Thèse présentée pour obtenir le grade de
DOCTEUR DE L'UNIVERSITE DE BRETAGNE SUD
Mention INFORMATIQUE

par

Igor SCHADLE

Titre de la thèse :

Sibylle : système linguistique d'aide à la communication pour les personnes handicapées

Laboratoire VALORIA EA2593 – Université de Bretagne Sud
VALORIA, BP 573, F-56017 Vannes Cedex, France
Rapport de recherche : VALORIA-CORAIL-2003-03

Soutenue le 18 décembre 2003 devant la commission d'examen composée de :

Denis Maurel	Rapporteur
Jean-Marc Toulotte	Rapporteur
Nadine Vigouroux	Examinatrice
Franck Poirier	Directeur de thèse
Jean-Yves Antoine	Co-directeur de thèse
Brigitte Le Pévédic	Co-directeur de thèse

Résumé

Ce mémoire présente un système d'aide à la communication pour personne handicapée. Le handicap concerne ici des personnes dont l'usage de la parole est altéré et les facultés motrices très réduites. Pour ces personnes, un moyen de communication alternatif est la composition de message assistée. Sur ordinateur, la saisie du texte est réalisée sur un clavier simulé à l'écran, via une commande « tout ou rien ». Ce moyen de communication pose cependant un problème d'excessive lenteur. Dans le système présenté, nous proposons deux aides complémentaires. La première est une prédiction de lettre qui permet de sélectionner plus rapidement les lettres sur le clavier simulé. Après chaque saisie, les lettres sont reclassées sur le clavier simulé afin de présenter en premier les lettres estimées comme les plus probables. Les estimations sont réalisées avec un modèle statistique n-gramme appliqué aux lettres. La deuxième aide, l'aide principale, est une prédiction de mot. Son rôle est d'économiser le nombre de saisies en complétant la fin des mots par l'intermédiaire d'une liste de mots. Pour établir ces prédictions, le modèle de langage que nous proposons intègre des connaissances d'ordre syntaxique aux modèles de langage stochastiques. En particulier ce modèle étend la taille du contexte pris en compte par ces derniers et ce, afin d'en améliorer les capacités prédictives. La prédiction commence par une analyse syntaxique de surface : étiquetage morpho-syntaxique puis segmentation de l'énoncé en *chunks*. C'est ensuite à partir de cette représentation que la probabilité d'apparition des mots est estimée. Ces travaux ont donné lieu à une application, Sibylle, actuellement utilisée par des enfants Infirmes Moteurs Cérébraux au centre de rééducation et réadaptation fonctionnelle de Kerpape.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Handicap et communication

Être privé de l'usage de la parole est un handicap qui peut être compensé avec la langue des signes ou par l'écriture. Cependant, lorsque le handicap est plus grave et atteint les facultés motrices, les modalités de communication deviennent limitées et nécessitent le recours à une aide externe qu' elle soit humaine ou matérielle. Il va sans dire que restaurer, ne serait-ce que partiellement, la fonction de communication, est essentiel pour un individu. Parler, communiquer, c'est permettre l'échange, base de toute société, mais c'est également pouvoir exprimer ses sentiments. Ce besoin de communiquer peut être vital pour des personnes dépendantes comme les enfants, les personnes âgées ou les personnes malades. Outre la perte d'une fonction, un handicap entraîne bien souvent la dépréciation de soi. C'est de plus un facteur d'exclusion dans une société gênée par le mot même d'handicapé.

Un tel handicap, perte ou forte altération de la parole associé à des problèmes moteurs, est la conséquence de multiples maladies. La sclérose latérale amyotrophique (SLA) est une affection dégénérative qui apparaît chez certaines personnes âgées. Elle se caractérise par une perte progressive des facultés motrices et finit par paralyser les muscles utiles à la phonation. Le malade garde tout au long de l'évolution une lucidité et une conscience indemne. L'infirmité motrice cérébrale (IMC) résulte de lésions cérébrales non évolutives survenues au cours de la vie prénatale. Ces lésions induisent des troubles du mouvement et peuvent associer des déficiences sensorielles ou intellectuelles. Le locked-in syndrome (LIS), syndrome d'enfermement, est un état neurologique rare consécutif généralement à un accident vasculaire ou à un traumatisme détruisant le tronc cérébral. Après une phase de coma, le locked-in syndrome se traduit par une paralysie complète qui entraîne une incapacité de parler. Jean-Dominique Bauby atteint de cette maladie suite à un accident, a écrit le livre devenu célèbre « Le scaphandre et le papillon » avec sa paupière gauche. Il y décrit ce sentiment d'un esprit libre enfermé dans un corps sans vie. Cette liste non exhaustive

montre que les causes sont diverses (génétique, accident, etc.), atteignent des populations de tout âge, et sont de gravité diverse. Pour chacune de ces maladies, le handicap physique est sévère, la communication est privée de son support oral habituel et les modalités d'interaction sont limitées. Dans bien des cas, la possibilité de guérison est faible et la rééducation fonctionnelle ne permet de restaurer que quelques gestes élémentaires.

Un moyen de communication alternatif est le recours aux systèmes de communication assistée ou AAC (Alternative and Augmentative Communication). On appelle système de communication assistée tout système de suppléance dont le rôle est d'augmenter ou de restaurer la fonction de communication perdue ou altérée. Si la communauté anglo-saxonne s'est accordée sur le terme de AAC, en France aucun terme n'est officialisé. Le terme d'aide technique, plus général, est également employé. Certains préfèrent la notion d'orthèse cognitive, en référence aux mots de prothèse et de téléthèse. La prothèse est un appareil servant à remplacer un membre ou organe perdu ou défaillant. La téléthèse est un dispositif qui donne au patient une certaine maîtrise de son environnement quotidien. Ses applications pratiques sont les commandes de contrôle de l'environnement. Quant à l'orthèse, elle désigne un dispositif superposé à un membre dans le but d'assister la fonction endommagée. D'une manière générale, ces termes caractérisent des outils permettant de communiquer et s'adressant à des personnes dont la fonction normale de verbalisation est diminuée.

Quel que soit le système de communication assistée, le principe est toujours le même : sélectionner les lettres ou mots ou plus généralement les symboles qui vont composer le message à émettre. Pour décrire plus précisément ces systèmes, nous proposons trois caractéristiques principales : le dispositif de pointage qui sert à la sélection des symboles, la nature des symboles utilisés (lettres, images, etc.) et le type de support utilisé (tableau électronique, outil informatique, etc.). Ces caractéristiques vont être détaillées successivement dans les paragraphes qui suivent.

Le rôle du dispositif de pointage est littéralement de remplacer le doigt pour permettre la désignation des lettres ou des mots sur un tableau par exemple. Ces dispositifs doivent nécessairement s'adapter au geste laissé libre par le handicap. Par exemple, la licorne est une tige légèrement recourbée vers le bas qui se fixe sur le front. Pour une personne tétraplégique, elle permet à l'aide des mouvements de la tête, de désigner, de tourner les pages d'un cahier ou encore d'utiliser un clavier. Dans ce dernier cas, la personne s'aide généralement d'un guide-doigt qui évite l'appui simultané sur plusieurs touches. Par ailleurs, il existe un large éventail de claviers adaptés au handicap : des claviers avec des touches plus grandes, d'autres où les touches sont moins nombreuses, etc. Lorsque le clavier est inutilisable, l'ordinateur et, dans une moindre mesure, tout appareil électronique, offrent d'autres modalités d'interaction. Selon les facultés motrices, il est possible d'utiliser la souris, un

joystick, un trackball. De plus, lorsque les capacités sont très réduites, des commandes de type « tout ou rien » comme le bouton poussoir peuvent suffire à manipuler l'ordinateur. Dans cette catégorie, le souffle peut être utilisé à l'aide d'un mécanisme sophistiqué appelé tout simplement commande par souffle. Enfin, des technologies plus récentes permettent de diriger directement le curseur de la souris à partir des mouvements des yeux (commande oculaire) ou à l'aide d'une pastille fixée sur le front. Si ces dernières technologies sont prometteuses, elles restent pour l'instant limitées car elles imposent de garder une posture fixe fatigante pour bien coordonner le mouvement des yeux ou de la tête avec celui du curseur à l'écran.

La deuxième caractéristique des systèmes d'aide à la communication assistée est la nature du jeu de symboles utilisés pour composer les messages. La plupart des systèmes utilisent l'alphabet orthographique, c'est-à-dire les 26 lettres de l'alphabet romain, plus quelques signes supplémentaires comme l'espace et les ponctuations. Cet usage n'est cependant pas exclusif. Par exemple, pour les enfants ne connaissant pas encore la lecture et l'écriture, l'alphabet orthographique peut être remplacé par l'alphabet phonétique. Bien que celui-ci pose quelques problèmes d'apprentissage pour des personnes qui entendent mais ne peuvent prononcer de sons, il offre cependant plusieurs avantages. Premièrement, le nombre de lettres pour écrire un mot en phonétique est généralement inférieur à celui nécessaire en alphabet orthographique, alors que ces deux alphabets ont une cardinalité comparable. Deuxièmement, sur certains appareils électroniques disposant d'une synthèse vocale comme le SYNTHE 4 (cet appareil est décrit dans le chapitre suivant), la synthèse est réalisée à partir de l'écriture en phonétique.

Les images ou icônes sont également très souvent utilisés, généralement par les enfants ou certaines personnes lourdement handicapées. Un intérêt de la représentation iconographique est qu'elle nécessite moins de symboles pour communiquer. Par exemple, un icône peut suffire pour exprimer la phrase complète « j'ai froid ». Cependant, le gain en nombre de symboles à taper est à relativiser par le temps nécessaire à la sélection des symboles ; un jeu d'images comprenant de quelques dizaines à plusieurs centaines de représentations. La représentation iconographique pose également le problème de la qualité des dessins. Il doit y avoir une bonne adéquation entre le dessin et l'idée à exprimer. Un objet concret comme une maison est facilement représentable. Les concepts abstraits sont déjà plus difficiles à représenter, de même pour les verbes qui désignent souvent une action alors qu'une image est statique. Dans ce dernier cas, l'ordinateur permet cependant d'utiliser des images animées. La liste des difficultés est encore longue et l'iconographie est un sujet de recherche à part entière. Enfin, bien que le jeu de symboles Bliss soit sûrement le système le plus connu et le plus utilisé, il n'existe pas à l'heure actuelle de système de symboles unifié. Ceci est un inconvénient pour le développement de leur utilisation. En effet, lorsqu'une personne change d'outil de communication assistée (par exemple un enfant qui possède son cahier d'images et

passe sur un logiciel de communication assistée sur ordinateur), elle doit souvent réapprendre la signification des images et leurs positions pour les sélectionner.

La dernière caractéristique d'un système d'aide à la communication concerne le support matériel qui affiche la liste des symboles. Ce support est important car il détermine des facteurs comme la disponibilité et l'efficacité.

Le support le plus rudimentaire est le tableau de lettre sur support papier ou le cahier d'images. Si ce support répond bien au critère de disponibilité car il peut être aisément transporté, son inconvénient est qu'il nécessite de la part de l'interlocuteur une aide active. Ce dernier doit en effet suivre pas à pas la composition du message.

Le deuxième type de support concerne les tableaux électroniques. Grâce à un périphérique d'entrée adapté, l'utilisateur sélectionne les symboles du tableau électronique. Un écran affiche la phrase en cours de saisie. Ces appareils permettent généralement de mémoriser les phrases écrites et disposent parfois d'une synthèse vocale. Outre qu'ils offrent d'avantages de modalités pour la sélection des symboles, l'aide active de l'interlocuteur n'est plus nécessaire. Ils comblent donc pour partie l'inconvénient du support papier et redonnent ainsi une certaine autonomie à la personne handicapée. Nombre de ces appareils peuvent être embarqués, fixés au fauteuil roulant par exemple, et répondent ainsi au critère de disponibilité.

L'outil informatique est le dernier support. L'ordinateur est un outil polyvalent qui offre de nombreuses interfaces d'accès. Il permet ainsi d'utiliser un dispositif de pointage adapté au mieux au geste laissé libre par le handicap. De plus, les logiciels développés sur ordinateur sont beaucoup plus performants que sur un appareil électronique, grâce à sa puissance de calcul. À l'heure actuelle, les ordinateurs sont encore lourds et encombrants, ils sont souvent utilisés en poste fixe. Par ailleurs, l'ordinateur portable est difficilement transportable à cause de son clavier qui prend une place inutile. À cela s'ajoute l'autonomie de ses batteries qui est encore insuffisante. Cependant, les évolutions technologiques dans le domaine de l'informatique sont rapides. Le développement d'appareils légers et peu encombrants comme les assistants personnels (PDA pour Personal Digital Assistant) ou les ardoises électroniques est prometteur.

Dans cette thèse, nous présentons un système d'aide à la communication utilisant le support informatique, nous allons donc maintenant exposer plus en détail le fonctionnement de la communication assistée sur ordinateur.

1.2 Communication assistée sur ordinateur

Dans la communication assistée sur ordinateur, on distingue généralement la partie matérielle de la partie logicielle. La partie matérielle est représentée par le

périphérique d'entrée, encore appelé interface d'accès. Cette interface dépend du geste libre laissé par le handicap. Son rôle est de remplacer le clavier réel, dispositif rendu inadapté. Il peut s'agir comme nous l'avons vu précédemment d'un joystick, d'une commande oculaire, d'une commande par souffle, d'un simple bouton poussoir, etc. La caractéristique principale de l'interface d'accès est le degré de liberté qu'elle autorise pour manipuler l'ordinateur. La partie logicielle, quant à elle, réalise à proprement parler le système d'aide à la communication. Elle est composée principalement du clavier simulé (clavier présenté à l'écran) et d'un traitement de texte, ou d'une simple zone d'édition pour afficher le texte en cours (Figure 1.1). Une synthèse vocale permet éventuellement de prononcer le texte composé.

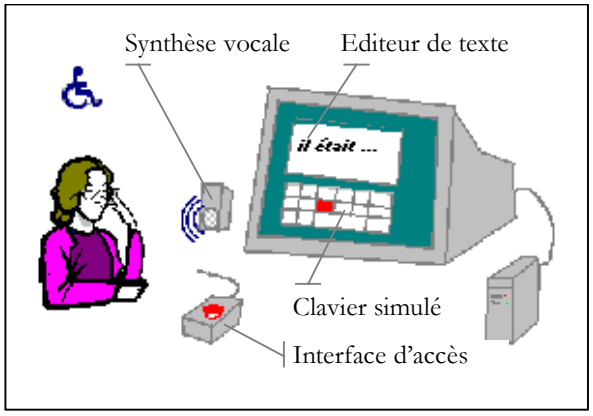


Figure 1.1 : Système de communication assisté sur ordinateur.

La saisie sur clavier simulé est étroitement liée au degré de liberté offert par l'interface matérielle. Dans le meilleur des cas, il est équivalent à celui de la souris et l'utilisateur manipule le pointeur de la souris. Il en va tout autrement pour les commandes « tout ou rien » comme le bouton poussoir qui sont l'équivalent d'un simple clic. Dans ce cas, la sélection des symboles ne peut être réalisée que par un système de défilement automatique du curseur. Le curseur met en surbrillance successivement les touches une à une ; lorsque le curseur désigne la lettre désirée, l'utilisateur n'a plus qu'à valider.

Le problème majeur des systèmes de communication assistée est la lenteur de composition, en particulier dans le cas des commandes « tout ou rien ». La tâche de saisie est généralement longue et fastidieuse. Une personne handicapée avec aide technique peut écrire jusqu'à 5 mots/minute alors que la communication orale est de 150 à 200 mots/minute. À titre de comparaison, le Tableau 1.1 présente la vitesse de quelques modalités de communication [Le Pévédic 1997], [Poirier 1994].

Type de communication	Vitesse (mots/min.)
Communication orale	150-200
Communication écrite	12-25
Saisie secrétaire	20-100
Saisie un doigt	11
Handicapé avec aide technique	5

Tableau 1.1 : Exemples de vitesses de communication (mots/minute)

Pour permettre une saisie plus rapide, deux approches sont envisageables. L'une cherche à accélérer la sélection sur le clavier simulé, l'autre à économiser le nombre de saisies. Ces deux approches sont complémentaires et doivent être envisagées en même temps. En effet, un système de communication assisté efficace réduit le nombre de saisies tout en facilitant la sélection des lettres qui restent à taper. Ces deux approches sont introduites ici rapidement, elles seront exposées de manière plus détaillées dans le Chapitre II sur l'état de l'art.

La première approche pour accélérer la vitesse de saisie cherche à faciliter la sélection des lettres sur le clavier simulé. Elle concerne plus particulièrement la sélection avec défilement automatique. Dans ce cas, l'objectif consiste à minimiser le nombre de défilements pour atteindre la lettre désirée. Ce problème est loin d'être trivial. Les travaux de [Cantegrit 2001] ont ainsi clairement montré que les systèmes de sélection habituellement proposés sont loin d'être optimaux. La méthode généralement proposée par les logiciels du commerce consiste à adopter un balayage des touches optimisé comme le défilement « ligne/colonne ». Une autre possibilité est d'organiser de manière plus judicieuse les touches sur le clavier en utilisant leur fréquence d'utilisation. Enfin, la dernière méthode, la plus sophistiquée, utilise des connaissances de plus haut niveau pour présenter les lettres les plus probables en fonction de ce que l'utilisateur a déjà tapé.

L'objectif de la deuxième approche consiste à économiser le nombre de saisies. Ceci peut être réalisé par différentes méthodes. Le rappel de phrases pré-enregistrées par touche de fonction est efficace mais limité. Il a son utilité pour les phrases types ou pour une communication d'urgence. Un moyen efficace est l'utilisation d'un système d'abréviations, en particulier si le système est capable de décoder l'abréviation [Ricco 2001], ou mieux, de créer des abréviations à la volée [Mc_Coy 1995]. Une autre méthode propose d'élaborer la communication à partir de scénarios ; la saisie est alors partiellement guidée par des schémas prédéfinis, l'utilisateur n'ayant plus qu'à saisir les mots-clé de la phrase [Richardet 1998]. Actuellement, les systèmes les plus performants sont fondés sur une prédiction de mot. Ils permettent

d'économiser jusqu'à 50 % des saisies (ce qui laisse donc encore la moitié des saisies à sélectionner rapidement). Le système VITIPI [Boissière 2000] écrit automatiquement les lettres dès qu'il n'y a plus d'ambiguïté, et le système HandiAS [Maurel 2001a] propose une liste de mots qui évite à l'utilisateur de taper la fin des mots.

Certains logiciels du commerce proposent déjà une liste de mots pour compléter la saisie du mot en cours. Ces logiciels disposent d'un lexique avec la fréquence des mots observée sur corpus. Lors de la saisie, le logiciel extrait du lexique les mots commençant par les premières lettres du mot tapé et les ordonne en fonction de leur fréquence. Par exemple, après la saisie de « le chat attrape la sou », une liste de propositions pourrait être : « sous, souvent, soupirer, soupe, sourds ». On peut aisément constater que cette liste contient des mots qui, replacés dans leur contexte, sont aberrants, et ce, à différents niveaux linguistiques :

- Morphosyntaxique : *le chat attrape la sourds**, ici *sourds* est incorrect pour non respect de l'accord en genre et en nombre
- Syntaxique : *le chat attrape la soupirer**, syntaxiquement, un verbe ne peut suivre un déterminant
- Sémantique : *le chat attrape la soupe**, au niveau du sens, la *soupe* ne peut être attrapée…

Afin d'améliorer l'efficacité de la prédiction, il est ainsi important de doter le système d'un modèle de langage. Au croisement de l'informatique et de la linguistique, se trouve la discipline du Traitement Automatique des Langues (TAL désormais) dont l'objet est la modélisation du langage en vue de son traitement informatique.

En TAL, il existe deux grandes approches pour modéliser le langage, l'approche symbolique et l'approche stochastique. De manière simplifiée, l'approche symbolique fait reposer sa modélisation sur un système à base de règles et de transformations, en nombres limités. L'approche stochastique, quant à elle, utilise des informations d'ordre statistique à partir d'observations sur de grandes masses de données. Si l'objectif est commun, modéliser le langage, les deux approches sont radicalement différentes. La première approche formalise la structure du langage, tandis que la seconde propose plutôt de faire émerger cette même structure à partir de données recueillies empiriquement. La différence ne tient donc pas tant de la conception du matériel étudié, le langage, mais plutôt de l'approche méthodologique. Par ailleurs, il est intéressant de constater que cette opposition se retrouve en informatique dans la modélisation de l'intelligence avec d'un côté l'intelligence artificielle symbolique proposant des systèmes à base de règles comme les systèmes experts et de l'autre des systèmes fondés sur l'apprentissage comme les réseaux de neurones ou la logique floue. Notons cependant qu'en TAL, ces deux approches ont tendance à se rejoindre avec l'apparition de nombreux systèmes hybrides.

Malgré les progrès réalisés, les systèmes d'aide à la communication peuvent être encore améliorés. Tel est l'enjeu des recherches actuelles dans ce domaine. La recherche tend vers une approche pluridisciplinaire associant psychologues, linguistes, ergonomes et informaticiens. Citons parmi les centres de recherche du domaine les universités de Dundee (Écosse), Stanford et Delaware (EU), du Royal Institut KTH (Suède) et en France, les laboratoires du LIM (Marseille), de l'IRIT (Toulouse) et du LI (Tours). Cette thèse, que nous allons maintenant présenter, se situe dans cet axe de recherche.

1.3 Objectifs généraux de la thèse – présentation du système Sibylle

Dans cette thèse, nous présentons un système de communication assistée sur ordinateur utilisant l'alphabet orthographique. Ce système, appelé Sibylle[1], est prévu pour les personnes dont les facultés motrices sont très réduites et qui utilisent comme interface d'accès le bouton poussoir.

Le premier objectif de cette thèse est de permettre une sélection rapide des lettres sur le clavier simulé. Nous proposons pour cela une prédiction de lettre, appelée SibyLettre. La liste des lettres prédites est affichée par le clavier simulé (Figure 1.2). Le deuxième objectif, l'objectif principal, est de concevoir un système de prédiction de mot efficace pour économiser le nombre de saisies. Celui-ci est appelé SibyMot et ses propositions sont accessibles à l'utilisateur par l'intermédiaire d'une liste de mots (Figure 1.2).

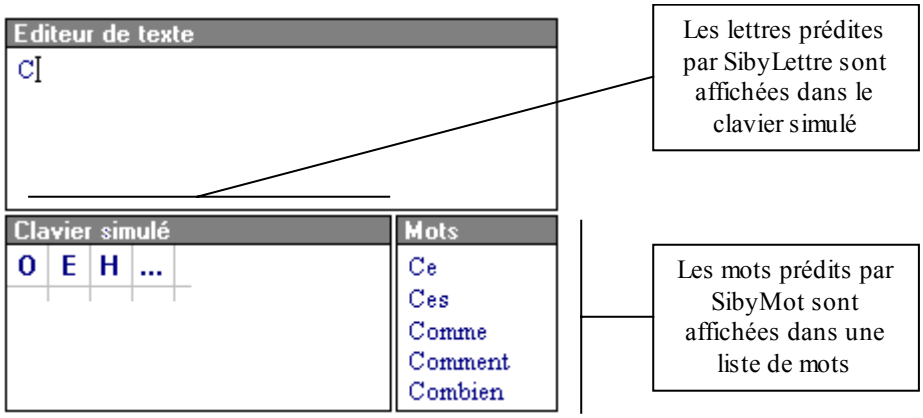


Figure 1.2 : Rôle des prédictions de mot et de lettre dans Sibylle

La partie SibyLettre s'adresse plus particulièrement aux personnes ayant recours à la sélection par défilement automatique. C'est une aide complémentaire à la prédiction de mot qui doit faciliter la saisie des lettres que SibyMot ne permet pas d'économiser.

Le rôle de SibyLettre est donc essentiel pour la saisie des mots inconnus du système et surtout au début des mots, là où le nombre de mots prédits est supérieur à celui que peut contenir la liste des mots affichés. Un des moyens d'optimiser la vitesse de sélection est de classer les lettres en fonction de leur fréquence d'usage dans la langue. Le balayage automatique commence ainsi par passer en revue les caractères les plus fréquents (typiquement l'espace, puis les lettres E, S, A, etc.). Le système que nous proposons, SibyLettre, est une généralisation de ce principe. Les lettres sont classées selon leur probabilité d'apparition, en fonction des lettres déjà tapées du mot en cours. Par exemple, après la saisie de la première lettre C, le système propose en priorité les lettre O et E, ce qui reflète les nombreux mots commençant par CO ainsi que l'usage fréquent du déterminant « ce » et de ses dérivés. L'ordre des lettres est actualisé sur le clavier simulé après chaque. Le clavier était statique, il devient dynamique.

L'aide principale à la saisie fournie par Sibylle est SibyMot. SibyMot est un module linguistique capable à la fois d'analyser une phrase et de délivrer des prédictions. À partir de sa propre analyse du contexte gauche de la phrase (les mots déjà tapés, hors le mot en cours de saisie), SibyMot réalise une estimation de la probabilité d'apparition de chacun des mots contenu dans son lexique. Les prédictions sont établies à partir du classement des mots en fonction de ces estimations. Les propositions sont affinées en élaguant tous les mots qui ne commencent pas par les premières lettres du mot en cours de saisie. Au niveau de l'interface, SibyMot est utilisé avec une liste de mots. Lorsque la saisie d'un nouveau mot commence, les n meilleures prédictions de SibyMot sont affichées. Si le mot souhaité n'est pas contenu dans la liste, l'utilisateur tape une nouvelle lettre. La liste est alors actualisée en affichant les meilleures propositions commençant par cette lettre. Notons que si un mot a déjà été proposé, celui-ci n'est plus proposé (cette option est paramétrable). Quand un nouveau mot est validé, soit par saisie complète, soit par sélection dans la liste, celui-ci est envoyé à SibyMot. SibyMot ré-analyse alors ce nouveau contexte et recalcule ses estimations. Une nouvelle liste de propositions est générée et le processus de saisie se poursuit.

Au niveau modélisation du langage, SibyMot adopte une approche hybride entre les approches symbolique et stochastique. Il s'agit d'un modèle de langage original, réalisant une analyse syntaxique de surface. Celle-ci est chargée d'affecter à chaque mot sa partie du discours (proche des catégories syntaxiques), son éventuelle déclinaison, son lemme. L'analyse segmente ensuite la phrase en groupes de mots. Le système utilise à la fois des informations d'ordre statistique (apprentissage sur corpus) et symbolique (comme les règles qui lui permettent de déterminer la segmentation ou les règles d'accord). L'objectif de ce modèle est de tirer le meilleur parti des techniques stochastiques, c'est à dire leur robustesse (capacité du système à maintenir un bon niveau d'analyse face à des inattendus, des mots inconnus …) et leur performance (à

[1] Sibylle est le nom donné aux prêtresses grecques chargées de prédire l'avenir

l'heure actuelle, la plupart des systèmes opérationnels comme les systèmes de reconnaissance vocale utilisent des modèles de langage stochastiques). L'ajout de connaissances symboliques sert à guider l'organisation des données, à faciliter la gestion des accords et ainsi d'améliorer l'efficacité globale du modèle de langage.

