post-traitement a une influence sur le taux de reconnaissance obtenu à l'issue de ce post-traitement : si trop peu de phrases sont prises en compte, l'intérêt de l'étape de post-traitement est limité mais, si trop de phrases sont prises en compte, elles risquent d'apporter du bruit et augmentent aussi le temps de post-traitement global.

La figure 4.8 montre l'évolution du taux de présence des mots à reconnaître, sur la base $Saisie_{app}$, en fonction du nombre de phrases considérées.

Le taux de présence des mots à reconnaître donne le pourcentage de mots à reconnaître qui sont présents dans au moins une des phrases de la liste, en ne considérant que les k premières phrases. Par exemple, en considérant 100 phrases, ce taux signifie que les mots à reconnaître se trouvent dans au moins une des 100 premières phrases de la liste des M-meilleures phrases.

Nous constatons qu'à partir d'un certain nombre de phrases, le taux de présence se stabilise, pour l'extraction manuelle comme pour l'extraction automatique des mots. Nous choisissons donc de restreindre la taille des listes considérées à leurs M*=200 meilleures phrases*.

4.5.3.2 Combinaison avec un modèle de langage de type n-gramme

Nous utilisons maintenant des modèles n-grammes et des modèles n-classes d'ordres 4 et 5 pour réordonner les 200 meilleures phrases, en combinant chacun de ces modèles avec le modèle biclasse$_{1000}$. Nous indiquons tout d'abord les statistiques sur ces modèles de langage avant de présenter les résultats obtenus avec ce post-traitement.

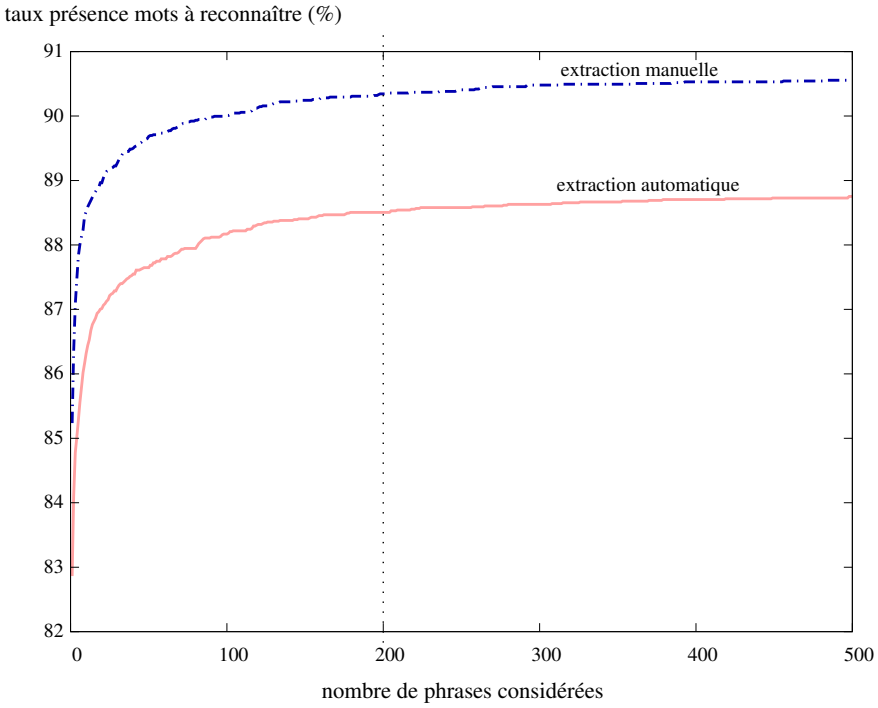


FIG. 4.8 – Évolution du taux de présence des mots à reconnaître, en fonction du nombre de phrases considérées.

La table 4.6 donne les statistiques sur les modèles de langage qui seront considérés pour le post-traitement.

TAB. 4.6 – Comparaison des modèles n-grammes et n-classes d'ordres 4 et 5.

Modèle de langage	Nb^{ML}_{param}	$Couv^{ML}_{n}$	PPL_{ML}
4-gramme	347 601	4,04 %	242
5-gramme	350 496	0,68 %	242
4-classe$_{1000}$	239 002	5,37 %	238
5-classe$_{1000}$	242 747	0,86 %	238

Nous constatons que les modèles d'ordre 5 n'apportent pas beaucoup plus d'informations que les modèles correspondants d'ordre 4. En effet, la perplexité est la même et la couverture, à l'ordre 5, des phrases de $Brown_{saisie}$ est très faible. De plus, le nombre de paramètres est aussi relativement stable.

La table 4.7 donne les taux de reconnaissance obtenus en combinant des modèles n-grammes et des modèles n-classes d'ordres 4 et 5, avec le modèle biclasse$_{1000}$. Nous rappelons également les taux de reconnaissance obtenus avec le modèle biclasse$_{1000}$ seul ainsi que ceux obtenus avec les modèles trigrammes et triclasses correspondants.

TAB. 4.7 – Combinaison du modèle biclasse$_{1000}$ avec des modèles n-grammes et n-classes d'ordres 4 et 5.

Extraction des mots	Modèle de langage	TRM
Manuelle	Biclasse$_{1000}$	88,37 %
	Trigramme	88,87 %
	Biclasse$_{1000}$ + 4-gramme	88,93 %
	Biclasse$_{1000}$ + 5-gramme	88,93 %
	Triclasse$_{1000}$	88,42 %
	Biclasse$_{1000}$ + 4-classe$_{1000}$	88,56 %
	Biclasse$_{1000}$ + 5-classe$_{1000}$	88,55 %
Automatique	Biclasse$_{1000}$	86,47 %
	Trigramme	86,86 %
	Biclasse$_{1000}$ + 4-gramme	87,06 %
	Biclasse$_{1000}$ + 5-gramme	87,06 %
	Triclasse$_{1000}$	86,50 %
	Biclasse$_{1000}$ + 4-classe$_{1000}$	86,81 %
	Biclasse$_{1000}$ + 5-classe$_{1000}$	86,73 %

L'utilisation des modèles 4-grammes et des modèles 4-classes permet d'améliorer le taux de reconnaissance, par rapport à celui obtenu avec le modèle biclasse (correspondant à une baisse relative du taux d'erreur de près de 5 %). En revanche, le passage à l'ordre 5 ne permet pas d'augmenter encore le taux de reconnaissance, ce qui pouvait déjà être supposé au vu de la faible couverture des 5-grammes et 5-classes sur $Brown_{saisie}$.

Néanmoins, le taux de reconnaissance obtenu avec le modèle 4-gramme (respectivement 4-classe) est relativement proche de celui obtenu lorsque le modèle trigramme (respectivement triclasse) est utilisé directement lors de la reconnaissance. La différence entre le taux de reconnaissance obtenu avec un modèle trigramme et le taux obtenu en combinant le modèle biclasse$_{1000}$ avec un modèle 4-gramme est un peu plus élevée lorsque l'extraction des mots est réalisée de manière automatique.

Il peut être intéressant de combiner le modèle biclasse$_{1000}$ avec le modèle 4-gramme. En effet, cela permet de faire une première passe de reconnaissance plus rapide, en utilisant le modèle biclasse directement pendant l'exploration du graphe de mots. Ensuite, le modèle 4-gramme est utilisé pour affiner le résultat de la reconnaissance, en réordonnant les phrases résultats. Un exemple d'application de cette approche peut être la saisie de phrases à la volée. Ainsi, une première reconnaissance est effectuée durant la saisie, puis cette reconnaissance est améliorée avec une étape de post-traitement.

Apport respectif des modèles de langage combinés Nous revenons sur la combinaison des deux modèles de langage et plus particulièrement sur leur impact relatif dans la décision finale. En effet, comme présenté dans l'équation 4.5, l'impact de chacun de ces modèles de langage est pondéré par un poids linguistique (γ et γ_{post}).

Lorsqu'un modèle n-gramme (d'ordre 4 ou 5) est combiné avec le modèle biclasse$_{1000}$, le poids associé au modèle biclasse$_{1000}$ est 0,2 et celui associé au modèle n-gramme est 0,3 (ces poids sont optimisés sur la base d'apprentissage, en prenant des valeurs entre 0 et 1). Comme aucun des deux modèles de langage n'a un poids prédominant, nous pouvons penser qu'ils doivent apporter des informations complémentaires.

Lorsqu'un modèle n-classe (d'ordre 4 ou 5) est utilisé, le poids de ce modèle est 0,4 alors que celui du modèle biclasse$_{1000}$ est 0,1. Cela est normal puisque ces deux modèles de langage modélisent les mêmes informations, le modèle 4-classe (ou 5-classe) étant plus précis que le modèle biclasse$_{1000}$.

4.5.3.3 Combinaison avec un modèle de langage morpho-syntaxique

Nous étudions maintenant la combinaison avec un modèle 7-classe à base de classes morpho-syntaxiques. Généralement, ces classes représentent des informations différentes de celles modélisées par des classes statistiques. Comme précédemment, ce modèle est utilisé en combinaison du modèle biclasse$_{1000}$, pour réordonner les listes de 200 meilleures hypothèses de phrases.

Dans la table 4.8, nous présentons les taux de reconnaissance obtenus : nous n'avons considéré ici que le cas où l'extraction des mots est réalisée de manière automatique.

TAB. 4.8 – Combinaison avec un modèle à base de classes morpho-syntaxiques.

Modèle de langage	TRM
Biclasse$_{1000}$	86,47 %
Biclasse$_{1000}$ + 7-classe$_{POS}$	87,11 %

Le modèle 7-classe$_{POS}$ construit contient 308 577 paramètres et est basé sur 152 classes morpho-syntaxiques (ou *POS*). Son utilisation, en combinaison avec le modèle biclasse$_{1000}$, permet d'améliorer le taux de reconnaissance, par rapport à celui obtenu lorsque le modèle biclasse$_{1000}$ est utilisé seul.

Les informations apportées par ce modèle de langage (de nature plutôt syntaxique) sont complémentaires à celles apportées par le premier. En effet, ces deux modèles de langage prennent part de manière relativement égale à la décision finale, puisque le poids du modèle biclasse$_{1000}$ est 0,3 et celui du modèle 7-classe$_{POS}$ est 0,2.

4.6 Bilan

Dans ce chapitre, nous avons présenté deux approches basées sur l'exploration du graphe de mots, pour améliorer la reconnaissance de phrases en utilisant des connaissances syntaxiques.

La première stratégie permet d'utiliser les modèles de langage directement lors de l'exploration du graphe de mots. Ils participent alors au choix de la segmentation de la phrase résultat.

Nous avons comparé les apport de modèles n-grammes et de modèles n-classes, en terme de performances mais aussi de place mémoire utilisée. Nous avons tout d'abord observé que l'utilisation d'un modèle unigramme permettait d'augmenter le taux de reconnaissance sur les mots, tout en occupant peu de place mémoire. Nous avons ensuite constaté qu'un modèle biclasse avec 1 000 classes statistiques permettait d'obtenir un taux de reconnaissance proche de celui obtenu par un modèle bigramme, tout en étant plus compact. Le passage aux modèles trigrammes ou triclasses n'a pas énormément amélioré les performances par rapport à l'augmentation de la taille du graphe de mots, nécessaire pour mémoriser le contexte des mots.

Nous avons aussi montré l'importance des poids γ et δ qui pondèrent l'impact du modèle de langage et celui des sur- et des sous-segmentations, respectivement. En effet, lorsque ces paramètres ne sont pas optimaux, les taux de reconnaissance obtenus sont 3 à 4 % inférieurs à ceux obtenus avec un réglage optimal des poids. La taille du corpus a aussi une influence sur les performances réalisées par les modèles de langage puisque les modèles de langage construits sur le corpus $Brown$ ont permis d'obtenir des taux de reconnaissance 1,5 à 2 % supérieurs à ceux obtenus avec les modèles créés sur le corpus $Susanne$.

Ainsi, le modèle de langage considéré comme le meilleur compromis entre place mémoire utilisée et performances obtenues est le modèle *biclasse*$_{1000}$, construit sur le corpus $Brown$. Son utilisation, durant la reconnaissance, a permis de diminuer de 43,9 % le taux d'erreur sur les mots, quand la segmentation est effectuée manuellement, et de 52,6 %, lorsque l'extraction des mots est automatique (relativement à une reconnaissance sans modèle de langage).

Afin d'utiliser des modèles de langage plus précis mais aussi de combiner des modèles de langage basés sur des informations complémentaires, nous avons proposé une approche pour réordonner une liste d'hypothèses de phrases. Pour ce faire, un modèle de langage de type bigramme (ou biclasse statistique) est utilisé pour générer les listes d'hypothèses de phrases, comme présenté précédemment. Puis, un autre modèle de langage (typiquement d'ordre supérieur à 3) est utilisé pour donner une nouvelle probabilité à chacune des phrases. Cette probabilité est combinée avec le score de chaque phrase et la liste de phrases est ensuite réordonnée.

L'utilisation d'un modèle 4-gramme, en combinaison du modèle biclasse$_{1000}$ a permis de réduire de nouveau le taux d'erreur sur les mots de 0,5 % (en absolu). L'utilisation d'un modèle 7-classe à base de classes morpho-syntaxiques (à la place du modèle 4-gramme) permet d'obtenir une réduction similaire du taux d'erreur.

De plus, nous avons observé que les deux modèles de langage combinés participaient de manière égale à la décision finale puisque leurs poids respectifs sont proches.

L'approche basée sur cette combinaison de modèles de langage permet d'utiliser un premier modèle de langage d'ordre 2 durant la reconnaissance puis d'affiner le résultat de la reconnaissance avec un modèle de langage plus précis, lors d'une phase de post-traitement syntaxique.

Deuxième partie

Exploitation de réseaux de confusion pour la reconnaissance de phrases

Introduction

Dans cette partie, nous nous appuyons cette fois sur une représentation originale issue de la reconnaissance automatique de la parole : les *réseaux de confusion* [Man00]. Cette représentation permet de mettre en évidence les confusions entre les mots, à chacune des positions de la phrase à reconnaître : ces zones sont appelées ensembles de confusion.

Cette représentation se base sur les probabilités *a posteriori* des mots. La probabilité *a posteriori* d'un mot correspond à la somme des probabilités de tous les chemins auxquels il appartient. Ces probabilités *a posteriori* peuvent alors être utilisées pour trouver la phrase résultat de la reconnaissance (aussi appelée hypothèse consensus), en cherchant à maximiser les probabilités *a posteriori* des mots de cette phrase. Les probabilités *a posteriori* peuvent également être utilisées comme indices de confiance sur la reconnaissance des mots, pour détecter d'éventuelles erreurs, par exemple.

Les travaux, que nous présentons dans cette partie, s'intéressent à ces deux cas d'utilisation.

Cette partie se compose de deux chapitres.

Dans le chapitre 5, nous présentons l'approche originale de reconnaissance basée sur les réseaux de confusion et qui s'appuie pour cela sur les probabilités *a posteriori* des mots, en cherchant à maximiser celles-ci. Nous expliquons tout d'abord le calcul des probabilités *a posteriori* des mots ainsi que la recherche de la meilleure phrase à partir de ces probabilités, dans le cadre de la reconnaissance automatique de la parole.

Nous expliquons ensuite les différences liées à la reconnaissance d'écriture manuscrite et nous proposons des modifications, pour pouvoir s'adapter à notre tâche de reconnaissance d'écriture. Nous comparons alors cette nouvelle tâche de reconnaissance avec l'approche MAP, plus classiquement utilisée en reconnaissance d'écriture.

Dans le chapitre 6, nous présentons une approche originale de détection d'erreurs potentielles au niveau des mots de la phrase résultat, obtenue par une phase de reconnaissance utilisant l'approche MAP. Cette détection s'appuie sur les probabilités *a posteriori* des mots résultats, utilisées comme indices de confiance. Nous proposons ensuite de corriger ces erreurs potentielles détectées, lorsque cette correction est possible. Lorsque la correction n'est pas possible, nous le signalons en rejetant le résultat de la reconnaissance.

Pour réaliser cette approche complète de détection et de correction des erreurs potentielles de reconnaissance, nous nous appuyons sur les probabilités *a posteriori* des mots ainsi que sur des classifieurs dédiés, de type *SVM*, et sur l'utilisation du rejet.

Nous comparons enfin les résultats obtenus avec cette approche, à ceux obtenus avec l'approche MAP. Nous comparons notamment ces résultats en terme de taux d'erreur sur les mots.

Chapitre 5

Utilisation de réseaux de confusion pour la reconnaissance de phrases

5.1 Introduction

Dans le chapitre 4, nous avons utilisé une méthode pour la reconnaissance de phrases, basée sur la maximisation de la probabilité de la phrase résultat. Cependant, le but de la reconnaissance de phrases est plutôt de maximiser le nombre de mots reconnus, en tirant partie des connaissances linguistiques sur la phrase.

Une nouvelle approche a été introduite pour la reconnaissance automatique de la parole ; elle est basée sur la minimisation du taux d'erreur sur les mots. Elle s'appuie, pour cela, sur une représentation des hypothèses de phrases sous la forme d'un *réseau de confusion*.

La nouvelle tâche de reconnaissance des phrases est donnée par la figure 5.1.

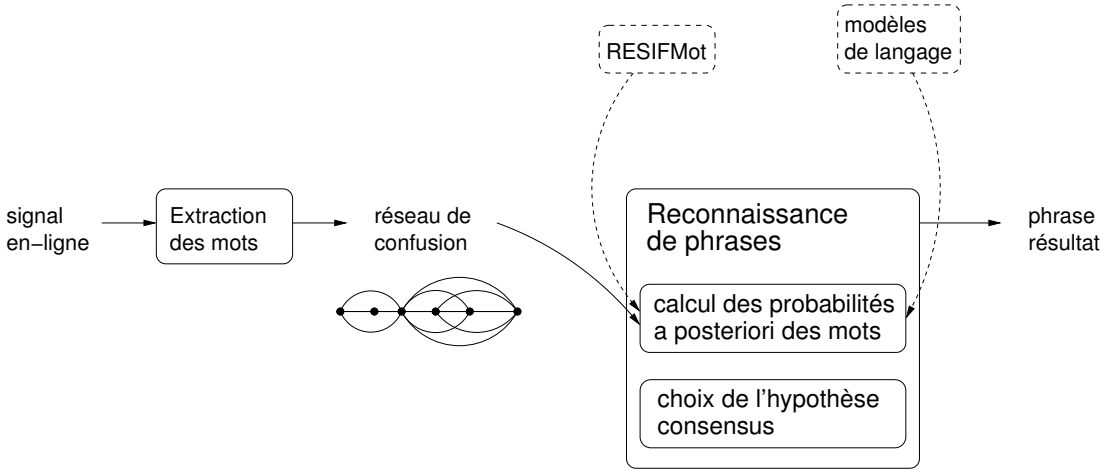


FIG. 5.1 – Principe de la reconnaissance de phrases, basée sur un réseau de confusion.

À partir du graphe de mots construit à l'issue de l'extraction des mots de la phrase, les probabilités *a posteriori* de chacun des mots de ce graphe sont calculées. Ces probabilités constituent la base de la représentation sous forme de réseau de confusion. Elles sont notamment utilisées pour trouver la meilleure hypothèse de phrase, aussi appelée *hypothèse consensus*.