Cette thèse s'effectue en collaboration avec le Centre Mutualiste de Rééducation et Réadaptation Fonctionnelle (CMRRF) de Kerpape. Le centre de Kerpape apporte sa compétence dans le domaine du handicap et son expérience dans les systèmes de communication assistée. Il a notamment participé aux projets d'aide à la communication PVI [Vaillant 1997] et Axelia [Abraham 2000] et contribue à la recherche dans le domaine du handicap. Son rôle est de servir de centre de test et de validation des différents outils qui sont issus du projet. Dans le cadre de cette collaboration, de nombreux patients sont des enfants IMC (Infirmes Moteurs Cérébraux), aux facultés motrices très réduites. Ils utilisent comme interface matérielle le bouton poussoir.

1.4 Organisation du manuscrit

La suite de ce manuscrit est organisée de la manière suivante. Nous avons introduit dans ce chapitre le contexte de ces travaux de thèse. Nous rappelons que dans ce cadre, le handicap est sévère, il concerne les personnes privées de l'usage de la parole et atteintes de lourds déficits moteurs. Pour ces personnes, un moyen de communication alternatif est le recours aux systèmes dits de communication assistée.

Dans le chapitre II, nous présentons un état de l'art de ces systèmes, en nous focalisant principalement sur les outils d'aide à la communication utilisant l'alphabet orthographique et le support informatique. Comme nous l'avons mentionné, il existe deux solutions complémentaires pour accélérer la vitesse de composition des messages. Le chapitre II aborde successivement les différentes techniques pour ces deux solutions. Pour la deuxième partie du chapitre consacrée aux systèmes de prédiction de mot, nous décrivons principalement les systèmes issus de différentes équipes de recherche. La différence entre ces systèmes se situe essentiellement au niveau des modèles de langage utilisés. Le chapitre II fait également état des systèmes existant dans le commerce.

Nous abordons ensuite dans le chapitre III le module destiné à la saisie rapide des lettres : SibyLettre. Ce chapitre présente d'abord le modèle prédictif utilisé par SibyLettre, puis les techniques d'apprentissage ayant été utilisées sur corpus. Cette présentation est suivie d'une première série d'évaluation du modèle et souligne l'intérêt de la prédiction de lettre. La deuxième partie du chapitre présente l'interface que nous avons conçue pour intégrer les prédictions de SibyLettre : le clavier simulé « dynamique ». Cette interface a été validée au centre de rééducation fonctionnelle de Kerpape par des enfants IMC.

Avant d'aborder la deuxième partie de Sibylle, SibyMot, nous présentons dans le chapitre IV un état de l'art des modèles de langage développés en Traitement Automatique des Langues (TAL). Ce chapitre présente d'abord rapidement l'approche symbolique. Nous exposons ensuite de manière plus détaillée l'approche stochastique et les techniques utilisées dans SibyMot.

Le chapitre V est consacré au modèle de langage conçu pour SibyMot. Dans une première partie, nous exposons les principales idées, ainsi que leurs motivations, qui ont guidé la conception du modèle de langage de SibyMot. Le fonctionnement de SibyMot est ensuite décrit en deux parties, avec d'une part, son fonctionnement en mode analyse, lorsqu'il reçoit un nouveau mot, et d'autre part son fonctionnement en prédiction. Ce chapitre se termine avec l'évaluation de SibyMot. Chaque partie de l'analyse est évaluée puis nous donnons les performances globales du système en prédiction.

Dans le chapitre VI, nous présentons l'interface de l'application Sibylle telle qu'elle est actuellement utilisée au centre de Kerpape. Nous y précisons quelques détails de l'ergonomie du logiciel comme l'auto-adaptation à la vitesse de défilement ou « le clic long » qui est une alternative au double-clic de la souris.

Enfin, ce mémoire se conclut avec le chapitre VII qui dresse un bilan des travaux de cette thèse et donne un ensemble de perspectives que nous envisageons pour la poursuite de notre travail.

Chapitre 2

Les systèmes de communication assistée

Nous présentons dans ce chapitre un état de l'art des systèmes de communication assistée qui entrent dans le cadre de notre système Sibylle. Aussi cette présentation est-elle principalement axée sur les systèmes utilisant le support informatique et l'alphabet orthographique.

Ce chapitre est organisé en trois parties. Après un bref rappel historique sur les systèmes d'aide à la communication (partie 2.1), nous exposons successivement l'état de l'art pour les deux approches qui accélèrent la saisie, d'abord les méthodes de sélection rapide (partie 2.2) puis l'économie de saisies (partie 2.3).

2.1 Rappels historiques

Le premier système de communication alternative, conçu comme tel, a sans nul doute été le langage des signes [Zangari 1994]. On en retrouve des traces déjà du temps de la Rome antique dans les travaux de Platon [Levinson 1974] ou encore en Italie au XVIe siècle, où il a été utilisé par les sourds (au XVIIIe siècle en France).

Cependant, ce n'est qu'à partir des années 1950 que la recherche sur les systèmes de communication alternatifs devient un champ de recherche à part entière. Ceci est partiellement dû, d'une part à l'augmentation du nombre de personnes ayant subi des dommages au cours de la deuxième guerre mondiale, et d'autre part aux efforts portés pour améliorer le taux de survie des enfants nés handicapés. Dans la même période, ce qui marque une grande avancée, la langue des signes est officiellement admise comme moyen de communication alternatif au langage oral (un peu plus tardivement en France). Son apprentissage a été progressivement intégré au système éducatif et la langue des signes est de plus en plus fréquemment employée avec les personnes ayant des problèmes d'audition.

À partir des années 70, les résultats issus de la recherche commencent à prendre une part prépondérante dans l'amélioration des systèmes d'aide à la communication. Par exemple, la découverte que les chimpanzés peuvent communiquer avec un système de symboles a été mise à profit pour réaliser des langages à base de symboles pour les personnes ayant des troubles cognitifs [Premack 1974]. Ceci a permis le développement des tableaux de symboles ou encore du langage morse qui sont toujours utilisés de nos jours. Ces aides ont prouvé leur efficacité chez les patients IMC (infirme moteur cérébral) et tout particulièrement chez les enfants.

Toujours dans les années 70, l'électronique a commencé à faire son entrée dans les systèmes d'aide. Par exemple, le système Talking Brooch, développé par Alan Newell et son équipe au Micro Centre de Dundee, permettait d'afficher des mots sur un tableau électronique accroché sur leur épaule [Shank 1977]. De nombreux systèmes de tableaux électroniques portables ont ainsi été développés mais ceux-ci étaient souvent limités dans leurs fonctionnalités. Ces systèmes employaient un clavier à l'image de celui des machines à écrire sur lequel l'utilisateur pouvait taper ses phrases. Généralement le résultat était imprimé ou affiché sur un écran LCD. Le communicateur Canon est un bon exemple de ce type de système. Il a été utilisé avec un certain succès pendant de nombreuses années. Ce type de système devint commercialement viable à la fin des années 70 et les communicateurs de ce genre existent toujours.

À la même époque, les premiers systèmes à balayage ont commencé à faire leur apparition comme le Bliss Symbol Scanner, Illuminaid ou Message Selector. Le système de symbole Bliss a fait son apparition au Canada pour des personnes ayant des déficiences mentales ne leur permettant pas d'écrire correctement avec l'alphabet orthographique. D'autres systèmes de langage symbolique ont suivi suite à des études sur l'iconographie. Le langage Bliss demeure cependant le plus populaire. C'est ensuite dans les années 80 que les premiers projets cherchant à traduire les symboles affichés dans leur équivalent textuel font leur apparition.

L'étape suivante dans l'avancée des aides techniques a été l'ajout d'une synthèse vocale. L'utilisateur a eu ainsi pour la première fois la possibilité d'utiliser une sortie vocale. Celle-ci a augmenté son autonomie : l'interlocuteur n'est plus obligé de lire ou de déchiffrer le message. La sortie vocale a également permis à la personne handicapée de passer des appels téléphoniques. Si les premières synthèses vocales étaient de qualité médiocre, les synthèses d'aujourd'hui atteignent une qualité acceptable.

Enfin, d'autres progrès scientifiques ont permis aux personnes disposant de capacités de mouvement limitées d'utiliser d'autres modalités d'interaction comme les mouvements de leurs yeux (commande oculaire) de leur souffle (commande par souffle) ou des systèmes de pointage comme les faisceaux lumineux.

Depuis les années 90, les systèmes de communication ne cessent de se perfectionner devenant toujours plus efficaces, adaptés et portables et aujourd'hui, de nombreux fabricants offrent un large éventail de systèmes d'aide à la communication.

2.2 La saisie sur clavier

Cette partie présente maintenant la saisie sur clavier. Nous commençons par exposer des méthodes de sélection simples sur tableaux de lettre, puis sur claviers électroniques, et enfin sur claviers simulés. Pour illustrer ces derniers, nous donnons des exemples de logiciels du commerce. Nous verrons, lors d'une étude plus approfondie de la saisie sur clavier simulé, que ces logiciels sont loin d'offrir une vitesse de saisie optimale.

Nous présentons ensuite trois méthodes issues de travaux de recherche permettant d'accélérer la saisie sur clavier. La première propose de mieux agencer les touches du clavier afin de faciliter la saisie, ce qui n'est pas le cas des claviers actuels. La deuxième technique étudiée est celle des claviers ambigus. Un clavier ambigu est un clavier disposant de peu de touches, chaque touche contenant plusieurs lettres, comme sur les téléphones portables. La troisième et dernière technique qui est présentée est celle utilisée par SibyLettre, la prédiction de lettre.

Cette partie se termine par une synthèse des différentes méthodes de sélection sur clavier.

2.2.1 Les tableaux de lettres

Un système rudimentaire peut parfois suffire à aider une personne handicapée et des solutions simples peuvent donner de bons résultats. Il en va ainsi pour le tableau de lettres à désignation manuelle (Figure 2.1). Si la personne dispose de suffisamment de contrôle de ses gestes et si la taille des cases est assez grande, elle peut elle-même désigner les lettres qui vont composer le mot.

A	B	C	D	E	
F	G	H	I	J	
K	L	M	N	O	
P	Q	R	S	T	
U	V	W	X	Y	Z

Figure 2.1 : Tableau de lettres à désignation manuelle

Ces tableaux sont aussi utilisés par les patients atteints de la maladie de Parkinson qui ont des difficultés à maîtriser le flot de leurs paroles. Généralement ces malades parlent trop vite, certains mots étant incompréhensibles. En les obligeant à

désigner sur de tels tableaux la première lettre de chaque mot, ils réduisent ce flot et parlent ainsi à une vitesse correcte. C'est par exemple le principe du système Alphalite [Easton 1994].

Chez certaines personnes handicapées, la paralysie des membres peut laisser la faculté de pointer mais réduire fortement l'ampleur du mouvement. Dans ce cas, la taille d'un tableau de lettres est parfois trop grande. Une solution consiste alors à doter le tableau d'un système de coordonnées (Figure 2.2). Au lieu de désigner directement la lettre, la personne indique une lettre par la valeur de ses coordonnées. Les numéros des coordonnées sont reportés dans un tableau qui occupe un espace plus réduit et permet donc une sélection avec une ampleur de mouvement plus faible. Notons que la saisie d'une lettre demande maintenant deux sélections, la valeur de son abscisse puis celle de son ordonnée.

	1	2	3	4	5	6
I	A	B	C	D	E	
II	F	G	H	I	J	
III	K	L	M	N	O	
IV	P	Q	R	S	T	
V	U	V	W	X	Y	Z

1	2	3	4	5	6
I	II	III	IV	V	

Figure 2.2 : Tableau de lettres avec coordonnées

Lorsque les facultés motrices sont très réduites comme dans le Locked-in Syndrom, la personne ne peut plus désigner elle-même les lettres. C'est alors à l'interlocuteur, ou à une tierce personne, de désigner les lettres. Les lettres du tableau sont parcourues une à une et la personne handicapée acquiesce lorsque le doigt pointe sur la lettre désirée. Ce mécanisme de sélection est cependant très lent. Pour réduire le temps de sélection d'une lettre dans ce type de saisie, certains tableaux adoptent un ordre des lettres très différent de l'ordre alphabétique. Les lettres sont classées par leur fréquence d'utilisation dans la langue ce qui place en priorité les lettres les plus utilisées et relègue en fin de tableau les lettres d'emploi rare. Un exemple d'ordre « fréquentiel » est donné dans la Figure 2.3. Le « e » qui est la lettre la plus fréquente de la langue française est ainsi placé en première position, tandis que le « b » deuxième lettre de l'ordre alphabétique est déplacé beaucoup plus loin.

E	S	A	N	I	
R	T	L	U	O	
D	C	P	M	V	
F	G	Q	B	H	
X	J	Y	K	Z	W

Figure 2.3 : Tableau avec ordre fréquentiel des lettres

C'est avec un tableau de ce type que Jean-Dominique Bauby atteint du Locked-in Syndrom suite à un accident, a écrit son livre « le scaphandre et le papillon », en quatre mois.

2.2.2 Les claviers électroniques

Les claviers électroniques sont généralement des appareils de taille réduite. Ces outils d'aide à la communication fonctionnent sur le même principe que les tableaux de lettres. L'utilisateur saisit une à une les lettres de son message. Cependant, en plus du tableau de lettres, ces appareils disposent d'un petit écran où s'affiche la phrase écrite. La plupart de ces appareils permettent également de mémoriser un certain nombre de phrases ce qui permet à la personne de préparer à l'avance son message, le mémoriser, puis le restituer en temps voulu, voire de l'utiliser plusieurs fois. Enfin, ces dispositifs peuvent être couplés avec une synthèse vocale qui lit leur message.

Synthé 4 est un exemple de ce type d'appareil. Léger (300 g) et de petite taille (10 x 14 cm dans sa version la plus réduite), il peut être fixé à un fauteuil roulant et suivre ainsi la personne dans ses déplacements (Figure 2.4). Une cinquantaine de messages sont enregistrables et restituables par synthèse vocale.

Figure 2.4 : L'appareil électronique SYNTHE 4

L'intérêt principal de ces dispositifs électroniques est que les fonctionnalités supplémentaires qu'ils apportent par rapport à un tableau de lettres, rendent plus d'autonomie à la personne handicapée ce qui améliore leur qualité de vie. Cependant, ils n'apportent pas de solutions au problème principal de ces aides, la vitesse de communication.

2.2.3 Les claviers simulés

Que ce soit sur support papier, sur appareil électronique ou sur ordinateur, la sélection de lettre nécessite un minimum de facultés motrices. Il faut pouvoir contrôler un geste avec une précision suffisante pour désigner une case (ou une touche) et ce, sur une étendue supérieure à l'amplitude de mouvement du doigt ou de la main. Pour les personnes ne disposant pas de ces capacités motrices, un clavier physique est une interface inadaptée. Le clavier simulé apporte une solution à ces personnes. Le clavier simulé, encore appelé clavier virtuel ou clavier à l'écran, est un clavier s'affichant sur l'écran d'un ordinateur et remplaçant le clavier traditionnel. La saisie d'une touche est réalisée en déplaçant le curseur sur la touche puis en validant (ou en laissant le curseur un certain temps sur la touche, selon le mode du logiciel). Le déplacement du curseur est quant à lui dirigé par l'interface physique la plus adaptée : souris, joystick, trackball, commande oculaire…

Les claviers simulés disposent également d'une fonctionnalité appelée défilement ou balayage automatique. Celle-ci est particulièrement utile lorsque l'interface matérielle ne permet que l'équivalent d'un simple clic, comme le bouton poussoir. Le rôle du défilement automatique est le même que celui de la tierce personne évoquée précédemment dans les tableaux de lettre : il désigne successivement chaque lettre du clavier et l'utilisateur n'a plus qu'à valider lorsque la lettre désirée est atteinte. La vitesse de défilement, durée qui s'écoule entre le passage d'une lettre à l'autre, dépend de l'utilisateur. Elle peut varier d'un peu moins d'une seconde (ce qui est très rapide) à plusieurs secondes. Une durée de deux ou trois secondes pour donner le temps à l'utilisateur de valider peut paraître excessivement longue pour un individu non handicapé. Cependant, dans le cas des maladies citées, il existe de nombreux troubles qui perturbent le contrôle du geste ne serait-ce que pour appuyer sur un simple bouton. Par exemple, l'infirmité motrice cérébrale (IMC) se caractérise par des trouble du tonus musculaire (spasticité), ainsi que par des mouvements lents, incontrôlés et involontaires lors de la réalisation d'actions motrices volontaires (athétose). De plus, d'autres troubles atteignent l'émotivité et une vitesse trop rapide entraîne une forme de stress lorsque le curseur s'approche de la touche à sélectionner. Dans ces conditions, la saisie sur clavier simulé est excessivement longue. Avec une vitesse de défilement de deux secondes, atteindre la quinzième touche du clavier nécessite une trentaine de secondes, et saisir un seul mot peut prendre plusieurs minutes.

Sélectionner au plus vite une touche sur un clavier revient à atteindre une case sur un tableau en un minimum de coups, et la méthode qui consiste à parcourir les touches une à une, appelée défilement linéaire, n'est certainement pas la plus efficace. La plupart des logiciels du commerce, comme ceux présentés par la suite, proposent une méthode plus rapide, le défilement ligne/colonne. Avec ce type de balayage, la sélection d'une touche est réalisée en deux étapes. La première étape consiste à sélectionner la ligne dans laquelle se trouve la lettre désirée. Pour cela, le défilement se déroule non pas sur les touches mais sur les lignes : toutes les lettres de la ligne sont mises en surbrillance (Figure 2.5). Lorsque la ligne est sélectionnée, le balayage reprend mais cette fois-ci en parcourant une à une les lettres de la ligne ce qui permet d'atteindre la lettre à taper (Figure 2.6).

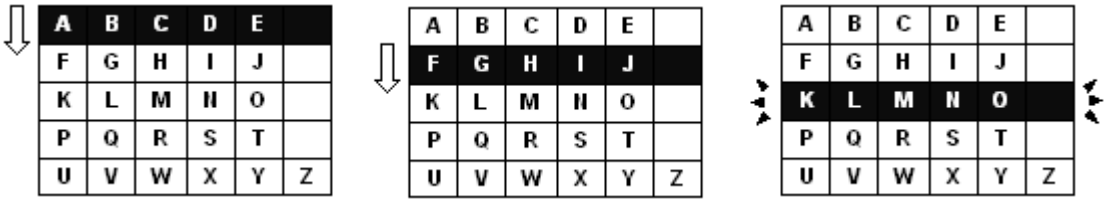


Figure 2.5 : Défilement ligne/colonne : sélection de la ligne

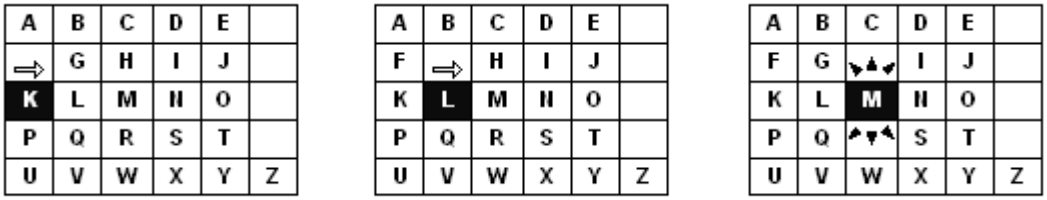


Figure 2.6 : Défilement ligne/colonne : sélection de la lettre

Le défilement ligne/colonne peut être considéré comme une méthode graphique pour désigner une lettre par ses coordonnées. L'inconvénient est le même que le tableau de lettres avec coordonnées : la sélection d'une lettre nécessite deux validations. En particulier, la saisie de la première lettre de chaque ligne nécessite deux validations très rapprochées qui peuvent être sources d'erreurs. Cependant, le défilement ligne/colonne est beaucoup plus rapide pour sélectionner une lettre et c'est sûrement le mode de défilement le plus utilisé.

Il existe un dernier mode de balayage des touches, le balayage bloc par bloc. Ce type de balayage correspond à une approche beaucoup plus méthodique pour atteindre une touche en un minimum de coups. Il utilise pour cela le principe de dichotomie : le clavier est découpé en blocs, les blocs en sous-blocs, le dernier niveau de granularité correspondant aux touches (Figure 2.7). Compte tenu du nombre de

touches d'un clavier standard (généralement une centaine, cf infra), ces trois niveaux d'imbrication suffisent. Cette fois-ci, la sélection d'une touche est réalisée en trois étapes, sélection d'un bloc, puis d'un sous-bloc et enfin d'une touche.

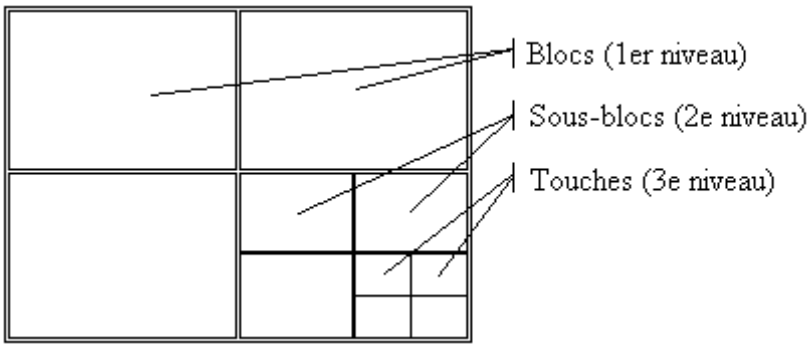


Figure 2.7 : Clavier simulé organisé en blocs

Normalement, ce mode de balayage est le plus rapide mais peu usité dans la pratique. En effet, ce balayage amplifie le problème du défilement ligne/colonne : il sollicite trois validations et ces validations sont très rapprochées. Comme le gain en temps est faible, il ne justifie pas la perte de « confort ».

2.2.4 Les logiciels du commerce

Les logiciels de clavier à l'écran se scindent en deux grandes catégories : les logiciels-claviers dont le rôle principal est d'émuler le clavier réel et les logiciels de communication plus dédiés à la saisie de messages.

Les logiciels-clavier du commerce (Wivik, Ken:x, Mouse Clav, etc.) sont une représentation plus ou moins fidèle du clavier réel (Figure 2.8). Ils contiennent les mêmes touches de fonction qu'un clavier réel pour diriger l'ordinateur.

Figure 2.8 : Logiciel de clavier à l'écran[2]

Lorsque la sélection des touches se fait en balayage automatique, la disposition des touches est modifiée pour adopter une forme plus tabulaire (Figure 2.9).

[2] « Clavier visuel » disponible sous Microsoft Windows XP

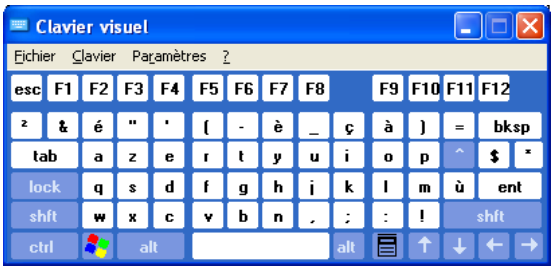


Figure 2.9 : Logiciel de clavier à l'écran : disposition pour le balayage automatique

L'intérêt principal de ces claviers est qu'ils peuvent être associés à n'importe quel type d'application nécessitant une saisie : écriture d'un mail, saisie de la comptabilité dans un tableur, etc. La critique majeure que l'on peut leur apporter est que leur ergonomie n'est pas adaptée. Elle reprend celle des claviers réels conçus pour la saisie dactylographique à deux mains. Parmi certaines aberrations, citons la présence double des touches SHIFT et ALT utilisées pour la combinaison de touches ou encore les larges touches en bordure réservées à l'origine au doigt le plus petit et le moins agile. De même, en balayage automatique, les touches les plus accessibles de la première ligne sont celles des touches de fonction fort peu utilisées, tandis que la barre espace, la touche la plus utilisée sur un clavier, est pratiquement la touche la moins accessible.

Les logiciels de communication sont davantage adaptés à la saisie. Ils contiennent essentiellement les touches des caractères alphanumériques, intègrent une zone de texte. Les touches supplémentaires correspondent à des commandes comme la synthèse vocale, la vitesse de défilement, etc. (Figure 2.10).

Figure 2.10 : Clavier virtuel de logiciel de communication

2.2.5 La vitesse de saisie en balayage automatique

Après avoir décrit différents systèmes de sélection sur clavier simulé, nous proposons dans ce paragraphe une étude plus formelle sur les paramètres qui interviennent dans la vitesse de sélection des touches en défilement automatique. Ces paramètres sont :

- la vitesse de défilement du balayage,
- le type de balayage (défilement linéaire, ligne/colonne ou bloc par bloc),
- le nombre de touches,
- la forme de la matrice des touches,
- l'ordre des touches dans le clavier.

Le premier paramètre, la vitesse de défilement, modifie directement la vitesse de sélection même si cette vitesse reste cependant étroitement liée aux capacités de l'utilisateur. Sous le terme de *type de balayage*, nous distinguons les différentes méthodes de parcours des touches sur le clavier simulé : à savoir principalement les défilements linéaire, ligne/colonne et bloc par bloc. La contribution du nombre de touches à la vitesse de sélection peut se résumer de manière triviale par la formule suivante « moins il y a de touches, plus l'accès aux touches est rapide ». La forme du clavier (carrée ou rectangulaire) peut elle aussi contribuer à accroître la vitesse en fonction du type de balayage. Enfin nous verrons l'importance de l'organisation des touches sur le clavier en fonction de leur fréquence d'utilisation.

La métrique principale utilisée dans cette étude est le nombre de défilements moyens (NDM). Il comptabilise le nombre de pas nécessaires, en moyenne, au balayage pour sélectionner une touche. Dans les résultats, nous rappellerons également le nombre de validations nécessaires. Comme nous l'avons déjà mentionné, valider est parfois difficile pour certaines personnes handicapées et peut être source d'erreur de saisie.