Dans ce chapitre, nous présentons notre approche de reconnaissance de phrases manuscrites, exploitant une représentation des phrases sous la forme d'un réseau de confusion.

Dans la section 5.2, nous introduisons l'état de l'art sur la reconnaissance de phrases basée sur des réseaux de confusion, en détaillant les différentes étapes de la construction et de l'exploitation de ces réseaux. Dans la section 5.3, nous proposons ensuite des modifications, à la fois sur la construction et sur l'exploitation des réseaux de confusion, afin d'adapter cette approche pour notre tâche de reconnaissance de phrases manuscrites. Enfin, dans la section 5.4, nous présentons les résultats des expérimentations menées.

5.2 État de l'art sur les réseaux de confusion

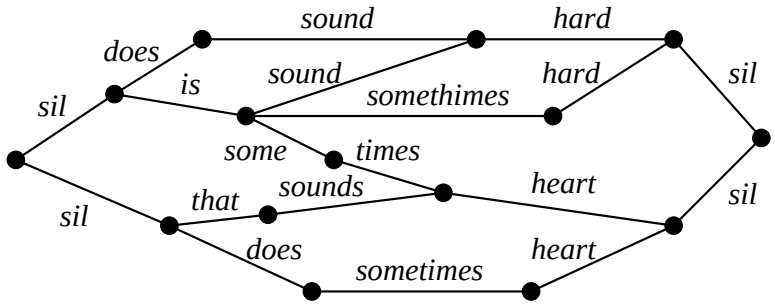
Dans cette section, nous présentons le principe général de la reconnaissance de phrases basée sur des réseaux de confusion, puis nous détaillons les différentes étapes de la construction d'un tel réseau de confusion. Nous finissons par donner les limites de cette approche, en indiquant également une extension proposée pour l'améliorer.

5.2.1 Principe général

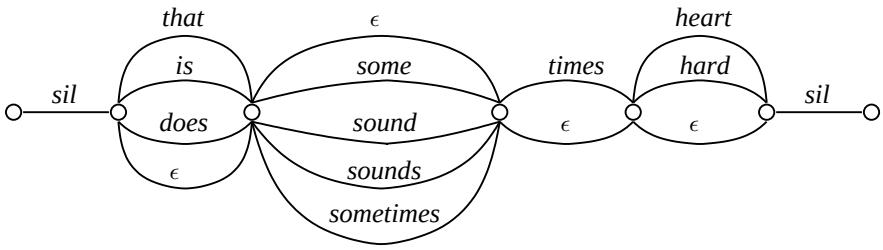
Dans [Man00], une nouvelle approche pour la reconnaissance de phrases a été proposée en reconnaissance automatique de la parole. Contrairement à l'approche MAP (voir chapitre 1, section 1.3), couramment employée pour la reconnaissance de phrases et qui vise à maximiser la probabilité de la phrase résultat, cette nouvelle approche est basée sur la minimisation du taux d'erreur sur les mots. Elle est alors plus proche du but de la reconnaissance de phrases qui n'est pas d'obtenir la phrase ayant la probabilité la plus élevée mais plutôt celle qui contient le plus de mots correctement reconnus (ou encore de minimiser les erreurs de substitution, de suppression et d'insertion).

Dans cette approche, l'ensemble des hypothèses de phrases est représenté par un *réseau de confusion* (voir figure 5.2(b)). Les nœuds d'un tel réseau représentent des classes d'équivalence, appelées *ensembles de confusion*, qui correspondent à des confusions entre des hypothèses de mots, à une position donnée de la phrase à reconnaître. Des nœuds adjacents sont alors reliés par autant d'arcs que d'hypothèses de mots. Par exemple, dans la figure 5.2(b), les mots *that*, *is*, *does* et ϵ (correspondant à une suppression) appartiennent au même ensemble de confusion.

À chacun des mots du réseau de confusion est associée une *probabilité a posteriori*, qui correspond à la somme de tous les chemins du graphe initial auxquels ce mot appartient. Ces probabilités *a posteriori* sont ensuite utilisées pour retrouver la meilleure hypothèse de phrase.



(a) Graphe de mots



(b) Réseau de confusion

FIG. 5.2 – Graphe de mots et réseau de confusion correspondant.

Un réseau de confusion est en fait obtenu à partir d'un graphe de mots. Dans la sous-section suivante, nous présentons les différentes étapes de la construction d'un réseau de confusion, à partir d'un graphe de mots, ainsi que le choix de la meilleure phrase, à partir des probabilités *a posteriori* des mots du réseau de confusion.

5.2.2 Construction du réseau de confusion

L'algorithme 5.1 présente les différentes étapes pour construire un réseau de confusion (voir figure 5.2(b)) à partir d'un graphe de mots (voir figure 5.2(a)) et pour retrouver la phrase qui minimise le taux d'erreur sur les mots.

Nous présentons ici l'algorithme utilisé en reconnaissance automatique de la parole. Nous verrons, dans la section 5.3, les modifications apportées pour la reconnaissance d'écriture.

Les étapes de cet algorithme sont détaillées dans les sous-sections suivantes.

5.2.2.1 Calcul des probabilités *a posteriori* des arcs

Les probabilités *a posteriori* de chacun des arcs du graphe de mots sont tout d'abord calculées. La probabilité du k^e arc associé au mot w_i correspond à la somme des probabilités des chemins qui passent par cet arc. Elle peut être calculée efficacement en utilisant

Algorithme 5.1 : Algorithme de construction du réseau de confusion et de reconnaissance de phrase [Man00].

Données :
G : le graphe de mots;

Résultat :
C_G : le réseau de confusion construit à partir du graphe G;
$W_{consensus}$: l'hypothèse consensus, correspondant à la phrase résultat;

début
§ 5.2.2.1 calcul des probabilités *a posteriori* des arcs du graphe G;
§ 5.2.2.2 création des ensembles de confusion;
§ 5.2.2.3 calcul des probabilités *a posteriori* des mots des ensembles de confusion;
§ 5.2.2.4 choix de l'hypothèse consensus $W_{consensus}$;
fin

l'algorithme *forward-backward* [BPSW70] et est donnée par l'équation suivante :

$$P_{post}(w_i^{(k)}) = \frac{\alpha(n_{t-1})\ P(w_i|w_j)^{\gamma}\ P(s_i|w_i)^{\zeta}\ \beta(n_t)}{\alpha(n_T)} \tag{5.1}$$

avec $w_i^{(k)}$ le k^e arc correspondant au mot w_i, $\alpha(n_{t-1})$ la probabilité *forward* du nœud n_{t-1}, $\beta(n_t)$ la probabilité *backward* du nœud n_t, $P(w_i|w_j)$ la probabilité donnée par le modèle de langage au bigramme $w_j w_i$ (de pondération associée γ) et $P(s_i|w_i)$ la probabilité acoustique donnée par le système de reconnaissance de parole au mot w_i, à partir du signal acoustique s_i (de pondération associée ζ).

La probabilité *forward* d'un nœud correspond à la somme des probabilités de tous les chemins entre le nœud initial et ce nœud, alors que la probabilité *backward* correspond à la somme des probabilités de tous les chemins entre ce nœud et le nœud final. La probabilité $P_{post}(w_i^{(k)})$ est normalisée par la somme des probabilités de tous les chemins du graphe de mots, représentée par $\alpha(n_T)$ (où T est le nombre de nœuds du graphe).

5.2.2.2 Création des ensembles de confusion

La création des ensembles de confusion est réalisée en alignant les hypothèses du graphe de mots initial.

Initialisation Tout d'abord, les ensembles de confusion sont initialisés de telle sorte que chacun ne contienne que les arcs ayant des temps de début et de fin ainsi que des étiquettes de mots identiques.

Regroupement intra-mot Cette étape a pour but de fusionner les ensembles de confusion qui ont la même étiquette de mot mais des temps de début et/ou de fin qui se chevauchent. Une

des difficultés de cette étape réside dans le fait qu'un même mot peut apparaître plusieurs fois dans la phrase à reconnaître. En effet, il ne faut pas regrouper des arcs ayant la même étiquette de mot mais correspondant à des positions différentes dans la phrase à reconnaître. La mesure de similarité, utilisée pour ce regroupement, s'appuie sur le chevauchement temporel des arcs.

Regroupement inter-mot Cette étape permet de regrouper les ensembles de confusion correspondant à des mots différents, c'est-à-dire qui sont des hypothèses de reconnaissance de mot différentes, à la même position donnée de la phrase à reconnaître. Pour cela, le calcul de similarité entre arcs se base également sur un chevauchement temporel ainsi que sur une similarité phonétique entre les mots qu'ils représentent.

Élagage des arcs Avant la création de ces classes d'équivalence, une étape d'élagage peut être utilisée afin de supprimer les arcs dont la probabilité *a posteriori* est trop faible. La probabilité de ces arcs est négligeable mais ils peuvent avoir une mauvaise incidence sur l'alignement des hypothèses de phrases. Leur suppression permet ainsi d'obtenir un meilleur alignement des hypothèses, tout en diminuant également la taille du graphe.

5.2.2.3 Calcul des probabilités *a posteriori* des mots

Une fois les ensembles de confusion créés, la probabilité *a posteriori* de chacun des mots w_i peut être calculée, à partir de celles des arcs dont l'étiquette correspond au mot w_i :

$$P_{post}(w_i) = \sum_{k=1}^{K} P_{post}(w_i^{(k)}) \tag{5.2}$$

avec $P_{post}(w_i^{(k)})$ la probabilité *a posteriori* du k^e arc correspondant au mot w_i, calculée par l'équation 5.1.

5.2.2.4 Choix de l'hypothèse consensus

Le réseau de confusion est maintenant entièrement construit. L'hypothèse de phrase $W_{consensus}$ qui maximise le nombre de mots correctement reconnus peut alors être choisie : elle est appelée *hypothèse consensus*. Il a été montré dans [Man00] que l'hypothèse de phrase qui minimise le taux d'erreur sur les mots peut être obtenue, en choisissant dans chaque ensemble de confusion, le mot dont la probabilité *a posteriori* est la plus élevée :

$$W_{consensus} = w^1_{consensus}\, w^2_{consensus} \ldots w^L_{consensus} \tag{5.3}$$

avec L le nombre d'ensembles de confusion et $w^l_{consensus}$ le mot défini par :

$$w^l_{consensus} = \arg \max_{w_i \in C_l} P_{post}(w_i) \tag{5.4}$$

avec C_l le l^e ensemble de confusion.

5.2.3 Limitations de l'approche initiale

L'alignement des arcs est effectué en considérant qu'un arc ne peut être aligné qu'avec un seul autre arc. Cependant, si un arc est long, il peut être préférable de l'aligner avec deux arcs courts car il peut correspondre à une sous-segmentation (ou les deux arcs courts à une sur-segmentation). Par exemple, dans le réseau de confusion de la figure 5.2(b), le mot *sometimes* devrait plutôt être aligné avec les mots *some* et *times*, ce qui est plus cohérent au niveau de la similarité acoustique.

Pour prendre en compte ce cas, [XZ05] ont proposé une modification dans la création des ensembles de confusion, afin de pouvoir aligner un arc avec deux autres arcs. Ainsi, le graphe de mots donné à la figure 5.2(a) permet d'obtenir le réseau de confusion de la figure 5.3, où le mot *sometimes* est maintenant aligné correctement avec les autres arcs.

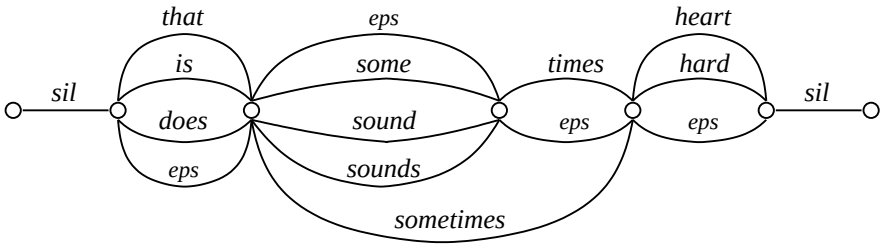


FIG. 5.3 – Réseau de confusion étendu.

Après avoir présenté l'état de l'art sur les réseaux de confusion, en reconnaissance automatique de la parole, nous décrivons maintenant leur utilisation pour la reconnaissance d'écriture ainsi que les modifications que nous avons apportées.

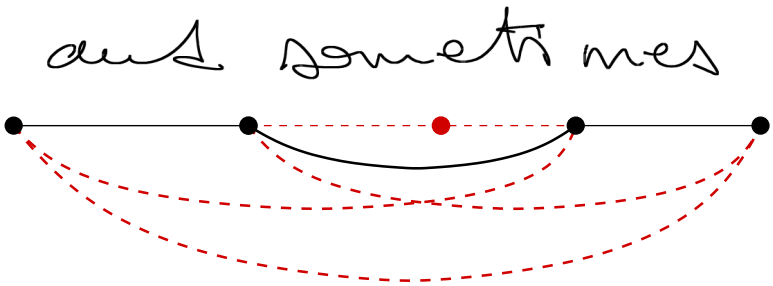
5.3 Extension pour la reconnaissance d'écriture manuscrite

Nous présentons les différences entre la reconnaissance de la parole et celle de l'écriture manuscrite, en ce qui concerne la représentation des hypothèses de phrases dans le graphe de mots. Ces différences entraînent des modifications dans le calcul des probabilités *a posteriori* des mots ainsi que dans celui de l'hypothèse consensus.

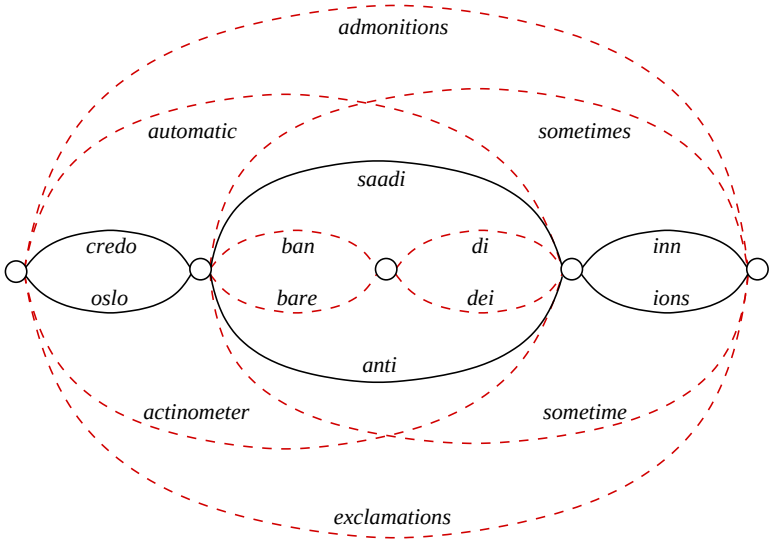
5.3.1 Différences liées à la reconnaissance d'écriture

Étant donné la structure de notre graphe de mots, l'étape de création des ensembles de confusion est immédiate. En effet, les listes de mots candidats associées à chacun des arcs (et données par notre système de reconnaissance de mots) correspondent aux ensembles de confusion. En réalité, c'est le cas lorsqu'il n'y a qu'une seule hypothèse de segmentation de la phrase : il n'existe alors qu'un seul arc entre deux nœuds consécutifs du graphe de mots.

Lorsque plusieurs hypothèses de segmentation sont présentes dans le graphe, nous obtenons un graphe de mots comme celui illustré par la figure 5.4(a) (en fait, seule une partie du graphe est donnée). Nous souhaitons alors obtenir l'ensemble de confusion correspondant à la figure 5.4(b).



(a) Partie d'un graphe de mots



(b) Ensemble de confusion correspondant

FIG. 5.4 – Exemple de partie d'un graphe de mots, obtenu en la reconnaissance d'écriture, et l'ensemble de confusion associé.

Dans cet ensemble de confusion, chacun des mots associés aux arcs correspond à un des mots candidats donnés par le système de reconnaissance de mots. Ces mots sont obtenus à partir du signal manuscrit correspondant aux arcs du graphe de mots (dans cet exemple, nous limitons à deux le nombre de réponses du système de reconnaissance de mots).

Afin de pouvoir considérer des ensembles de confusion comme ceux présentés par la figure 5.4(b), nous modifions le calcul des probabilités *a posteriori* des mots, en étendant les travaux présentés dans [XZ05] (voir aussi sous-section 5.2.3).

Dans les sous-sections suivantes, nous présentons tout d'abord les modifications apportées au niveau du calcul de la probabilité *a posteriori* des mots. Ces probabilités sont directement associées aux mots du graphe de mots puisque le graphe correspond au réseau de confusion, dans notre cas. Nous donnons ensuite les modifications que cela engendre, lors du calcul de l'hypothèse consensus.

5.3.2 Nouveau calcul des probabilités *a posteriori* des mots

Comme le réseau de confusion correspond au graphe de mots et que les ensembles de confusion s'obtiennent immédiatement (voir sous-section 5.3.1), les probabilités *a posteriori* sont directement associées aux mots du graphe. Comme précédemment, le calcul des probabilités *a posteriori* s'appuie sur l'algorithme *forward-backward*.

Nous donnons tout d'abord le nouveau calcul des probabilités *forward* et *backward*, sur les ensembles de confusion étendus, et permettant également de prendre en compte la longueur des chemins. Ces probabilités seront ensuite utilisées pour calculer les nouvelles probabilités *a posteriori* de chacun des mots du graphe.

5.3.2.1 Calcul des probabilités *forward* α

L'algorithme 5.2 donne le calcul de la probabilité *forward* de chacun des mots du graphe.

Le score $score(s_i|w_i)$ est normalisé par rapport aux scores minimal et maximal parmi ceux des mots ayant le même nombre de traits descendants que w_i. Comme ce score correspond essentiellement au score $score_{lexique}(w_i)$ du mot w_i, il correspond à la distance d'édition minimale pour obtenir ce mot (voir chapitre 1, section 1.4). Il est donc dépendant de la longueur du mot puisqu'un mot long nécessite généralement plus d'opérations d'édition qu'un mot court. D'où cette normalisation.

Pour rappel, la probabilité *forward* d'un mot w_i correspond à la somme des probabilités des chemins allant du début du graphe jusqu'au mot w_i considéré. La figure 5.5 indique les mots pris en compte dans le calcul de la probabilité *forward* du mot *straight*.

La probabilité *forward* d'un mot w_i est ainsi calculée à partir de celle de chacun des mots w_j, appartenant à une des listes de mots candidats d'un des arcs a_{t-y-x}^{t-y}. Comme, dans notre méthode de génération du graphe de mots, un nœud ne peut pas avoir plus de 3 arcs prédécesseurs et plus de 3 arcs successeurs (voir chapitre 3), les valeurs de x et y sont telles que $x, y \in \{1, 2, 3\}$. Néanmoins, comme l'algorithme présenté est générique, il peut s'appliquer aux cas où plus de 3 arcs successeurs ou prédécesseurs sont présents.

Les arcs a_{t-y-x}^{t-y} considérés sont les prédécesseurs du nœud n_{t-y}, qui correspond au nœud de début de l'arc a_{t-y}^{t} auquel appartient le mot w_i (*straight*, dans l'exemple de la figure 5.5).

Nous souhaitons également prendre en compte la longueur des chemins (calculée en nombre de mots), afin de ne pas défavoriser les chemins comportant beaucoup de mots. Pour cela, lors du calcul de la probabilité *a posteriori* du mot w_i, nous décomposons les probabilités *forward*

Algorithme 5.2 : Algorithme de calcul des probabilités *forward* des mots.

Données :
G : le graphe de mots;
$P(w_i|w_j)$: la probabilité du bigramme $w_j w_i$, donnée par le modèle de langage;
$score_{norm}(s_i|w_i)$: le score normalisé du mot w_i;

Résultat :
α : les probabilités *forward* des mots du graphe G;

début
  pour chaque *nœud* n_t *du graphe* G*, dans un ordre topologique* **faire**
    pour chaque *arc* a_{t-y}^{t} *prédécesseur du nœud* n_t **faire**
      pour chaque *mot candidat* w_i *de l'arc* a_{t-y}^{t} **faire**
        pour chaque *arc* a_{t-y-x}^{t-y} *prédécesseur du nœud* n_{t-y} **faire**
          pour chaque *mot candidat* w_j *de l'arc* a_{t-y-x}^{t-y} **faire**
            pour chaque *longueur de chemin* l_k *de* $\alpha[t-y][x][j]$ **faire**
              $\alpha[t][y][i][k+1] +=$
              $\alpha[t-y][x][j][k] * P(w_i|w_j)^{\gamma} * score_{norm}(s_i|w_i)^{\zeta}$;
fin

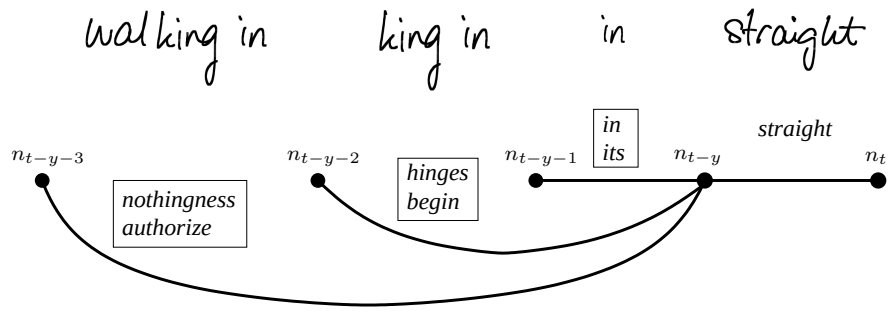


FIG. 5.5 – Mots pris en compte dans le calcul de la probabilité *forward* α du mot *straight*.

selon les différentes longueur de chemins allant du nœud de début du graphe jusqu'au nœud n_t pour lequel la probabilité *forward* α est calculée.

5.3.2.2 Calcul des probabilités *backward* β

L'algorithme 5.3 donne le calcul des probabilités *backward* des mots du graphe.

Le calcul s'effectue de manière similaire à celui des probabilités *forward*. Cependant, comme la probabilité *backward* d'un mot correspond à la somme des probabilités des chemins

Algorithme 5.3 : Algorithme de calcul des probabilités *backward* des mots.

Données :
G : le graphe de mots;
$P(w_i|w_j)$: la probabilité du bigramme $w_j w_i$, donnée par le modèle de langage;
$score_{norm}(s_i|w_i)$: le score normalisé du mot w_i;

Résultat :
β : les probabilités *backward* des mots du graphe G;

début
  pour chaque *nœud* n_t *du graphe* G*, dans un ordre topologique inverse* **faire**
    pour chaque *arc* a_{t-y}^{t} *prédécesseur du nœud* n_t **faire**
      pour chaque *mot candidat* w_i *de l'arc* a_{t-y}^{t} **faire**
        pour chaque *arc* a_t^{t+x} *prédécesseur du nœud* n_t **faire**
          pour chaque *mot candidat* w_j *de l'arc* a_t^{t+x} **faire**
            pour chaque *longueur de chemin* l_k *de* $\beta[t+x][x][j]$ **faire**
              $\beta[t][y][i][k+1] +=$
              $\beta[t+x][x][j][k] * P(w_j|w_i)^{\gamma} * score_{norm}(s_j|w_j)^{\zeta}$;
fin

allant de ce mot jusqu'à la fin du graphe, nous parcourons les nœuds du graphe, en commençant par celui de fin. Le calcul de la probabilité *backward* d'un mot w_i s'effectue alors à partir des probabilités *backward* des mots w_j, appartenant à une des listes de mots candidats d'un des arcs a_t^{t+x} (avec $x \in \{1, 2, 3\}$), comme illustré par la figure 5.6, pour le mot *straight*.

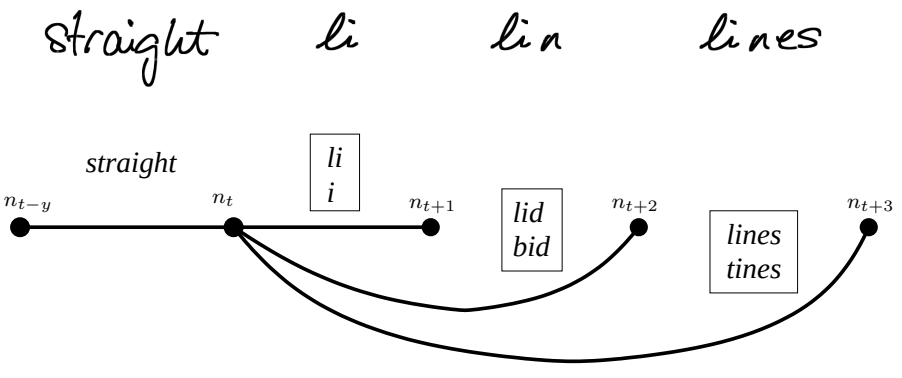


FIG. 5.6 – Mots pris en compte dans le calcul de la probabilité *backward* β du mot *straight*.

Comme pour les probabilités *forward*, les probabilités *backward* sont décomposées selon les différentes longueurs de chemins allant du nœud n_t considéré jusqu'au nœud de fin du graphe.

5.3.2.3 Calcul des probabilités *a posteriori* $P_{post}(w_i)$

En utilisant les probabilités *forward* et *backward* calculées précédemment, l'algorithme 5.4 donne le calcul des probabilités *a posteriori* de chacun des mots du graphe.

Algorithme 5.4 : Algorithme de calcul des probabilités *a posteriori* des mots.

Données : G : le graphe de mots;
$P(w_i|w_j)$: la probabilité du bigramme $w_j w_i$, donnée par le modèle de langage;
$score_{norm}(s_i|w_i)$: le score normalisé du mot w_i;
α : les probabilités *forward* des mots du graphe G;
β : les probabilités *backward* des mots du graphe G;
$denom$: la somme des probabilités de tous les chemins du graphe;

Résultat :
$P_{post}(w_i)$: la probabilité *a posteriori* associée à chacun des mots w_i du graphe G;

début
  pour chaque *nœud n_t du graphe G, dans un ordre topologique* **faire**
    pour chaque *arc a_{t-y}^{t} prédécesseur du nœud n_t* **faire**
      pour chaque *mot candidat w_i de l'arc a_{t-y}^{t}* **faire**
        pour chaque *arc a_{t-y-x}^{t-y} prédécesseur du nœud n_{t-y}* **faire**
          pour chaque *mot candidat w_j de l'arc a_{t-y-x}^{t-y}* **faire**
            pour chaque *longueur de chemin l_k de $\alpha[t-y][x][j]$* **faire**
              pour chaque *longueur de chemin l_l de $\beta[t][y][i]$* **faire**
                $P_{post}(w_i) += \alpha[t-y][x][j][k] * P(w_i|w_j)^{\gamma} * score_{norm}(s_i|w_i)^{\zeta} * \beta[t][y][i][l] * \delta^{k+l}$;
        $P_{post}(w_i) = \frac{P_{post}(w_i)}{denom}$;
fin

En reprenant la figure 5.5, la probabilité *a posteriori* du mot *straight* est obtenue en combinant la probabilité *backward* de ce mot aux probabilités *forward* de chacun des mots w_j, appartenant aux arcs précédant l'arc auquel appartient *straight*. Cela permet ainsi de prendre en compte tous les chemins passant par le mot *straight*, à partir des chemins arrivant à ce mot et de ceux y commençant. La longueur de chacun des chemins considérés est alors la somme des longueurs des sous-chemins se terminant par le mot *straight* et des longueurs de ceux débutant par ce mot. La longueur des chemins pondère le paramètre δ, qui permet de prendre en compte les sur- et les sous-segmentations.

Les *probabilités a posteriori* de chacun des mots sont enfin obtenues en normalisant la

somme des probabilités des chemins auxquels ils appartiennent par rapport à la somme de tous les chemins du graphe de mots (représentée par la variable $denom$, dans l'algorithme 5.4).

Il reste alors à choisir l'hypothèse consensus, à partir des probabilités *a posteriori* des mots.

5.3.3 Nouveau calcul de l'hypothèse consensus

Les ensembles de confusion ayant été modifiés, il n'est plus toujours possible de choisir, dans chacun d'eux, le mot ayant la probabilité *a posteriori* la plus élevée. En effet, quand les ensembles de confusion sont de la forme de celui donné par la figure 5.4, ils représentent, non plus des confusions sur des mots, mais des confusions sur une sous-partie de la phrase à reconnaître. Il faut alors identifier le meilleur chemin dans chacun des ensembles de confusion, en s'appuyant sur les probabilités *a posteriori* des mots qui le composent.

Pour retrouver le meilleur chemin dans un ensemble de confusion, nous utilisons l'algorithme de Viterbi qui génère la liste des M meilleurs chemins. Nous prenons alors, comme score d'un chemin, le produit des probabilités *a posteriori* des mots qui le compose.

L'hypothèse consensus est ainsi obtenue en choisissant, dans chacun des ensembles de confusion, le chemin dont le score est le plus élevé. L'équation 5.3 devient alors :

$$W_{consensus} = W^1_{consensus}\, W^2_{consensus} \ldots W^L_{consensus} \tag{5.5}$$

avec L le nombre d'ensembles de confusion et $W^l_{consensus}$ la sous-partie de la phrase définie par :

$$W^l_{consensus} = \arg \max_{W^l \in C_l} \prod_{j=1}^{N} P_{post}(w_j^{(l)}) \tag{5.6}$$

avec C_l le l^e ensemble de confusion et $W^l = w_1^{(l)} \ldots w_N^{(l)}$ un chemin (composé de N mots) de l'ensemble de confusion C_l.

Après avoir détaillé les extensions apportées à la construction et à l'exploitation des réseaux de confusion, nous présentons, dans la section suivante, les expérimentations menées sur leur utilisation pour la reconnaissance de phrases manuscrites.

5.4 Expérimentations

L'objectif des expérimentations présentées dans cette section est de comparer la reconnaissance de phrases basée sur le consensus (détaillée dans ce chapitre) avec celle utilisant l'approche MAP (introduite au chapitre 4).

Nous reviendrons également sur l'influence de la longueur des phrases, dans le calcul des probabilités *a posteriori* des mots, puisqu'une partie des modifications que nous avons apportées au calcul de ces probabilités est destinée à prendre en compte la longueur des phrases.

5.4.1 Comparaison avec la reconnaissance basée sur l'approche MAP

La table 5.1 donne les taux de reconnaissance (TRM) obtenus avec chacune des deux approches considérées, sur les mots de la base de test $Saisie_{test}$ (les différents paramètres ont été préalablement optimisés sur la base d'apprentissage $Saisie_{app}$, indépendamment pour chacune des deux approches). De plus, le modèle de langage utilisé dans ces deux algorithmes est un modèle bigramme appris sur le corpus $Brown_{app}$.

TAB. 5.1 – Comparaison des tâches de reconnaissance basées soit sur un réseau de confusion, soit sur un graphe de mots.

Extraction des mots	Approche de reconnaissance	TRM
Manuelle	MAP	88,48 %
	Consensus	88,23 %
Automatique	MAP	86,51 %
	Consensus (δ_{opt})	85,38 %
	Consensus ($\delta = 0$)	83,86 %

Quand l'extraction des mots est réalisée manuellement, la différence entre les deux taux de reconnaissance est assez faible (0,25 %). Cette différence pourrait être expliquée par la normalisation du score $score_{norm}(s_i|w_i)$ qui n'est peut-être pas optimale. De plus, ce score n'étant pas de nature probabiliste, sa combinaison avec les probabilités données par le modèle de langage n'est peut-être pas non plus optimale.

Lorsque l'extraction des mots est réalisée de manière automatique, la stratégie de segmentation utilisée correspond à la meilleure stratégie présentée dans le chapitre 3.

Nous considérons alors deux variantes de l'approche consensus. Dans le premier cas, le nombre de mots dans les phrases n'est pas considéré, ce qui correspond à un poids $\delta = 0$. dans le second cas, la prise en compte du nombre de mots dans les phrases (pour le calcul des probabilités *a posteriori* des mots) est optimisée, ce qui correspond à δ_{opt}.

Les taux de reconnaissance obtenus avec les deux approches utilisant le consensus sont inférieurs à celui obtenu en utilisant l'approche MAP. Cette plus grande différence entre les taux de reconnaissance, par rapport aux résultats obtenus dans le cas de la segmentation manuelle, est due à la plus grande complexité des ensembles de confusion. En effet, lorsque la segmentation est effectuée manuellement, les ensembles de confusion sont simples puisqu'ils correspondent aux listes de mots candidats associées à chaque arc du graphe de mots. De plus, comme toutes les phrases ont la même longueur, les éventuels problèmes de sur- et de sous-segmentations n'entrent pas en compte.

Néanmoins, la prise en compte du nombre de mots dans les phrases permet d'améliorer le taux de reconnaissance sur les mots, d'un peu plus de 1 %. Dans la sous-section suivante, nous revenons sur l'influence de ce paramètre.

5.4.2 Prise en compte de la longueur des phrases

La figure 5.7 montre l'évolution du taux de reconnaissance sur les mots de la base d'apprentissage, en fonction des paramètres ζ et δ (le paramètre γ est fixé à 1, comme suggéré dans [Man00]).

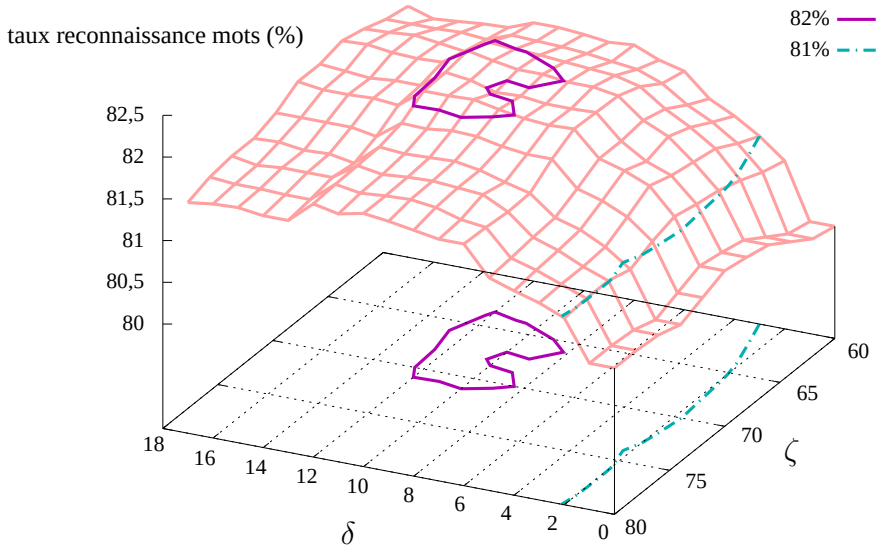


FIG. 5.7 – Influence de la prise en compte de la longueur des phrases, dans le calcul des probabilités *a posteriori* des mots.

Nous constatons plus particulièrement l'impact du paramètre δ, qui permet de pondérer les sur- et sous-segmentations, en prenant en compte le nombre de mots dans les phrases. En effet, l'optimisation de ce dernier permet d'améliorer le taux de reconnaissance d'un peu plus de 1 %, comme nous l'avions déjà vu sur la base de test, dans la sous-section précédente. Les résultats obtenus sur la base d'apprentissage, avec les paramètres optimaux appris sur celle-ci, sont confirmés sur la base de test : cela montre les bonnes capacités de généralisation de l'optimisation sur la base d'apprentissage.

La prise en compte du nombre de mots dans les phrases permet de ne pas favoriser les mots appartenant à des phrases courtes. En effet, les scores des phrases courtes étant généralement plus élevés que ceux des phrases plus longues (comme le score d'une phrase est calculé à partir de celui des mots la constituant), les phrases courtes peuvent avoir une trop grande importance dans le calcul des probabilités *a posteriori* des mots (pour rappel, la probabilité *a posteriori* d'un mot est la somme des probabilités des phrases auxquelles il appartient). Ainsi, en pénalisant les phrases courtes, cela permet d'obtenir une plus grande indépendance du score des phrases vis-à-vis du nombre de mots qu'elles comportent.

5.5 Bilan

Dans ce chapitre, nous avons présenté une nouvelle approche pour la reconnaissance de phrases. Cette approche est basée sur une représentation des phrases sous la forme d'un réseau mettant en évidence les confusions existantes entre les mots présents dans un graphe de mots initial. La reconnaissance s'appuie alors sur le calcul des probabilités *a posteriori* des mots, correspondant à la somme des probabilités des phrases auxquelles chacun des mots appartient. La phrase résultat, appelée aussi hypothèse consensus, est alors obtenue en minimisant le taux d'erreur sur les mots et non plus en minimisant le taux d'erreur sur les phrases, comme c'est le cas dans l'approche MAP classique.

Cette technique ayant été introduite pour la reconnaissance automatique de la parole, nous avons proposé des modifications pour l'adapter à notre tâche de reconnaissance de phrases manuscrites. Ces modifications ont concerné le calcul des probabilités des mots en étendant ce calcul afin de prendre en compte les confusions entre plusieurs segmentations possibles d'une même partie du signal manuscrit de la phrase. Le calcul des probabilités *a posteriori* proposé permet également de prendre en compte le nombre de mots dans les phrases considérées. Le choix de l'hypothèse consensus a lui aussi été étendu, pour s'adapter au nouveau calcul des probabilités *a posteriori*.

Les résultats obtenus en utilisant l'approche proposée sont inférieurs à ceux obtenus avec l'approche MAP. Cependant, les modifications apportées pour la prise en compte de la longueur des phrases, dans le calcul des probabilités *a posteriori* des mots, permettent d'améliorer les résultats par rapport à l'approche initiale basée sur les réseaux de confusion.

Il n'est pas très intéressant d'utiliser l'approche consensus pour effectuer la reconnaissance des phrases. Cependant, comme cette stratégie de reconnaissance de phrases s'appuie sur une approche différente de l'approche MAP, il peut être intéressant de l'utiliser pour détecter des risques de confusion ou pour affiner la reconnaissance réalisée avec l'approche MAP. C'est ce que nous présentons dans le chapitre suivant.