2.2.5.1 Vitesse de défilement et nombre de touches

La vitesse de défilement du curseur dépend essentiellement des capacités de l'utilisateur liées à son handicap (maîtrise du geste, facultés oculomotrices, etc.) ou encore de phénomènes de fatigue dus à la pénibilité de la tâche. Ce paramètre dépend donc plus des facultés de l'utilisateur que du clavier ou du balayage, c'est pourquoi il ne sera pas pris en compte dans la suite de cette étude. Notons cependant que la complexité de la tâche de sélection augmente avec les différents types de balayage, du défilement linéaire (le plus simple) au défilement bloc par bloc (le plus complexe). À vitesse de défilement égale, le défilement ligne/colonne et bloc par bloc sont donc susceptibles de provoquer plus de fautes de frappes que le défilement linéaire ou

entraîner des phénomènes accélérés de fatigue. Une étude expérimentale serait donc plus appropriée pour analyser les effets de la vitesse de défilement.

Quant au paramètre du nombre de touches, nous supposons qu'il a été déterminé et qu'il s'agit d'optimiser la vitesse pour ce nombre donné. Il est régit par deux lois contradictoires : d'un côté il s'agit d'offrir de nombreuses fonctionnalités, et de l'autre de réduire le nombre de touches pour accélérer la saisie. Pour des critères d'efficacité, il est ainsi légitime de se demander si un clavier d'une centaine de touches (comme un clavier réel) est pertinent là où seulement une trentaine de touches suffisent (les lettres de l'alphabet et quelques caractères supplémentaires comme l'espace). Dans cet ordre d'idée, on peut observer l'omission des lettres accentuées dans certains claviers simulés. *A contrario*, un texte sans lettre accentuée peut être considéré comme un message dégradé, mal formé et ne pas satisfaire l'utilisateur. Au final, le choix devrait être toujours laissé à l'utilisateur.

2.2.5.2 Type de balayage

Pour accéder à une touche du clavier en défilement automatique, la méthode la plus simple est de parcourir séquentiellement chaque touche, jusqu'à atteindre la touche désirée. Ce balayage, appelé défilement linéaire (ou séquentiel) est le plus simple mais aussi le plus long. Pour un clavier de T touches, en considérant que chaque touche est équiprobable, le *nombre de défilements moyen* est donné par la formule suivante :

$$NDM = \frac{T + 1}{2} \tag{2.1}$$

Soit, pour un clavier de 64 touches, 32,5 défilements.

En balayage ligne/colonne (ou colonne/ligne), la sélection se fait en deux temps. Dans un premier temps, le balayage s'effectue ligne par ligne, jusqu'à atteindre la ligne contenant la touche désirée. Puis, une fois la ligne sélectionnée, le balayage reprend dans la ligne en parcourant chacune de ses touches. Pour un tel balayage, le *nombre de défilements moyen* devient :

$$NDM = \frac{L + 1}{2} + \frac{C + 1}{2} \tag{2.2}$$

où L est le nombre de lignes du clavier et C le nombre de colonnes

Pour permettre la comparaison avec (1.1), il convient d'exprimer le nombre de colonnes et de lignes en fonction du nombre de touches. Si l'on considère une matrice carrée (organisation optimale, cf infra), $C = L = \sqrt{T}$, (1.2) devient :

$$NDM = \frac{\sqrt{T} + 1}{2} \times 2 \quad (2.3)$$

Un petit exercice rapide avec un clavier de 64 touches (T = 64) permet de montrer l'intérêt du balayage ligne/colonne par rapport au balayage séquentiel. Dans ce cas (et toujours en considérant l'équiprobabilité des touches), le *nombre de défilements moyen* est de 32,5 en défilement linéaire pour seulement 9 en défilement ligne/colonne. Ce gain majeur explique que le balayage ligne/colonne soit fréquemment utilisé dans les claviers simulés.

Il existe un dernier type de balayage qui permet encore d'améliorer la vitesse d'accès aux touches, il s'agit du balayage bloc par bloc. Dans ce balayage, le clavier est composé de blocs, eux-mêmes composés de sous-blocs permettant d'accéder finalement aux touches. Cette fois-ci, la sélection d'une touche est réalisée en trois étapes, sélection d'un bloc, puis d'un sous-bloc et enfin d'une touche. L'organisation optimale de ce type de balayage est obtenue lorsque le clavier contient autant de blocs qu'un bloc ne contient de sous-blocs et qu'un sous-bloc de touches. Par exemple, pour 64 touches, le clavier doit être décomposé en 4 blocs de 4 sous-blocs de 4 touches. Quand cette organisation est respectée, le *nombre de défilements moyen* est :

$$NDM = \frac{\sqrt[3]{T} + 1}{2} \times 3 \quad (2.4)$$

De manière formelle, la formule du *nombre de défilements moyen* peut être généralisée en introduisant explicitement le nombre de niveaux utilisés pour sélectionner une touche (un en défilement linéaire, deux en ligne/colonne, etc.). Pour N niveaux, un clavier est organisé de manière optimale si chaque niveau est composé de $\sqrt[N]{T}$ sous-niveaux. La formule suivante donne le nombre de défilements moyen généralisé pour une organisation optimale et l'équiprobabilité des touches :

$$NDM = \frac{\sqrt[N]{T} + 1}{2} \times N \quad (2.5)$$

Le tableau suivant (Tableau 2.1) donne quelques exemples de *nombre de défilements moyen* selon le type de balayage (défilements linéaire, ligne/colonne et bloc par bloc) et pour différentes tailles de clavier. La dernière colonne Niveaux rappelle le nombre de pas utilisés avant d'accéder à une touche, ce qui correspond au nombre d'appuis nécessaires pour sélectionner une touche.

Type de balayage	Nombre de touches			Niveaux
	25-27	64	121-125	
Défilement linéaire	13,5	32,5	62,5	1
Défilement ligne/colonne	6	9	12	2
Défilement bloc par bloc	4,5	7,5	6	3

Tableau 2.1 : NDM selon le type de balayage et la taille du clavier

Les chiffres de ce tableau montre le saut qualitatif entre le défilement linéaire et le défilement ligne/colonne. Le défilement ligne/colonne est cinq fois plus rapide sur un clavier standard (NDM = 12 pour 62,5 au défilement linéaire) et même deux fois plus rapide sur un tableau d'une vingtaine de lettres. Quant au défilement bloc par bloc, l'ajout d'une validation supplémentaire le pénalise et son utilisation devient intéressante à partir d'un très grand nombre de touches (cent et plus) comme c'est le cas pour un clavier standard.

2.2.5.3 Matrice des touches

Lors des calculs précédents, nous avons signalé que nous supposions une forme optimale de la matrice des touches, par exemple une matrice carrée pour le balayage ligne/colonne. Bien que ce paramètre ait peu d'influence, il peut cependant devenir non négligeable en cas de forme disproportionnée.

Un clavier simulé recopiant la forme du clavier réel « 102 touches » permet d'illustrer ce cas. Pour un tel clavier, les touches sont disposées sur 6 lignes avec des lignes de longueur variable allant jusqu'à 21 touches. Ainsi, en balayage ligne/colonne, le *nombre de défilements moyen* pour la disposition 6 lignes x 17 colonnes (= 102 touches) est de 12,5 contre seulement 11 avec une disposition 10 x 10, soit une différence supérieure à 10 %.

Il convient bien sûr de noter que la forme du clavier obéit également à des critères ergonomiques.

2.2.5.4 Fréquence d'utilisation des touches

Le dernier paramètre étudié concerne la fréquence d'utilisation des touches. La plupart des claviers simulés des logiciels du commerce ne tiennent pas compte de ce paramètre en présentant les lettres dans l'ordre « AZERTY » ou alphabétique. Pourtant, l'idée de classer les lettres selon leur fréquence n'est pas nouvelle et les tableaux de lettres cartonnés utilisent déjà l'ordre « ESARIN » correspondant à l'ordre décroissant des fréquences d'apparition des lettres en français.

Les travaux de [Cantegrit 2001] proposent une étude sur le *nombre de défilements moyen* en fonction de l'ordre des lettres (AZERTY, alphabétique et fréquentiel), la forme de la matrice des touches (3 x 10, 4 x 8 et 5 x 6) et le type de balayage (défilement séquentiel et ligne/colonne). Le clavier est supposé contenir les 26 lettres de l'alphabet ainsi que l'espace et quelques ponctuations. Le tableau suivant (Tableau 2.2) donne les valeurs obtenues pour une matrice 5 x 6.

	Équiprobabilité	Ordre fréquentiel
Défilement linéaire	15,5	6,86
Défilement ligne/colonne	10,83	4,29

Tableau 2.2 : Nombre de défilements moyen pour un clavier de *lettres* disposé en 5 x 6

Ces résultats rappellent l'intérêt du défilement ligne/colonne sur le défilement linéaire (*le nombre de défilements moyen* en utilisant l'ordre fréquentiel est près de 7 en défilement linéaire contre un peu plus de 4 en défilement ligne/colonne). Ils montrent surtout que classer les lettres en fonction de leur fréquence est beaucoup plus pertinent que supposer leur équiprobabilité. Le *nombre de pas moyen* passe ainsi de 10,83 à 4,29 en défilement ligne/colonne.

Cette étude a aussi porté sur les ordres standards 1) AZERTY disposé en 3 x 10 et 2) alphabétique en 4 x 8. En tenant compte de la probabilité des lettres, les résultats donnent respectivement 7,27 et 6,19 en défilement ligne/colonne à comparer au 4,29 obtenu en utilisant l'ordre fréquentiel. Ces résultats montrent que l'ordre AZERTY est nettement moins performant (et donc moins adapté) que l'ordre fréquentiel.

Pour permettre le calcul du *nombre de défilements moyen*, il faut maintenant tenir compte de la probabilité d'utilisation de chaque touche et de leur position relative :

$$NDM = \sum_{i=1}^{T} P(t_i) \times D(t_i) \qquad (2.6)$$

où $P(t_i)$ est une estimation de la probabilité de saisie de la touche t_i, et $D(t_i)$ le nombre de défilements nécessaires pour l'atteindre.

D'après cette dernière formule (1.6), pour minimiser le *nombre de défilements moyen*, les lettres dont la probabilité d'apparition $P(t_i)$ est la plus élevée doivent être placées sur les touches dont l'accès nécessite le moins de défilements $D(t_i)$. Ainsi, sur clavier statique, le *nombre de défilements moyen* est minimal pour l'ordre fréquentiel, ce qui n'est ni le cas de l'ordre alphabétique, ni de l'ordre AZERTY.

Cette idée peut elle-même être prolongée en utilisant non plus les fréquences des lettres hors contexte, mais les fréquences en contexte, en fonction des saisies

précédentes. D'un clavier statique, on passe alors à un clavier dynamique. C'est le principe de SibyLettre que nous présenterons par la suite.

2.2.6 Les claviers issus de la recherche

Nous venons de montrer qu'il est possible de rationaliser les claviers simulés actuels pour obtenir des gains substantiels en vitesse de saisie. Nous allons maintenant présenter trois méthodes issues de travaux de recherche dont l'objectif est d'accélérer la saisie sur clavier.

2.2.6.1 Organisation optimisée des touches

Le problème de la saisie rapide sur clavier n'est pas nouveau et ne concerne pas que les personnes handicapées. Les claviers AZERTY (ou QWERTY dans les pays anglo-saxons) ont un ordre des touches peu adapté à la saisie rapide. Cet ordre correspond même à une préoccupation opposée. Il a été conçu pour que les marteaux qui tapaient les lettres des machines à écrire ne s'entrechoquent trop fréquemment. Ainsi les lettres formant des paires ou des triplés très fréquents dans la langue ont été éloignées pour des raisons mécaniques, alors que leur saisie est au contraire plus rapide si elles sont rapprochées. Les claviers d'ordinateur, et maintenant les claviers virtuels, héritent donc d'une disposition de touches qui n'a plus lieu d'être.

Le clavier Dvorak est un clavier dont la disposition des touches a été étudiée pour la saisie rapide. Les touches sont organisées afin de minimiser le déplacement des doigts entre les rangées, de rapprocher des lettres se succédant fréquemment… Si un tel ordre n'est pas directement exploitable pour accélérer la saisie en balayage automatique, il pourrait par contre être utilisé pour minimiser le déplacement de la souris. L'idée du clavier Dvorak est d'organiser de manière optimale les touches. L'ordre qu'il propose n'est cependant pas le meilleur. À l'heure actuelle, il n'existe pas un ordre considéré comme optimal pour la saisie. Un calcul systématique pour chaque organisation de clavier dépasse les capacités de calcul actuelles et des heuristiques sont employées dans les algorithmes. Nous présentons dans les paragraphes qui suivent deux de ces « claviers optimisés ».

2.2.6.1.1 Le clavier KNITS

L'équipe de recherche du centre Enkidu à New York a développé un clavier réarrangé. Le constat initial est le même sur l'ordre inadapté des claviers QWERTY. L'objectif de leur recherche est de créer un clavier virtuel adapté à la sélection de touche pour personne handicapée [Lesher 2000].

La méthode qu'ils proposent pour trouver une organisation optimale (ou du moins proche) est la suivante. À chaque transition entre deux touches du clavier est affecté

un coût qui dépend principalement de la distance entre le centre des deux touches. L'efficacité d'un clavier est calculée en sommant le coût de ces transitions, à partir des fréquences ce co-occurrence des lettres recueillies sur corpus (3 millions de mots dans leur étude). À partir d'un clavier au format donné (grille 5 x 6 par exemple) et d'une disposition initiale aléatoire des touches, l'algorithme proposé permute chaque couple de lettres. Après chaque permutation, l'efficacité du clavier est recalculée. Si le coût total obtenu est plus faible, la permutation est conservée. Cet algorithme est connu sous le nom de n-opt [Lin 1965] (n = 2 dans le cas de la permutation de couples). L'algorithme appliqué avec une permutation de triplets (n = 3) donne des résultats proches. Leur étude conclut que l'ordre trouvé est très proche de l'ordre optimal.

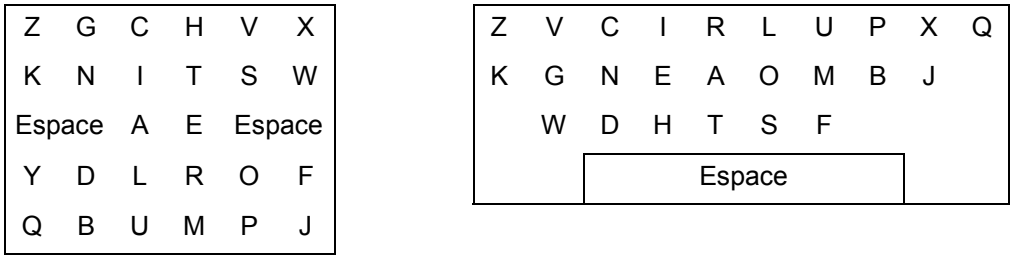


Figure 2.11 : Le clavier optimisé KNITS, matrices 5 x 6 et 4 x 10

La Figure 2.11 présente les dispositions obtenues pour deux formes de clavier, en grille 5 x 6 et en clavier standard sur 4 lignes. Pour la grille 5 x 6, l'ordre des lettres KNITS (en référence à la deuxième ligne) réduit de 32 % le coût des déplacements par rapport à l'ordre alphabétique. Dans la configuration 4 x 10, la réduction est de 35 % par rapport à l'ordre QWERTY.

2.2.6.1.2 Le clavier Metropolis

Le clavier Metropolis [Zhai 2000] est issu du centre de recherche IBM Almaden (San-José, Etats-Unis). Ce clavier est destiné à des appareils utilisant la combinaison d'un clavier virtuel et d'un dispositif de pointage pour la saisie, par exemple pour un assistant numérique (ou PDA, Personnal Digital Assistant). L'objectif sous-jacent est cependant le même : à savoir minimiser la longueur des déplacements entre touches.

Le modèle utilisé pour estimer l'efficacité d'un clavier diffère légèrement de l'étude précédente. Il reprend celui proposé par I. S. Mac Kenzie et son équipe [Mac Kenzie 1995]. Le principal changement réside dans l'estimation du coût de transition entre deux touches. La fonction de coût utilise la loi de Fitts [Fitts 1954], [Ferrand 1997] bien connue en ergonomie des interfaces. Celle-ci stipule que le temps de déplacement d'un point initial à une zone d'arrivée (l'espace occupé par une touche par exemple) dépend de la distance à parcourir et de la taille de la zone de destination, selon la formule :

$$T = a + b \, \mathrm{Log}_2 \, (D_{ij} \, / \, W_j + 1) \qquad (2.7)$$

Où T est la durée du déplacement, D_{ij} la distance entre les deux points, W_j la taille de la cible (Figure 2.12) et a, b deux constantes dépendant des capacités de l'utilisateur.

Figure 2.12 : Les paramètres de la loi de Fitts

En estimant la probabilité P_{ij} de taper la touche j après la touche i pour chaque couple de lettres (i, j) (avec l'espace), on peut obtenir le temps moyen de saisie t d'une lettre :

$$t = \sum_{i=1}^{27} \sum_{j=1}^{27} P_{ij} \times [a + b \, \mathrm{Log}_2 (D_{ij} \, / \, W_j + 1)] \qquad (2.8)$$

L'algorithme d'ordonnancement est quant à lui radicalement différent de celui vu précédemment pour le clavier KNITS. Le clavier est considéré comme un système dynamique où chaque touche interagit avec les autres. Le degré d'interaction entre deux touches dépend de leur probabilité de co-occurrence selon une fonction reprenant la loi de Fitts. L'objectif de minimiser la longueur des déplacements revient alors à chercher le système avec un minimum d'energie. Sans entrer dans les détails, l'algorithme appliqué est celui de Metropolis, de type Monte Carlo, couramment utilisé en physique statistique pour déterminer un état d'énergie minimale [Binder 1988]. C'est de cet algorithme que le nom du clavier Metropolis est extrait.

La Figure 2.13 illustre un des claviers obtenus. Celui-ci a la particularité d'avoir les voyelles organisées le long des médianes d'un triangle virtuel. La touche Espace, le caractère le plus utilisé, est au centre.

X	K	W	M	U	Q	‘
C	H	T	O	F	Z	
J	I	E	Esp	N	G	B
V	R	S	A	D	Ret	
,	X	P	L	Y	Sh	

Figure 2.13 : Clavier Metropolis

Les résultats sont présentés dans le Tableau 2.3 (et extraits de [Zhai 2000]). Le clavier Metropolis y est comparé au clavier de référence QWERTY ainsi qu'à d'autres

claviers utilisant un ordre optimisé des touches : le clavier FITALY (produit commercial développé par Textware Solutions) et le clavier OPTI [Zhang 1998].

Clavier	Performance (mots par minute)
QWERTY	30
FITALY	36
OPTI	38
Metropolis	43

Tableau 2.3 : Performances de claviers optimisés

Bien que les résultats semblent être clairement en faveur du clavier Metropolis (43 mots/minute, contre 38 mots/minute pour le clavier OPTI et 30 pour l'ordre QWERTY), la comparaison est à relativiser. Chaque clavier a été conçu selon des méthodes d'estimation de coût différentes et avec des estimations de cooccurrences de lettres différentes. En particulier, la loi de Fitts est plus difficile à appliquer pour des touches rectangulaires comme la barre d'espace. Or ce caractère représente entre 15 % à 20 % des caractères d'un texte ! Ceci souligne la difficulté de la modélisation de la saisie.

Le clavier FITALY et OPTI ont été également évalués auprès d'utilisateurs. L'évaluation du clavier FITALY a donné 44,4 mots/min contre 28,2 pour le clavier QWERTY [TextWare 1998]. Quant à l'évaluation du clavier OPTI, [Mac Kenzie 1999] rapporte une vitesse de 44,3 mots/minute. Toutes ces études semblent donc s'accorder sur une vitesse autour de 30 mots/minute pour le clavier QWERTY et près de 45 mots/minute pour un clavier optimisé, soit un gain de près de 50 % en vitesse de saisie.

2.2.6.2 Les claviers ambigus

La saisie de lettre sur téléphone portable est un autre problème dont les solutions sont exploitables pour la saisie sur clavier simulé. Sur un téléphone, chaque touche contient trois ou quatre lettres. Le nombre d'appuis successifs sur la même touche permet la sélection entre les différentes lettres de cette touche. Pour taper deux fois la même lettre, il faut attendre un certain laps de temps avant de ré-appuyer sur la même touche.

Le système T9™[3] utilisé aujourd'hui sur la plupart des téléphones portables est une petite révolution : il permet d'appuyer une seule fois sur la touche de la lettre désirée même si celle-ci n'est pas en première position sur la touche. Le système se

[3] Le système T9™ est développé par la société Tegic Communications. Le sigle T9 signifie « Taper avec 9 touches ».

trouve face à une ambiguïté (la touche correspond à plusieurs lettres) qu'il lève en contexte avec les lettres précédentes du mot en cours. L'algorithme est un parcours de graphe composé par les mots du lexique. La réponse donnée correspond au mot le plus fréquent dans la liste des possibles.

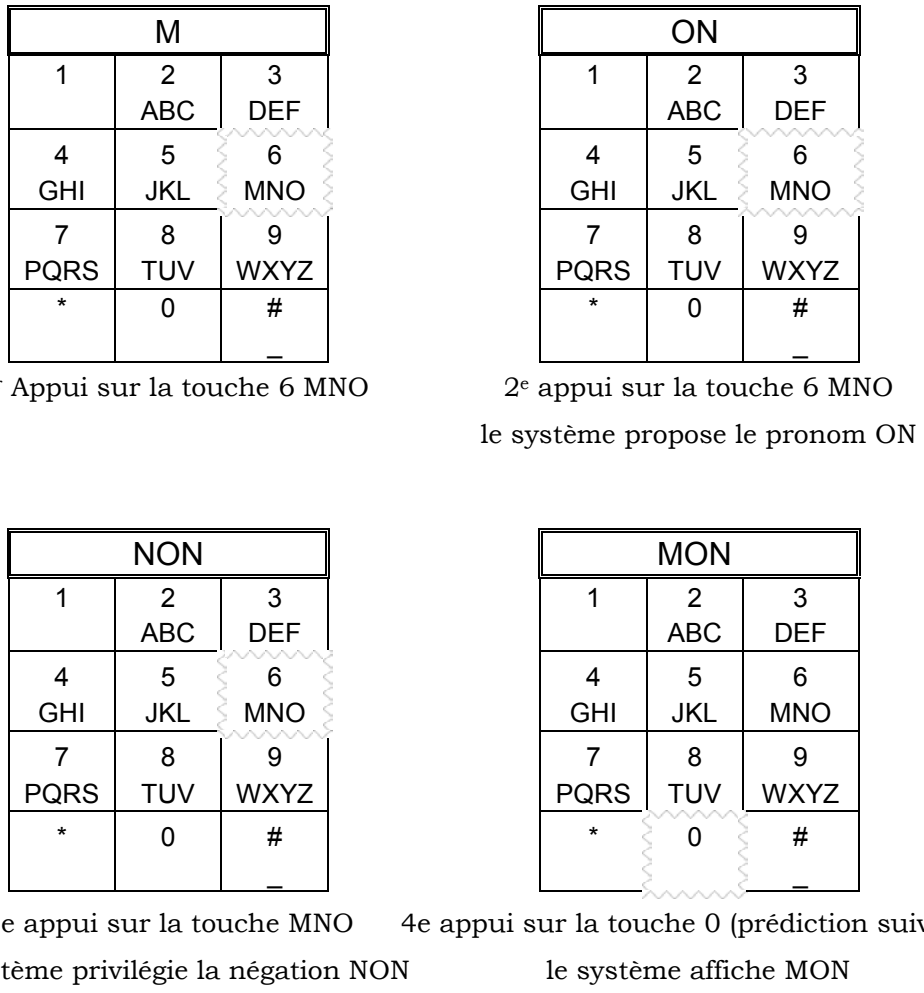


Figure 2.14 : La saisie en 4 appuis du mot « mon » avec le système T9™

La Figure 2.14 illustre la saisie du déterminant « MON » sur le système T9™, les trois lettres qui composent ce mot se trouvent sur la même touche : la touche 6 (qui contient les lettres MNO). Trois appuis sur cette touche vont donner successivement M, ON, NON. Sur cet exemple, le mot affiché « NON » n'est pas le mot attendu « MON ». Un quatrième appui sur la touche 0 (prédiction suivante) permet finalement d'obtenir le déterminant possessif MON. Il a donc fallu seulement 4 appuis au lieu de 6 (sans compter les pauses intermédiaires puisque les lettres sont sur la même touche) pour composer le mot. Les performances annoncées du système T9™ sont de 1,0051 appuis par lettre.

Dans le domaine de la recherche, ces claviers sont connus sous le nom de claviers ambigus. Cette technique est directement utilisable dans les claviers virtuels. Comme nous l'avons déjà indiqué, la saisie est plus rapide sur un clavier disposant de moins de touches.

2.2.6.2.1 Le système T9™ adapté

La société Tegic Communications a reçu le droit du National Institutes of Health (institut national de la santé) des Etats-Unis d'exploiter sous son visa, le potentiel du système T9™ pour les systèmes d'aide à la communication [Kushler 1998]. La plate-forme matérielle choisie pour ce projet est celle du système Vanguard, développé par la Prentke Romich Company (PRC). Cette dernière société fabrique déjà de nombreuses aides techniques. En particulier, le système Vanguard supporte de multiples interfaces d'accès qui peuvent être utilisées en association avec le système T9™.

Dans le cadre de ce projet, T9™ deviendrait T8 (taper avec huit touches) ce qui correspond aux huit directions pour la sélection par joystick par exemple. En balayage automatique sur ce clavier de huit touches, les performances seraient de 3,4 défilements par lettre en utilisant le défilement linéaire. Le système est prévu pour intégrer une prédiction de mot abaissant à 2,9 le nombre de défilements par touche [Kushler 1998].

2.2.6.2.2 Le système UKO

L'objectif du projet UKO (Unbekanntes KommunicationObjekt, ie objet de communication non identifié) du département informatique de l'université de Coblence en Allemagne, est d'élaborer une plate-forme d'aide à la communication basée sur un clavier ambigu.

Pour déterminer le mot correct, UKO utilise un modèle de langage à quatre couches [Harbusch 2003] :

1. Le modèle *stop mot* contient la liste des mots les plus usités (quelques centaines de mots). La fréquence associée à ces mots est considérée constante.
2. Le modèle *local* est actualisé avec les saisies de l'utilisateur. Les mots les plus récents voient leur probabilité amplifiée.
3. Le modèle *spécifique au domaine* contient des informations propres au thème du texte. Les informations ont été collectées sur des textes de domaine prédéfini.
4. Le *modèle de langage général* utilise les fréquences de la base de données CELEX, soit 360 000 mots pour l'allemand ou 160 000 mots pour l'anglais [Baayen 1995].

Une autre caractéristique du clavier UKO est le classement des touches sur le clavier. Au lieu de suivre l'ordre alphabétique (comme dans les téléphones par exemple), les lettres sont organisées afin de simplifier la tâche de désambiguïsation. Enfin, le nombre de touches est configurable et des claviers de taille très réduite (3 touches) peuvent être créés. Dans la Figure 2.15, le clavier UKO contient six touches dont quatre pour les lettres ; dans le cadre gauche apparaît la saisie en cours et dans celui de droite les mots possibles. Les touches sont parcourues cycliquement en défilement automatique. L'accès à la liste de mots est réalisé par l'intermédiaire d'un deuxième bouton physique.

Figure 2.15 : Le clavier ambigu UKO

Le clavier UKO a été testé avec une jeune fille IMC de quinze ans [Kühn 2001]. L'expérience a porté sur l'écriture d'une dizaine de phrases avec d'un côté le clavier UKO, de l'autre le système à base d'icônes habituellement utilisés par la personne. Le gain réalisé en temps a été de 18 % avec le clavier UKO, la vitesse de frappe passant de 5,1 mots par minute à 6,2 mots par minute. Une partie du gain obtenu a été réalisé sur les mots inconnus du système à base d'icônes et présents dans le lexique CELEX.

2.2.6.3 La prédiction de lettre

La dernière solution présentée concerne la prédiction de lettre. Dans les tableaux de lettres, nous avons déjà évoqué l'intérêt du recours à l'ordre fréquentiel des lettres pour faciliter la saisie. Son principe est simple, favoriser l'accès aux lettres les plus utilisées. Cependant, l'idée de classer les lettres selon leur fréquence peut être encore poussée plus avant : présenter prioritairement, cette fois-ci en contexte, les lettres les plus probables. En effet, les premières lettres de l'ordre fréquentiel « hors contexte » E et S correspondent, en français, aux marques flexionnelles du féminin et du pluriel, ces lettres sont donc fréquentes surtout en fin de mot. D'un autre côté, la fréquence relative du E en début de mot est beaucoup plus faible. Il est donc légitime de penser que la prédiction de lettre puisse améliorer la vitesse de sélection des lettres.

À notre connaissance, il existe très peu de systèmes utilisant la prédiction de lettre. Nous exposons ici deux méthodes qui permettent de prédire les lettres. La première est fondée sur les connaissances lexicales, la seconde sur la fréquence de co-occurrence de n-uplets de lettres.

2.2.6.3.1 Prédiction de lettre fondée sur le lexique

Une première méthode pour réaliser la prédiction de lettre consiste à construire l'arborescence d'un lexique et à factoriser les fréquences associées à chaque mot. C'est ce que réalise le système proposé par [Ménier 2001] du laboratoire VALORIA de l'Université de Bretagne Sud. Ce système utilise un TRIE ou Tree RetrIEvial [Knuth 1968] pour représenter le lexique, comme illustré par la Figure 2.16 .

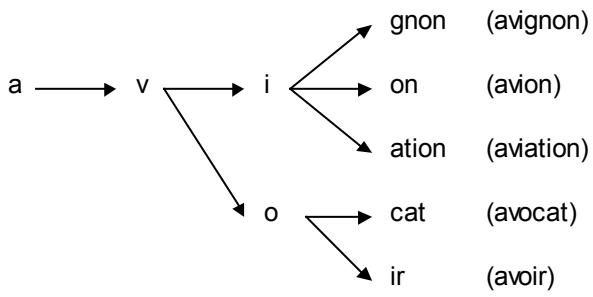


Figure 2.16 : Organisation lexicale TRIE

Chaque nœud de l'arborescence est pondéré par la fréquence qui correspond à la somme des fréquences des nœuds fils. Un nœud terminal utilise la fréquence du mot (hors contexte) qu'il représente. Lors du processus de saisie, la saisie en cours correspond à un nœud de l'arborescence (sauf si le mot est inconnu). Les probabilités des lettres sont estimées à partir des fréquences du nœud en cours et de celle des nœuds fils.

2.2.6.3.2 Prédiction de lettre basée le modèle statistique n-gramme

Cette deuxième méthode est fondée sur le modèle statistique n-gramme. Cette méthode sera exposée plus en détail lors de la présentation de SibyLettre (Chapitre III) qui l'utilise. De manière succincte, le modèle n-gramme permet d'estimer la probabilité d'une lettre en fonction des *n-1* dernières lettres. Par rapport à l'approche précédente, cette méthode permet de s'affranchir du lexique et reste donc applicable dans le cas des mots inconnus.

Dans [Toulotte 1986], l'auteur compare différentes méthodes d'accélération pour un système d'aide à la communication. La méthode la plus rapide utilise de manière combinée la catégorie syntaxique du mot à taper et des bigrammes de lettre de début de mot.

Un autre exemple d'application de ce modèle est le système HandiAS que nous présenterons plus en détail dans la partie suivante. Le système HandiAS utilise conjointement une liste de trois mots pour économiser le nombre de saisies et une

liste de trois lettres [Le Pévédic 1997]. Les lettres affichées sont celles estimées les plus probables par un bigramme de lettres (n = 2).

2.2.7 Synthèse des systèmes de saisie sur clavier

Pour conclure la partie consacrée à la saisie sur clavier, nous proposons une rapide synthèse des performances obtenues par les différents systèmes de sélection de lettre en balayage automatique. Le clavier de référence proposé contient une trentaine d'items (les 26 lettres de l'alphabet plus l'espace et quelques caractères supplémentaires ou de commandes). Les données présentées dans le tableau suivant sont extraites des différentes études pré-citées et sont exprimées en *nombre de défilements* pour accéder aux lettres (en moyenne).

Clavier	Défilement linéaire	Défilement ligne / colonne
Clavier AZERTY 3 x 10		7,27[4]
Clavier Alphabétique 4 x 8		6,19[4]
Clavier équiprobable 5 x 6[5]	15	6
Clavier fréquentiel 5 x 6	6,86[4]	4,29[4]
Clavier ambigu T8	3,4[6]	

Tableau 2.4 : Performances de claviers pour la sélection en balayage automatique

Il est souvent difficile de comparer les performances annoncées par divers systèmes tant les conditions d'évaluation sont différentes (corpus d'apprentissage, de test, langue de référence…). Les résultats présentés dans ce tableau n'échappent pas à la règle, et nous nous limiterons à deux remarques.

La première est qu'à mode de balayage identique (le défilement linéaire par exemple), on peut établir une hiérarchie des systèmes composée de 3 classes correspondant à des sauts qualitatifs en terme de performance :

(Clavier standard AZERTY ou alphabétique) < (Clavier fréquentiel) < (Clavier ambigu)

La deuxième remarque est en forme de question. Si l'on envisage les deux meilleurs systèmes à savoir un clavier fréquentiel utilisant le balayage ligne / colonne (4,29 défilements/lettre) et 2) le clavier ambigu avec défilement linéaire (3,4 défilements/lettre), on peut se poser la question du meilleur système, compte tenu de la réserve émise sur les résultats. De plus, à ce niveau de performance, d'autres critères entrent en jeu. En particulier, lors de la saisie d'un mot, le clavier ambigu

[4] Résultat extrait de [Cantegrit 2001] et exposé en 2.2.5 « La vitesse de saisie en balayage automatique ».
[5] Le clavier « équiprobable » est un clavier théorique où la fréquence d'utilisation de chaque touche est supposée équiprobable.
[6] Résultat extrait de [Kushler 1998] et exposé en 2.2.6.2.1 « Le système T9™ adapté ».

affiche des mots intermédiaires qui ne correspondent pas aux lettres attendues ce qui peut être perturbant. Par contre, l'usage du défilement linéaire, qui ne nécessite qu'une validation pour sélectionner une lettre, est un atout.

Nous reviendrons sur ces résultats lors de l'évaluation de SibyLettre pour les comparer avec notre système SibyLettre fondé sur une prédiction de lettre.

Après avoir exposé les différentes méthodes de sélection sur clavier, nous allons maintenant présenter les méthodes qui permettent d'économiser le nombre de saisies.

2.3 L'économie de saisies

À l'heure actuelle, seulement quelques logiciels du commerce permettent de compléter automatiquement la fin des mots grâce à un lexique. Ces logiciels sont présentés dans le paragraphe suivant.

Cette aide reste cependant rudimentaire au vu des systèmes développés dans les centres de recherche. La plupart des systèmes d'aide à la communication évolués sont issus de laboratoires de recherche disposant d'un département ou d'une équipe spécialisé en linguistique ou Traitement Automatique des Langues. Les systèmes appliqués au domaine du handicap tirent alors parti de modèles linguistiques évolués. Nous présenterons ainsi successivement les systèmes PAL, Profet, Compansion, Kombe, VITIPI et HandiAS.

2.3.1 Les logiciels du commerce

2.3.1.1 Prédiction de mot

Pour permettre une saisie plus rapide, quelques logiciels de clavier à l'écran disposent d'une prédiction de mot. On peut citer par exemple les logiciels Wivik, OnScreen et Clavicom. Ces logiciels affichent une liste des mots, généralement en haut du clavier (Figure 2.17). La sélection d'une de ces touches « mot » permet de compléter le mot en cours de saisie avec le mot contenu dans la touche. Après chaque lettre tapée par l'utilisateur, les propositions sont actualisées pour tenir compte de la saisie en cours (affichage de mots commençant par les premières lettres tapées).

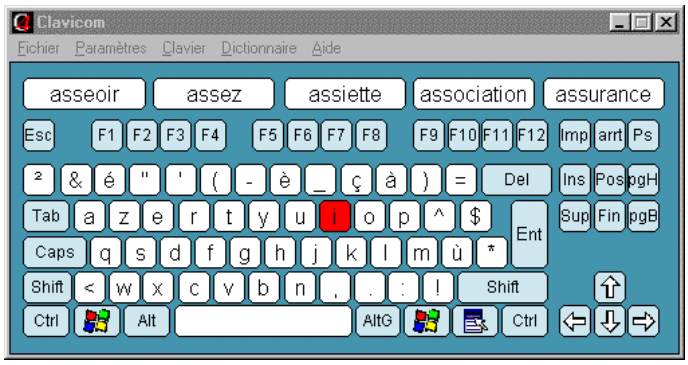


Figure 2.17 : Clavicom[7], exemple de clavier à l'écran avec prédiction de mot

Quel que soit le logiciel du commerce, le principe de la prédiction de mot est toujours le même. Ces logiciels ont à disposition un dictionnaire doté d'informations de fréquence. À partir du préfixe donné par la saisie en cours, les mots potentiels sont extraits du lexique puis classés en fonction de leur valeur fréquentielle. Les meilleurs mots, ceux dont la fréquence est la plus élevée, sont affichés. Il s'agit donc d'une prédiction hors contexte. Par exemple, la prédiction ne gère pas les accords en genre et en nombre, mêmes élémentaires, comme l'accord du nom ou de l'adjectif avec le déterminant qui précède.

En dehors de la qualité du dictionnaire (le nombre de mots qu'il contient et l'efficacité des informations fréquentielles), il existe deux critères qui différencient les prédictions de mot des logiciels du commerce. La première est l'adaptation (ou non) à l'utilisateur (actualisation des fréquences en fonction des mots tapés par l'utilisateur). La deuxième est la possibilité de compléter le dictionnaire. Sur ce dernier point, notons que l'ajout de mots à un lexique est souvent difficile pour une personne disposant de capacités d'interaction limitées avec l'ordinateur. Aussi, certains logiciels permettent d'ajouter automatiquement au dictionnaire les mots qui leur sont inconnus. Cette fonctionnalité n'est cependant pas sans inconvénients. En effet, les dictionnaires se retrouvent vite « pollués » par de nombreux mots correspondant à des saisies incorrectes liées aux fautes de frappe.

2.3.1.2 Abréviations

Une autre méthode pour économiser le nombre de saisies est l'utilisation d'abréviations. Certains logiciels permettent ainsi de stocker des paires (abréviation, mot) et dès qu'une abréviation est détectée, elle est remplacée par son mot équivalent. Cette méthode a cependant deux inconvénients. D'abord elle nécessite de la part de l'utilisateur une charge cognitive accrue. En effet celui-ci doit mémoriser la liste des abréviations. Ensuite, le nombre d'abréviations qui peuvent être créées est rapidement

[7] Clavicom a été développé par Handicap International

limité. Par exemple, si « abr » est utilisé pour « abréviation », cette abréviation ne peut plus être employée pour « abricot », « abriter », « abrutissant », etc.

2.3.2 PAL et les systèmes du Micro Centre de Dundee

Le centre de recherche en informatique appliquée (Applied Computer Studies Division, ACSD) est une branche du département de mathématiques et d'informatique du Micro Centre de l'université de Dundee en Écosse. Ce centre regroupe l'équipe de recherche la plus importante de toute la Grande-Bretagne travaillant dans le domaine des systèmes d'aide à la communication pour personne handicapée. Le professeur Alan Newell est la figure emblématique de ce centre. Lui et son équipe ont obtenu de nombreux succès autant dans le domaine de la recherche que dans les systèmes commerciaux. Leurs recherches s'orientent plus particulièrement vers la conception de logiciels sur ordinateur et leur utilisation par des personnes ayant des troubles de la parole ou du langage.

2.3.2.1 PAL (Predictive Adaptive Lexicon)

L'objet du projet PAL est davantage la réalisation d'un système de prédiction qu'un système d'aide. Ce n'est que lorsque PAL est associé à un traitement de texte qu'il peut être utile à la communication en facilitant la saisie de textes. C'est un des principaux projets du Micro Centre.

Le principe de PAL est de délivrer une liste de mots à partir des premières lettres d'un mot. Pour cela, il combine une fréquence des mots hors contexte et une fréquence d'utilisation récente des mots. Si le mot désiré n'appartient pas à la liste affichée, l'utilisateur doit taper la lettre suivante. PAL refait une liste de propositions et ainsi de suite. Un des avantages de PAL est que les fréquences sont actualisées avec la saisie des mots de l'utilisateur, ce qui le rend de plus en plus performant.

PAL a fait l'objet de nombreuses expériences pour tester son efficacité. Celles-ci ont montré que PAL permet d'économiser jusqu'à 50 % des saisies, ou en d'autres termes que l'utilisateur ne tape, en moyenne, qu'une lettre sur deux [Wright 1994]. Si l'on transposait, dans une certaine mesure, l'économie de saisie en économie de temps, on peut considérer que la personne s'exprime ainsi deux fois plus vite. Ceci est particulièrement vrai chez les personnes lourdement handicapées pour lesquelles la saisie est difficile. Il a été également montré que PAL peut aider de manière significative les enfants ayant des difficultés avec l'orthographe. En utilisant PAL de manière répétée avec l'affichage des mots correctement orthographiés, les enfants ré-apprennent l'orthographe [Booth 1990].

2.3.2.2 Syntax PAL

Pendant l'évaluation de PAL, il a été montré que les utilisateurs, surtout les enfants, faisaient des fautes autant de grammaire que d'orthographe [Murray 1991]. Pour éviter ces fautes, PAL a été augmenté d'une grammaire, appelée Syntax PAL. Cette amélioration a eu deux conséquences : non seulement les personnes font moins de fautes, mais en plus elles utilisent des structures de phrase plus complexes [Morris 1993]. Syntax PAL aide en effet les personnes à ajouter certains mots qui étaient précédemment omis. Ces mots sont essentiellement des mots grammaticaux comme les déterminants.

2.3.2.3 CHAT (Conversation Helped by Automatic Talk)

CHAT propose une autre approche pour améliorer la communication. Plutôt que de chercher à trouver le mot suivant le plus probable, son rôle est d'analyser la structure de la phrase et du discours en cours pour prédire des groupes de mots en relation avec la conversation. Les propositions sont pré-enregistrées puis restituées de manière aléatoire pour varier les expressions. La liste des phrases connues contient, en particulier, nombre de formules de politesse, d'invite et de fin de dialogue. Contrairement aux systèmes fondés sur la seule prédiction de mots, CHAT permet des réponses rapides dans une conversation comme l'accord ou la désapprobation. Plusieurs modes de dialogue sont disponibles pour exprimer des sentiments comme la joie, la tristesse, l'incompréhension…

Le principe du système est qu'un dialogue suit un schéma conversationnel strict : ouverture, puis discussion orientée sur un sujet et enfin fermeture [Gumperz 1982]. À chacune de ces parties correspond une structure et des phrases types. Le discours est représenté par la succession d'états dans un réseau de transitions (ATN, Augmented Transition Network) [Woods 1970]. À chaque étape du dialogue, plusieurs chemins peuvent être empruntés, chacun représentant une évolution possible de la conversation. Pour l'instant, CHAT est surtout efficace pour les deux parties les plus simples, l'ouverture et la fermeture du dialogue. Il peut être complété par les systèmes PAL et TOPIC aux plus bas niveaux.

CHAT a d'abord été testé avec des personnes non handicapées [Alm 1988] puis avec quatre sujets handicapés [Brophy 1991]. Dans les deux séries d'expériences, la communication était axée sur des conversations simples entrant dans les schémas prédéfinis. Les remarques des personnes testées ont été positives sur la vitesse de communication et l'expressivité.

2.3.2.4 TOPIC

Le rôle de CHAT est d'aider la communication à un haut niveau mais ses performances demeurent limitées. Une conversation est souvent peu structurée et peut « partir dans tous les sens ». Cependant, certaines parties tendent à être les mêmes et peuvent être réutilisées [Reichman 1985]. Par exemple, une bonne nouvelle, les résultats de son équipe favorite sont autant de sujets de conversation utilisés par la même personne avec des interlocuteurs différents. Il est donc possible de mémoriser des phrases ou des portions de phrase pour les restituer ultérieurement. Ceci pose cependant de nombreux problèmes pour retrouver des phrases stockées dans une base. L'objectif de TOPIC est de faciliter cette recherche [Alm 1989]. Le principe de ce prototype est d'utiliser une interface évoluée pour l'accès à la base de données. Chaque phrase correspond à un enregistrement associé à un thème qui permet la restitution de phrases de même thématique ou des références croisées. À l'utilisation, TOPIC prédit des phrases extraites de la base qui ont le plus de chances d'être appropriées. La sélection est déterminée par la fréquence d'utilisation, le contexte et des scripts définis par l'utilisateur pour lui permettre de suivre un schéma de discussion sur un sujet précis [Alm 1989].

2.3.2.5 TalksBack

TalksBack, [Broumley 1990], est un autre système qui se situe à l'intermédiaire entre CHAT et PAL. Son objectif est identique à celui de TOPIC, c'est-à-dire restituer des phrases pré-enregistrées. Les informations qu'il utilise pour y parvenir concernent les interlocuteurs les plus fréquents, les situations type. Pendant une conversation, l'utilisateur sélectionne son interlocuteur, le sujet de la discussion, le type de phrase (une question par exemple), et si possible la situation. TalksBack recherche alors dans sa base une phrase et la suggère. Si la phrase correspond, elle est directement envoyée à une synthèse vocale. Dans l'autre cas, une nouvelle recherche est opérée. Si aucune phrase proposée n'est validée, le système finit par rendre la main pour permettre la saisie conventionnelle. Lorsqu'une nouvelle phrase est tapée, l'utilisateur entre les informations complémentaires qui permettront de la réutiliser.

2.3.3 Profet et Multi-Talk

L'institut royal de technologie ou KTH (Kungl Tekniska Högskolan) de Stockholm en Suède dispose d'un département de musique et de communication très actif dans le domaine des systèmes d'aide à la communication. Bien que la thématique de ce département ne soit pas à l'origine axée sur le handicap, des efforts substantiels sont menés dans cette direction. Ses deux principales réalisations sont le système de prédiction de mot Profet et le système d'aide à la communication Multi-Talk.

2.3.3.1 Profet

Profet est un système prédictif qui peut être utilisé, comme PAL, avec un traitement de texte. Lorsque le système est activé, ses prédictions sont affichées dans une fenêtre surgissante (pop-up window).

Pour établir ses prédictions, la première version de Profet, utilise trois types d'information : un lexique principal de 7 000 mots classés par fréquence, une liste de 7 300 bigrammes et des lexiques thématiques [Hunnicutt 1986]. Lorsque la saisie d'un mot commence (avant la saisie de la première lettre), Profet délivre ses prédictions à partir des bigrammes de mots. Dès qu'une lettre est tapée, le système se réfère au lexique principal. Les lexiques thématiques sont activés par l'utilisateur au début de la saisie d'un texte. Les entrées de ces lexiques viennent compléter les prédictions établies précédemment. Lorsqu'un mot nouveau apparaît, il est inséré automatiquement dans le lexique thématique spécifié par l'utilisateur.

Dans une version plus récente, le modèle de prédiction a été revu. Il combine maintenant deux modèles n-gramme (cf. chapitre IV), l'un portant sur les mots (avec n = 2), l'autre portant sur les étiquettes syntaxiques des mots (avec n = 3). Le jeu d'étiquettes contient environ 150 items avec les informations morphologiques [Carlberger 1997]. Le modèle de prédiction se voit ainsi augmenté d'information de nature syntaxique.

Taille de liste	% de saisies économisées
Liste de 1 mot	32,9 %
Liste de 5 mots	46,0 %

Tableau 2.5 : Évaluation du système Profet

Le Tableau 2.5 présente les résultats obtenus par Profet en pourcentage de saisies économisées et en fonction de la taille de la liste des mots. Profet affiche ainsi un taux de 46 % de saisies économisées pour la taille conventionnelle de cinq mots.

2.3.3.2 Multi-Talk

Multi-Talk est un système d'aide à la communication prédictif utilisant la synthèse vocale développée au KTH. Pour composer une phrase, l'utilisateur définit d'abord son type (question, information, commande). Ce type permet de définir la structure grammaticale de la phrase à écrire et l'intonation pour la synthèse vocale [Hunnicutt 1993]. La phrase est ensuite normalement saisie avec, par exemple, un système de prédiction de mots comme Profet.

2.3.4 Compansion

Le système Compansion [Demasco 1992] a été developpé à l'université de Delaware. Il permet à l'utilisateur de taper son message en style télégraphique avec des mots non fléchis. Le système transforme ensuite la séquence de mots en phrase syntaxiquement et sémantiquement bien formée. Par exemple, l'utilisateur saisit « 5 apple eat John » et Compansion traduit en « 5 apples were eaten by John ». Il est possible que pour une phrase donnée, Compansion trouve plusieurs solutions. Dans ce cas, les différentes phrases sont proposées à l'utilisateur qui doit sélectionner la phrase désirée. Cette méthode, qui prend en entrée un message « compressé » et reconstitue un message expansé, d'où le nom du système (compressed-expansion), permet ainsi d'économiser un certain nombre de saisies.

Dans Compansion, la transformation est réalisée en trois étapes [Mc_Coy 1995] :

- L'analyseur syntaxique est chargé d'attribuer une étiquette syntaxique à chaque mot de la phrase en entrée, puis de segmenter cette phrase en groupe de mots
- L'analyseur sémantique construit ensuite une représentation sémantique de la phrase à partir des mots lexicaux des groupes précédemment déterminés. Celui-ci est fondé sur le formalisme des grammaires de cas [Fillmore 1968]
- Enfin, un traducteur reconstruit une phrase syntaxiquement et sémantiquement correcte à partir de la représentation sémantique

À partir du même module linguistique, deux autres systèmes de communications ont été développés : un système de prédiction de mot et un système permettant d'écrire des abréviations « à la volée » que nous allons détailler.

Comme nous l'avons déjà vu, l'expansion d'abréviations est une technique qui peut être utilisée dans les systèmes de communication. Cependant cette technique a deux limites : la mémorisation des abréviations et le nombre limité d'abréviations qui peuvent être créées. Le système proposé ici permet de s'affranchir de ces deux limites puisque les abréviations sont créées à la volée. Pour réaliser ceci, l'hypothèse émise est qu'un petit nombre de règles est employé pour créer des abréviations. Par exemple, une abréviation est constituée des premières lettres du mot ou du mot sans les voyelles, ou une combinaison des deux. À partir de ces règles et du lexique, il est ainsi possible de reconstituer l'ensemble des mots que peut désigner l'abréviation. Le système utilise ensuite son module linguistique pour déterminer le mot le plus approprié dans la phrase en cours.

2.3.5 Kombe

Kombe est un système de composition de phrase assistée conçu au Laboratoire d'informatique de Marseille (LIM) [Richardet 1998]. À l'origine, Kombe est un des

projets du programme européen TIDE (Technology Initiative for Disable and Elderly People), programme destiné à promouvoir le développement de technologies au service des personnes âgées et handicapées. Dans ce cadre, Kombe a été développé plus spécialement pour des patients atteints de sclérose amyotrophique latérale.

L'interface du système Kombe est composée principalement d'un clavier simulé, d'une liste de mots et d'une zone d'édition de texte. L'utilisateur peut soit valider un mot de la liste soit taper une lettre sur le clavier simulé. À l'inverse des systèmes présentés jusqu'à présent, la liste de mots contient tous les mots possibles trouvés par le système. Cependant, les mot proposés vérifient que la phrase est bien formée d'un point de vue lexical, syntaxique, sémantique et pragmatique. Par exemple dans le contexte d'une relation patient-médecin, après le début de phrase « j'ai beaucoup de difficultés à plier le », Kombe présente les 7 mots « bras, coude, doigt, genou, majeur, pied, pouce ».

Pour établir cette liste de mots, Kombe utilise le module linguistique ILLICO. ILLICO [Pasero 1994] est un système qui permet de guider un utilisateur dans la composition de phrase à partir de l'analyse de la partie gauche de la phrase. Le système comporte quatre modules :

1. Un lexique qui contient la liste des mots (ou des icônes)
2. Une grammaire qui spécifie les structures linguistiques ou iconiques correctes
3. Un ensemble de règles sémantiques qui produisent des représentations sémantiques
4. Un modèle conceptuel qui exprime en terme de domaine et de relations les présuppositions lexicales associées avec les mots

Ce dernier module suppose que le domaine (ou plus simplement le thème) du texte de l'utilisateur soit défini au début de la saisie. Dans le cas de Kombe, les thèmes développés sont orientés vers le dialogue patient-médecin (santé, hygiène, mobilité, etc.).

2.3.6 VITIPI

Le système français VITIPI [Boissière 1990] (Versatile Interpretation of Text Input By Persons with Impairments) a été développé par l'Institut de Recherche en Informatique de Toulouse (IRIT).

Le principe de VITIPI pour économiser les saisies est légèrement différent de ceux présentant une liste de mots. En effet, d'après [Boissière 2000], cette liste perturbe l'utilisateur lors de la saisie. À la place, VITIPI propose d'insérer automatiquement les lettres des mots dès qu'il n'y a plus d'ambiguïté. Par exemple, si l'on considère un lexique minimal « d-irect-eur, d-irect-rice, d-irect-ion », après la saisie de la lettre « d »,

tous les mots du lexique commençant par « d » ont une séquence commune « irect » qui peut donc être affichée automatiquement. Pour que le système soit efficace, VITIPI travaille avec un lexique réduit d'un peu plus de 5 000 mots. Le système accepte cependant automatiquement les nouveaux mots.