Chapitre 6

Détection et correction d'erreurs de reconnaissance

6.1 Introduction

Malgré l'ajout de connaissances linguistiques pour améliorer la reconnaissance des phrases, les systèmes de reconnaissance ne sont pas parfaits et il reste des erreurs dans la phrase résultat, c'est-à-dire la phrase en sortie du système de reconnaissance. Il peut alors être intéressant de mesurer la confiance que l'on a en chacun des mots résultats, pour pouvoir identifier ceux sur lesquels il subsiste un doute. Cela permet alors de mettre ces mots en évidence, pour un utilisateur (pour faciliter sa tâche de relecture), ou encore d'essayer de corriger les erreurs potentielles détectées.

Dans ce chapitre, nous proposons une approche originale pour *détecter* les risques d'erreurs de reconnaissance, en s'appuyant sur les *probabilités a posteriori* des mots de la phrase résultat, obtenue par l'approche MAP. La figure 6.1 illustre cette approche.

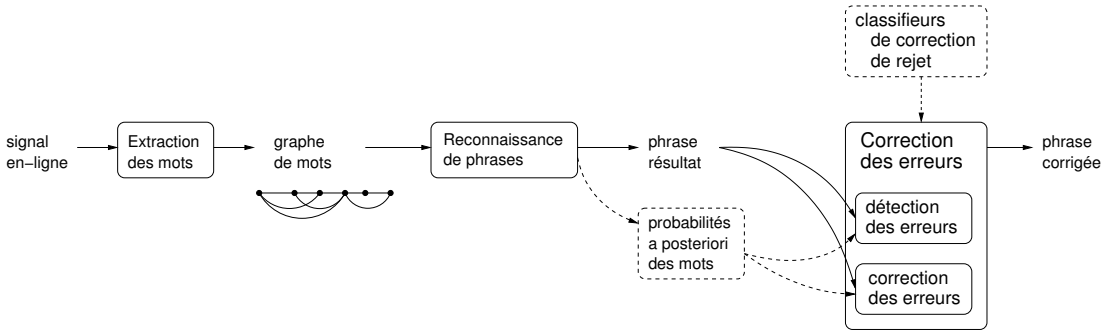


FIG. 6.1 – Principe de la détection et de la correction des erreurs de reconnaissance.

Une phase de *correction* permet ensuite de confirmer le résultat de la reconnaissance du mot considéré ou de remettre en cause sa reconnaissance, en lui préférant un autre mot. Pour

cela, des classifieurs dédiés à cette tâche sont utilisés. Enfin, nous ajoutons un *mécanisme de rejet*, pour mesurer la confiance sur le classifieur de correction. Cela permet alors de rejeter la reconnaissance des mots si la confiance en le résultat du classifieur de correction n'est pas assez élevée. Ainsi, nous n'essaierons pas de corriger les erreurs que les classifieurs dédiés ne peuvent pas corriger.

Dans ce chapitre, nous exposons notre approche permettant la détection et la correction d'erreurs de reconnaissance, en utilisant les probabilités *a posteriori* des mots comme indices de confiance.

Nous présentons tout d'abord un rapide état de l'art sur les mesures de confiance pouvant être utilisées sur les mots, dans la section 6.2. Puis, nous introduisons notre approche, dans la section 6.3, en nous focalisant notamment sur les étapes de détection et de correction des erreurs. Nous présentons finalement, dans la section 6.4, le mécanisme permettant de rejeter les erreurs potentielles que nous ne pourrons pas corriger, avant de détailler, dans la section 6.5, les résultats obtenus sur les expérimentations menées pour valider l'approche proposée.

6.2 État de l'art sur les mesures de confiance au niveau des mots

Les mesures de confiance utilisées pour la reconnaissance de phrases portent généralement sur les mots de ces phrases. Elles intègrent en général des informations sur le contexte des mots dans la phrase.

Peu de mesures de confiance sont utilisées en reconnaissance de phrases manuscrites.

Dans [Mar04], des anti-modèles de mots sont construits à partir des anti-modèles de lettres, pour rejeter les mots de la phrase résultat en s'appuyant sur un seuil fixé. Aucune phase de reconnaissance supplémentaire n'est réalisée sur les mots ainsi rejetés.

La mesure de confiance utilisée dans [BZB06] s'appuie sur l'intégration d'un modèle de langage durant le processus de reconnaissance. Différentes valeurs des paramètres γ et δ (utilisés pour pondérer l'apport du modèle de langage et compenser les problèmes de segmentation, respectivement) sont utilisées pour générer des hypothèses de phrases. La mesure de confiance des mots est alors directement déduite de leur nombre d'occurrences, parmi les hypothèses ainsi générées. Les mots sont alors rejetés si leur indice de confiance est inférieur à un seuil fixé *a priori*.

Enfin, le système de reconnaissance de textes hors-ligne présenté dans [BHB06] utilise le vote de plusieurs systèmes de reconnaissance, comme mesure de confiance sur les mots. Si un des systèmes n'est pas d'accord avec les autres, le mot est rejeté. Les mots rejetés sont alors soumis à une nouvelle étape de reconnaissance, en utilisant pour cela les mêmes systèmes de reconnaissance.

D'autres mesures de confiance plutôt basées sur des mots isolés (ou au niveau des caractères) sont discutées dans [PSP06]. Ces mesures de confiance se basent essentiellement sur des travaux en reconnaissance automatique de la parole, ce que nous présentons maintenant.

En reconnaissance automatique de la parole, beaucoup de systèmes associent une mesure de confiance aux mots de la phrase résultat. Ces mesures sont généralement basées sur le graphe de mots ou la liste des M-meilleures hypothèses de phrases [WSMN01].

Les probabilités *a posteriori* des mots sont ainsi souvent utilisées [EW00]. Elles peuvent aussi être combinées à d'autres informations telles que la densité du graphe [KS97] ou encore des caractéristiques acoustiques et linguistiques [WBR+97]. Un réseau de neurones peut alors être utilisé pour combiner ces informations.

Dans [HO06], des informations telles que la position du mot dans la phrase résultat, le nombre de mots de cette phrase ou encore la moyenne et la variance des probabilités *a posteriori* des mots sont combinées aux probabilités *a posteriori*, en utilisant des SVMs.

6.3 Détection et correction d'erreurs basées sur les probabilités *a posteriori* des mots

Nous avons choisi d'utiliser les *probabilités a posteriori* des mots de la phrase résultat, comme mesure de confiance pour détecter et corriger d'éventuelles erreurs de reconnaissance.

Dans cette section, nous présentons tout d'abord le principe général de notre approche (voir sous-section 6.3.1). Nous nous focalisons ensuite sur les deux types d'erreurs de reconnaissance considérées (voir sous-sections 6.3.2 et 6.3.3), en précisant pour chacune les étapes de détection et de correction.

6.3.1 Principe général

La détection des erreurs est basée, d'une part, sur le calcul de la meilleure phrase en utilisant l'approche MAP (voir algorithme 4.1, chapitre 4) et, d'autre part, sur l'utilisation des probabilités *a posteriori* des mots de cette phrase. Le calcul de ces probabilités *a posteriori* est le même que celui décrit par l'algorithme 5.4, dans le chapitre 5.

L'algorithme 6.1 détaille notre approche de détection et de correction des erreurs.

La probabilité *a posteriori* de chacun des mots de la phrase résultat est utilisée pour détecter une éventuelle erreur de reconnaissance. Nous considérons deux cas d'erreurs potentielles, pouvant se produire pour le mot w_i :

- un autre mot de la liste de mots candidats l_i (à laquelle il appartient) a une probabilité *a posteriori* plus élevée que celle de w_i (§ 6.3.2) ;
- la probabilité *a posteriori* de w_i est considérée comme trop faible (§ 6.3.3).

Nous associons alors, à chaque type d'erreur détectée, une *fonction de correction*, qui vise à choisir entre le mot résultat et un des autres mots de la liste l_i à laquelle il appartient.

La phrase corrigée est alors obtenue à partir de chacun de ses mots corrigés.

Dans cette approche, nous ne remettons pas en cause la segmentation de la phrase résultat ; cela fera notamment partie des perspectives envisagées pour ces travaux.

Algorithme 6.1 : Algorithme de détection et de correction des erreurs de reconnaissance.

Données :
$\widehat{W}_{MAP}$: la phrase résultat, obtenue en utilisant l'approche MAP;
$P_{post}(w_i)$: la probabilité *a posteriori* du mot w_i;
σ_{post} : le seuil sur les probabilités *a posteriori* des mots;
w_{max} : le mot qui a la probabilité maximale, dans la liste de mots candidats l_i (liste à laquelle le mot w_i appartient);
w_{max_2} : le mot ayant la deuxième probabilité la plus élevée, dans la liste l_i;
$correction_{1,2}(w_i, w_k)$: les fonctions de correction pour choisir entre les mots w_i et w_k;

Résultat :
$\widehat{W}$: la phrase résultat corrigée;

début
  pour chaque *mot* w_i *de la phrase* $\widehat{W}_{MAP}$ **faire**
§ 6.3.2 **si** $P_{post}(w_i) \neq P_{post}(w_{max})$ **alors**
    $\widehat{w_i} = correction_1(w_i, w_{max})$;
§ 6.3.3 **sinon si** $P_{post}(w_i) < \sigma_{post}$ **alors**
    $\widehat{w_i} = correction_2(w_i, w_{max2})$;
  $\widehat{W} = \widehat{w_1}\,\widehat{w_2} \ldots \widehat{w_N}$;
fin

6.3.2 Mots de probabilité *a posteriori* non-maximale

6.3.2.1 Détection

Dans ce premier cas, nous considérons qu'il peut y avoir une erreur de reconnaissance si la probabilité *a posteriori* du mot résultat w_i n'est pas la probabilité la plus élevée, parmi celles des mots appartenant à la liste de mots candidats l_i (liste à laquelle appartient le mot résultat w_i). Cela signifie qu'il existe au moins un autre mot de la liste l_i dont l'indice de confiance est plus élevé que celui de w_i.

Pour détecter ce type d'erreur, nous comparons la probabilité *a posteriori* du mot résultat w_i avec celle du mot w_{max}, qui a la probabilité la plus élevée. La figure 6.2 illustre un exemple de ce type d'erreur.

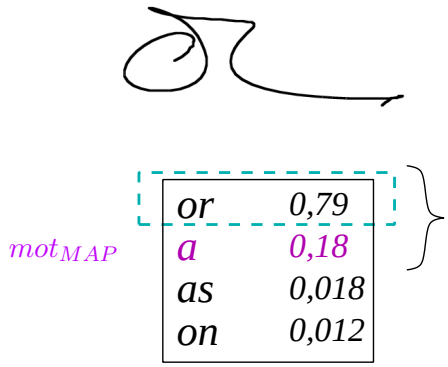


FIG. 6.2 – Exemple d'erreur potentielle détectée, pour un mot de probabilité *a posteriori* non-maximale.

Dans cet exemple, le mot *a* est le mot résultat (appelé aussi mot_{MAP}, sur la figure) alors que le mot correct est *or* (encadré en pointillé, sur la figure), ce dernier ayant une probabilité *a posteriori* plus élevée.

6.3.2.2 Correction

Pour proposer une correction du mot détecté comme potentiellement en erreur, nous considérons que le mot correct est soit le mot résultat w_i, soit le mot w_{max} qui a la probabilité *a posteriori* maximale (voir figure 6.2). Les expérimentations montrent que, dans environ 85 % des cas, lorsque le mot correct est présent dans la liste, il correspond à l'un de ces deux mots.

Pour choisir parmi ces deux mots, un classifieur dédié est appris. Il comporte 6 entrées (3 caractéristiques pour chacun des deux mots) et 2 sorties (chacune représente un des mots).

Les 3 caractéristiques utilisées pour chaque mot sont les suivantes :

- sa probabilité *a posteriori* $P_{post}(w_i)$;
- son score lexical normalisé $score_{lexique\_norm}$;
- son score graphique normalisé $score_{graphique\_norm}$.

Le score lexical (voir chapitre 1, sous-section 1.4) est normalisé par rapport aux scores lexicaux minimal et maximal des mots ayant le même nombre de traits descendants. Cette normalisation est effectuée parce que ce score est dépendant du nombre de traits descendants du mot (voir chapitre 1, sous-section 1.4). Le score graphique (voir chapitre 1, sous-section 1.4) est normalisé de la même façon.

Ces caractéristiques permettent de considérer à la fois des informations *globales* sur le graphe (les probabilités *a posteriori* des mots, qui correspondent à la somme des probabilités des chemins auxquels appartiennent les mots) et des informations *locales* sur les mots (leurs scores lexicaux et graphiques).

6.3.3 Mots de probabilité *a posteriori* faible

6.3.3.1 Détection

Dans ce deuxième cas, le mot résultat w_i est considéré comme potentiellement en erreur si sa probabilité *a posteriori* est jugée trop faible (sa probabilité est la probabilité maximale de sa liste de mots l_i). Nous considérons ainsi que la confiance en le résultat de la reconnaissance n'est pas assez fiable.

Pour détecter ce type d'erreur, nous comparons la probabilité *a posteriori* du mot w_i par rapport à un seuil σ_{post}. Si la probabilité est inférieure à ce seuil, le mot est alors considéré comme une potentielle erreur. La figure 6.3 illustre un exemple de ce type d'erreur.

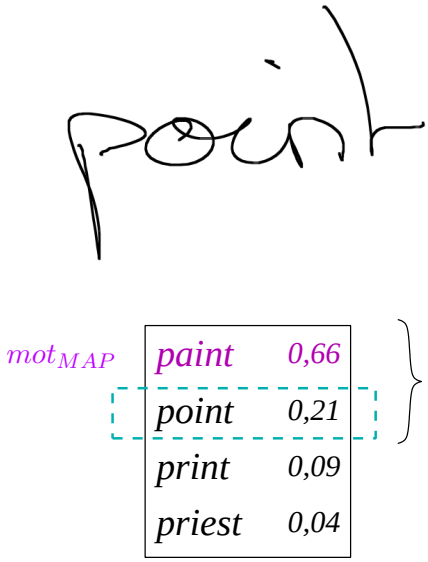


FIG. 6.3 – Exemple d'erreur potentielle détectée, pour un mot de probabilité *a posteriori* faible.

Dans cet exemple, le mot *paint* est le mot résultat mais sa probabilité *a posteriori* est en-dessous du seuil σ_{post}, fixé à 0,8 (voir sous-section 6.5.3).

6.3.3.2 Correction

La proposition de correction de l'erreur détectée est réalisée de la même façon que dans le cas précédent, en utilisant un nouveau classifieur dédié, permettant de choisir entre deux mots.

Le choix se porte alors soit sur le mot résultat w_i, soit sur le mot w_{max2} ayant la deuxième probabilité *a posteriori* la plus élevée (voir figure 6.3). En effet, le mot résultat a la probabilité *a posteriori* la plus élevée de la liste l_i, dans ce cas d'erreur détectée. De même que précédemment, les expérimentations ont montré que, lorsque le mot correct se trouve dans la liste, il correspond à un de ces deux mots dans à peu près 83 % des cas.

Les caractéristiques utilisées pour chacun de ces mots sont les mêmes que précédemment ; le classifieur appris possède également 6 entrées et 2 sorties.

Dans l'approche de correction que nous avons présentée, nous avons considéré que le mot correct se trouvait parmi les deux mots en entrée du classifieur de correction. En fait, dans environ 15 % des cas, le mot correct ne correspond à aucun de ces deux mots. Nous ajoutons alors un *mécanisme de rejet* qui permet de détecter ces cas de figure et ainsi de ne pas essayer de corriger les mots qui ne pourront pas l'être.

Nous expliquons cette extension, dans la section suivante.

6.4 Ajout d'un mécanisme de rejet des mots absents

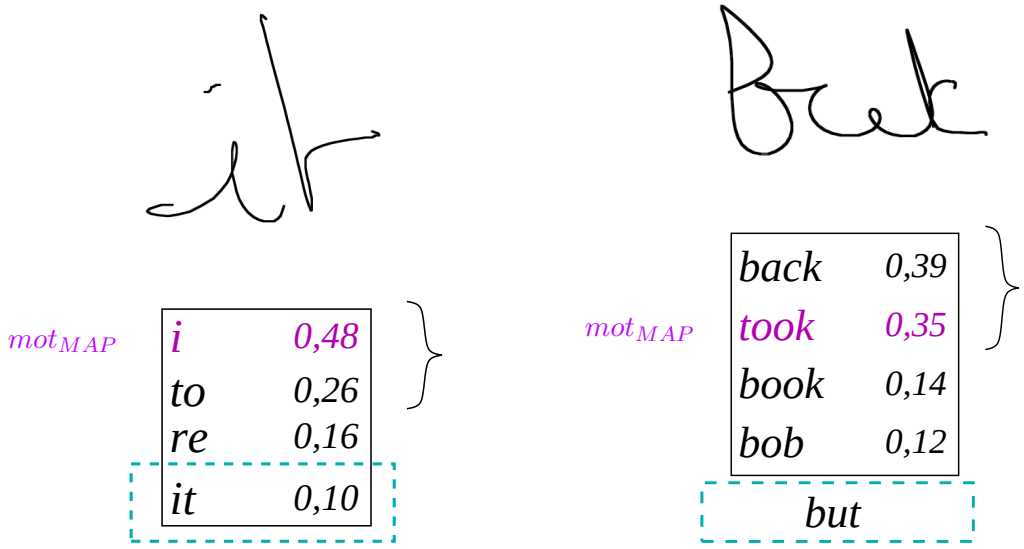
Pour justifier l'introduction du mécanisme de rejet de certains mots détectés, nous commençons par présenter les restrictions introduites pour le traitement des erreurs, dans l'approche précédente. Nous expliquons ensuite l'utilisation du rejet pour étendre cette approche.

6.4.1 Limitations de l'approche précédente

Dans les deux stratégies de correction présentées, nous supposons que le mot correct se trouve parmi les deux mots donnés en entrée de chacun des classifieurs dédiés. En fait, le mot correct peut soit être parmi les autres mots de la liste l_i (voir figure 6.4(a)), soit ne pas être dans cette liste (voir figure 6.4(b)). Ainsi, au vu des expérimentations, dans 40 % des cas, le mot correct correspond à un des autres mots de la liste alors que, dans 60 % des cas, il est absent de cette liste de mots candidats.

Dans l'exemple donné par la figure 6.4(a), *it* est le mot correct mais les deux mots utilisés pour la correction sont *i* et *to*. De même, dans l'exemple donné par la figure 6.4(b), le mot correct *but* n'est pas un des deux mots utilisés pour la correction (qui sont *took* et *back*). De plus, le mot *but* n'apparaît même pas parmi les mots de la liste.

Cette simplification a été faite car la majorité des mots corrects se trouve parmi les deux mots utilisés pour la correction. Ainsi, si nous ne considérons que les mots détectés en erreur pour lesquels la liste l_i contient le mot correct, dans 71,35 % des cas, le mot correct est un des deux mots utilisés pour la correction, dans le cas de la segmentation manuelle. Ce pourcentage est de 85,08 % lorsque la segmentation est automatique.



(a) Mot *it* présent dans la liste (b) Mot *but* absent de la liste

FIG. 6.4 – Exemples de mots à rejeter.