L'intérêt de VITIPI est également sa capacité à gérer les mots inconnus. Un mot inconnu peut être de deux types. Soit le mot n'appartient pas au vocabulaire, soit il comporte une ou plusieurs fautes de frappe ou d'orthographe. Face à un mot hors lexique, VITIPI dispose d'un système d'inférence analogique dont le rôle est de trouver un mot dont le comportement est similaire à un mot appartenant déjà au vocabulaire. Supposons que le début de mot *administratr* a été entré. S'il n'appartient pas au vocabulaire, le système peut faire l'hypothèse que la lettre « r » a pu se substituer au « e » car elles sont voisines sur le clavier et donc afficher *administrateur*. Mais, il peut également produire le mot *administratrice* par analogie avec des mots ayant un comportement similaire (*direct-eur, direct-rice, institut-eur*, *institut-rice*). Quand le système est placé devant l'autre situation, c'est-à-dire une faute dans le mot, il peut avoir à faire avec quatre situations d'erreurs différentes qui sont :

- La *substitution* : cas où deux lettres sont inversées. Par exemple dans *trpisième* le « p » a été tapé à la place du « o ». Dans VITIPI, la distance qui sépare deux touches sur le clavier a été définie. Si la distance est faible (1 touche), le mot est corrigé.
- L'élision **:** cas où une lettre a été omise. Par exemple, *quproquo* au lieu de *quiproquo.* Ce cas peut être résolu si le mot entré n'appartient pas au lexique mais que l'ajout d'une lettre correspond à un mot du lexique.
- L'insertion **:** cas où une lettre a été ajoutée.
- La faute d'orthographe. Pour gérer ces fautes un expert comportant deux cents règles d'orthographes du français a été créé.

Le choix d'une hypothèse dépend de plusieurs facteurs qui évoluent au cours de l'utilisation du système en fonction de la réussite ou des échecs des hypothèses antérieures. Le système s'adapte ainsi à l'utilisateur en favorisant la solution correspondant à l'erreur la plus fréquente.

Une évaluation de VITIPI a donné une prédiction de 26 à 37 % des lettres selon la taille du lexique et une correction de 72 % des fautes de frappe et 75 % des fautes d'orthographe.

2.3.7 HandiAS

Le système HandiAS a été développé à l'Institut de Recherche en Informatique de Nantes (IRIN) [Le Pévédic 1997], son développement se poursuit au Laboratoire d'Informatique de l'Université de Tours [Maurel 2001b].

HandiAS est un système de prédiction de mot. Pour établir la liste des mots, HandiAS utilise une méthode de prédiction morphosyntaxique évolutive, à partir du contexte gauche. Il utilise une technique hybride, symbolique et statistique afin de permettre dans le cadre du vocabulaire courant d'une personne, la prédiction de la fin d'un mot. Le système fonctionne à partir de dictionnaires fréquentiels et à partir de la modélisation d'un ensemble de structures du français auxquels sont affectées des probabilités d'utilisation. HandiAS doit rechercher dans un dictionnaire la liste de mots et de lettres les plus fréquents dans la (ou les) catégorie syntaxique la plus probable tout en respectant le début de la saisie. En cas d'échec, l'utilisateur saisit une nouvelle lettre et le système poursuit sa recherche. L'étude de [Le Pévédic 1997] s'est organisée autour de trois axes. Le premier est la création d'un dictionnaire électronique comportant des indications morphologiques, séquentielles et syntaxiques. La constitution du dictionnaire a été conditionnée par l'étude statistique de la fréquence d'utilisation des mots dans la langue française. Le second est une étude syntaxique de la langue qui détermine un modèle de language et permet une prédiction de la catégorie grammaticale la plus adéquate en fonction du contexte gauche. Enfin, un dernier module permet au système de s'adapter à l'utilisateur au fur et à mesure de son utilisation. Cette adaptation intervient d'une part au niveau du module de prédiction syntaxique et d'autre part au niveau du dictionnaire initial.

L'ensemble des résultats des tests effectués montre que le système HandiAS donne de meilleurs résultats que le module syntaxique seul ainsi que le module fréquentiel seul. Le système HandiAS, qui est la combinaison du module syntaxique et du module fréquentiel, permet d'économiser 43 % des saisies.

2.3.8 Synthèse des systèmes économisant le nombre de saisies

Tous les systèmes exposés dans cette partie ont le même objectif : écrire une phrase en entrant un minimum de caractères. Les moyens pour y parvenir sont cependant très différents. Dans cette synthèse, nous proposons quatre caractéristiques principales d'un système de composition assistée :

- La correction des fautes (de frappe ou d'orthographe).
- L'adaptation à l'utilisateur. Cette adaptation peut se faire principalement de deux manières : modification des fréquences d'usage des mots (voire des structures syntaxiques) et apprentissage des mots inconnus.
- L'interface avec l'utilisateur.
- Le modèle de prédiction.

La correction automatique des fautes permet d'économiser le nombre de saisies de manière indirecte : soit en évitant à l'utilisateur la correction, soit (si le mot n'est pas corrigé) en évitant une interprétation erronée à l'analyseur pour la prédiction du mot suivant. Dans les systèmes présentés seul VITIPI propose une correction automatique.

On peut également noter que l'utilisation d'une liste de mot permet de réduire le nombre de fautes d'orthographe comme le montre l'expérience avec PAL. Le système Sibylle n'utilise pas non plus de correcteur. Nous pensons cependant qu'à terme, les systèmes de communications assistés devront intégrer une correction automatique comme certains traitements de texte par exemple. Parmi les techniques de correction autres que VITIPI, on peut citer le correcteur TYPO du système UNIX [Morris 1975] et le projet Céline [Ménézo 1996].

Le deuxième point concerne l'adaptation à l'utilisateur. De manière générale, une amélioration significative des performances est observée lorsque le système dispose d'informations fréquentielles qui sont ajustées avec les saisies de l'utilisateur [Boissière 2001] [Ménier 2001]. L'adaptation concerne également l'apprentissage des mots inconnus. Sur ce point, les systèmes présentés utilisent des lexiques de taille limitée (inférieure à 10 000 mots). Ceci nous semble une limite. Si comme l'indique [Rivenc 1979] « il existe un noyau minimal de vocabulaire général commun d'environ 1100 à 1200 mots », cette vision est cependant réductrice des capacités langagières d'un individu. De plus l'intégration de mot nouveau est souvent délicate à réaliser lorsque le système associe au mot des informations supplémentaires comme son lemme, sa catégorie syntaxique, etc. Dans HandiAS, les mots nouveaux doivent être validés en fin de session, ce qui nous paraît contraignant. Dans SibyMot, le lexique contient plus de 50 000 lemmes. En tout état de cause, le lexique ne doit pas être une limite pour le système.

Dans les systèmes présentés, l'économie de saisie est réalisée au niveau de l'interface de différentes manières : en complétant la fin des mots avec une liste de mot limitée (HandiAS, Profet, PAL) ou non limitée (Kombe), de manière automatique (VITIPI), etc. Il n'y a pas de meilleure interface *a priori*. Le point qui nous semble important de souligner ici est que l'évaluation d'un système de communication assistée doit clairement distinguer l'évaluation du modèle prédictif de l'évaluation de l'interface.

La dernière caractéristique concerne le modèle de langage. Nous rappelons qu'en TAL, il existe deux approches principales symbolique et stochastique. Dans les systèmes présentés Kombe et Compansion utilisent l'approche symbolique. L'intérêt de cette approche est de construire une représentation de la phrase qui permet, dans le cas de Kombe, de présenter une liste de mots qui sont « corrects » d'un point de vue syntaxique et sémantique avec le reste de la phrase. Cependant cette approche impose d'utiliser les structures syntaxiques modélisées, ce qui rend la saisie souvent contraignante. À l'inverse, la modélisation stochastique (de type n-gramme, cf. Chapitre IV) utilise seulement des informations fréquentielles des derniers mots pour établir ses prédictions. Elle est donc plus souple d'utilisation au risque de proposer des mots incorrects (par exemple en cas d'accord avec un mot du début de la phrase). PAL et Profet sont des exemples de cette approche. Pour Kombe et Compansion, il

n'existe pas à notre connaissance d'évaluation quantitative alors que les systèmes PAL, Profet, ou HandiAS (qui utilise une approche hybride) proposent des gains en saisie de 46 % à 50 %. De plus, les modèles probabilistes sont également très utilisés en Reconnaissance Automatique de la Parole (RAP), autre domaine qui emploie des modèles de langage prédictifs. C'est pourquoi nous avons choisi pour SibyMot une modélisation fondée principalement sur l'approche stochastique. Les modèles probabilistes seront présentés de manière détaillée dans le chapitre IV.

Dans ce chapitre, nous avons exposé deux approches pour accélérer la saisie de texte. La première consiste à permettre une sélection rapide des touches sur le clavier simulé. Dans Sibylle, cette approche correspond à SibyLettre exposé dans le chapitre suivant. La deuxième concerne l'économie de saisie. Ce rôle est attribué à SibyMot qui sera présenté au chapitre V.

Chapitre 3

Prédiction de lettre – SibyLettre

Ce chapitre présente SibyLettre, notre système de prédiction de lettre qui permet la saisie rapide des lettres sur le clavier simulé. Ce chapitre est organisé en trois parties. La première partie expose le module de prédiction de lettre : le modèle utilisé puis son évaluation quantitative. Dans la deuxième partie, nous présentons le clavier simulé « dynamique » que nous avons développé pour utiliser la prédiction de lettre. Celui-ci a été évalué auprès d'enfants IMC au centre de Kerpape. Enfin, ce chapitre se termine par un bilan de SibyLettre et une comparaison avec les autres systèmes de sélection rapide vus dans le chapitre précédent.

3.1 SibyLettre

3.1.1 Modélisation

Dans le cadre de Sibylle, il serait possible d'établir une prédiction de lettre à partir d'une approche lexicale, soit, encore mieux, à partir des prédictions délivrées par SibyMot. Cependant, un des rôles attribué à SibyLettre est d'être efficace même lorsque l'utilisateur saisit un mot nouveau, ce qui exclut cette approche. Nous avons donc opté pour une prédiction indépendante du lexique et choisi le modèle statistique n-gramme. La probabilité d'apparition des lettres est alors estimée à partir des *n-1* dernières lettres du mot en cours. Ce modèle a pour avantage d'être robuste et est couramment utilisé en ingénierie linguistique (étiquetage morphosyntaxique, reconnaissance vocale, etc.).

3.1.1.1 Modèle de langage n-gramme

Nous présenterons le modèle n-gramme de manière plus détaillée dans le chapitre IV sur la modélisation du langage. Ce modèle est donc décrit ici de manière succincte pour expliquer SibyLettre.

Le modèle n-gramme est issu de la théorie de l'information. Celle-ci suppose que la probabilité d'apparition d'un événement w_m dépend de toute la séquence passée $W = w_1,\ldots,w_{m-1}$. Compte tenu du grand nombre de séquences possibles, ce résultat n'est pas directement exploitable. Le modèle n-gramme propose de limiter le contexte à n observations. Ainsi, la probabilité d'apparition d'un événement peut être donnée de manière satisfaisante par une combinaison de probabilités basée sur les $n-1$ observations précédentes :

$$P(W) \cong \prod_{i=1}^{m} P(w_i \mid w_{i-n+1},\ldots,w_{i-1}) \tag{3.1}$$

L'approximation est d'autant meilleure que le paramètre n est élevé.

Dans le cadre de notre prédiction de lettre, les observations sont les lettres et la séquence $w_1,\ldots,w_{m-1}$ renommée $l_1,\ldots,l_{m-1}$ correspond au début du mot en cours. Cette séquence est connue et notre problème consiste à classer les lettres en fonction des probabilités $P(l_m \mid l_{m-n+1},\ldots,l_{m-1})$, ou encore des fréquences d'occurrence $F(l_{m-n+1},\ldots,l_m)$ observées sur corpus.

3.1.1.2 Estimation des probabilités n-gramme

La nature statistique du modèle pose le problème de l'estimation des probabilités pour les événements jamais observés. Sa bonne résolution conditionne les capacités de généralisation du modèle n-gramme. Parmi les techniques employées pour obtenir des estimations fiables (Chapitre IV), nous avons utilisé pour SibyLettre une technique de repliement : le module de prédiction classe les lettres en fonction des probabilités d'ordre n non nulles puis, pour les lettres restantes, il fait appel aux probabilités d'ordre $n-1$ et ainsi de suite jusqu'à un classement complet.

3.1.1.3 Représentativité des données

Un autre problème lié à la nature statistique du modèle n-gramme concerne la représentativité des données. En effet, l'estimation des probabilités repose sur les fréquences observées lors de l'apprentissage sur corpus. Ce corpus doit donc être représentatif de la tâche à accomplir. Dans le cas de SibyLettre, il s'agit de la probabilité d'apparition d'une lettre en fonction des lettres précédentes, le problème de la représentativité du vocabulaire est donc *a priori* limité.

3.1.2 Apprentissage

Le corpus utilisé pour estimer les probabilités est celui du journal Le Monde[8]. Les données représentent tous les articles du journal sur une période de 5 ans (1995 à 1999). La prédiction ne portant que sur les lettres des mots, seules les lettres et le caractère « espace » ont été conservés, soit 65 caractères (cette cardinalité élevée s'explique par la présence des lettres étrangères). Nous avons également converti les majuscules en minuscules. Le corpus ainsi obtenu contient 600 millions de caractères, ce qui correspond à 110 millions de mots pour 400 000 formes fléchies. Outre sa grande taille, ce corpus présente l'avantage de permettre une estimation de la fréquence des mots (et des séquences de lettres) en usage dans la langue.

3.1.3 Évaluation

Pour les évaluations quantitatives que nous avons réalisées, chacune des cinq années a servi de corpus de test (autour de 20% des données), l'apprentissage étant réalisé sur les quatre autres années. Cinq séries de tests ont ainsi été obtenues, ceci afin d'effectuer une validation croisée [Allen 1997]. Les résultats étant très proches, seule la moyenne est présentée par la suite.

Pour la partie évaluation qui suit, nous rappelons que le paramètre n définit la longueur de la séquence et $n-1$ la taille de la fenêtre contextuelle.

3.1.3.1 Évaluation de la prédiction

3.1.3.1.1 Perplexité

La mesure de perplexité est couramment utilisée pour évaluer les modèles statistiques. Elle permet de mesurer la complexité de la tâche étant donné un modèle [Boite 2000] et réciproquement de mesurer l'efficacité d'un modèle pour une tâche donnée. Cette perplexité est définie comme étant 2^H, où H est la mesure d'entropie :

$$H = \sum_h P(h) \sum_w P(w/h) \log_2 P(w/h) \qquad (3.2)$$

où h une séquence du modèle, $P(h)$ la probabilité d'apparition de la séquence h et $P(w/h)$ la probabilité d'un événement w en fonction d'une séquence h (dans SibyLettre, la probabilité d'apparition d'une lettre en fonction des premières lettres du mot).

Le Tableau 3.1 présente les valeurs de perplexité estimées sur le corpus pour différentes valeurs de n.

[8] « Le Monde » Text Corpus Version 1.0 Years 1995 to 1997 & 1998 and 1999

n	1	2	3	4	5
Perplexité	17,4	10,2	6,9	5,0	4,1

Tableau 3.1 : Perplexité en fonction de n

La mesure de perplexité est difficile à interpréter. Cependant, l'évolution de la perplexité de 17,4 (n = 1) à 4,1 (n = 5) montre que l'efficacité du modèle s'accroît de manière significative avec la taille du contexte.

Ces résultats peuvent également être mis en relation avec la perplexité des modèles de langage utilisant le même modèle statistique (n-gramme de mots). [Roukos 1996] cite ainsi des valeurs de perplexité de 247 pour un trigramme de mots (n = 3) estimé sur de l'anglais général, de 105 en style journalistique et de 20 dans le domaine de la radiologie. Ceci rappelle que la prédiction de mot est une tâche complexe offrant de nombreux choix possibles et laisse à supposer que la prédiction de lettre peut être envisagée comme une bonne aide complémentaire (rôle défini pour SibyLettre).

3.1.3.1.2 Prédiction moyenne (PM)

La pertinence de la prédiction délivrée par le modèle peut également être montrée par une mesure de prédiction moyenne (PM). Celle-ci consiste à relever, lors de la phase de test, le rang de la lettre attendue dans la liste proposée. Par exemple, avec « COM » le système délivre la liste « M, P, B, … ». Si le mot à saisir est « COM…PTER », la lettre attendue est P et le rang dans la liste est 2. Un calcul de moyenne délivre la prédiction moyenne :

$$PM = \frac{1}{N} \times \sum_{i=1}^{N} Rang(i) \qquad (3.3)$$

où N est le nombre d'observations et $Rang(i)$ le rang obtenu à l'observation i.

La Figure 3.1 donne la prédiction moyenne obtenue pour les différentes valeurs de n ainsi qu'une valeur limite (cf. infra) :

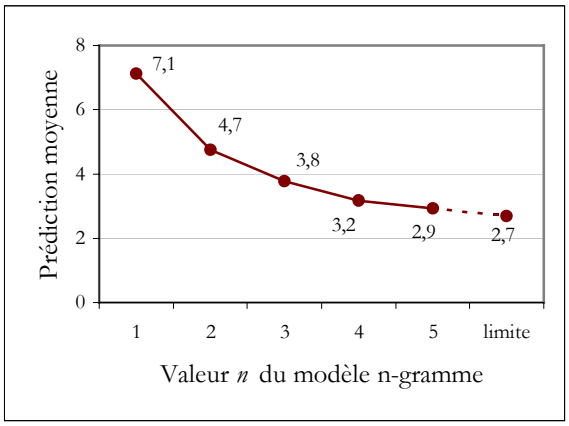


Figure 3.1 : Prédiction moyenne en fonction de n

Si l'on rappelle que pour n = 1, le modèle n-gramme est basé sur la seule fréquence des lettres dans la langue (et ne tient donc pas compte du contexte), on peut constater que l'ordre fréquentiel est loin d'être optimal. La prédiction moyenne varie ainsi de 7,1 pour n = 1 à 2,9 pour n = 5, soit un gain de 59 %. Autrement dit, l'utilisation du contexte gauche du mot en cours permet d'améliorer considérablement la prédiction utilisée par le clavier simulé.

On peut également remarquer que le gain obtenu pour chaque valeur de n supplémentaire décroît rapidement. Ce gain est de 2,4 entre n = 1 et n = 2 et seulement de 0,3 entre n = 4 et n = 5. Ceci s'explique par la décroissance rapide du nombre d'occurrences des mots en fonction de leur longueur (Tableau 3.2), décroissance qui limite l'influence d'une prise en compte d'un contexte plus long. En effet, pour deux valeurs m et m+1 de n, les prédictions délivrées par le modèle sont identiques pour les lettres dont la position est inférieure à m. Par exemple, pour m = 4 et m+1 = 5, les prédictions sont identiques sur les trois premières lettres des mots soit 50,3 % des lettres (Tableau 3.3). Le gain n'est donc réalisé que sur 49,7 % des lettres.

Longueur	1	2	3	4	5
% mots	9,0	21,9	13,4	10,0	8,9
% cumulé	9,0	30,9	44,3	54,3	63,2

Tableau 3.2 : Occurrences des mots en fonction de leur longueur

Position	1	2	3	4	5
% lettres	17,3	17,3	15,7	12	9,6
% cumulé	17,3	34,6	50,3	62,3	71,9

Tableau 3.3 : Occurrences des lettres en fonction de leur position dans le mot

La remarque précédente nous permet également de calculer une prédiction moyenne limite que le modèle peut au mieux approcher. Pour toutes les lettres dont la position est inférieure au n du meilleur modèle (dans notre cas 5), on utilise les prédictions de ce modèle ; pour les autres lettres, on suppose que la prédiction est idéale (rang égal à 1). La prédiction moyenne obtenue pour n = 5 donne une valeur limite de 2,7. On remarquera que le résultat obtenu pour n = 5 (2,9) est déjà très proche de cette valeur limite.

3.1.3.1.3 Prédiction moyenne en fonction de la position dans le mot (*PMP*)

Dans le cadre du projet Sibylle, en dehors des mots inconnus, la prédiction de lettre sera utilisée pour la saisie du début des mots. Ceci correspond à un contexte faible pour la prédiction de lettre. Aussi avons nous cherché à évaluer la perte de performances observée sur le début des mots.

Au calcul de prédiction moyenne défini précédemment, le paramètre « position de la lettre dans le mot » est ajouté. Sur l'exemple « COM…PTER » avec la liste « M, P, B, … », le rang de la lettre dans la liste est 2 et la position dans le mot 4. Le calcul de moyenne est effectué pour chaque position j :

$$PMP(j) = \frac{1}{N} \times \sum_{i=1}^{N} Rang(i, j) \tag{3.4}$$

où N est le nombre d'observations et $Rang(i, j)$ le rang obtenu à l'observation i pour la position j.

Les résultats sont donnés par la Figure 3.2 pour les 5 premières lettres des mots et pour les valeurs de n de 1 à 5. Par exemple, pour la 4è lettre des mots, la *PMP* est de 7,3 avec n = 1 (1ère série, en haut) ; pour cette même position, la *PMP* est de 1,8 avec n = 5 (dernière série, en bas).

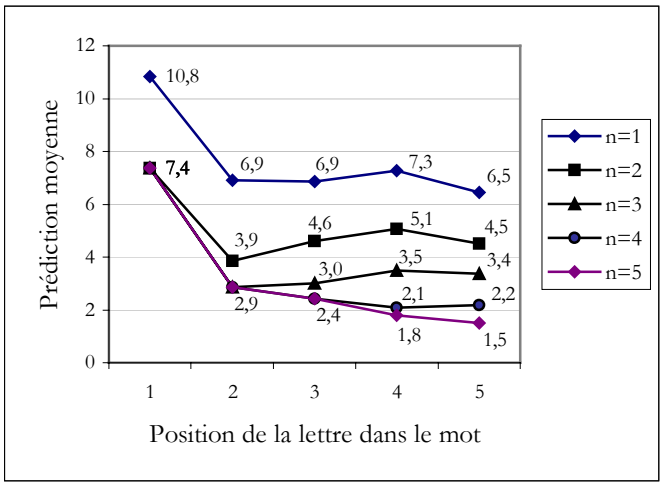


Figure 3.2 : Prédiction moyenne en fonction de *n* et de la position de la lettre dans le mot

L'analyse de chaque série de données (chaque valeur de *n*) permet de constater que la principale difficulté réside dans la prédiction de la première lettre. On observe un écart net entre la prédiction moyenne de la première lettre des mots et les suivantes ($PMP(1) = 7{,}4$ et $PMP(2) = 2{,}9$ pour $n > 2$). Nous reviendrons sur cette remarque dans les perspectives pour proposer une ébauche de solution à ce problème. Notons cependant qu'une prise en compte du contexte (ici le fait que l'on est en début de mot) par notre système permet tout de même une amélioration de la prédiction par rapport à l'ordre fréquentiel statique, la *PMP* passe en effet de 10,8 (1ère lettre des mots, n=1) à 7,4 (n>1), soit un gain de 31 %.

A l'inverse, la prédiction est efficace dès la deuxième lettre. Ceci permet d'envisager une collaboration entre les deux systèmes de prédiction (lettre et mot) sur les mots courts. Nous reviendrons également plus en détail sur cette remarque dans la partie perspectives.

3.2 Clavier dynamique

Les logiciels de clavier simulé habituels ne disposent pas de prédiction dynamique des lettres. Avec la prédiction contextuelle (réalisée après chaque saisie), l'ordre des lettres devient variable et implique une interface gérant cet aspect dynamique. L'ajout au clavier simulé d'une liste de lettres affichant les premières propositions serait envisageable. Afin de ne pas surcharger l'interface, nous avons opté pour un système de clavier dynamique.

3.2.1 Fonctionnement

Après chaque saisie, le début du mot en cours est fourni au module de prédiction. En retour, celui-ci délivre la liste des lettres classées en fonction de leur probabilité d'apparition. La liste contient également un caractère « fin de mot », symbolisé par l'espace. Le clavier simulé est alors réorganisé pour tenir compte de la prédiction. Le curseur est remis en position initiale (en haut à gauche du clavier) et le défilement automatique reprend.

Les lettres accentuées ne sont pas différenciées des autres. Elles bénéficient de la prédiction et sont naturellement classées en fonction de leur probabilité d'apparition.

Le balayage utilisé est le défilement linéaire. Ce balayage est nécessité par l'aspect dynamique qui rend le défilement ligne/colonne inadapté. En effet, en terme de charge cognitive, le défilement ligne/colonne impose de connaître à l'avance la ligne sur laquelle se trouve la lettre désirée, ce qui n'est pas le cas ici.

Le Tableau 3.4 et la Figure 3.3 présentent une simulation de la saisie des premières lettres du mot « COMP…TER ». Pour la première lettre, le contexte est vide (Ø), la liste proposée est D, L, P, A, E, C, … le curseur doit défiler 6 fois pour atteindre la lettre désirée. Après cette première lettre, la saisie est plus rapide. Avec le contexte C, la lettre présentée en premier est la lettre O souhaitée. Un seul défilement et l'utilisateur n'a plus qu'à valider cette proposition (la remise en position initiale du curseur est comptée comme un défilement). Les deux lettres suivantes sont saisies presque aussi rapidement, en deux défilements.

Contexte	Propositions	Saisie
Ø	DLPAE**C**…	C
C	**O**EH…	O
CO	N**M**U…	M
COM	M**P**B…	P

Tableau 3.4 : Contexte et propositions pour la saisie de COMP…TER

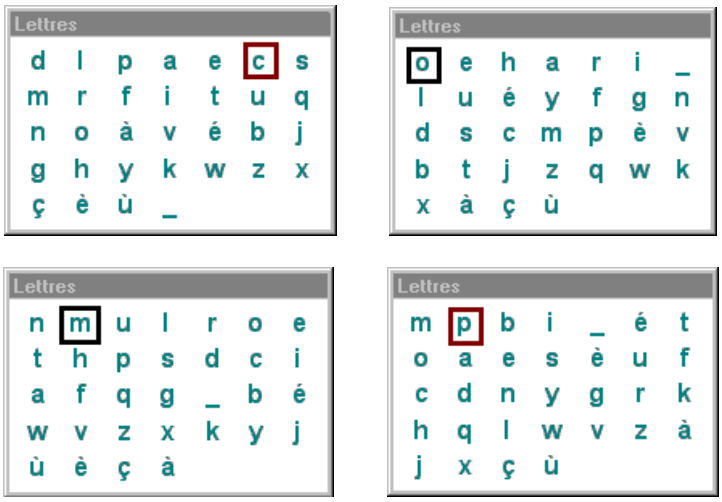


Figure 3.3 : Exemple de réorganisation dynamique sur le début de mot COMP…TER

Sur cet exemple, la saisie a été réalisée successivement en 6, 1, 2, 2 défilements. Ces valeurs sont à rapprocher avec celles obtenues théoriquement par la *PMP* (pour n = 5, les résultats donnaient : *PMP*(1..4) = 7,4 ; 2,9 ; 2,4 et 1,8).

3.2.2 Évaluation

3.2.2.1 Évaluation quantitative

Après l'évaluation des capacités prédictives du modèle n-gramme, nous avons tout naturellement cherché à mesurer le gain apporté par cette prédiction sur le temps de sélection. À cet effet, nous avons comparé les différents modes de sélection possibles.

Ces modes de sélection peuvent être caractérisés par deux principaux paramètres. Ces paramètres sont le type de balayage (défilement linéaire ou défilement ligne/colonne) et l'organisation des lettres sur le clavier simulé. Nous avons distingué trois types d'organisation :

- L'ordre aléatoire (noté 0) ne tient compte ni de la prédiction ni de la fréquence des lettres. L'ordre alphabétique et l'ordre « azerty » entrent dans cette catégorie.
- L'ordre fréquentiel est organisé en fonction de la fréquence des lettres observées dans la langue, indépendamment de leur contexte d'apparition. Cet ordre correspond au cas n = 1 du modèle n-gramme.
- Enfin, l'ordre dynamique, réorganisé après chaque saisie (cas n > 1)

Pour le défilement ligne/colonne, la matrice des touches est supposée carrée.

La métrique utilisée est le nombre de défilements moyen. Nous en rappelons ici l'équation :

$$NDM = \sum_{i=1}^{T} P(t_i) \times D(t_i) \qquad (3.5)$$

$P(t_i)$ est la probabilité d'apparition de la touche t_i, $D(t_i)$ est le nombre de défilements nécessaires pour accéder à la touche t_i. Il dépend de la position de la touche dans la matrice et du type de balayage. T est le nombre de touches.

Les résultats sont donnés dans le Tableau 3.5 avec les deux balayages et les différentes organisations (les ordres statiques et dynamiques ont été regroupés). La dernière colonne notée Val rappelle le nombre de validations nécessaires pour sélectionner une lettre. Par exemple, on peut lire que le nombre de défilements moyen pour le défilement ligne/colonne avec une organisation fréquentielle (colonne n = 1) est de 4,3.

	statique	dynamique					
	0	n = 1	n = 2	n = 3	n = 4	n = 5	Val
Défilement linéaire	*33*	7,1	4,7	3,8	3,2	2,9	1
Défilement ligne/colonne	*9*	4,3	-	-	-	-	2

Tableau 3.5 : *NDM* en fonction du balayage et de l'ordre des lettres

Ces résultats mettent clairement en évidence l'intérêt de la prédiction de lettre. Avec n = 5, le *NDM* obtenu par SibyLettre est de 2,9. Par rapport aux systèmes habituels, comme le logiciel présenté dans la Figure 3 (ordre 0 avec défilement ligne/colonne *NDM* = 9), le gain est considérable. Par rapport au mode de sélection optimal sur clavier statique (ordre fréquentiel avec défilement ligne/colonne), le gain est de 32 % (2,9 contre 4,3). Ce gain est d'autant plus appréciable que la sélection nécessite deux fois moins de validations.

Ces résultats permettent également de mettre en évidence certaines observation de [Cantegrit 2001]. Ainsi, il apparaît clairement que les ordres alphabétique et « azerty » habituellement proposés ne sont pas adaptés (ordre 0 dans le Tableau 3.5, résultats en italique). Dans le cas d'un clavier statique, cette synthèse met également en évidence l'efficacité du défilement ligne/colonne sur le défilement linéaire. Ainsi, pour l'ordre fréquentiel (colonne n = 1), le nombre de défilements moyen est de 7,1 pour le défilement linaire contre 4,3 pour le défilement ligne/colonne, soit un gain de 42 %. Ce gain permet de justifier la validation supplémentaire introduite par le défilement ligne/colonne dans le cas des ordres statiques mais reste cependant nettement moins efficace que notre système SibyLettre.

Enfin, les résultats obtenus par SibyLettre ont été comparés aux autres systèmes de sélection de lettre sur clavier simulé (Tableau 3.6) exposés dans le chapitre précédent. Ces résultats sont exprimés comme dans le tableau précédent en *Nombre de Défilements Moyen* pour accéder à une touche.

Clavier	Défilement linéaire	Défilement ligne / colonne
Clavier AZERTY 3 x 10		7,3
Clavier Alphabétique 4 x 8		6,2
Clavier équiprobable 5 x 6	15	6
Clavier fréquentiel 5 x 6	6,9	4,3
Clavier ambigu T8	3,4	
SibyLettre	2,9	

Tableau 3.6 : Performance comparée de SibyLettre en *NDM*

Comme nous l'avions déjà indiqué précédemment, il est difficile de comparer des résultats obtenus dans des conditions expérimentales différentes. Cependant, on peut observer un saut qualitatif entre, d'une part, le clavier ambigu T8 (3,4) et SibyLettre (2,9) et, d'autre part, les autres méthodes (4,3 pour le défilement ligne/colonne sur clavier fréquentiel 5 x 6).

On peut également remarquer que les trois méthodes les plus rapides ont chacune un inconvénient. Le défilement ligne/colonne nécessite deux validations, le clavier ambigu n'affiche le mot souhaité qu'en fin de saisie et SibyLettre utilise un clavier dynamique qui ne permet pas de connaître à l'avance la position de la touche désirée.

3.2.2.2 Évaluation qualitative

Cette dernière évaluation est consacrée à l'évaluation de l'interface auprès de personnes handicapées. L'évaluation a porté sur le clavier dynamique. Cette évaluation est essentiellement qualitative. L'application a été testée avec trois enfants IMC, à l'école du CMRRF de Kerpape. Ces enfants ont l'habitude d'utiliser les aides logicielles.

Deux des enfants ont accueilli très favorablement le logiciel avec le clavier dynamique. Le troisième a préféré retourner à un mode de sélection statique. Ce dernier résultat était cependant prévisible car cet enfant souffre de problèmes de poursuite oculaire ; ce problème de vue rend l'aspect dynamique très perturbant. Pour les deux premiers enfants, les résultats sont très positifs. La phase d'apprentissage à cette nouvelle aide a été simple et rapide. Le gain apporté par SibyLettre est essentiellement apprécié en termes de « confort », le défilement linéaire et son unique validation est particulièrement apprécié. Dans le cadre de l'école, le personnel

encadrant a constaté que ces enfants composent plus de textes et font moins de fautes d'orthographe.

Depuis cette évaluation, le logiciel SibyLettre a été installé sur deux postes à l'école de Kerpape où il est utilisé régulièrement. Le logiciel est également employé en ergothérapie.

3.3 Conclusion

Pour permettre une sélection plus rapide des touches sur le clavier simulé, SibyLettre propose de remplacer le clavier habituel avec défilement ligne/colonne par un clavier dynamique avec prédiction de lettre. Cette méthode est plus rapide et l'accroissement de la charge cognitive lié à l'aspect dynamique du clavier est compensé par le fait que SibyLettre est plus efficace. En particulier, on peut mentionner que les utilisateurs composent plus de textes et font moins de fautes.

Pour la prédiction de lettre, nous avons utilisé le modèle n-gramme. L'utilisation du modèle n-gramme dans ce cadre n'est pas nouvelle, cependant nous avons réalisé une étude approfondie sur ce modèle, étude de $n = 1$ à $n = 5$ sur un très large corpus (100 millions de mots). En particulier, nous avons montré que les performances de la prédiction s'améliorent de manière significative de $n = 1$ (7,1 *NDM*) à $n = 5$ (2,9 *NDM*) et que cette dernière valeur est proche d'une valeur limite à 2,7 *NDM*.

Chapitre 4

Modélisation du langage

Dans le cadre de l'aide à la saisie de texte, il est important de doter le système d'un modèle de langage pour prédire de manière efficace le mot suivant. Ce chapitre est donc un exposé des techniques de modélisation du langage, en introduction à notre système de prédiction SibyMot. Cette présentation a trois objectifs : donner un état de l'art des techniques de modélisation du langage, resituer les modèles des systèmes d'aide à la communication présentés dans le chapitre II et décrire certains des modèles et techniques utilisés par SibyMot. Ce chapitre commence par présenter l'approche symbolique utilisée par des systèmes comme Kombe ou Compansion. Le reste du chapitre est ensuite consacrée de manière plus détaillée à l'approche stochastique retenue pour SibyMot et sur laquelle sont fondés également des systèmes comme PAL ou Profet.

4.1 L'approche symbolique

Chomsky est le premier à avoir adopté une démarche logique et mathématique pour proposer une description formelle de la langue. Son objectif est de déterminer l'ensemble fini des règles qui permettent à la fois d'analyser et générer l'infinité des phrases de la langue, et celles-là seulement. Chomsky commence par classer les grammaires de réécriture [Chomsky 1957]. Nous rappelons qu'une grammaire de réécriture est définie par un quadruplet <Vt, Va, P, R>, où :

- Vt (Vocabulaire Terminal) est un ensemble fini de symboles terminaux (mots du langage)
- Va (Vocabulaire Auxiliaire) est un ensemble fini de symboles non terminaux
- P (P pour Phrase) est un symbole auxiliaire particulier qui sert d'axiome de départ

- R (Règles de réécriture) est un ensemble de règles de réécriture (ou règles de production) de la forme $\alpha \rightarrow \beta$ avec α et $\beta \in (Vt \times Va)^*$

Chomsky classe ces grammaires en quatre catégories (régulières, hors-contexte, contextuelles et non contraintes) avec une relation d'inclusion stricte (régulières $\subset$ hors-contexte $\subset$ ...). Le critère de classification est la forme des règles de production :

- Grammaires régulières (ou grammaires de Kleene), type 3 : $\alpha \in Va$ et $\beta \in (Va)$ ou $\beta \in (Va \cup Vt)$, par exemple : A $\rightarrow$ aB (A, B $\in Va$ et a $\in Vt$). Ces grammaires peuvent être modélisées par un automate d'états finis. Elles permettent, par exemple, l'analyse lexicale mais sont insuffisantes pour traiter tous les aspects de la syntaxe.
- Grammaires hors-contexte (ou indépendantes du contexte, en anglais *Context Free Grammar* ou CFG), type 2 : $\alpha \in Va$ et $\beta \in (Va \cup Vt)^+$, par exemple : A $\rightarrow$ AB. Ces grammaires n'expriment aucune contrainte sur le membre droit. Elles sont dites indépendantes du contexte car le membre gauche peut être réécrit sous la forme du membre droit indépendamment du contexte (ex : A $\rightarrow$ aAb). Ces grammaires permettent en particulier à définir les langages de programmation.
- Grammaires contextuelles (ou sensibles au contexte, en anglais *Context Sensitive Grammar* ou CSG), type 1 : $\alpha \in (Va \cup Vt)^m$ et $\beta \in (Va \cup Vt)^n$ avec $m \leq n$. Ces grammaires tiennent leur nom du fait que les règles peuvent s'écrire de la forme $uxv \rightarrow uyv$ où $u \ldots v$ est appelé le contexte $x \in Va$, $y \in (Va \cup Vt)^+$, et $u, v \in (Va \cup Vt)^*$. Par exemple : aAb $\rightarrow$ aABb réécrit A en AB dans le contexte a..b. Ces grammaires peuvent rendre compte de la complexité du langage naturel mais il n'existe pas d'algorithme pour les manipuler.
- Grammaires non contraintes, type 0 : la seule contrainte est que α ne soit pas vide et contienne au moins un symbole auxiliaire. Par exemple : AB $\rightarrow$ A. Selon [Sabah 1988] ces grammaires sont trop peu structurées pour pouvoir servir de grammaire, de plus, toutes les grammaires qui ont été écrites pour des langues naturelles peuvent être réécrites en utilisant le formalisme des règles contextuelles, même si l'utilisation des règles des grammaires non contraintes permet une écriture plus souple.

Suite à cette classification, Chomsky énonce les inconvénients de ces grammaires. Les deux premières (régulières et hors-contexte) sont insuffisantes du point de vue de la capacité générative. Par exemple, elles ne permettent pas de rendre compte de dépendances croisées comme dans les constructions avec l'adverbe *respectivement*. De plus, quel que soit le type, ces grammaires ne peuvent pas exprimer l'ambiguïté d'une phrase. Par exemple, la phrase « *la peur des ennemis* » peut signifier que les ennemis ont peur ou que l'on a peur des ennemis. De même, ces grammaires ne permettent pas de relier des phrases de sens proche. L'exemple classique donné pour ce dernier

cas est la relation entre tournures active et passive : « *Jean aime Marie* » et « *Marie est aimée par Jean* ».

Pour dépasser ces limites, Chomsky propose la grammaire générative. Cette grammaire distingue deux structures : la structure « de surface » (la phrase telle qu'elle est écrite) et la structure « profonde » (la phrase telle qu'elle est « pensée »). La structure « profonde » est décrite par une grammaire syntagmatique, ou autrement appelée grammaire de constituants (grammaire de type 2 ou 3). Pour permettre le passage de la structure profonde à la structure de surface, il propose des mécanismes de « transformation ». Ainsi, l'ambiguïté d'une phrase comme « la peur des ennemis » est liée au fait que deux phrases de structures profondes différentes obtiennent, après transformations, une même structure de surface. À l'inverse une phrase de même structure profonde peut produire des phrases différentes selon les dérivations utilisées (comme le cas actif/passif). [Kuno 1962] et [Woods 1970] proposent des analyseurs issus de cette grammaire transformationnelle.

Suite à ces travaux, de nombreux formalismes de description linguistique ont vu le jour visant à enrichir l'expressivité des grammaires de réécriture.

Les recherches en logique et en programmation (qui ont par ailleurs donné le langage Prolog) ont produit les grammaires dites « logiques » [Gazdar 1979]. Ce type de grammaire est à l'origine de grammaires telles que les grammaires de métamorphose [Colmerauer 1975] ou les grammaires de clauses définies (ou DCG) [Pereira 1980]. Dans le chapitre II, le système Kombe que nous avons décrit est issu d'une version des grammaires de métamorphose [Richardet 1998]. Dans cette approche, les règles de réécriture sont conçues comme des clauses. Une séquence de mots est correcte si elle correspond à un théorème déductible des axiomes de départ. Dans ce parallèle, l'analyse syntaxique devient le développement de la preuve de ce théorème. Une phrase syntaxiquement et sémantiquement correcte est appelée « expression bien formée ».

Dans l'idée de s'affranchir de la structure de surface, [Fillmore 1968] a présenté les grammaires de cas qui renforcent la composante sémantique dans l'analyse de la phrase. L'analyse s'articule autour du verbe, l'action, et les autres constituants se voient affectés de cas reflétant leur rôle sémantique par rapport à l'action. Comme exemples de cas, on peut citer le cas « agent », celui qui est à l'origine de l'action, « patient » celui qui reçoit l'action, « instrument » ce avec quoi est faite l'action, etc. Ce formalisme est par exemple adapté aux langues dont l'ordre des mots est variable (contrairement à l'anglais) [Schadle 1999]. Dans les systèmes de communication assistée, Compansion utilise une représentation sémantique de la phrase fondée sur ce formalisme.

Les grammaires d'unification sont le dernier type de grammaires que nous évoquons dans cette partie sur l'approche symbolique. Apparues dans les années quatre-vingt, elles s'opposent aux travaux de Chomsky en rejetant les transformations et en réhabilitant le rôle de la structure de surface et du lexique. En particulier ces grammaires adoptent le même type de représentation (structure de traits) pour le lexique, la syntaxe et la sémantique. La figure ci-dessous représente la phrase « Jean aime Marie » mais cette fois-ci en utilisant le formalisme des structures de traits.

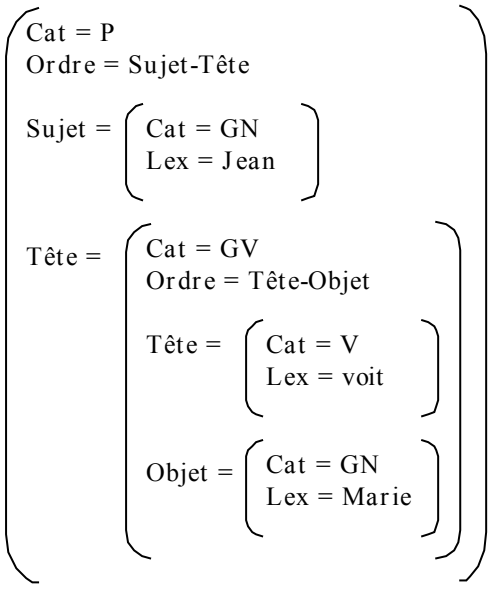


Figure 4.1 : Exemple de structure de traits

Parmi les grammaires d'unification, on peut citer principalement les quatre grammaires suivantes [Abeillé 1993] :

- La grammaire lexicale fonctionnelle (LFG) [Kaplan 1982] est issue de travaux en psycholinguistique. Elle abandonne les transformations jugées peu plausibles et réhabilite les fonctions grammaticales pour permettre l'expression d'universaux linguistiques comme la notion de sujet ou de passif.
- La grammaire syntagmatique généralisée (GPSG) [Gazdar 1985] vient de travaux en linguistique mathématique dont le but était de construire un modèle équivalent à une grammaire hors contexte. En particulier, il s'agissait de montrer qu'il n'y a pas de propriété dans les langues qui nécessitent un modèle plus puissant.
- La grammaire syntagmatique guidée par les têtes (HPSG) [Pollard 1994] a été conçue pour intégrer de manière plus explicite des différents niveaux linguistiques : phonétique, syntaxe et sémantique.
- La grammaire d'arbres adjoints (TAG) [Joshi 1987] [Abeillé 2000b] est issue à la fois de travaux mathématiques et linguistiques. Il s'agit d'une part de

formaliser la composante récursive de la grammaire par l'opération d'adjonction et d'autre part de lexicaliser intégralement la grammaire.

Dans le chapitre II, nous avons déjà évoqué le problème des contraintes imposées par l'utilisation de telles grammaires : les mots proposés correspondent aux contraintes définies par la grammaire (cf. Kombe). Ceci est en partie lié au problème de couverture : il n'existe actuellement aucune grammaire utilisable couvrant la totalité des phénomènes syntaxiques de la langue. Par ailleurs, il faut noter que l'accroissement d'une grammaire est loin d'être trivial. Il augmente souvent de manière considérable la complexité de l'analyse (ou de la génération) et l'ajout de règles remet parfois en cause des règles préexistantes. Enfin, l'essor des méthodes stochastiques [Charniak 1993], que nous présentons dans la partie suivante, a mis en évidence les difficultés de ces approches à décrire et à traiter l'ensemble très diversifié des variations autour des phénomènes linguistiques et des constructions attestées en corpus.

La présentation de l'approche symbolique a permis d'introduire des notions linguistiques comme la notion d'analyse et de représentation syntaxique. Nous avons également décrit de manière succincte les théories sur lesquelles se fondent des systèmes comme Kombe ou Compension. Nous allons maintenant exposer de manière plus détaillée l'autre approche, l'approche stochastique. Notons qu'il existe de nombreux systèmes hybrides combinant les deux approches symbolique et stochastique comme les grammaires stochastiques que nous présentons dans la partie suivante.

4.2 L'approche stochastique

Contrairement à l'approche symbolique, l'approche stochastique utilise un minimum de connaissances *a priori* sur la langue. Le but des modèles de langage probabilistes est d'extraire les régularités de la langue à partir de données apprises généralement sur de larges corpus. Ces connaissances sont notamment représentées sous la forme de fréquences de suites de mots. De plus, là où l'approche symbolique cherche à construire un arbre syntaxique pour vérifier la validité d'une séquence de mots, l'objectif de l'approche stochastique est plutôt de maximiser la probabilité d'apparition d'une séquence de mots.

Le modèle que l'on peut considérer comme le modèle de référence dans la modélisation stochastique du langage est le modèle statistique n-gramme. Celui-ci permet d'estimer la probabilité d'apparition d'un mot en fonction des derniers mots (généralement les deux derniers mots sont utilisés). Cette taille de contexte peut paraître très réduite pour prédire le mot suivant, elle permet pourtant d'obtenir de bons résultats. Le contexte pris en compte par le modèle est par ailleurs une des problématiques de la modélisation stochastique du langage et d'autres modèles

utilisent des tailles de contexte plus larges, typiquement la phrase et le document en cours. De manière plus générale, on peut dire que plusieurs niveaux de contexte semblent intervenir dans la langue : local, phrastique et thématique (document). Aussi, nous avons choisi de présenter les modèles de langage probabilistes en les regroupant par le type du contexte. Nous commençons par exposer le modèle n-gramme et les autres modèles qui réalisent leurs estimations à partir des derniers mots (partie 4.3) puis ceux qui modélisent toute la phrase (partie 4.4) et enfin ceux qui extraient des informations du document en cours (partie 4.5).

Le modèle n-gramme et plus généralement les modèles probabilistes posent un deuxième problème principal, celui de l'estimation des séquences non observées sur le corpus d'apprentissage. Les techniques pour obtenir des estimations fiables seront donc abordées à la suite de la présentation des modèles (partie 4.6).

Enfin, ce chapitre se terminera (partie 4.7) par la présentation de la mesure habituellement utilisée pour l'évaluation des modèles probabilistes, la mesure d'entropie, déjà succinctement décrite dans SibyLettre.

4.3 Le modèle n-gramme et ses dérivés

4.3.1 Modèle n-gramme

Le modèle n-gramme est issu de la théorie de l'information [Jelinek 1976]. Celle-ci suppose que la probabilité d'apparition d'un événement dépend de toute la séquence d'événements passés. La probabilité d'occurrence d'une séquence de N mots $W = w_1,\ldots,w_N$ est ainsi décrite comme le produit des probabilités conditionnelles :

$$P(W) = \prod_{i=1}^{N} P(w_i \mid w_1,\ldots,w_{i-1}) \qquad (4.1)$$

où $P(w_i \mid w_1,\ldots,w_{i-1})$ est la probabilité d'apparition du mot w_i connaissant les i-1 premiers mots de la séquence W. La suite des mots précédemment émis $w_1,\ldots,w_{i-1}$ est également appelée historique du mot w_i et notée h_i, ou encore de manière plus générale h.

Compte tenu du nombre de séquences possibles, ce résultat n'est pas directement exploitable. L'estimation de ces probabilités nécessiterait en effet un trop grand nombre de données. Une bonne approximation consiste à limiter le contexte à n observations. Ce modèle est appelé modèle n-gramme. Il exprime le fait que la probabilité d'apparition d'une séquence peut être donnée de manière satisfaisante par une combinaison de probabilités fondée sur les *n-1* observations précédentes :

$$P(W) \cong \prod_{i=1}^{N} P(w_i \mid w_{i-n+1},\ldots,w_{i-1}) \qquad (4.2)$$

L'approximation est d'autant meilleure que le paramètre n est élevé. Dans les cas $n = 1$, $n = 2$ et $n = 3$, on parle respectivement d'uni-gramme, bi-gramme et tri-gramme. Pour $n = 1$, la taille du contexte est nulle, la probabilité délivrée est donc hors contexte. Par exemple, les logiciels du commerce avec une liste de mots, tels que nous les avons présentés dans le chapitre II, classent ces mots selon leur probabilité hors contexte, ils sont donc une illustration de l'uni-gramme.

La probabilité d'apparition d'un mot après une séquence donnée est estimée de la manière suivante[9] :

$$P(w_i \mid w_{i-n+1},\ldots,w_{i-1}) = \frac{C(w_{i-n+1},\ldots,w_{i-1},w_i)}{C(w_{i-n+1},\ldots,w_{i-1})} \qquad (4.3)$$

où le $C(.)$ est le nombre d'occurrences de l'argument entre parenthèses. $C(w_{i-n+1},\ldots,w_{i-1})$ est le nombre d'occurrences du contexte et $C(w_{i-n+1},\ldots,w_{i-1},w_i)$ celui du mot w_i après ce même contexte. L'expression précédente divise donc la fréquence d'apparition de la séquence cherchée par la fréquence de son contexte.

Si V est la taille du vocabulaire, le nombre de paramètres du modèle à estimer est V^n : la probabilité de V mots est à déterminer pour V^{n-1} contextes possibles.

La principale difficulté du modèle n-gramme est liée à l'espace des paramètres. Par exemple, pour un vocabulaire V = 10 000 et le modèle tri-gramme $n = 3$, le nombre de paramètres est de 10^{12}. Ainsi, quelle que soit la taille du corpus d'apprentissage se pose le cas de séquences non observées qui se verront affecter d'une probabilité nulle alors qu'elles sont possibles. Les deux modèles que nous allons maintenant présenter, le modèle n-classe et le modèle n-POS, permettent de résoudre, en partie, ce problème.

4.3.2 Modèle n-classe

Pour réduire l'espace des paramètres à estimer, une solution consiste à regrouper les mots par classes d'équivalence, ces classes pouvant être fixées *a priori* ou déterminées par apprentissage. C'est ce que propose le modèle n-classe [Jardino 1993], [Witschel 1993], [Jardino 2000]. La probabilité d'apparition d'un mot w_i, à la suite d'un historique h_i, est alors exprimée comme la combinaison de la probabilité du

[9] Pour être plus précis, cette méthode est l'estimation par maximum de vraisemblance. Il existe d'autres méthodes comme l'estimation par maximum *a posteriori* ou l'estimation Bayésienne qui ne sont pas abordées dans le cadre de ce mémoire. Pour plus d'informations on pourra se référer à [Spargins 1965], [Mood 1974].

mot dans sa classe c_i par la probabilité d'occurrence de cette classe en fonction des classes des n-1 derniers mots :

$$P(w_i \mid h_i) = P(w_i \mid c_i) \times P(c_i \mid c_{i-n+1},\ldots,c_{i-1}) \tag{4.4}$$

La probabilité d'un mot dans sa classe $P(w_i \mid c_i)$ est estimée par la formule suivante :

$$P(w_i \mid c_i) = \frac{C(w_i)}{C(c_i)} \tag{4.5}$$

où $C(w_i)$ est la fréquence du mot w_i sur le corpus d'apprentissage et $C(c_i)$ le nombre d'occurrences de la classe c_i.