Une solution, pour corriger plus d'erreurs, serait alors de considérer plus de mots parmi lesquels trouver le mot correct. Cela se traduit par une augmentation du nombre d'entrées et de sorties des classifieurs utilisés pour la correction. Il est alors nécessaire de disposer de plus de données afin d'apprendre des classifieurs suffisamment fiables. Comme le volume de données dont nous disposons et correspondant aux mots détectés est assez faible, il n'est actuellement pas envisageable d'augmenter le nombre de mots à considérer. Cela pourra néanmoins faire partie des perspectives concernant l'amélioration de la correction des erreurs potentielles.

Une seconde voie complémentaire consiste à utiliser un *mécanisme de rejet*. Ce mécanisme permet de mesurer la confiance, non plus sur les mots résultats, mais sur le classifieur utilisé pour la correction. En effet, s'il est jugé que ce classifieur ne pourra pas corriger le mot résultat, le mot est rejeté.

Nous présentons maintenant les modifications que cela entraîne.

6.4.2 Principe de l'approche utilisant le rejet

L'algorithme 6.2 étend l'algorithme 6.1, pour inclure le mécanisme de rejet des mots qui ne pourront *a priori* pas être corrigés.

Dans chacun des deux cas d'erreurs détectées, nous regardons d'abord si le mot pourra être corrigé, en utilisant la fonction de rejet. Si le mot n'est pas rejeté, la correction peut avoir lieu comme précédemment (en utilisant les mêmes classifieurs).

Algorithme 6.2 : Algorithme de détection et de correction des erreurs de reconnaissance, incluant un mécanisme de rejet.

Données :
$\widehat{W}_{MAP}$: la phrase résultat en utilisant l'approche MAP;
$P_{post}(w_i)$: la probabilité *a posteriori* du mot w_i;
σ_{post} : le seuil sur les probabilités *a posteriori* des mots;

w_{max} : le mot qui a la probabilité maximale dans la liste de mots candidats l_i;
w_{max_2} : le mot ayant la deuxième probabilité la plus élevée, dans la liste de mots candidats l_i;
$correction_{1,2}(w_i, w_k)$: les fonctions de correction pour choisir entre les mots w_i et w_k;

$rejet_{1,2}(w_i, w_k)$: les fonctions qui rejettent éventuellement le mot w_i, à partir des informations sur les mots w_i et w_k;

Résultat :
$\widehat{W}$: la phrase résultat corrigée et pouvant comporter des mots rejetés;

début
  pour chaque *mot* w_i *de la phrase* $\widehat{W}_{MAP}$ **faire**
    si $P_{post}(w_i) \neq P_{post}(w_{max})$ **alors**
      si $rejet_1(w_i, w_{max})$ **alors**
        $\widehat{w_i} = REJET$;
      sinon
        $\widehat{w_i} = correction_1(w_i, w_{max})$;
    sinon si $P_{post}(w_i) < \sigma_{post}$ **alors**
      si $rejet_2(w_i, w_{max2})$ **alors**
        $\widehat{w_i} = REJET$;
      sinon
        $\widehat{w_i} = correction_2(w_i, w_{max2})$;
  $\widehat{W} = \widehat{w_1}\,\widehat{w_2} \ldots \widehat{w_N}$;
fin

Pour rejeter les mots qui ne pourront pas être reconnus, nous utilisons à nouveau deux classifieurs dédiés à chacun des deux types d'erreurs considérées. Cela correspond à l'architecture SC (classifieur spécialisé), présentée dans [Mou07]. Ce choix permet de pouvoir utiliser les classifieurs dédiés aux corrections des erreurs et déjà appris, pour l'approche précédente.

Chacun des deux classifieurs de rejet comporte 6 entrées, correspondant aux 3 caractéristiques de chacun des deux mots considérés (soit w_i et w_{max}, soit w_i et w_{max2}). Ce sont donc les mêmes entrées que précédemment.

En ce qui concerne les sorties, bien qu'il y en ait également 2, elles ne correspondent pas aux classes utilisées précédemment. En effet, la première sortie correspond à une classe de rejet et la deuxième à une classe d'acceptation. Ainsi, la décision de rejeter un mot est prise en fonction de la classe attribuée au mot w_i, par le classifieur de rejet correspondant.

Après avoir détaillé notre approche pour détecter et corriger des erreurs sur la phrase résultat (obtenue avec l'approche MAP), nous présentons maintenant les résultats des expérimentations menées pour valider cette approche.

6.5 Expérimentations

L'objectif des expérimentations menées, et présentées dans cette section, est d'améliorer les performances du système de reconnaissance, en diminuant le taux d'erreur sur les mots. Pour cela, nous utilisons l'approche de détection et de correction des erreurs, à laquelle nous ajoutons également le mécanisme de rejet. Ce dernier mécanisme permet d'augmenter la fiabilité du système de reconnaissance, en rejetant les mots qui ne peuvent pas être corrigés.

Nous mentionnons tout d'abord les classifieurs permettant la correction des mots ainsi que ceux servant au rejet et nous présentons également les mesures de performance utilisées.

6.5.1 Cadre expérimental

6.5.1.1 Classifieurs utilisés

Pour la correction comme pour le rejet, nous avons utilisé des *systèmes à vastes marges* [CV95] (SVM, pour *Support Vector Machines*) avec des noyaux gaussiens. Nous avons choisi ces classifieurs pour leur efficacité et leur fiabilité ainsi que pour leur capacité d'apprentissage lorsque les classes sont déséquilibrées (en terme de nombre d'exemples d'apprentissage).

La table 6.1 détaille les bases d'apprentissage extraites à partir de la base $Saisie_{app}$ et utilisées pour l'apprentissage des classifieurs de correction et de rejet, pour chacun des deux types d'erreurs potentielles considérées.

Les mots des bases utilisées pour l'apprentissage des classifieurs dédiés à la correction sont les mots de la base $Saisie_{app}$ (voir chapitre 3, sous-section 3.4.1.1), tels que le mot correct est un des deux mots utilisés pour la correction (cela correspond aux $nb_{mots\_présents}$ mots, définis dans la sous-section suivante). De même, pour l'apprentissage des classifieurs dédiés au rejet, la base $Saisie_{app}$ est également utilisée, en ne considérant cette fois que les mots à rejeter.

TAB. 6.1 – Nombre de mots dans les bases, pour l'apprentissage des classifieurs dédiés à la correction et au rejet des erreurs potentielles de reconnaissance.

Extraction	Détection par probabilité non-maximale		Détection par probabilité faible	
des mots	$Base^{nonMax}_{correct}$	$Base^{nonMax}_{rejet}$	$Base^{faible}_{correct}$	$Base^{faible}_{rejet}$
Manuelle	108	60	341	270
Automatique	118	81	417	334

6.5.1.2 Mesures de performance

Taux de reconnaissance sur les mots détectés pour la correction ($TRMD$) Il donne le pourcentage de mots correctement reconnus, parmi les mots qui ont été détectés comme potentiellement en erreur (dans un des deux types d'erreur considérés) :

$$TRMD = \frac{nb^{recos}_{mots\_détectés}}{nb_{mots\_détectés}} \times 100 \tag{6.1}$$

avec $nb_{mots\_détectés}$ le nombre de mots détectés comme potentiellement en erreur et $nb^{recos}_{mots\_détectés}$ le nombre de ces mots qui sont correctement reconnus.

Taux de reconnaissance sur les mots présents dans la correction ($TRMP$) Il mesure le pourcentage de mots correctement reconnus mais cette fois seulement parmi les mots détectés pour lesquels le mot correct correspond à un des deux mots utilisés pour la correction (c'est-à-dire ceux qui peuvent être correctement reconnus par le classifieur de correction) :

$$TRMP = \frac{nb^{recos}_{mots\_présents}}{nb_{mots\_présents}} \times 100 \tag{6.2}$$

avec $nb_{mots\_présents}$ le nombre de mots détectés, pour lesquels le mot correct se trouve parmi les mots utilisés pour la correction, et $nb^{recos}_{mots\_présents}$ le nombre de ces mots qui sont correctement reconnus.

Taux de rejet sur les mots ($TRjM$) Ce taux permet de connaître le pourcentage de mots qui sont rejetés par le mécanisme ajouté :

$$TRjM = \frac{nb^{rejetés}_{mots}}{nb_{mots}} \times 100 \tag{6.3}$$

avec $nb^{rejetés}_{mots}$ le nombre de mots rejetés, en utilisant les classifieurs de rejet, et nb_{mots} le nombre de mots de la base de test $Saisie_{test}$.

Taux d'erreur sur les mots (TEM_{rej}) Pour pouvoir prendre en compte la notion de rejet, ce taux est modifié par rapport à celui présenté précédemment (voir chapitre 3, sous-section 3.4.1.2) :

$$TEM_{rej} = 100 - TRM - TRjM \tag{6.4}$$

avec TRM le taux de reconnaissance sur tous les mots de la base de test $Saisie_{test}$ et $TRjM$ le taux de rejet défini précédemment et calculé également sur la base $Saisie_{test}$.

Performance ($Perf$) Nous l'utilisons pour mesurer les taux de reconnaissance obtenus avec les classifieurs de correction, lorsque ceux-ci sont combinés avec les classifieurs de rejet correspondants. Comme la performance mesure le taux de reconnaissance parmi les mots en entrée des classifieurs de correction, les mots considérés sont tels que le mot correct se trouve parmi les deux mots en entrée du classifieur. La performance $Perf$ correspond alors au taux de reconnaissance sur les mots présents $TRMP$, précédemment défini.

Taux de vraie acceptation (TAR) Ce taux est utilisé pour évaluer le rejet introduit. Le taux de vraie acceptation (*True Acceptance Rate*, en anglais) permet de connaître le pourcentage de mots qui doivent être acceptés et qui le sont réellement. Ce taux doit donc être élevé.

Dans notre cas, les mots à accepter sont les $nb_{mots\_présents}$. Le TAR est alors donné par :

$$TAR = \frac{nb^{acceptés}_{mots\_présents}}{nb_{mots\_présents}} \times 100 \tag{6.5}$$

avec $nb^{acceptés}_{mots\_présents}$ le nombre de mots présents qui sont acceptés par le classifieur de rejet.

Taux de fausse acceptation (FAR) À l'instar du TAR, le FAR (*False Acceptance Rate*, en anglais) permet d'évaluer le rejet utilisé. Il mesure, cette fois, le pourcentage de mots à rejeter mais qui sont acceptés. Ce taux doit ainsi être le plus bas possible.

Dans notre cas, les mots à rejeter sont les mots détectés pour lesquels le mot correct ne se trouve pas parmi les mots utilisés pour la correction. Nous avons ainsi la définition du FAR :

$$FAR = \frac{nb^{acceptés}_{mots\_détectés} - nb^{acceptés}_{mots\_présents}}{nb_{mots\_détectés} - nb_{mots\_présents}} \times 100 \tag{6.6}$$

avec $nb^{acceptés}_{mots\_détectés}$ le nombre de mots détectés qui sont acceptés par le classifieur de rejet et $nb^{acceptés}_{mots\_présents}$ le nombre de mots présents qui sont acceptés par ce classifieur.

6.5.2 Correction des mots de probabilité *a posteriori* non-maximale

Dans ce premier cas d'erreur, près de 3 % des mots de la base de test $Saisie_{test}$ sont détectés (2,64 % des mots, dans le cas de la segmentation manuelle, et 3,13 % des mots, pour la segmentation automatique). Cela permet la détection de près de 15 % des erreurs, en utilisant l'approche MAP pour la reconnaissance (14,46 % des erreurs, pour l'extraction manuelle des mots, et 14,92 % des erreurs, pour l'extraction automatique).

Dans la table 6.2, nous présentons les taux de reconnaissance obtenus sur les mots détectés ainsi que sur les mots présents, afin d'évaluer la correction sur les mots concernés par cette stratégie de détection.

TAB. 6.2 – Taux de reconnaissance pour la correction des mots non-maximaux.

Extraction des mots	Approche de reconnaissance	$TRMD$	$TRMP$
Manuelle	MAP	36,90 %	57,41 %
	Détection/correction	46,43 %	72,22 %
Automatique	MAP	35,68 %	60,17 %
	Détection/correction	42,71 %	72,03 %

L'amélioration obtenue dans le cas de la segmentation manuelle est un peu au-dessus de celle obtenue pour la segmentation automatique. Ainsi, le taux d'erreur parmi les mots présents est réduit de 34,77 %, pour l'extraction manuelle des mots, et de 29,78 %, lorsqu'ils sont extraits automatiquement.

6.5.3 Correction des mots de probabilité *a posteriori* faible

Dans ce deuxième cas d'erreur, 9,60 % des mots de la base $Saisie_{test}$ sont détectés, pour la segmentation manuelle, et 11,80 % de ces mots, pour la segmentation automatique. Pour l'extraction manuelle des mots, cela permet de détecter 49,11 % des erreurs alors que, pour leur extraction automatique, 47,20 % des erreurs sont ainsi sélectionnées.

Dans la table 6.3, nous présentons les taux de reconnaissance obtenus, pour la correction des erreurs de ce type qui ont été détectées.

TAB. 6.3 – Taux de reconnaissance pour la correction des mots faibles.

Extraction des mots	Approche de reconnaissance	$TRMD$	$TRMP$
Manuelle	MAP	41,08 %	73,61 %
	Détection/correction	44,52 %	79,77 %
Automatique	MAP	46,07 %	82,97 %
	Détection/correction	47,80 %	86,09 %

L'augmentation relative des taux de reconnaissance est meilleure pour la segmentation manuelle. La réduction du taux d'erreur sur les mots présents est alors de 23,34 %, pour cette extraction manuelle des mots, et de 18,32 %, pour leur extraction automatique.

Une des raisons des performances plus faibles, dans le cas de la segmentation automatique, peut être expliquée par la segmentation de la phrase. D'une part, cette segmentation peut ne pas être entièrement correcte. D'autre part, les ensembles de confusion sont maintenant plus

complexes, pour prendre en compte les différentes segmentations alternatives présentes dans le graphe de mots (voir chapitre 5).

La figure 6.5 donne un exemple de graphe de mots qui comporte un seul ensemble de confusion étendu.

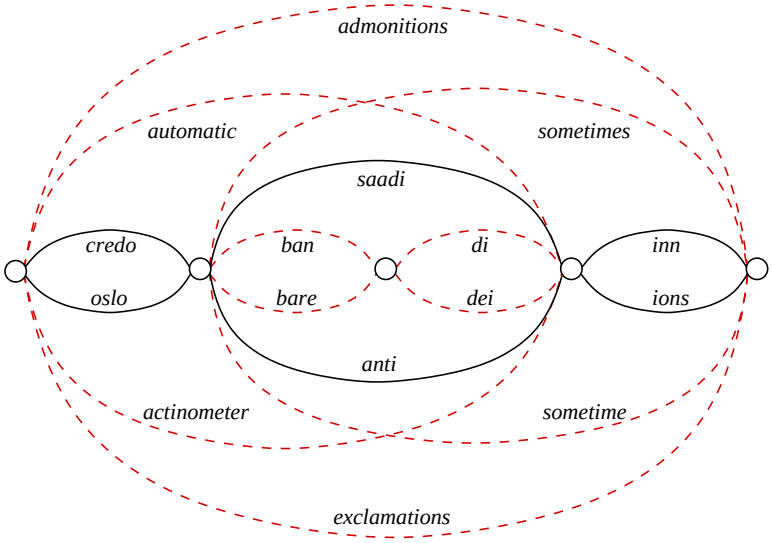


FIG. 6.5 – Exemple de graphe de mots avec un ensemble de confusion étendu.

Dans cet exemple, il y a plus de chemins alternatifs considérés que s'il y avait seulement les trois ensembles de confusion correspondant à la meilleure hypothèses de segmentation (arcs en trait plein, sur la figure). Ainsi, comme la probabilité *a posteriori* d'un mot correspond à la somme des probabilités de tous les chemins auxquels appartient ce mot, normalisée par la somme des probabilités de tous les chemins, la probabilité *a posteriori* des mots sera plus faible dans ce type d'ensemble de confusion.

Par exemple, nous supposons que tous les chemins du graphe de mots de la figure 6.5 sont équiprobables : ce graphe en compte 34 (8 lorsque seuls les arcs de la meilleure segmentation sont considérés). Comme 14 chemins passent par le mot *credo*, sa probabilité *a posteriori* est 0,29, alors qu'elle aurait été égale à 0,5 (4 chemins) si seuls les ensembles de confusion simples étaient considérés (en trait plein, sur la figure).

Les mots longs sont encore plus désavantagés. En effet, la probabilité *a posteriori* du mot *sometimes*, par exemple, est 0,06 puisque seuls 2 chemins passent par ce mot.

En conséquence, même si les mots de ces ensembles de confusion complexes sont correctement reconnus, leur probabilité sera généralement inférieure au seuil σ_{post}. Il faudrait alors modifier l'étape de détection pour ne pas les sélectionner, en n'utilisant plus un seuil absolu mais peut-être un seuil relatif à la probabilité *a posteriori* du mot w_{max2} ayant la deuxième probabilité *a posteriori* la plus élevée. Cet affinement des seuils fera l'objet de perspectives, concernant la détection des potentielles erreurs.

6.5.4 Ajout du mécanisme de rejet des mots absents

Avant de présenter les résultats sur l'ajout de classifieurs de rejet dédiés à chacun des cas, nous expliquons tout d'abord la manière dont se fait le choix des meilleurs classifieur de rejet.

6.5.4.1 Choix des classifieurs de rejet

Pour ajouter le mécanisme de rejet présenté, il faut tout d'abord choisir le classifieur de rejet le plus performant pour notre tâche, parmi ceux qui ont été appris. Les différents classifieurs appris diffèrent par les valeurs des paramètres propres aux SVMs ainsi que par la proportion de mots à reconnaître et à rejeter qui sont considérés lors de l'apprentissage [Mou07]. En effet, faire varier la taille des bases d'exemples (c'est-à-dire les mots à accepter) et des bases de contre-exemples (c'est-à-dire les mots à rejeter) permet de donner plus ou moins d'importance à l'une ou l'autre de ces deux classes.

Pour choisir le classifieur de rejet le plus adapté à notre tâche, nous utilisons des courbes ROC (donnant le TAR en fonction du FAR) ainsi que des courbes indiquant la performance en fonction du FAR, comme proposé dans [Mou07].

La *courbe ROC* permet de trouver le meilleur compromis entre les vraies et les fausses acceptations, l'idéal étant de n'avoir aucune fausse acceptation (FAR=0 %) et d'avoir toutes les vraies acceptations (TAR=100 %).

La *courbe* $Perf/FAR$ est utilisée pour choisir le classifieur de rejet en considérant le taux de reconnaissance obtenu par la classifieur de correction associé. En effet, les performances de ce classifieur de correction peuvent être dégradées s'il y a des faux rejets (*FRR*), c'est-à-dire des mots qui sont rejetés alors qu'ils devraient être acceptés.