La probabilité d'apparition d'une classe $P(c_i \mid c_{i-n+1},\ldots,c_{i-1})$ est calculée de manière analogue à celle de la probabilité d'apparition d'un mot $P(w_i \mid w_{i-n+1},\ldots,w_{i-1})$ dans le modèle n-gramme :

$$P(c_i \mid c_{i-n+1},\ldots,c_{i-1}) = \frac{C(c_{i-n+1},\ldots,c_{i-1},c_i)}{C(c_{i-n+1},\ldots,c_{i-1})} \tag{4.6}$$

Si V est la taille du vocabulaire et C le nombre de classes, le nombre de paramètres du modèle à estimer est de $V \times C^n$. L'espace des paramètres est donc fortement réduit par rapport à celui du modèle n-gramme V^n, si l'on considère C très inférieur à V.

En ce qui concerne la classification des mots, celle-ci est généralement réalisée de manière automatique, à partir d'un critère d'optimisation. Les principaux critères utilisés sont le maximum de vraisemblance, la validation croisée et l'information mutuelle [Brown 1992], [Kneser 1991], [Jelinek 1990].

L'intérêt principal de ce modèle est la réduction de l'espace de paramètres. Ceci permet d'obtenir des estimations plus fiables ou bien, pour un espace de paramètres sensiblement égal, d'accroître la taille de l'historique pris en compte.

4.3.3 Modèle n-POS

L'idée du modèle n-POS (Part Of Speech) [Niesler 1996] [Pastor 1998] est proche de celle du modèle n-classe, réduire l'espace des paramètres en utilisant des classes. Les classes correspondent généralement à des catégories grammaticales (noms communs, adjectifs, etc.) ou des classes sémantiques (noms des mois, prénoms, etc.). Ce modèle introduit donc implicitement des notions syntaxiques voire sémantiques. À la

différence du modèle n-classe, où chaque mot appartient à une et une seule classe, un mot peut maintenant appartenir à plusieurs classes. Par exemple, le mot *le* peut désigner un article défini ou un pronom personnel. Pour tenir compte de cette ambiguïté, la probabilité d'apparition d'un mot après une séquence est calculée pour chacune des classes potentielles du mot :

$$P(w_i \mid h_i) = \sum_{t_i \in T w_i} P(w_i \mid t_i) \times P(t_i \mid t_{i-n+1}, \ldots, t_{i-1}) \qquad (4.7)$$

où t_i est une partie du discours (encore appelée étiquette syntaxique ou *tag*) associée au mot w_i, et T_{wi} l'ensemble de toutes les étiquettes syntaxiques possibles du mot.

L'estimation des probabilités $P(w_i \mid t_i)$ et $P(t_i \mid t_{i-n+1}, \ldots, t_{i-1})$ suit le même principe que les modèles précédents :

$$P(w_i \mid t_i) = \frac{C(w_i, t_i)}{C(t_i)} \qquad (4.8)$$

$$P(t_i \mid t_{i-n+1}, \ldots, t_{i-1}) = \frac{C(t_{i-n+1}, \ldots, t_i)}{C(t_{i-n+1}, \ldots, t_{i-1})} \qquad (4.9)$$

Pour estimer un modèle n-POS, l'étiquetage de chaque mot du corpus par ses vraies classes est nécessaire.

4.3.4 Modèle morphologique

Le modèle morphologique est considéré comme un raffinement du modèle n-POS [El-Bèze 1990], [Cerf-Danon 1991]. Il utilise la notion de lemme (forme canonique d'un mot, par exemple le masculin singulier pour les noms ou l'infinitif pour les verbes) et distingue les mots grammaticaux (comme les conjonctions ou les prépositions) des mots lexicaux (noms, verbes, adjectifs) porteurs de sens. L'intérêt du lemme est de rapprocher les différentes formes fléchies d'un même mot. Quant à la distinction mot grammatical, mot lexical, elle apparaît dans une composante n-lemmes portant seulement sur les lemmes des mots lexicaux. L'expression du modèle n-POS devient :

$$P(w_i \mid h_i) = \sum_{t_i \in T w_i} P(w_i \mid t_i) \times [\lambda_{ti}\, P(t_i \mid t_{i-n+1}, \ldots, t_{i-1}) + (1 - \lambda_{ti})\, P_M(w_i \mid t_i, h_i)] \qquad (4.10)$$

où λ_{ti} est un paramètre d'interpolation dépendant de la classe à prédire. La différence avec le modèle n-POS est l'ajout d'une composante P_M appelée composante morphologique. P_M est définie comme suit :

$$P_M(w_i \mid t_i, h_i) = \sum_{t_i \in Tw_i} P(w_i \mid t_i, l_i) \times P(l_i \mid l_{i-j}, \ldots, l_{i-k}) \qquad (4.11)$$

où l_i désigne le lemme du mot w_i. $l_{i-j}, \ldots, l_{i-k}$ sont non pas les lemmes des n derniers mots mais les lemmes des n derniers mots porteurs de sens, typiquement les mots lexicaux. Il est à noter que cette composante ajoute au modèle des connaissances d'ordre sémantique et permet d'augmenter la taille du contexte pris en compte dans la prédiction.

Pour l'apprentissage des paramètres du modèle, le corpus doit contenir deux informations pour chaque mot : sa classe (POS) et son lemme. En supposant que chaque couple (mot, POS) identifie un seul lemme, la phase d'étiquetage du corpus peut se limiter à celle de la partie du discours, le lemme étant ensuite ajouté de manière déterministe.

4.3.5 Optimisations de l'approche n-gramme

Le principal problème du modèle n-gramme est l'estimation des paramètres et les modèles n-classe et n-POS apportent des solutions partielles à cette difficulté. Un autre inconvénient du modèle n-gramme est de ne prendre en compte qu'un contexte à très courte distance (2 mots pour le modèle tri-gramme). Il existe de nombreuses approches dans la littérature essayant d'améliorer le modèle n-gramme, nous en présentons quelques-unes dans ce paragraphe.

Dans le but de réduire le nombre de paramètres à estimer, [Brugnara 1996] a proposé un modèle nommé *scalable n-gram* qui cherche à supprimer les paramètres qui affectent peu les performances du modèle. Avec ce modèle, certaines séquences de mots sont estimées par des modèles d'ordre m inférieur à celui de référence n. En réduisant les paramètres des modèles 4-grammes et 5-grammes [Kneser 1996] a ainsi obtenu des performances meilleures qu'un modèle 3-gramme.

Dans l'exemple du contexte « *les chercheurs* » et « *les chercheurs d'aujourd'hui* », on s'attend à obtenir une probabilité similaire pour le mot « *inventent* ». Pour pallier ce problème, une approche utilisant des n-grammes distants a été proposée [Langlois 1999]. Le modèle combine plusieurs n-grammes qui extraient leur contexte à des distances variables dans l'historique. Par exemple, un modèle trigramme distant d'ordre 2 prédit le mot w_i en se basant sur w_{i-3}, w_{i-2}. Notons que dans cette

modélisation, le n-gramme classique est tout simplement un n-gramme distant d'ordre 1.

Le modèle permugramme [Schukat 1995] est une autre généralisation des modèles n-gramme. Il combine différents sous-modèles n-grammes agissant sur des permutations différentes de l'historique du mot à prédire. Cette approche, comme la précédente, permet de tenir compte de dépendances entre mots qui ne sont pas immédiatement adjacents.

Une autre classe de modèles propose d'établir la prédiction non plus sur des mots mais sur des séquences de longueur variable. Ces modèles apprennent des séquences comme *pomme de terre* ou *premier ministre* qui peuvent ensuite être considérées comme des unités dans un modèle n-gramme [Jelinek 1990]. [Deligne 1996] a ainsi proposé une approche de ce type en utilisant un modèle multi-gramme et [Zitouni 2000] en se fondant sur la mesure d'information mutuelle pour déterminer les séquences pertinentes. La notion de séquence variable a également son utilité dans l'aide à la saisie de texte. En effet, en prédisant des unités de longueur supérieure, un certain nombre de saisies peuvent être économisées.

Ces modèles améliorent souvent l'efficacité d'un modèle n-gramme classique. Cependant les approches restent très proches, elles se limitent malgré tout à des contextes très courts (autour de 3 ou 4 mots) et n'utilisent pas (ou peu) de connaissances linguistiques.

4.4 Les modèles de phrase

4.4.1 Modèle de grammaire probabiliste

Une des difficultés du modèle n-gramme en raison de son contexte réduit est qu'il ne permet pas d'exploiter les contraintes syntaxiques imposées par le début de la phrase. Par exemple, dans la phrase « les pommes de l'archiduchesse sont cuites », la contrainte d'accord qui porte sur le mot « cuites » ne peut être réalisée qu'avec le mot « pommes » qui se situe dans une position bien antérieure dans la phrase.

Les grammaires formelles introduites par Chomsky permettent d'exprimer ce type de contrainte grammaticale. D'où l'idée de combiner l'approche stochastique avec l'approche symbolique en décrivant des règles grammaticales probabilistes. Cependant la création d'une telle grammaire probabiliste est loin d'être triviale, en particulier en ce qui concerne les hypothèses d'indépendance qui doivent être établies pour l'estimation des règles à appliquer [Allen 1997]. Un exemple d'ajustement des probabilités pour une grammaire contextuelle est proposé par [Baker 1979]. [Gillet 1998] présente une approche combinant modèle n-gramme et grammaire hors-contexte stochastique.

4.5 Les modèles de document

4.5.1 Modèle cache

Le modèle cache propose une approche différente pour la prédiction de mot. Au lieu de se limiter à un contexte gauche réduit, il cherche à étendre la taille du contexte, typiquement le texte en cours. L'idée d'utiliser un contexte étendu provient du jeu de Shannon développé par IBM. L'objectif du jeu de Shannon est relativement simple : il s'agit de prédire pour une séquence de mots donnée le mot suivant. Le jeu commence avec un contexte nul puis s'étend à un mot, deux mots, etc. jusqu'à atteindre une taille de paragraphe. Ce jeu a ainsi montré qu'un humain est plus performant qu'un modèle tri-gramme, connaissant un historique complet [Rosenfeld 1994]. En particulier, 40 % des mots à prédire étaient contenus dans l'historique. Le modèle cache utilise donc le fait que les mots apparus dans le contexte ont plus de chances de réapparaître par la suite. L'estimation des probabilités d'apparition de chaque mot est ainsi uniquement calculée à partir des fréquences des M derniers mots observés [Kuhn 1988], [Kuhn 1990], [Kupiec 1989] :

$$P(w_i \mid h_i) = \frac{1}{M} \sum_{m=1}^{M} \delta(w_i \mid w_{i-m}) \qquad (4.12)$$

avec $\delta(w \mid v) = 1$ si et seulement si $w = v$, 0 sinon.

Le modèle trigger [Ney 1994] [Tillmann 1996] est une variante à ce modèle. Dans ce dernier, les mots de l'historique servent de déclencheur pour les mots à prédire. Le modèle trigger utilise des couples de mots (w_i, w_j), les triggers, qui ont une forte corrélation entre eux. Si le mot w_i apparaît dans les derniers mots, il augmente la probabilité du mot w_j. Les couples sont obtenus à partir d'un corpus d'apprentissage, en se fondant généralement sur l'information mutuelle comme critère d'optimalité [Rosenfeld 1992].

Ces deux modèles, cache et trigger, sont surtout utilisés en étant combinés à d'autres modèles, tels que les modèles n-gramme ou n-POS, pour renforcer la probabilités de mots à partir d'un contexte plus long. Par exemple, [Tillmann 1996] propose une combinaison avec le modèle n-gramme en utilisant l'interpolation linéaire (développé plus loin dans le chapitre en 4.6.4) :

$$P(w_i \mid h_i) = \lambda_1 P_{Gram}(w_i \mid h_i) + \lambda_2 P_{Cach}(w_i \mid h_i) + \lambda_3 P_{Trig}(w_i \mid h_i) \qquad (4.13)$$

où $P_{Gram}(.)$, $P_{Cach}(.)$ et $P_{Trig}(.)$ sont respectivement les probabilités délivrées par les modèles n-gramme, cache et trigger. λ_1, λ_2 et λ_3 sont les paramètres d'interpolation calculés par apprentissage, leur somme est égale à 1.

4.5.2 Modèle s'adaptant au thème du document

Toujours dans la problématique d'extraire de l'information du document en cours, il existe d'autres modèles qui modifient un plus grand nombre de paramètres que les modèles cache et trigger.

En partant de l'approche n-gramme classique, le modèle *mixture n-gram* propose d'interpoler plusieurs sous-modèles n-grammes [Iyer 1994], [Kneser 1995], [Clarkson 1997]. Chaque sous-modèle possède ses propres paramètres dont les estimations ont été obtenues par apprentissage sur des corpus regroupés par thème. Au fur et à mesure que la longueur du document augmente, les paramètres d'interpolation évoluent afin de s'ajuster aux sous-modèles identifiés comme se rapprochant le plus du thème en cours.

L'approche précédente a cependant un inconvénient : elle amplifie le problème du modèle n-gramme concernant l'espace des paramètres à estimer puisque les paramètres sont estimés sur autant de thèmes que le modèle en dispose. Pour éviter ce problème, [Chen 1997] et [Khudanpur 2000] proposent de scinder le vocabulaire en trois parties : le vocabulaire général qui est toujours le même, le vocabulaire du thème et le vocabulaire hors-thème. Ainsi seuls les mots du thème en cours voient leur probabilité estimée sur des données spécifiques. Quant à l'estimation des mots hors-vocabulaire elle est pondérée par un facteur qui amoindrit leur probabilité d'apparition.

Le dernier modèle que nous évoquerons dans cette partie est le modèle SLA (*Semantic Latent Analysis*) [Bellegarda 1997]. Ce modèle est issu de la recherche d'information (IR *Information Retrieval*) [Deerwester 1990] et son approche est très différente du modèle n-gramme. Initialement conçu pour l'indexation et la recherche de document, ce modèle définit une matrice de correspondance W où chaque ligne est associée à un mot du lexique L et chaque colonne correspond à un des D documents. Ceci suppose que le corpus d'apprentissage est préalablement scindé en documents. Chaque valeur de la matrice $W_{i,j}$ donne ainsi le nombre d'occurrences du mot w_i dans le document d_j. Cette matrice est ensuite décomposée en trois termes suivant la méthode de décomposition en valeurs singulières :

$$W = U\,S\,V^T \tag{4.14}$$

où S est une matrice carrée $R \times R$ appelée *espace sémantique*, U est une matrice de dimension $L \times R$ qui représente les mots du lexique dans l'espace sémantique S, V est une matrice de dimension $R \times D$ qui représente les documents dans S et ^{T} dénote une matrice transposée. Dans ce modèle, deux mots sont considérés comme sémantiquement proches si les vecteurs u_i et u_j qui les représentent dans U ont une

distance métrique faible. La probabilité d'apparition d'un mot w_i dans un contexte h (l'ensemble des mots du documents en cours) est définie comme suit :

$$P(w_i \mid h,S) = P(w_i \mid \delta_{i-1}) \tag{4.15}$$

où la condition sur S est de montrer que la probabilité d'apparition du mot dépend de la décomposition précédente (4.16). La probabilité $P(w_i \mid \delta_{i-1})$ est mesurée par une normalisation de la distance entre le vecteur u_i ($u_i \in U$) du mot à prédire w_i et l'ensemble des vecteurs représentatifs de δ_{i-1} dans l'espace S, où δ_{i-1} est considéré comme un pseudo-document. Cette probabilité reflète ainsi la pertinence du mot dans le contexte h. Comme on peut le constater, une différence notable avec le modèle n-gramme est que l'ordre d'apparition des mots dans l'historique n'est pas pris en compte. Ce modèle est par contre un mauvais prédicteur pour les mots outils, c'est pourquoi cette approche est généralement combinée à un autre modèle comme le modèle n-gramme [Bellegarda 1997].

En conclusion à cette présentation sur les modèles de langage probabilistes, on peut constater que, malgré les améliorations proposées, le modèle n-gramme demeure le modèle de référence de la modélisation stochastique. Les modèles s'adaptant au document offrent selon nous des perspectives intéressantes en restant toutefois combinées au modèle n-gramme.

Récemment des modèles comme [Chelba 2000] proposent une alternative en élargissant la taille du contexte du modèle n-gramme aux n-1 derniers syntagmes grâce à l'introduction de connaissances syntaxiques. C'est cette alternative que nous avons choisie pour SibyMot et qui sera présentée en partie introductive à celui-ci dans le chapitre V.

4.6 Estimation fiable des paramètres

Après avoir présenté les différents modèles probabilistes, nous abordons maintenant le problème de l'estimation des données manquantes. Comme nous l'avons déjà évoqué, le nombre de paramètres à estimer est très important et nombre de séquences de mots envisageables pourront n'avoir jamais été rencontrées sur le corpus d'apprentissage. Cette difficulté est liée au problème de l'éparpillement des données [Allen 1997] : un grand nombre de données est concentré sur un faible nombre de cas. Dans le cas des séquences jamais observées, l'estimation par maximum de vraisemblance délivre une probabilité nulle ce qui n'est pas souhaitable. Ce problème est connu sous le nom de « *zero frequency problem* » [Witten 1991] et sa bonne résolution conditionne les capacités de généralisation du modèle. Une solution

générale consiste à prélever à l'ensemble des événements observés une fraction de la masse de probabilité et à la redistribuer aux événements non observés.

Dans cette partie les conventions de notation sont définies comme suit :

- w désigne le mot à prédire et h son contexte, par extension hw désigne la séquence du contexte suivie du mot à prédire
- la probabilité d'apparition d'un mot w en fonction de son contexte h est ainsi notée $P(w \mid h)$

Nous introduisons également les notions de *fréquence relative* et de *fréquence conditionnelle réduite* :

- la *fréquence relative* d'un mot sachant un contexte h est définie suivant le principe de maximum de vraisemblance par : $f(w \mid h) = C(hw) \;/\; C(h)$. Ce qui correspond à la probabilité d'apparition d'un mot définie en (4.3).
- la notion de *fréquence conditionnelle réduite* est utilisée pour exprimer le fait qu'une fraction de la masse des probabilités est déduite de celle des événements observés. Ainsi, pour une séquence hw observée, sa *fréquence conditionnelle réduite* f^* est inférieure à sa *fréquence relative* f :

$$0 \leq f^*(w \mid h) \leq f(w \mid h) \quad \forall\ hw \in V^n \tag{4.17}$$

Ainsi, les méthodes exposées par la suite donnent toutes une estimation de la *fréquence conditionnelle réduite* f^* et redistribuent la différence avec la *fréquence relative* f sur les séquences jamais observées.

4.6.1 Lissage par seuil

La méthode la plus simple à mettre en œuvre est le lissage par seuil (*floor discounting*). Cette technique consiste à donner une même valeur ε aux événements non observés :

$$P(w \mid h) = \begin{cases} f^*(w \mid h) & \text{si } C(hw) > 0 \\ \varepsilon & \text{sinon} \end{cases}$$

Si $C_0(h)$ est le nombre de fois où $C(hw) = 0$ (i.e. le nombre d'événements non observés) alors la masse de probabilité à retrancher est $\varepsilon C_0(h)$. La fréquence relative est donc exprimée comme suit :

$$f^* = P(w \mid h) \cdot (1 - \varepsilon C_0(h))$$

De plus, dans le cas où le contexte h n'a jamais été observé $C(h) = 0$, le lissage par seuil est calculé sur la fréquence des mots hors contexte (donc l'uni-gramme). La constante ε s'applique cette fois-ci aux mots du vocabulaire de fréquence nulle (V_0 étant leur nombre) :

$$f^* = P_{1\text{-gram}}(w) \cdot (1 - \varepsilon V_0)$$

avec : $$P_{1\text{-gram}}(w) = C(w) \;/\; V$$

Le lissage par seuil est intéressant car il est simple à implanter et est peu coûteux en terme de stockage. Cependant, en affectant la même constante aux séquences inconnues, il ne permet pas de les différencier ce qui n'est pas très satisfaisant pour la prédiction.

4.6.2 Lissage linéaire

Contrairement à la méthode précédente, le lissage linéaire (*linear discounting*) permet une estimation différenciée des événements non observés :

$$P(w \mid h) = \begin{cases} f(w \mid h) \cdot (1-\lambda) & \text{si } C(hw) > 0 \\ \lambda \cdot \alpha(h,w) & \text{sinon} \end{cases}$$

$$\alpha(w \mid h) = \begin{cases} \dfrac{C(w)}{\sum_{k, C(hw_k)=0} C(w_k)} (1 - \varepsilon V_0) & \text{si } C(w) > 0 \\ \varepsilon & \text{sinon} \end{cases}$$

Ces équations doivent être interprétées comme suit. Soit $(1-\lambda)$ la masse de probabilité accordée aux événements observés (légèrement inférieure à 1). La quantité λ est donc la probabilité attribuée aux événements non observés. Pour exprimer que la probabilité varie en fonction d'un mot w inconnu après h, la quantité λ est pondérée par un coefficient $\alpha(h,w)$ dépendant de w. Ce coefficient $\alpha(h,w)$ est directement proportionnel à sa fréquence hors contexte $C(w)$ moyennée par la somme des fréquences hors contexte des mots inconnus après h (pour s'assurer que $\sum_w \alpha(h,w) = 1$). Enfin, le coefficient α est pondéré par un coefficient $(1 - \varepsilon V_0)$ pour les mots du vocabulaire de fréquence nulle.

4.6.3 Lissage absolu

La méthode par lissage absolu (*absolute discounting*) diffère du lissage linéaire sur deux points. D'abord, la probabilité retirée à chaque événement observé est fixe, contrairement au lissage linéaire où elle est proportionnelle à la probabilité de l'événement (d'où le nom de ces méthodes). Par contre, et c'est le deuxième point, la masse globale retirée aux événements observées est variable :

$$P(w \mid h) = \begin{cases} f(w \mid h) - \dfrac{d}{C(h)} & \text{si } C(hw) > 0 \\ \lambda(h) \cdot \alpha(h,w) & \text{sinon} \end{cases}$$

Avec :

$$\lambda(h) = d\,\frac{C_+(h)}{C(h)}$$

où $C_+(h)$ désigne le nombre de successeurs de h (événements observés) et d une constante inférieure à 1.

L'intérêt de cette méthode est d'introduire un critère de fiabilité dépendant du contexte h. La masse $\lambda(h)$ retranchée est maximale lorsqu'il y a autant d'événements observés que le nombre d'occurrences de l'historique (i.e. tous les mots observés après h sont différents).

4.6.4 Lissage par interpolation

Cette méthode a été proposée initialement par [Jelinek 1976]. Elle se distingue des précédentes dans la mesure où elle ne modifie pas directement les paramètres estimés par le maximum de vraisemblance. Il s'agit d'une combinaison linéaire de paramètres dont au moins un des termes n'est pas nul. Dans le cas d'un modèle tri-gramme, la probabilité d'apparition d'un mot ainsi redéfinie devient :

$$P(w_i \mid h) = \lambda_3 P_{3\text{-}gram}(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 P_{2\text{-}gram}(w_i \mid w_{i-1}) + \lambda_1 P_{1\text{-}gramg}(w_i) + \lambda_0 \qquad (4.18)$$

où λ_j sont les paramètres d'interpolation avec $\sum \lambda_j = 1$. Il existe plusieurs méthodes pour calculer les paramètres d'interpolation. On pourra se référer à [Boite 2000] pour différentes méthodes d'estimation de ces paramètres. Parmi ces méthodes, nous citerons en particulier l'algorithme EM (*deleted-interpolation*) que nous avons utilisé dans SibyMot.

Nous avons présenté plusieurs techniques de lissage qui permettent de résoudre partiellement au problème principal du modèle n-gramme : l'estimation des données manquantes. Parmi les quatre méthodes présentées, lissage par seuil, linéaire, absolu

et par interpolation, nous avons utilisée cette dernière, le lissage par interpolation, couramment utilisée dans la littérature.

4.7 Évaluation d'un modèle probabiliste

La dernière partie de ce chapitre concerne la mesure de perplexité utilisée en modélisation probabiliste pour évaluer un modèle que nous avons déjà employée dans l'évaluation de la prédiction de lettre.

Pour mesurer la perplexité, on introduit l'entropie issue de la théorie de l'information. Dans le cadre de la modélisation du langage, le principe de l'entropie consiste à calculer la moyenne de la log-probabilité sur un corpus de test. Soit P la vraie distribution que nous ne connaissons pas, l'entropie H est définie comme suit, pour un corpus de test de k mots :

$$H = \lim_{k \to +\infty} \sum_{x} P(x) \log_2 P(x) \tag{4.19}$$

Pour déterminer la probabilité, on utilise l'estimation délivrée par le modèle de langage notée $\tilde{P}$. Si N est le nombre d'échantillons du corpus de test, la log-probabilité lPN s'exprime alors :

$$lP_N = -\frac{1}{N} \sum_{i=1}^{N} \log_2 \tilde{P}(w_i \mid h_i) \tag{4.20}$$

et la perplexité :

$$PP = 2^{lPN} \tag{4.21}$$

$$PP = \left(\prod_{i=1}^{N} \tilde{P}(w_i \mid h_i) \right)^{-\frac{1}{N}} \tag{4.22}$$

Cette mesure est difficile à interpréter. L'étude de cas extrême permet cependant d'en donner une interprétation. Si l'on considère une tâche parfaitement prédictible (probabilité estimée à 1), alors la perplexité vaut 1. À l'inverse un modèle sans pouvoir prédictif attribue une équiprobabilité aux événements. Dans ce cas la perplexité est égale à la taille du vocabulaire V. La perplexité est donc comprise entre 1 et V, une faible valeur reflétant un fort pouvoir de prédiction. Par extension, si la valeur de la perplexité est K, la difficulté de la prédiction est équivalente à choisir un mot par K mots équiprobables. Il est important de noter que la perplexité mesure les capacités

prédictives en fonction d'une tâche donnée et réciproquement elle permet de mesurer la difficulté d'une tâche pour un modèle donné.

La mesure de perplexité est largement employée pour l'évaluation des modèles de langage. Une alternative a été proposée dans [Bimbot 1997] qui permet de comparer des modèles probabilistes à des modèles non probabilistes. Il s'agit d'une adaptation du jeu de Shannon qui consiste à proposer à partir d'un contexte, une liste de mots candidats. Chacun des mots candidats étant affecté d'un poids, on évalue la qualité du modèle comme la moyenne géométrique des poids accordés à la solution correcte pour chaque contexte. Cette dernière mesure entre bien dans le cadre de l'évaluation des systèmes d'aide à la communication qui présentent une liste de mots comme notre système Sibylle.

Dans ce chapitre nous avons commencé par présenter l'approche symbolique puis, de manière plus détaillée l'approche stochastique. Nous avons présenté de nombreux modèles de langage probabilistes en les classant par la taille du contexte pris en compte pour la prédiction. Puis nous avons détaillé des techniques utiles pour la modélisation stochastique : des méthodes de lissage pour l'estimation des données manquantes et des mesures pour l'évaluation. Le chapitre suivant expose le modèle utilisé pour notre prédiction de mot en justifiant le choix de notre approche par rapport aux modèles exposés dans ce chapitre.