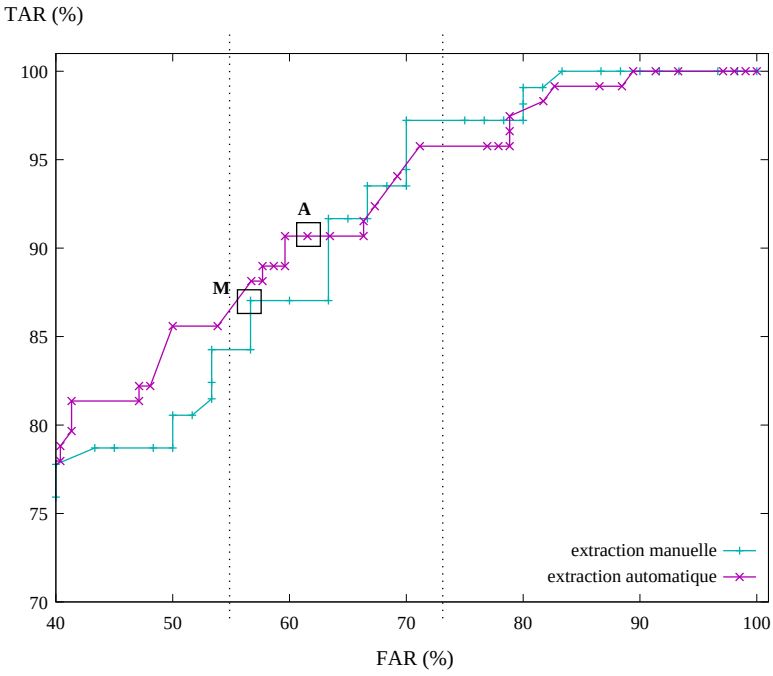
Sur les courbes utilisées par la suite, seuls les meilleurs classifieurs (parmi plus de 1 000 classifieurs appris) seront représentés.

Nous utilisons ces deux courbes pour trouver les classifieurs de rejet qui permettent le meilleur compromis entre le TAR et le FAR, tout en ne diminuant pas trop les performances des classifieurs de correction associés ; cela permet de considérer également le meilleur compromis entre la performance et le FAR.

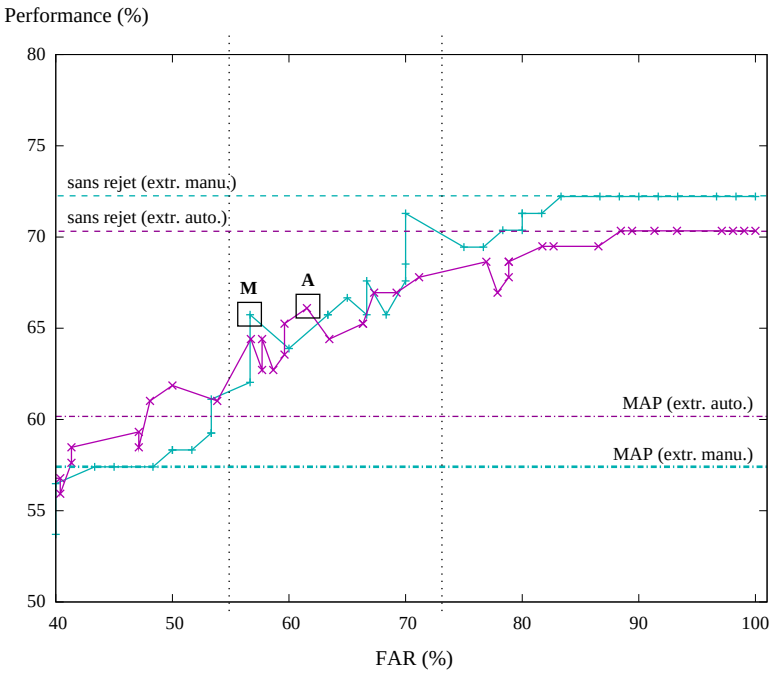
6.5.4.2 Mots de probabilité *a posteriori* non-maximale

La figure 6.6 donne les courbes *ROC* (voir figure 6.6(a)) et $Perf/FAR$ (voir figure 6.6(b)), utilisées pour choisir le classifieur de rejet associé aux erreurs du premier type. Sur chacune des courbes, les classifieurs choisis sont encadrés et étiquetés M et A, selon que la segmentation est manuelle ou automatique.

Le FAR est meilleur pour la segmentation manuelle (56,62 %, contre 61,52 % pour la segmentation automatique) mais le TAR est moins bon (87,06 % contre 90,68 %). Néanmoins, comme les performances sont proches, le fait que le TAR soit inférieur n'a pas d'influence sur le taux de reconnaissance parmi les mots présents (ce taux correspond à la performance). Cela



(a) Courbe *ROC*



(b) Courbe $Perf/FAR$

FIG. 6.6 – Rejet sur les mots de probabilité non-maximale.

peut s'expliquer par le fait que les mots qui ont été rejetés, alors qu'ils auraient dû être acceptés, ne sont vraisemblablement pas correctement reconnus par la suite.

Cela montre l'intérêt d'utiliser conjointement la courbe *ROC* et la courbe $Perf/FAR$, pour choisir les classifieurs de rejet.

6.5.4.3 Mots de probabilité *a posteriori* faible

La courbe *ROC* (voir figure 6.7(a)) et la courbe $Perf/FAR$ (voir figure 6.7(b)), utilisées pour choisir le classifieur de rejet associé aux erreurs du deuxième type, sont illustrées par la figure 6.7. Comme précédemment, les classifieurs choisis sont encadrés et étiquetés M et A, selon le type de segmentation considéré.

La différence entre les performances obtenues avec chacun des types de segmentation est plus importante que précédemment. Le fait que la performance soit plus élevée pour l'extraction automatique des mots vient de la présence d'une majorité de mots correctement reconnus, parmi les mots détectés (dû aux ensembles de confusion plus complexes, comme discuté dans la sous-section 6.5.3).

Les TAR obtenus sont plus élevés que précédemment (95,02 % pour la segmentation manuelle et 98,57 % pour la segmentation automatique) mais les FAR sont moins bons (71,40 % en extraction manuelle et 83,40 % en extraction automatique).

6.5.5 Bilan : combinaison des corrections avec rejet des mots absents

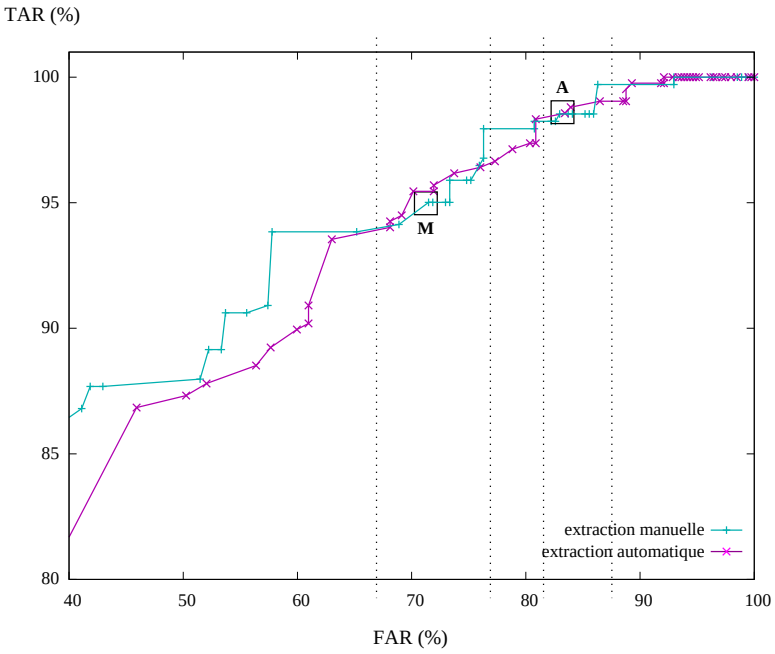
Nous présentons maintenant les résultats globaux, obtenus en combinant les deux stratégies de détection et de correction des erreurs, associées ou non avec leur stratégie de rejet correspondante. La détection cumulée des deux types d'erreurs permet tout d'abord de sélectionner 63,57 % des erreurs de reconnaissance, pour la segmentation manuelle, et 62,12 % des erreurs, pour la segmentation automatique (sur la base $Saisie_{test}$).

La table 6.4 regroupe les différents taux obtenus.

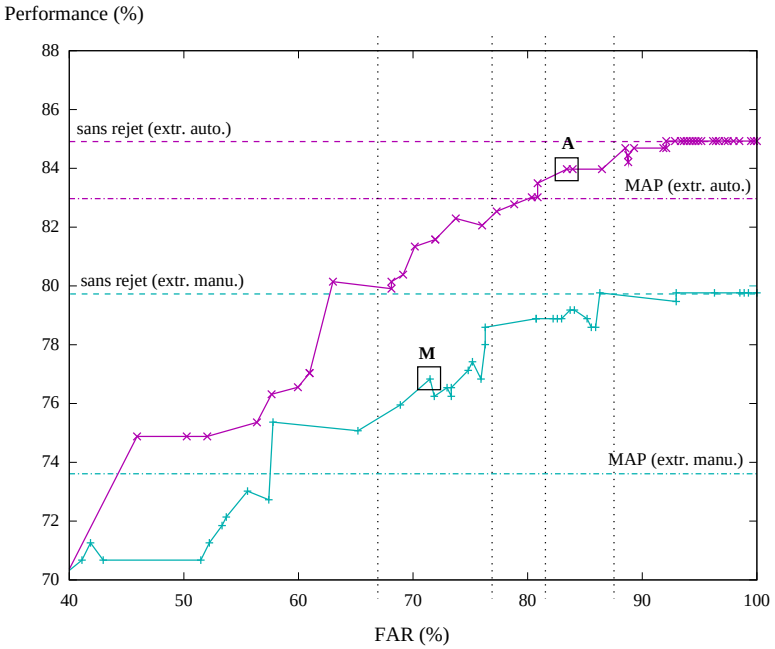
TAB. 6.4 – Taux de reconnaissance pour la correction des erreurs.

Extraction des mots	Approche de reconnaissance	TRM	$TRjM$	TEM_{rej}
Manuelle	MAP	88,48 %	0,00 %	11,52 %
	Détection/correction	89,06 %	0,00 %	10,94 %
	Détection/correction avec rejet	88,79 %	2,37 %	**8,84 %**
Automatique	MAP	86,51 %	0,00 %	13,49 %
	Détection/correction	86,94 %	0,00 %	13,06 %
	Détection/correction avec rejet	86,77 %	1,71 %	**11,52 %**

En ce qui concerne les *taux de reconnaissance*, l'augmentation absolue est de 0,6 %, pour la segmentation manuelle, et un peu plus de 0,4 %, pour la segmentation automatique : cela représente une baisse du taux d'erreur sur les mots de 5,03 % et 3,19 %, respectivement.



(a) Courbe *ROC*



(b) Courbe $Perf/FAR$

FIG. 6.7 – Rejet sur les mots de probabilité faible.

Le taux de reconnaissance atteignable correspond à une bonne correction de toutes les erreurs détectées, en ne considérant que les erreurs qui peuvent être corrigées. Ainsi, pour l'extraction manuelle des mots, il est de 90,62 % alors qu'il est de 88,37 %, pour l'extraction automatique. Comparativement à ces taux de reconnaissance atteignables, les taux de reconnaissance sont améliorés d'environ 25 %.

Lorsque les *stratégies de rejet* sont prises en compte, nous constatons que l'amélioration du taux de reconnaissance est plus faible (de 0,3 % et 0,25 %, en absolu). Cependant, les mots rejetés sont distingués des mots en erreur, les deux taux de rejet étant 2,37 % et 1,71 %, respectivement pour la segmentation manuelle et la segmentation automatique. Ainsi, le taux d'erreur est diminué : la baisse relative est alors importante, puisqu'elle est de 23,26 %, pour la segmentation manuelle, et de 14,60 %, pour la segmentation automatique.

6.6 Bilan

Dans ce chapitre, nous avons présenté une approche permettant de détecter et de corriger des erreurs restantes, à l'issue de la reconnaissance de phrases basée sur une approche MAP. Pour cela, nous utilisons les probabilités *a posteriori* des mots de la phrase MAP, en tant qu'indices de confiance sur le résultat de leur reconnaissance.

Nous avons pris en compte deux types d'erreurs potentielles. La première concerne le cas où la probabilité *a posteriori* du mot résultat ne correspond pas à la probabilité *a posteriori* maximale, dans la liste de mots candidats à laquelle le mot appartient. Dans le second cas, la probabilité *a posteriori* du mot résultat est maximale mais elle est inférieure à un seuil fixé *a priori* : elle est donc jugée comme non suffisamment fiable.

La correction des erreurs ainsi détectées est effectuée grâce à des SVMs, appris pour corriger chacun des deux types d'erreurs. Pour cela, des informations locales et globales sur les mots sont utilisées. Le résultat final consiste alors à choisir entre le mot résultat et un des autres mots de la liste de mots candidats à laquelle il appartient.

Nous avons finalement introduit un mécanisme de rejet afin de pouvoir identifier les erreurs de reconnaissance qui ne pourront pas être corrigées par l'approche proposée. Cela permet également de pouvoir mettre en évidence ces mots, pour effectuer un traitement supplémentaire (mise en évidence dans une interface de saisie ou encore nouvelle passe de reconnaissance).

Nous avons comparé l'apport de l'approche proposée par rapport à l'approche MAP utilisée pour la reconnaissance. Quelle que soit la méthode de segmentation utilisée (manuelle ou automatique), notre approche a permis de corriger des erreurs de reconnaissance et ainsi de diminuer le taux d'erreur sur les mots : la baisse relative est proche de 5 % (segmentation manuelle) et de 3 % (segmentation automatique).

Les résultats obtenus se sont révélés meilleurs lorsque la segmentation est effectuée manuellement. En effet, comme dans ce cas il n'y a pas d'erreurs sur la segmentation, le mot correct se trouve parmi les mots candidats de la liste à laquelle appartient le mot résultat considéré. En revanche, lorsque la segmentation est effectuée automatiquement, il peut y avoir des erreurs de segmentation, lesquelles ne sont pas corrigées par notre approche. Le mot résultat ne peut alors pas être corrigé car le mot correct n'apparaît pas dans sa liste de mots associée.

L'utilisation du mécanisme de rejet permet de diminuer de nouveau le taux d'erreur sur les mots, même si cela entraîne une légère baisse du taux de reconnaissance (par rapport à la correction sans rejet). La baisse relative, par rapport à la reconnaissance avec l'approche MAP sans correction, est relativement intéressante puisqu'elle est proche de 23 % (segmentation manuelle) et de 15 % (segmentation automatique).

Nous avons également dégagé des perspectives tout au long de l'étude de cette approche de détection et de correction d'erreurs de reconnaissance. Tout d'abord, nous pourrons inclure la détection des mauvaises segmentations, soit directement lors de l'étape de détection, soit en utilisant les classifieurs de rejet. En effet, les erreurs de segmentation correspondent typiquement aux cas où le mot correct ne se trouve pas dans la liste de mots candidats. De plus, pour la détection des erreurs potentielles liées à une probabilité *a posteriori* faible, nous avons vu qu'un seuil fixe n'était pas adapté au cas de la segmentation automatique. Il faudra alors affiner ce seuil, afin de ne pas détecter trop de fausses erreurs. Enfin, lors de la phase de correction, il pourrait être intéressant de considérer plus de deux mots en entrée des classifieurs dédiés. Comme nous l'avons déjà mentionné, cela nécessitera l'augmentation des bases d'exemples, afin de pouvoir apprendre les classifieurs dédiés de manière fiable.

Ces différentes perspectives permettront d'affiner et d'approfondir l'approche présentée pour la détection et la correction d'erreurs potentielles.

Conclusion générale

Conclusion

Dans cette thèse, nous avons abordé la reconnaissance d'écriture manuscrite, en nous plaçant au niveau de la reconnaissance de textes manuscrits en-ligne. L'objectif était alors de proposer un système complet de reconnaissance de phrases, en se basant sur le système existant de reconnaissance de mots, RESIFMot. Pour cela, nous avons étudié deux problématiques, liées à la reconnaissance de phrases :

- la *segmentation* des phrases en mots ;
- l'interprétation des mots en contexte de phrase, en intégrant des informations linguistiques grâce à des *modèles de langage*.

Durant notre étude, nous avons pris en compte la combinatoire liée à ces deux problématiques afin de limiter, d'une part, le nombre d'hypothèses de segmentation considérées et, d'autre part, la taille des modèles de langage utilisés.

Nous avons également proposé des méthodes pouvant facilement s'adapter à une évolution du système de reconnaissance de mots. De plus, les méthodes proposées sont généralisables à l'utilisation d'autres systèmes de reconnaissance de mots.

Dans la première partie de cette thèse, nous avons utilisé une représentation des phrases sous forme de graphe de mots. Ces graphes rassemblent à la fois les hypothèses de segmentation et les hypothèses de mots correspondant aux segmentations.

Nous nous sommes d'abord intéressées à l'extraction des mots des phrases, afin de construire les graphes de mots. Dans la première étape de cette segmentation, nous avons proposé une méthode basée sur la caractérisation des espaces entre traces consécutives du tracé de la phrase manuscrite. Cette caractérisation s'appuie sur le calcul d'une distance qui permet d'absorber les variations de style d'écriture et ainsi de s'affranchir d'une phase coûteuse de pré-traitements.

Dans une deuxième étape, nous avons ajouté un indice de confiance sur le résultat de la caractérisation des espaces inter-traces, afin de pouvoir proposer des hypothèses de segmentation alternatives. Cela nous a permis d'étendre l'approche précédente, en utilisant cet indice de confiance pour ajouter des hypothèses supplémentaires de segmentation afin de pouvoir résoudre d'éventuels problèmes de sur- et de sous-segmentation de la phrase. La stratégie proposée permet d'ajouter les hypothèses de segmentation les plus pertinentes, tout en contrôlant la combinatoire liée à la taille du graphe de mots.

Les expérimentations menées ont montré que l'approche présentée respectait ces dernières contraintes puisque les segmentations correctes sont quasiment toutes présentes dans les graphes

de mots (à près de 99 %), ces graphes ayant une taille raisonnable (en moyenne, deux alternatives de segmentation sont présentes pour chacun des mots de la phrase à reconnaître).

Pour pouvoir exploiter au mieux les graphes de mots ainsi construits, nous avons présenté deux approches permettant de tirer partie de l'utilisation de modèles de langage, au cours de la reconnaissance de phrases. Ces deux approches reposent sur l'approche MAP, qui cherche à maximiser la probabilité de la phrase résultat de la reconnaissance.

La première approche proposée permet de combiner les informations syntaxiques (données par les modèles de langage) avec les informations graphiques et lexicales (données par le système de reconnaissance de mots), directement lors de l'exploration du graphe de mots, en prenant en compte l'importance relative de chacune de ces informations ainsi que celle des éventuels problèmes de segmentation.

Dans la deuxième approche proposée, nous avons chercher à tirer partie de la spécificité de différents modèles de langage (plus grande précision, modélisation de connaissances différentes…). Pour cela, nous avons utilisé des modèles de langage, à la fois lors de l'exploration du graphe de mots (comme dans l'approche précédente) mais aussi dans une phase de post-traitement, pour réordonner les hypothèses de phrases issues de cette exploration du graphe. Cela permet l'utilisation de modèles de langage avec des historiques plus longs et qui ne peuvent être intégrés directement durant la reconnaissance sur le graphe de mots.

Les expérimentations menées sur l'intégration des modèles de langage ont permis de comparer des modèles unigrammes, bigrammes et trigrammes ainsi que des modèles biclasses et triclasses, basés sur des classes construites à partir de critères statistiques. Cette comparaison a été faite tant au niveau de la qualité de ces modèles que de leur apport dans la phase de reconnaissance. Nous avons ainsi montré l'influence de la taille du corpus utilisé pour l'apprentissage des modèles de langage ainsi que l'importance de la prise en compte du poids respectif des différentes informations (graphiques, lexicales et syntaxiques) ainsi que des éventuels problèmes de segmentation. L'optimisation de ces paramètres a permis de réduire de moitié le taux d'erreur sur les mots. Nous avons également mis en évidence l'intérêt des modèles biclasses puisqu'ils permettent d'obtenir des performances proches de celles obtenues avec des modèles bigrammes tout en étant plus compacts (en moyenne, deux fois plus petits).

Nous avons finalement étudié l'apport de modèles 4-grammes et de modèles 7-classes à base de classes morpho-syntaxiques, pour réordonner des hypothèses de phrases produites en intégrant un modèle biclasse (à base de classes statistiques) durant la reconnaissance. La combinaison de chacun de ces deux modèles de langage, avec le modèle biclasse, a encore permis d'améliorer les performances du système de reconnaissance (réduction relative du taux d'erreur de près de 5 %). Nous avons également constaté que les deux modèles de langage combinés participaient de manière égale à la reconnaissance finale.

Les travaux concernant l'approche de segmentation des phrases ont été réalisés en collaboration avec François Bouteruche et ont été publiés dans [QBA07]. Les travaux portant sur l'étude de l'intégration de modèles de langage ont été présentés dans [QAC05, QA06].

Dans la deuxième partie de cette thèse, nous avons exploré une nouvelle voie pour aborder l'intégration de connaissances linguistiques, en reconnaissance d'écriture.

Nous nous sommes alors basées sur une représentation originale des hypothèses de phrases, sous la forme d'un réseau de confusion. Cette technique, issue de la reconnaissance automa-

tique de la parole, permet de mettre en évidence les ambiguïtés sur les mots des hypothèses de phrases. La reconnaissance s'appuie alors sur le calcul des probabilités *a posteriori* des mots et la phrase résultat (appelée hypothèse consensus) est celle qui maximise les probabilités *a posteriori* de ses mots.

Nous avons tout d'abord proposé des modifications et des extensions de l'approche initialement utilisée en reconnaissance automatique de la parole, pour pouvoir l'adapter aux spécificités de la reconnaissance d'écriture et de notre tâche de reconnaissance, plus particulièrement. Cette approche de reconnaissance par consensus s'est révélée légèrement moins performante que l'approche MAP. Cependant, nous avons observé que les erreurs commises par les deux approches étaient différentes. Nous avons donc cherché à tirer partie de leurs spécificités.

Pour cela, nous avons alors proposé une approche permettant de détecter de potentielles erreurs de reconnaissance sur les mots de la phrase, résultat de la reconnaissance par l'approche MAP, en utilisant les probabilités *a posteriori* de ses mots en tant qu'indices de confiance. Nous avons considéré deux cas dans lesquels des erreurs potentielles de reconnaissance pouvaient apparaître. Nous avons alors proposé une stratégie de correction, basée sur l'utilisation de classifieurs de type SVM dédiés à chacun de ces deux cas, permettant de confirmer le résultat de la reconnaissance ou de le remettre en cause.

Nous avons finalement ajouté un mécanisme de rejet, basé lui aussi sur des classifieurs dédiés, pour cette fois rejeter le résultat de la reconnaissance s'il est considéré comme non correct et qu'il ne pourra pas être corrigé par l'étape de correction précédemment présentée.

Les expérimentations menées ont tout d'abord permis de réduire le nombre d'erreurs de reconnaissance, en utilisant seulement la correction sur les erreurs potentielles détectées (en moyenne, baisse relative du taux d'erreur sur les mots de 4 %). L'utilisation de l'étape de rejet a permis de réduire encore ces erreurs, en rejetant également certains résultats de reconnaissance qui peuvent alors être mis en évidence ou soumis à une nouvelle étape de reconnaissance (en moyenne, baisse relative du taux d'erreur de 20 %, par rapport à l'approche MAP).

Les travaux exploitant une représentation des hypothèses sous la forme d'un réseau de confusion ont été publiés dans [QA07, QA08].

Perspectives

Nous proposons maintenant des perspectives d'extension du système actuel de reconnaissance de phrases manuscrites.

Lors de la segmentation des phrases, nous détectons la zone du corps des mots, pour pouvoir calculer la distance entre deux traces d'une phrase considérée. Il pourrait être intéressant d'utiliser cette zone de corps, dans le système de reconnaissance de mots (qui se base aussi sur une zone de corps mais calculée seulement sur le mot considéré). Cela permettrait, également au niveau de la reconnaissance des formes, de tirer partie du contexte de phrase dans lequel est écrit le mot, ce qui serait en particulier bénéfique pour les mots courts.

Une des contraintes sur la segmentation concernait une extension possible vers une approche « à la volée », c'est-à-dire au fur et à mesure de l'écriture. De même, nous pourrions

étendre la reconnaissance de phrases à une reconnaissance « à la volée », en étudiant plus particulièrement les modifications nécessaires au niveau de l'intégration des modèles de langage.

Concernant l'intégration de connaissances linguistiques au niveau syntaxique, il pourrait être intéressant d'utiliser des approches plus structurelles, en complément des approches statistiques. En l'occurrence, des techniques issues notamment du traitement automatique des langues (TAL) commencent à être utilisées dans des systèmes de reconnaissance automatique de la parole [HSG06].

L'utilisation de telles informations linguistiques serait encore plus bénéfique dans le cadre de la reconnaissance de textes complets, c'est-à-dire composés de plusieurs phrases. Dans ce cas, il peut aussi être intéressant de tirer partie d'informations sur le thème du texte, en utilisant alors des modèles de langage spécialisés pour un thème donné [IO99] ou en utilisant des modèles de langage à base de cache [KM90]. En effet, ces modèles de langage permettent de prendre en compte dynamiquement ce qui a déjà été écrit mais il faut alors être capable de gérer les éventuelles erreurs de reconnaissance qui peuvent s'introduire dans le cache.

Pour pouvoir bénéficier encore plus de l'apport des informations graphiques (sur la forme des tracés) et linguistiques (au niveaux lexical et syntaxique), nous pourrions nous intéresser à une meilleure combinaison de ces données, en étudiant des méthodes de fusion d'informations. Ces méthodes seraient bénéfiques à la fois pour la reconnaissance par approche MAP et pour la reconnaissance par approche consensus. Dans l'approche consensus, cela permettrait ainsi de mieux combiner les informations, lors du calcul des probabilités *a posteriori* des mots.

En ce qui concerne l'approche consensus utilisée pour la reconnaissance de phrases, nous pourrions également envisager d'autres méthodes de calcul de la probabilité des phrases, à partir des probabilités *a posteriori* des mots. Cela pourrait permettre d'améliorer les résultats, notamment lorsque la segmentation est réalisée automatiquement et que les ensembles de confusion, sur lesquels se base le calcul des probabilités des phrases, sont plus complexes.

Nous pouvons finalement améliorer, à différents niveaux, la méthode de détection et de correction des erreurs potentielles de reconnaissance. Nous pourrions tout d'abord affiner la détection des erreurs potentielles, dans le cas où les probabilités *a posteriori* des mots sont comparées à un seuil fixé *a priori*. En effet, il pourrait être plus efficace d'utiliser une détection plus souple, peut-être de manière plus locale au contexte des mots considérés. Cela éviterait de détecter un trop grand nombre de fausses erreurs.

Concernant maintenant la phase de correction des éventuelles erreurs détectées, nous pourrions également remettre en cause le résultat en considérant cette fois plus de deux mots, en entrée des classifieurs dédiés et utilisés pour cette tâche.

Finalement, en plus de remettre en cause des éventuelles erreurs de reconnaissance, nous pourrions également reconsidérer la segmentation de la phrase résultat (obtenue par l'approche MAP). Pour cela, les mots rejetés peuvent être utilisés puisque les erreurs de segmentation se trouvent parmi les cas d'erreurs potentielles détectées et qui doivent être rejetées. Il faudrait alors trouver une méthode pour discriminer, parmi les erreurs potentielles rejetées, celles qui correspondent à des erreurs de segmentation de celles qui correspondent à des erreurs de reconnaissance. Les erreurs de reconnaissance pourraient également faire l'objet d'un traitement particulier, que ce soit une mise en évidence pour faciliter la relecture du résultat à un utilisateur ou encore une nouvelle passe de reconnaissance, en utilisant d'autres informations.

Annexes

Notations

Modèles de langage

$\phi(h_i)$	: Classe d'équivalence de l'historique h_i
$Couv_n^{ML}$	: Couverture des phrases saisie, à l'ordre n du modèle de langage ML
c_i	: Classe du mot w_i
C	: Nombre de classes de mots
H_{ML}	: Entropie croisée du modèle de langage ML
h_i	: Historique d'un mot w_i
ML	: Modèle de langage
n	: Ordre d'un modèle de langage
Nb_{param}^{ML}	: Nombre de paramètres du modèle de langage ML
$P(w_i\|w_{i-n+1}^{i-1})$	: Probabilité donnée par un modèle n-gramme à la suite de n mots w_{i-n+1}^{i}
POS	: Partie du discours (*part of speech*)
PPL_{ML}	: Perplexité du modèle de langage ML
V	: Vocabulaire

Reconnaissance de mots manuscrits

l_i	: Liste de mots candidats donnée par RESIFMot, pour le signal manuscrit s_i
m	: Nombre de mots candidats dans les listes données par RESIFMot
$score_{graphique}$	: Taux d'adéquation global donné à un mot candidat, par RESIFMot
$score_{lexique}$	: Distance d'édition minimale donné à un mot candidat, par RESIFMot
$score(s_i\|w_i)$	: Score donné au mot w_i résultat de la reconnaissance du signal manuscrit s_i

Reconnaissance de phrases manuscrites

γ	: Poids associé au modèle de langage
δ	: Poids pour gérer les problèmes de sur- et sous-segmentation
M	: Nombre de phrases dans les listes résultat de la reconnaissance de phrases
nb_{arcs}	: Nombre d'arcs dans le graphe de mots
$nb_{arcs}^{présents}$	: Nombre d'arcs du graphe correspondant à des mots à reconnaître

nb_{mots} : Nombre de mots à reconnaître
$nb_{mots}^{présents_m}$: Nombre de mots présents parmi les m mots des listes de RESIFMot
nb_{mots}^{recos} : Nombre de mots correctement reconnus dans les phrases résultat
nb_{mots}^{subst} : Nombre de substitutions de mots entre les phrases à reconnaître et résultat
nb_{mots}^{suppr} : Nombre de suppressions de mots entre les phrases à reconnaître et résultat
$Saisie_{app}$: Base d'apprentissage des phrases manuscrites
$Saisie_{test}$: Base de test des phrases manuscrites
nb_{traces} : Nombre de traces manuscrites
$\hat{W} = w_1^N$: Phrase de N mots, résultat de la reconnaissance par approche MAP
TEM : Taux d'erreur sur les mots
TPM : Taux de présence des mots à reconnaître, dans le graphe de mots
TRM : Taux de reconnaissance sur les mots

Extraction de mots manuscrits

Δx_{ref}^{nouv} : Distance en x entre les traces T_{ref} et T_{nouv}
$\sigma_{reconsider}$: Seuil pour la reconsidération de la classification des espaces inter-traces
DG : Densité du graphe de mots
$diff_{top2}$: Indice de confiance sur la classification d'un espace inter-traces E_{ref}^{nouv}
E_{ref}^{nouv} : Espace entre les traces T_{ref} et T_{nouv}
GRB : Groupe référence de base, composé de traces précédemment écrites
P_{nouv}^{pg} : Point le plus à gauche de la trace nouvellement écrite
P_{ref}^{pd} : Point le plus à droite de la trace de référence
T_{nouv} : Trace nouvellement écrite
T_{ref} : Trace de référence
TPA : Taux de présence des arcs à reconnaître, dans le graphe de mots

Graphes de mots

a_t^{t+x} : Arc du graphe de mots, entre les nœuds n_t et n_{t+x}
G : Graphe de mots
n_t : Nœud du graphe de mots, d'indice t
T : Nombre de nœuds du graphe de mots

Réseaux de confusion

$\alpha(n_t)$: Probabilité *forward* du nœud n_t
$\beta(n_t)$: Probabilité *backward* du nœud n_t
ζ : Poids associé à la probabilité $P(s_i|w_i)$, dans le calcul de la probabilité *a posteriori* du mot w_i
C_G : Réseau de confusion construit à partir du graphe de mots G

L	: Nombre d'ensembles de confusion
$P_{post}(w_i)$	: Probabilité *a posteriori* du mot w_i
$W_{consensus}$	: Hypothèse consensus

Détection et correction d'erreurs

σ_{post}	: Seuil sur les probabilités *a posteriori* des mots, pour la détection d'erreurs
$Base^{nonMax}_{correct}$	: Base d'apprentissage du classifieur de correction, quand une erreur est détectée par la probabilité *a posteriori* non maximale du mot résultat
$Base^{nonMax}_{rejet}$	: Base d'apprentissage du classifieur de rejet, quand une erreur est détectée par la probabilité *a posteriori* non maximale du mot résultat
$Base^{faible}_{correct}$	: Base d'apprentissage du classifieur de correction, quand une erreur est détectée par la probabilité *a posteriori* trop faible du mot résultat
$Base^{faible}_{rejet}$	: Base d'apprentissage du classifieur de rejet, quand une erreur est détectée par la probabilité *a posteriori* trop faible du mot résultat
FAR	: Taux de fausses acceptations (*false acceptance rate*)
mot_{MAP}	: Mot de la phrase résultat de la reconnaissance par approche *MAP*
$nb_{mots\_détectés}$	: Nombre de mots détectés comme potentielles erreurs de reconnaissance
$nb^{recos}_{mots\_détectés}$	: Nombre de mots détectés qui sont correctement reconnus
$nb^{acceptés}_{mots\_détectés}$	: Nombre de mots détectés qui sont acceptés par le classifieur de rejet
$nb_{mots\_présents}$	: Nombre de mots détectés et pour lesquels le mot correct est présent parmi les mots utilisés pour la correction
$nb^{recos}_{mots\_présents}$	: Nombre de mots présents qui sont correctement reconnus
$nb^{acceptés}_{mots\_présents}$	: Nombre de mots présents qui sont acceptés par le classifieur de rejet
$nb^{rejetés}_{mots}$	: Nombre de mots rejetés
$Perf$	: Performance correspondant au $TRMP$
TAR	: Taux de vraies acceptations (*true acceptance rate*)
TEM_{rej}	: Taux d'erreur sur les mots, lorsqu'un mécanisme de rejet est présent
$TRjM$	: Taux de rejet sur les mots
$TRMD$	: Taux de reconnaissance sur les mots détectés à corriger
$TRMP$	: Taux de reconnaissance sur les mots présents dans ceux à corriger

Annexe A

Interface de saisie de phrases manuscrites

FIG. A.1 – Écran d'accueil de l'application de saisie de phrases manuscrites.

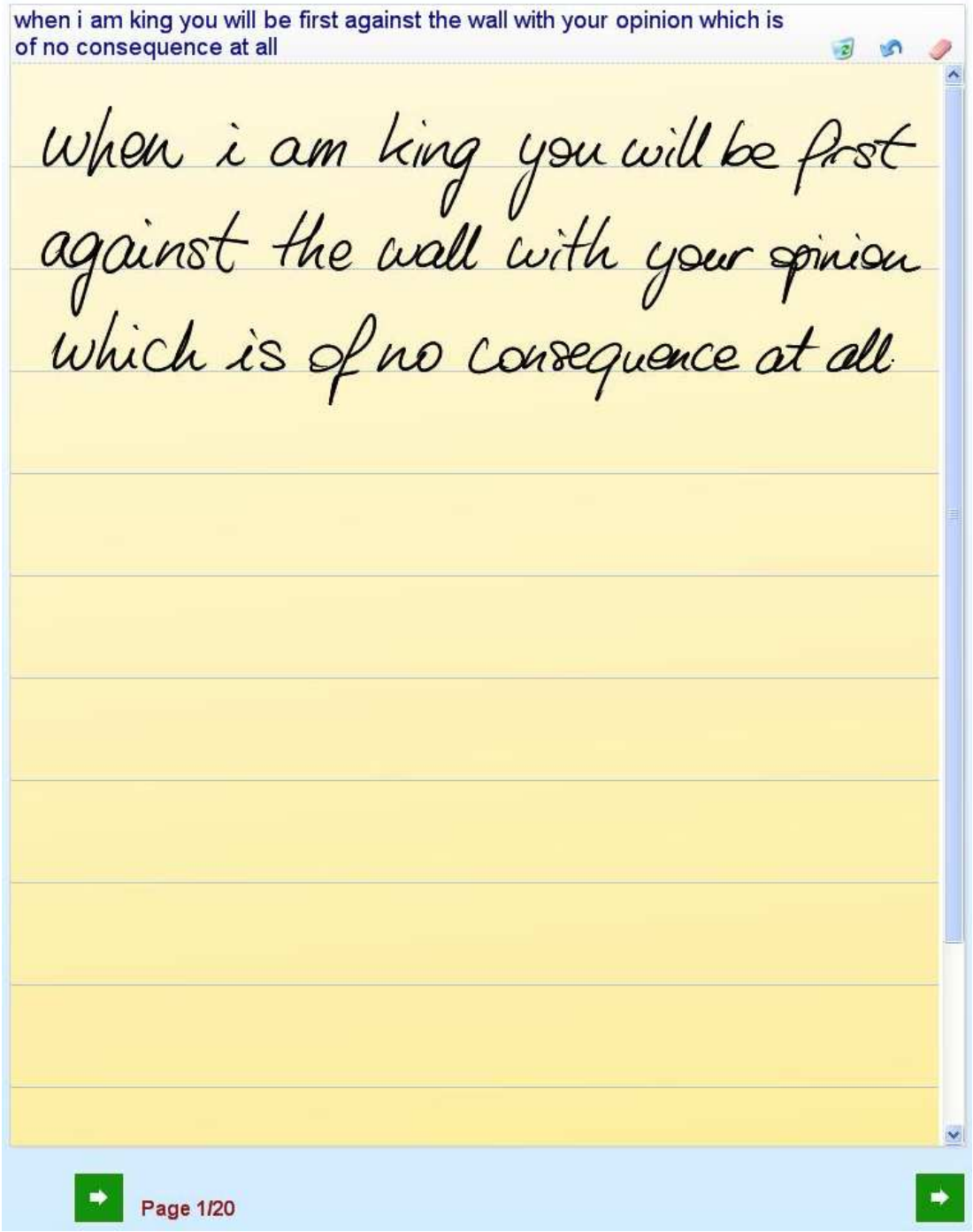


FIG. A.2 – Exemple de saisie d'une phrase manuscrite.

Bibliographie

Publications personnelles

[QA06] S. Quiniou and E. Anquetil. A Priori and A Posteriori Integration and Combination of Language Models in an On-line Handwritten Sentence Recognition System. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR'06)*, pages 403–408, La Baule, France, October 2006.

[QA07] S. Quiniou and E. Anquetil. Use of a Confusion Network to Detect and Correct Errors in an On-line Handwritten Sentence Recognition System. In *9th International Conference on Document Analysis and Recognition (ICDAR'07)*, pages 382–386, Curitiba, Brasil, September 2007.

[QA08] S. Quiniou and E. Anquetil. Utilisation de réseaux de confusion pour la reconnaissance de phrases manuscrites en-ligne. In *16ème Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle (RFIA'08)*, pages 101–108, Amiens, France, Janvier 2008.

[QAC05] S. Quiniou, E. Anquetil, and S. Carbonnel. Statistical language models for on-line handwritten sentence recognition. In *8th International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 516–520, Seoul, Korea, August-September 2005.

[QBA07] S. Quiniou, F. Bouteruche, and E. Anquetil. Word Extraction Associated with a Confidence Index for On-Line Handwritten Sentence Recognition. In *13th Conference of the International Graphonomics Society (IGS'07)*, pages 52–55, Melbourne, Australia, November 2007.

Publications de l'état de l'art

[Anq97] E. Anquetil. *Modélisation et reconnaissance par la logique floueă : application à la lecture automatique en-ligne de l'écriture manuscrite omni-scripteur*. Informatique, Université de Rennes1, Rennes, France, Janvier 1997.

[BA04] J. Blanchard and T. Artières. On-Line Handwritten Documents Segmentation. In *9th International Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*, pages 148–153, Tokyo, Japan, October 2004.

[BB05] R. Bertolami and H. Bunke. Multiple Handwritten Text Line Recognition Systems Derived from Specific Integration of a Language Model. In *8th International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 521–525, Seoul, Korea, August 2005.

[BCP+90] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2) :79–85, June 1990.

[BDVJ03] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(6) :1137–1155, March 2003.

[BHB06] R. Bertolami, B. Halter, and H. Bunke. Combination of Multiple Text Line Recognition Systems with a Recursive Approach. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR'06)*, pages 61–65, La Baule, France, October 2006.

[BM05] S. Broman and M.Kurimo. Methods for Combining Language Models in Speech Recognition. In *9th European Conference on Speech Communication and Technology (Eurospeech'05)*, pages 1317–1320, Lisbon, Portugal, September 2005.

[BPdSL92] P.F. Brown, V.J. Della Pietra, P.V. de Souza, and J.C. Lai. Class-Based N-Gram Models of Natural Language. *Computational Linguistics*, 18(4) :467–479, December 1992.

[BPP96] A.L. Berger, S.A Della Pietra, and V.J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1) :39–71, January 1996.

[BPSW70] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Function of Markov Chains. *The Annals of Mathematical Statistics*, 41(1) :164–171, February 1970.

[BSH04] C.M. Bishop, M. Svensen, and G.E. Hinton. Distinguishing Text from Graphics in On-Line Handwritten Ink. In *9th Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*, pages 142–147, Tokyo, Japan, October 2004.

[BZB06] R. Bertolami, M. Zimmermann, and H. Bunke. Rejection strategies for offline handwritten text recognition. *Pattern Recognition Letters*, 27 :2005–2012, 2006.

[Car05] S. Carbonnel. *Intégration et modèlisation de connaissances linguistiques pour la reconnaissance d'écriture manuscrite en-ligne*. Information, INSA de Rennes, Rennes, France, Décembre 2005.

[CG98] S.F. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Computer Science TR-10-98, Harvard University, Cambridge, Massachussetts, August 1998.

[Cha01] E. Charniak. Immediate-Head Parsing for Language Models. In *39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, pages 124–131, Toulouse, France, July 2001.

[Cho57] N. Chomsky. *Syntactic Structures*. Mouton, 1957.

[CJ00] C. Chelba and F. Jelinek. Structured Language Modeling. *Computer Speech and Language*, 14(4) :283–332, October 2000.

[CL00] Z. Chen and K.-F. Lee. A New Statistical Approach to Chinese Pinyin Input. In *38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, pages 241–247, Hong-Kong, China, October 2000.

[Cla99] P. Clarkson. *Adaptation of Statistical Language Models for Automatic Speech Recognition*. Computer science, Cambridge University, Cambridge, United Kingdom, 1999.

[CV95] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3) :273–297, September 1995.

[Dam64] F.J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. *ACM Transactions on Asian Language Information Processing*, 7(3) :171–176, March 1964.

[DB95] S. Deligne and F. Bimbot. Language Modeling by Variable Length Sequences : Theoretical Formulation and Evaluation of Multigrams. In *20th International Conference on Acoustics, Speech and Signal Processing (ICASSP'95)*, pages 169–172, Detroit, United States, May 1995.

[DD72] J.N. Darrroch and D.Ratcliff. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5) :1470–1480, October 1972.

[EW00] G. Evermann and P.C. Woodland. Large Vocabulary Decoding and Confidence Estimation Using Word Posterior Probabilities. In *25th International Conference on Acoustics, Speech and Signal Processing (ICASSP'00)*, pages 1655–1658, Istanbul, Turkey, June 2000.

[FK79] W.N. Francis and H. Kucera. *Brown Corpus Manual*. Brown University, Providence, United States, 1979.

[For73] G.D. Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3) :268–278, March 1973.

[Goo01] J.T. Goodman. A Bit of Progress in Language Modeling. Machine Learning and Applied Statistics MSR-TR-2001-72, Microsoft Research, Redmond, United States, August 2001.

[HAH+93] X. Huang, F. Alleva, H.W. Hon, M.Y. Hwang, K.F. Lee, and R. Rosenfeld. The SPHYNX-II Speech Recognition System : an Overview. *Computer Speech and Language*, 7(2) :137–148, April 1993.

[HGS07] S. Huet, G. Gravier, and P. Sébillot. Morphosyntactic Processing of N-Best Lists for Improved Recognition and Confidence Measure Computation. In *10th European Conference on Speech Communication and Technology (Eurospeech'07)*, pages 1741–1744, Antwerp, Belgium, August 2007.

[HO06] D. Hillard and M. Ostendorf. Compensating for Word Posterior Estimation Bias in Confusion Networks. In *31st International Conference on Acoustics, Speech and Signal Processing (ICASSP'06)*, pages 1153–1156, Toulouse, France, May 2006.

[HSG06] S. Huet, P. Sébillot, and G. Gravier. Utilisation de la linguistique en reconnaissance de la parole : un état de l'art. Publication interne 1804, IRISA, Rennes, France, Mai 2006.

[HW01] K. Hacioglu and W. Ward. Dialog-Context Dependent Modeling Combining N-Grams and Stochastic Context-Free Grammars. In *26th International Conference on Acoustics, Speech and Signal Processing (ICASSP'01)*, pages 537–540, Salt Lake City, United States, May 2001.

[IO99] R.M. Iyer and M. Ostendorf. Modeling Long Distance Dependence in Language : Topic Mixtures versus Dynamic Cache Models. *IEEE Transactions on Speech and Audio Processing*, 7(1) :30–39, January 1999.

[IOR94] R. Iyer, M. Ostendorf, and J.R. Rohlicek. Language Modeling with Sentence-Level Mixtures. In *7th Workshop on Human Language Technology (HLT'94)*, pages 82–87, Plainsboro, United States, October 1994.

[Jel00] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, 2000.

[JNS01] A.K. Jain, A.M. Namboodiri, and J. Subrahmonia. Structure in On-line Documents. In *6th International Conference on Document Analysis and Recognition (ICDAR'01)*, pages 844–848, Seattle, United States, September 2001.

[JWS+95] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchman, and N. Morgan. Using a Stochastic Context-Free Grammar as a Language Model for Speech Recognition. In *20th International Conference on Acoustics, Speech and Signal Processing (ICASSP'95)*, pages 189–192, Detroit, United States, May 1995.

[Kat87] S.M. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3) :400–401, March 1987.

[Kla98] D. Klakow. Log-Linear Interpolation of Language Models. In *5th, International Conference on Spoken Language Processing (ICSLP'98)*, pages 1695–1699, Sydney, Australia, November 1998.

[KM90] R. Kuhn and R.D. Mori. A Cache-Based Natural Language Model for Speech Reproduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(6) :570–583, June 1990.

[KN95] R. Kneser and H. Ney. Improved Backing-Off for M-Gram Language Modeling. In *20th International Conference on Acoustics, Speech and Signal Processing (ICASSP'95)*, pages 181–184, Detroit, United States, May 1995.

[KP02] D. Klakow and J. Peters. Testing the Correlation of Word Error Rate and Perplexity. *Speech Communication*, 38(1) :19–28, September 2002.

[KS97] T. Kemp and T. Schaaf. Estimating Confidence Using Word Lattices. In *5th European Conference on Speech Communication and Technology (Eurospeech'97)*, pages 827–830, Rhodes, Greece, September 1997.

[Kuk92] K. Kukich. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4) :377–439, December 1992.

[LB07] M. Liwicki and H. Bunke. Handwriting Recognition of Whiteboard Notes – Studying the Influence of Training Set Size and Type. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 21(1) :83–98, February 2007.

[LSB06] M. Liwicki, M. Scherz, and H. Bunke. Word Extraction from On-Line Handwritten Text Lines. In *18th International Conference on Pattern Recognition (ICPR')*, pages 929–933, Hong-Kong, China, August 2006.

[MA06] H. Mouchère and E. Anquetil. A Unified Strategy to Deal with Different Natures of Reject. In *18th International Conference on Pattern Recognition (ICPR'06)*, pages 792–795, Hong-Kong, China, 2006.

[Man00] L.L. Mangu. *Finding Consensus in Speech Recognition*. Computer science, Johns Hopkins University, Baltimore, United States, April 2000.

[Mar04] S. Marukatat. *Une approche générique pour la reconnaissance de signaux écrits en-ligne*. Informatique, Université Paris 6, Paris, France, Avril 2004.

[MB01] 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 (IJPRAI)*, 15(1) :65–90, February 2001.

[MB05] F. Morin and Y. Bengio. Hierarchical Probabilistic Neural Network Language Model. In *10th International Workshop on Artificial Intelligence and Statistics (AIStats'05)*, pages 246–252, Bridgetown, Barbados, January 2005.

[MLN98] S. Martin, J. Liermann, and H. Ney. Algorithms for Bigram and Trigram Word Clustering. *Speech Communication*, 24(1) :19–37, April 1998.

[Mou07] H. Mouchère. *Étude des mécanismes d'adaptation et de rejet pour l'optimisation de classifieurs : application à la reconnaissance de l'écriture manuscrite en-ligne*. Informatique, INSA de Rennes, Rennes, France, Décembre 2007.

[MS99] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, United States, May 1999.

[MS02] I. S. MacKenzie and R. W. Soukoreff. Text Entry for Mobile Computing : Models and Methods, Theory and Practice. *Human-Computer Interaction*, 17(2) :147–198, 2002.

[NEK94] H. Ney, U. Essen, and R. Kneser. On Structuring Probabilistic Dependencies in Stochastic Language Modelling. *Computer Speech and Language*, 8(1) :1–38, January 1994.

[Nie97] T. Niesler. *Category-Based Statistical Language Models*. Computer science, University of Cambridge, Cambridge, United Kingdom, June 1997.

[ONA97] S. Ortmanns, H. Ney, and X. Aubert. A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition. *Computer Speech and Language*, 11(1) :43–72, January 1997.

[Oud03] L. Oudot. *Fusion d'informations et adaptation pour la reconnaissance de textes manuscrits dynamiques*. Informatique, Université de Pierre & Marie Curie, Paris, France, Décembre 2003.

[Owe00] V. Owei. Natural Language Querying of Databases : an Information Extraction Approach in the Conceptual Query Language. *International Journal of Human-Computer Studies*, 53(4) :439–492, October 2000.

[Per05] F. Perraud. *Modélisation du langage naturel appliquée à la reconnaissance de l'écriture manuscrite en-ligne*. Informatique, Université de Nantes, Nantes, France, décembre 2005.

[PS00] R. Plamondon and S.N. Srihari. On-Line and Off-Line Handwriting Recognition : A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :63–84, January 2000.

[PSP06] J.F. Pitrelli, J. Subrahmonia, and M.P. Perrone. Confidence Modeling for Handwriting Recognition : Algorithms and Applications. *International Journal on Document Analysis and Recognition (IJDAR)*, 8(1) :35–46, April 2006.

[PVGM06] F. Perraud, C. Viard-Gaudin, and E. Morin. Language Independent Statistical Models for On-Line Handwriting Recognition. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR'06)*, pages 435–440, La Baule, France, October 2006.

[Ros94] R. Rosenfeld. *Adaptive Statistical Language Modeling : a Maximum Entropy Approach*. Computer science, Carnegie Mellon University, Pittsburgh, United States, April 1994.

[Sam95] G. Sampson. *English for the Computer : The Susanne Corpus and Analytic Scheme*. Clarendon Press, Oxford, 1995.

[SG00] H. Schwenk and J.-L. Gauvain. Combining Multiple Speech Recognizers using Voting and Language Model Information. In *6th International Conference on Spoken Language Processing (ICSLP'00)*, pages 915–918, Pekin, China, October 2000.

[SG04] H. Schwenk and J.L Gauvain. Neural Network Language Models for Conversational Speech Recognition. In *8th International Conference on Speech and Language Processing (ICSLP'04)*, pages 1215–1218, Jeju Island, Korea, October 2004.

[Sha48] C.E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27 :379–423, 623–656, July, October 1948.

[SK97] S.N. Srihari and E.J. Kuebert. Integration of Hand-written Address Interpretation Technology into the United States Postal Service Remote Computer Reader System. In *4th International Conference on Document Analysis and Recognition (ICDAR'97)*, pages 892–896, August 1997.

[SO00] M. Siu and M. Ostendorf. Variable N-Grams and Extensions for Conversational Speech Language Modeling. *IEEE Transactions on Speech and Audio Processing*, 8(1) :63–75, January 2000.

[STHKN95] E.G. Schukat-Talamazzini, R. Hendrych, R. Kompe, and H. Niemann. Permugram Language Model. In *4th, European Conference on Speech Communication and Technology (Eurospeech'95)*, pages 1773–1776, Madrid, Spain, September 1995.

[Sto02] A. Stolcke. Srilm - An Extensible Language Modeling Toolkit. In *7th International Conference on Spoken Language Processing (ICSLP'02)*, pages 901–904, Denver, United States, September 2002. disponible à «http ://www.speech.sri.com/projects/srilm/».

[SWR+03] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning Structure from Freeform Handwritten Notes. In *7th International Conference on Document Analysis and Recognition (ICDAR'03)*, pages 60–65, Edinburgh, Scotland, August 2003.

[VBB04] A. Vinciarelli, S. Bengio, and H. Bunke. Offline Recognition of Unconstrained Handwritten Texts using HMMs and Statistical Language Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6) :709–720, June 2004.

[WBR+97] M. Weintraub, F. Beaufays, Z. Rivlin, Y. Konig, and A. Stolcke. Neural-Network Based Measures of Confidence for Word Recognition. In *22nd International Conference on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 887–890, Munich, Germany, April 1997.

[WSMN01] F. Wessel, R. Schlüter, K. Macherey, and H. Ney. Confidence Measures for Large Vocabulary Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3) :288–298, March 2001.

[XZ05] J. Xue and Y. Zhao. Improved Confusion Network Algorithm and Shortest Path Search from Word Lattice. In *30th International Conference on Acoustics, Speech and Signal Processing (ICASSP'05)*, pages 853–856, Philadelphia, United States, March 2005.

[YIS03] H. Yamamoto, S. Isogai, and Y. Sagisaka. Multi-Class Composite N-Gram Language Model. *Speech Communication*, 41(2-3) :369–373, October 2003.

[YSR+05] M. Ye, H. Sutanto, S. Raghupathy, C. Li, and Michael Shilman. Grouping Text Lines in Freeform Handwritten Notes. In *8th International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 367–371, Seoul, Korea, August 2005.

[ZB04a] M. Zimmermann and H. Bunke. N-Gram Language Models for Offline Handwritten Text Recognition. In *9th International Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*, pages 203–208, Tokyo, Japan, October 2004.

[ZB04b] M. Zimmermann and H. Bunke. Optimizing the Integration of a Statistical Language Model in HMM based Offline Handwritten Text Recognition. In *17th International Conference on Pattern Recognition (ICPR'04)*, pages 541–544, Cambridge, United Kingdom, August 2004.

[ZCB06] M. Zimmermann, J.-C. Chappelier, and H. Bunke. Offline Grammar-Based Recognition of Handwritten Sentences. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(5) :818–821, May 2006.

Résumé

Avec l'émergence des périphériques mobiles tels que les assistants personnels ou les Tablet PC, le besoin de reconnaître l'écriture manuscrite est apparu, afin d'exploiter au mieux ces périphériques qui remplacent l'usage de la souris et du clavier par un stylet. La tâche de reconnaissance d'écriture est cependant complexe et la reconnaissance de textes manuscrits ne peut pas se limiter à la reconnaissance de ses caractères. Il est nécessaire d'utiliser des connaissances linguistiques tant au niveau des mots (grâce à des dictionnaires) qu'au niveau des phrases (à l'aide de modèles de langage).

L'objectif de ces travaux est de construire un système complet de reconnaissance de phrases, en se basant sur un système existant de reconnaissance de mots. Pour cela, deux axes de recherche sont abordés : d'une part, la segmentation de la phrase en mots et, d'autre part, l'intégration de connaissances linguistiques au niveau de la phrase, pour prendre en compte le contexte syntaxique des mots.

La première étape consiste à extraire les mots de la phrase. Pour ce faire, nous présentons une méthode basée sur la caractérisation des espaces entre les tracés manuscrits : cette caractérisation est apprise de manière automatique. Nous étendons cette méthode pour pouvoir considérer des segmentations alternatives, tout en tenant compte de la combinatoire liée au nombre total d'hypothèses de segmentation. Nous utilisons pour cela des indices de confiance sur le résultat de la caractérisation précédente, afin de pouvoir remettre en cause certaines de ces caractérisations d'espaces inter-traces.

La deuxième étape concerne la phase de reconnaissance des mots des phrases. L'objectif est d'utiliser des connaissances linguistiques pour prendre en compte des informations sur le contexte syntaxique des mots. Nous utilisons pour cela des modèles de langage statistiques, permettant de représenter les enchaînements possibles de mots.

Nous cherchons tout d'abord à intégrer ces modèles de langage le plus efficacement possible, lors de la phase de reconnaissance. Nous avons ainsi étudié l'impact de différents modèles de langage puis nous avons combiné ces différents modèles, afin de tirer partie de leurs spécificités.

Afin de pouvoir identifier les erreurs de reconnaissance restantes, nous présentons finalement une méthode originale permettant la détection puis la correction ou le rejet de ces erreurs. L'approche proposée s'appuie sur une représentation des phrases sous forme de réseaux de confusion (technique issue de la reconnaissance automatique de la parole).

L'utilisation des techniques présentées permet de réduire de façon importante le nombre d'erreurs de reconnaissance, parmi les mots des phrases.