Chapitre 5

Prédiction de mot – SibyMot

Ce chapitre est consacré au modèle de langage que nous avons élaboré pour SibyMot. Dans la première partie, nous commençons par expliquer l'approche que nous avons souhaitée pour SibyMot : augmenter la taille du contexte pris en compte pour la prédiction en intégrant des connaissances syntaxiques. Nous décrivons ensuite les principes du modèle. Dans la deuxième partie du chapitre, nous décrivons de manière formelle la prédiction dans le système SibyMot. Celle-ci se décompose en deux sous-parties, une pré-analyse qui s'apparente à une analyse syntaxique de surface (*shallow parsing*), puis l'étape de prédiction proprement dite. La troisième partie du chapitre présente les lexiques et corpus utilisés pour l'acquisition des données du modèle. Enfin, la quatrième et dernière partie est consacrée à l'évaluation. Chaque partie du système est évaluée et au final nous présentons les capacités prédictives de SibyMot.

5.1 Un modèle pour le système SibyMot

5.1.1 Un objectif pour la modélisation probabiliste : étendre la taille du contexte

Une manière de classer les modèles de langage probabilistes que nous avons décrits dans le chapitre précédent est de les regrouper selon le type de contexte qu'ils utilisent pour établir leurs prédictions. Selon cette classification, on pourrait distinguer ces modèles en trois catégories : les modèles utilisant un contexte très court de quelques mots comme le modèle n-gramme, les modèles fondés sur une grammaire probabiliste prenant en compte toute la phrase et enfin les modèles s'appuyant sur tout le document en cours.

Pour le modèle de SibyMot, cette dernière classe de modèles, les modèles s'adaptant au document, a été mise à l'écart. En effet, comme nous l'avons souligné

lors de leur présentation, ils ne représentent pas une réelle alternative au modèle n-gramme. Soit ils modifient dynamiquement les paramètres (modèle mixture n-gramme), soit ils s'utilisent de manière combinée avec celui-ci (modèles cache et trigger).

En ce qui concerne les modèles hybrides fondés sur une grammaire probabiliste, leur intérêt est de permettre de modéliser toute la phrase. Cependant, à l'heure actuelle, ces modèles sont difficiles à appliquer pour la prédiction, d'autant plus dans le cas où la phrase est en cours de saisie.

Ainsi, la modélisation probabiliste du langage semble pour l'instant rencontrer des difficultés à étendre la taille du contexte. Les modèles purement probabilistes, en décalant l'historique (modèle n-gramme distant), en regroupant (modèles à séquence variable), ou en permutant (modèle perm-gramme) améliorent les performances du modèle n-gramme mais ne permettent pas réellement d'étendre la taille du contexte. Nous allons maintenant présenter le modèle SLM (Structured Language Model) de Jelinek qui présente une alternative intéressante pour la modélisation probabiliste.

5.1.2 Le modèle n-têtes de Jelinek

Frederick Jelinek a apporté de nombreuses contributions aux modèles de langage probabilistes. Dans ses articles les plus récents [Jelinek 2000] [Jelinek 2002], il propose un modèle original qui s'appuie non plus sur les derniers mots mais sur les dernières « têtes ». Pour présenter son modèle, il commence par s'interroger sur la prédiction du mot *after* dans la phrase suivante :

the contract ended with a loss of 7 cents after trading as low as 9 cents.

Avec un modèle tri-gramme, la prédiction de *after* est réalisée avec le contexte *7 cents*. Pourtant, les deux mots qui semblent les plus pertinents pour la prédiction sont plutôt *contract* et *ended*. En représentant l'analyse de la phrase (Figure 5.1), on peut constater que ces deux mots sont les représentants, appelés têtes, des deux derniers constituants.

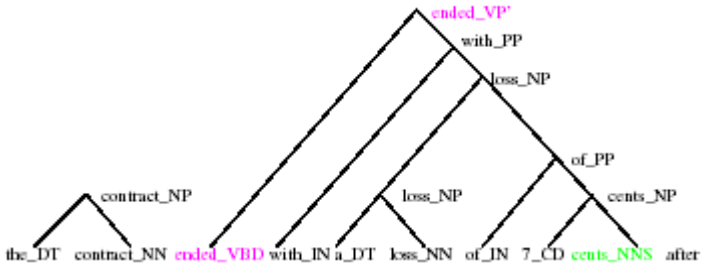


Figure 5.1 : Analyse partielle de la phrase : les nœuds terminaux sont les mots étiquetés avec leur POS, tandis que les non terminaux sont marqués par leur tête et leur étiquette non terminale

On peut remarquer que cette analyse est proche de l'analyse que fournirait une grammaire formelle. Cependant, à la différence des modèles fondés sur une grammaire, le modèle proposé ici ne cherche pas à construire une analyse complète. Pour établir ses prédicionts, le SLM (Structured Language Model) passe par une analyse partielle. Pour ce faire, il utilise trois modules :

- Un *prédicteur* qui prédit le mot suivant. La probabilité d'apparition d'un mot est : $P(w_i \mid H) = P(w_i \mid h_0, h_{-1})$ où h_0 et h_{-1} sont les deux dernières têtes.
- Un *étiqueteur* qui définit la partie du discours du nouveau mot. La probabilité d'assignation de l'étiquette ti est : $P(t_i \mid H) = P(t_i \mid w_i, h_0.tag, h_{-1}.tag)$ où $h_0.tag$, $h_{-1}.tag$ sont les étiquettes données aux deux dernières têtes.
- Un *constructeur* qui incrémente l'analyse partielle de la phrase. La probabilité d'une telle analyse (*parse*) p_i est : $P(p_i \mid H) = P(h_i \mid h_0, h_{-1})$.

Nous n'entrerons pas dans le détail de ce modèle, pour une information complète on pourra se référer à l'article très complet [Chelba, Jelinek 2000]. Notons cependant que le modèle utilise 40 POS, 52 étiquettes non-terminales et qu'un peu plus d'une centaine d'opérations de construction sont possibles. L'apprentissage des paramètres du modèle a été réalisé sur le corpus Penn TreeBank [Marcus 1995], corpus annoté manuellement et contenant les représentations syntaxiques des phrases. L'évaluation a donné une perplexité de 154 contre 167 à un modèle tri-gramme. En combinant les deux modèles avec un paramètre d'interpolation à 0,36 pour le modèle tri-gramme, la perplexité a été baissée à 148, soit une réduction de 11 % par rapport au modèle tri-gramme de référence.

Le modèle SLM correspond à notre approche. Il cherche à extraire des informations dans un contexte plus large en introduisant de la syntaxe, mais tout en conservant la robustesse qui fait l'intérêt des modèles probabilistes. Pour le modèle de SibyMot, nous avons également la notion de « tête ». Cependant, à l'inverse du modèle SLM qui détermine les têtes à partir d'une analyse partielle, notre modèle utilise la notion de *chunks* que nous allons maintenant présenter.

5.1.3 Introduction aux *chunks*

La notion de *chunk* qui est centrale à notre modèle est présentée par [Abney 1991] avec la phrase suivante :

I begin with an intuition : when I read a sentence, I read it a chunk at a time

Phrase qu'il segmente de la manière suivante :

[*I begin*] [*with an intuition*] : [*when I read*] [*a sentence*], [*I read it*] [*a chunk*] [*at a time*]

La définition que donne Abney d'un *chunk* repose sur les arguments psycholinguistiques de [Gee 1983]. Ces regroupements syntagmatiques sont supposés pertinents dans le processus de production de la parole. En particulier, ils rendent compte des pauses et des changements d'intonation. La segmentation de la phrase en *chunks* est utilisée par Abney pour faciliter la tâche de l'analyse syntaxique [Abney 1996]. En effet, une fois la phrase ainsi découpée, il devient plus simple d'établir les relations entre ces groupes de mots. Depuis, la notion de *chunks* a largement été réutilisée en linguistique informatique : pour la reconnaissance de la parole [Hirose 2001], pour l'analyse syntaxique [Ejerhed 1996] [Vergne 2000], pour l'étiquetage syntaxique de texte *tout-venant* [Abeillé 2000] [Clément 2001], pour la compréhension de la parole [Goulian 2002] [Antoine 2003].

Lors de la conférence CoNLL 2000, une tâche de segmentation a été proposée [Tjong Kim Sang 2000]. Les étiquettes à assigner aux *chunks* étaient au nombre de 11, sans mention de leur fonction syntaxique. Un exemple de segmentation à produire est donné par la phrase suivante :

NP[He] VP[reckons] NP[the current account deficit] VP[will narrow] PP[to] NP[only £ 1.8 billion] PP[in] NP[September] .

Les participants à cette évaluation disposaient du même corpus d'entraînement et leurs segmenteurs ont été évalués sur le même corpus de test (de plus de 200 000 mots soit 100 000 *chunks*). Le meilleur segmenteur [Kudoh 2000] a obtenu un score de 93.45 % en précision et 93.51 % en rappel. On peut cependant regretter que les articles de cette conférence relatifs aux segmenteurs manquent d'explications sur le fonctionnement des différents systèmes.

5.1.4 Principes de SibyMot

Nous allons maintenant présenter les principes de SibyMot. Nous commençons par exposer les deux idées fortes dans SibyMot : prédire avec des *chunks* et prédire avec

des lemmes. Puis nous donnons un aperçu des différentes étapes qui permettent la prédiction. Les parties suivantes exposeront de manière détaillée ces étapes.

5.1.4.1 Prédire avec des *chunks*

Pour introduire le modèle *n-chunks* de SibyMot, nous proposons, à la manière de Jelinek et d'Abney, d'illustrer les capacités prédictives des chunks sur des exemples. Les deux phrases suivantes sont respectivement les premières du corpus « vingt mille lieues sous les mers » et du corpus « Le Monde ». La définition que nous avons utilisée pour les chunks est ici maximale. En particulier les adjectifs post-posés aux noms sont regroupés dans le même chunk, et ce, afin d'augmenter la taille du contexte pris en compte. Pour les mêmes raisons, les prépositions sont rattachées aux groupes nominaux pour former des groupes prépositionnels. Les têtes des *chunks* sont marquées par le signe *.

[L'année* 1866] [fut marquée*] [par un événement* bizarre] [,] [un phénomène* inexpliqué] [,] [que*] [personne*] [n'a sans doute oublié*]

[La dernière semaine*] [à la bourse*] [de Paris*] [aura été*] [le reflet* fidèle] [de l'année* tout entière] [,] [vraiment morose*]

Le tableau suivant établit, pour les têtes de *chunks*, un contexte tri-gramme et tri-*chunks*. Pour les groupes prépositionnels, la préposition est marquée entre parenthèses dans le contexte n-*chunks*.

Phrase 1			Phrase 2		
3-gramme	3-chunks	à prédire	3-gramme	3-chunks	à prédire
∅ l'	∅ ∅	année	∅ la	∅ ∅	semaine
1866 fut	∅ année	marquée	à la	∅ semaine	bourse
par un	marquée (par)	événement	bourse de	bourse (de)	Paris
, un	événement ,	phénomène	Paris aura	bourse Paris	été
, que	, que	personne	été le	Paris été	reflet
sans doute	que personne	oublié	de l'	reflet (de)	année
			, vraiment	année ,	morose

Tableau 5.1 : Comparatif des contextes n-gramme et n-chunks

Par exemple, pour la première phrase, le contexte n-*chunks* « ∅ année » (au lieu de « 1866 fut ») permet de prédire *marquée*. De même, le contexte « événement , » (vs « , un ») prédit *phénomène*, etc.

Dans SibyMot, cette prédiction est appelée prédiction *inter-chunks*. Cette prédiction utilise les têtes des n-1 derniers *chunks* et sert à prédire les mots identifiés eux-mêmes comme tête *lexicale* de *chunk*. De plus, nous attribuons une place particulière à la préposition compte tenu de son rôle dans la fonction d'un syntagme : dans le cas du groupe prépositionnel, la préposition est également utilisée pour la prédiction *inter-chunks*.

Notons d'ores et déjà que nous établissons une différence entre les mots grammaticaux et les mots lexicaux, telle que définie dans Grévisse §141e :

> (Cit.5.1) *« On peut réunir la préposition, les deux espèces de conjonction et l'introducteur sous le nom de mots-outils. – L'appellation mots grammaticaux rassemble ces mots-outils, les déterminants et les pronoms ou, d'une façon plus générale, tous les mots dont le rôle est plutôt grammatical que lexical (il en va ainsi des verbes auxiliaires et de certains adverbes). »* [Goosse 2000]

Comme nous le verrons par la suite, SibyMot prédit l'étiquette grammaticale à venir. Les mots lexicaux, très nombreux dans leur classe, nécessitent donc une prédiction particulièrement efficace. De plus, les mots lexicaux sont généralement plus longs que les mots grammaticaux d'où un gain potentiel en économie de saisies plus important. Ceci explique que certains *chunks*, comme les conjonctions de subordination ou les pronoms relatifs, qui sont composés de mots grammaticaux, ne bénéficient pas de la prédiction *inter-chunks*.

Pour prédire au mieux les mots en exploitant la segmentation en *chunks*, nous avons relevé un deuxième type de prédiction. Lors de notre définition des *chunks*, nous avons mentionné que l'adjectif post-posé appartient au même *chunk* que le nom qu'il qualifie. Dans SibyMot, cet adjectif est prédit à partir de la tête du *chunk* auquel il est rattaché. De manière plus générale, les mots lexicaux d'un *chunk* placés après la tête sont prédits en fonction de cette tête. Ceci constitue notre deuxième type de prédiction, appelée prédiction *intra-chunks*. Cette prédiction est illustrée dans la phrase suivante par les mots suivis du signe + (* marque la tête des *chunks*) :

[L'année* 1866+] [fut marquée*] [par un événement* bizarre+] [,] [un phénomène* inexpliqué+] [,] [que*] [personne*] [n'a sans doute oublié*]

Sur cet exemple, « bizarre » et « inexpliqué » bénéficient de cette prédiction. Cette prédiction s'applique aussi par exemple à l'adverbe dans un groupe verbal « [il] [mange* rapidement+] [sa soupe] ».

Pour tous les autres mots de la phrase, aucun autre type de prédiction particulière n'est appliquée. Cependant, nous verrons par la suite comment ces mots bénéficient également d'une prédiction contextuelle.

5.1.4.2 Prédire avec des lemmes

Le deuxième choix important fait pour SibyMot est de privilégier le lemme comme unité lexicale plutôt que le graphème. Par exemple, la probabilité d'une forme fléchie est exprimée comme la probabilité combinée de son lemme et de sa flexion : P(la{lemme:le, flexion:féminin singulier}) = P_{lem}(lemme=le) $\times P_{flx}$(flexion=féminin singulier).

Les motivations de ce choix sont les mêmes que celles exprimées par [Cerf-Danon 1991] pour le modèle morphologique (cf. 4.3.4) qui intègre une composante tri-lemmes. Contrairement à l'anglais, le français dispose d'un système flexionnel très riche. Ceci est particulièrement vrai pour les verbes dont on peut dénombrer plus de 40 formes conjuguées (hors formes composées). Par exemple, un dictionnaire constitué de 5 000 adjectifs, 10 000 noms et 5 000 verbes correspond à moins de 50 000 formes fléchies en anglais et plus de 300 000 en français. [Cerf-Danon 1991] donne un rapport de 7 formes pour 1 lemme en français et un rapport de seulement 2 pour 1 en anglais. L'utilisation du lemme a donc pour but dans SibyMot de réduire l'espace des paramètres à estimer et ainsi d'augmenter la fiabilité des estimations.

5.1.4.3 Principe de la prédiction

Après avoir exposé les deux choix principaux pour SibyMot (prédire avec les *chunks* et avec les lemmes), nous décrivons dans cette partie les principes de fonctionnement du modèle de SibyMot afin d'en donner un aperçu général. Le modèle sera expliqué plus formellement par la suite.

Pour permettre une prédiction fondée sur les *chunks*, le processus de prédiction nécessite une étape préliminaire d'analyse du contexte gauche de la phrase. SibyMot est ainsi composé de deux modules : un *analyseur* chargé de construire une représentation de la phrase en *chunks* et un *prédicteur* prenant en entrée la structure précédente et délivrant les prédictions.

En ce qui concerne l'analyseur, la segmentation d'un énoncé en syntagmes correspond en TAL à une *analyse de surface* (*shallow parsing*). Elle se différencie de l'analyse syntaxique qui construit une représentation complète de la phrase avec les fonctions syntaxiques. Dans notre système, l'analyse de surface est chargée de déterminer pour chaque mot son lemme, son étiquette grammaticale, sa (ses) flexion(s) et son rôle dans la prédiction (préposition, tête, mot lexical post-posé). De plus, au niveau du *chunk*, elle délivre l'étiquette associée au *chunk* ainsi que la « flexion du *chunk* » qui correspond à celle de la tête. La figure suivante donne un exemple de représentation délivrée par l'analyseur :

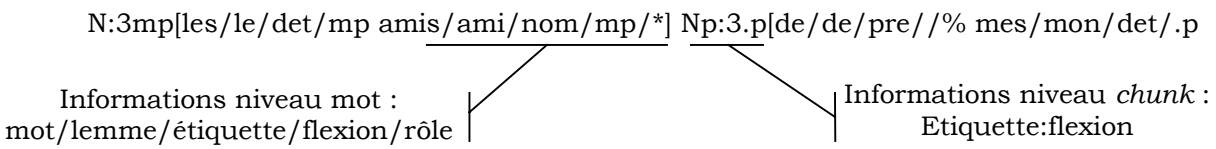


Figure 5.2 : Exemple de représentation délivrée par l'analyseur

On peut d'ores et déjà constater que l'étiquette grammaticale attribuée au mot ne contient pas d'information morpho-syntaxique et que les flexions peuvent être ambiguës. Le « rôle » attribué aux mots est utilisé pour les différents types de prédiction (inter-chunks, intra-chunks) tels que nous les avons définis dans la partie « prédire avec les chunks ».

À partir de cette structure, la prédiction est ensuite réalisée en cinq étapes :

1. Prédiction des segmentations. La première étape de la prédiction consiste à donner une estimation pour chaque étiquette grammaticale et pour chaque chunk. Par exemple, sur l'exemple précédent, déterminer la probabilité d'apparition d'un nom ou d'un adjectif (assez forte), d'un groupe verbal, etc.
2. Prédiction des relations. L'étape suivante est chargée de déterminer la relation du mot à prédire avec les chunks précédents, en particulier pour la prédiction inter-chunks
3. Prédiction des accords. Cette étape réalise une estimation pour chaque flexion potentielle du mot à prédire en fonction de l'étiquette grammaticale. Le fait d'avoir précédemment établi les relations permet la gestion des accords entre les chunks (comme un groupe verbal avec le groupe nominal qui le précède)
4. Prédiction des lemmes. En parallèle à l'étape précédente, la probabilité de chaque lemme est estimée selon les principes de prédiction n-chunks énoncés
5. Prédiction des mots. Enfin, à partir des estimations sur les lemmes et les flexions, ces probabilités sont combinées pour calculer la probabilité de chaque mot du lexique de SibyMot.

La prédiction dans SibyMot est donc réalisée en deux étapes, avec un *analyseur* et un *prédicteur*. On peut rapprocher cette décomposition avec le modèle SLM de Jelinek présenté plus avant. Le SLM construit une représentation de la phrase avec deux modules, un *étiqueteur* puis un *constructeur,* ce qui correspond aux deux sous-étapes de notre *analyseur* (étiquetage puis segmentation). Le modèle de SibyMot et du SLM utilisent ensuite cette représentation en entrée d'un *prédicteur* qui délivre les prédictions finales.

Avant d'exposer plus formellement l'analyse et la prédiction dans SibyMot, les paragraphes suivant présentent les jeux d'étiquettes (catégories grammaticales des

mots et classes des chunks) utilisés par ces deux modules. Nous présentons également la *grammaire des chunks* qui joue un rôle important autant en analyse lors de la segmentation qu'en prédiction.

5.1.5 Définition des étiquettes grammaticales

Chaque système d'étiquetage possède son propre jeu d'étiquettes. Le choix des étiquettes retenues est très variable, il dépend par exemple des choix linguistiques ou encore du détail de l'analyse morphologique. Le nombre des étiquettes est lui aussi extrêmement variable. Celui-ci peut être réduit et se limiter aux catégories grammaticales principales (adjectifs, noms, verbes, etc.) mais aussi très fin et donc très étendu comme dans le cadre de l'action Grace (plus de 300 étiquettes) [Rajman 1997].

Dans notre système, d'un côté le jeu doit être relativement réduit pour permettre un fort taux d'étiquetage correct par l'analyseur. De l'autre, il doit être assez fin pour permettre une prédiction efficace (une part de l'estimation de la probabilité d'apparition des lemmes dépend de la fréquence dans leur classe). Le jeu d'étiquettes retenu pour SibyMot contient 99 étiquettes (dont un pseudo tag $ de début et fin de phrase). Le Tableau 5.2 donne un échantillon du jeu d'étiquettes.

Étiquette	Définition
`adjcom`	adjectifs
`advcom`	adverbes
`nomcom`	noms communs
`nomprp`	noms propres
`advneg`	adverbe de négation « ne »
`advpas`	auxiliaire de négation « pas, point, etc. »
`numann, numjou`	expressions numériques année, jour (dans les dates)
`nomcommoi, nomcomjou`	liste des mois, des jours
`propercnjsuj`	pronoms personnels conjoints sujet (je, tu, etc.)
`propercnjcpl`	pronoms personnels conjoints complément (le, lui, etc.)
`veravr(cnj|inf|ppr|pps)`	auxiliaire *avoir* (conjugué, infinitif, p. présent et passé)
`veretr(cnj|inf|ppr|pps)`	auxiliaire *être*
`vermod(cnj|inf|ppr|pps)`	verbes modaux (ou assimilés)
`vercom(cnj|inf|ppr|pps)`	autres verbes

Tableau 5.2 : Échantillon du jeu d'étiquettes grammaticales de SibyMot

Comme nous l'avons déjà mentionné, les étiquettes ne contiennent pas d'indice morphologique à l'exception des catégories des verbes qui distinguent les formes conjuguées des infinitifs, participes présents et participes passés. Concernant les verbes, on peut remarquer que nous avons distingué quatre classes : les auxiliaires *être* et *avoir*, les modaux et les autres verbes. Pour permettre une prédiction plus efficace, il serait intéressant également de distinguer d'autres classes comme les verbes d'état, les verbes transitifs et intransitifs mais les lexiques à notre disposition ne contiennent pas ces informations (nous reviendrons par la suite sur le lexique de SibyMot).

Les étiquettes ne portent pas non plus d'information sur le rôle syntaxique (sujet, COD, etc.). Les seules exceptions concernent certaines catégories de pronoms où les formes des mots diffèrent selon la fonction. Un exemple est donné avec les pronoms personnels conjoints où l'étiquette `propercnjsuj` identifie les formes sujets (je, tu, etc.) et `propercnjcpl` les autres fonctions (le, lui, etc.).

Les étiquettes `advneg` (l'adverbe de négation « ne ») et `advpas` (les auxiliaires de négation « pas, point, guère, etc. ») sont une illustration de classes *ad hoc* créées pour la prédiction. Par exemple, après un verbe conjugué, la probabilité d'apparition de la négation « pas » sera très différente des autres adverbes selon la présence ou non de la négation « ne » devant le verbe. Ces classes supplémentaires ne sont cependant pas un problème pour l'étiquetage puisqu'elles n'introduisent pas d'ambiguïtés supplémentaires.

Nous avons également crée des classes particulières pour les expressions numériques. Par exemple dans « les deux personnes » et « les 2 personnes », *deux* est étiqueté `adjcrd` (adjectif cardinal) et *2* `adjcrdnum` (adjectif cardinal sous forme numérique). Ceci s'explique par les conventions d'utilisation particulières des expressions numériques. De même, les étiquettes `numann` et `numjou` identifient respectivement l'année et le quantième du mois dans les expressions de date. Par ailleurs, les noms de mois sont regroupés dans la classe `nomcommoi` et les jours dans la classe `nomcomjou`. Exemple : « le mardi/nomcomjou 2/numjou décembre/nomcommoi 1980/numann ». Ceci facilite la reconnaissance des expressions de date lors de l'étape de segmentation.

Nous terminerons la présentation du jeu d'étiquettes en signalant que la plupart des classes sont des classes « fermées » : la liste des mots contenus dans ces classes est close. Seulement une dizaine de classes comme les noms communs `nomcom`, les adjectifs `adjcom` constitue la liste des classes « ouvertes ». Nous verrons lors du problème de l'étiquetage des mots inconnus le rôle des classes ouvertes.

Notons que par la suite, nous utiliserons généralement des noms d'étiquette simplifiés (par exemple `det` pour déterminant, `adj` pour adjectif, etc.).

5.1.6 Grammaire des chunks

Dans SibyMot, les *chunks* sont définis par une « grammaire des *chunks* ». Cette grammaire joue un rôle central lors de l'étape de segmentation des énoncés et lors de la prédiction pour déterminer l'étiquette du *chunk* et l'étiquette grammaticale du mot à prédire.

Tout comme les mots, les *chunks* sont identifiés par une étiquette. Par rapport aux jeux d'étiquette habituellement proposés pour les *chunks* (par exemple 11 lors de la conférence CoNLL, cf. supra), notre jeu d'étiquettes est plus étendu (61 étiquettes). En effet, dans SibyMot, nous utilisons les *chunks* en prédiction et l'étiquette attribuée au *chunk* doit donc caractériser de manière assez fine la nature des éléments qui le composent. D'abord de manière assez générale, l'étiquette attribuée au *chunk* est proche de l'étiquette de sa tête. Par exemple, pour le groupe du verbe conjugué, nous avons quatre étiquettes « `Vac, Vec, Vmc, Vcc` » qui reprennent la distinction « verbe *être*, verbe *avoir*, verbes modaux et autres ». Dans le même ordre d'idée, chaque *chunk* contenant un type de pronom différent va être marqué par une étiquette différente (`PR` pour le pronom relatif, `PP` pour le pronom personnel, etc.). Ensuite les étiquettes portent des traits renseignant sur la composition du *chunk*. Par exemple, pour le groupe adjectival étiqueté `A`, nous attribuons l'étiquette `As` lorsqu'il s'agit d'un superlatif : « c'est As[le plus grand] que je connaisse ». Le Tableau 5.3 liste quelques-unes des étiquettes des *chunks*.

Étiquette	Définition
`A, As, Ac`	groupe adjectival, superlatif, comparatif
`F`	groupe de l'adverbe
`N, Np`	groupe du nom, groupe prépositionnel
`Vac, Vec, Vmc, Vcc`	groupe du verbe conjugué (verbe avoir, être, modaux, autres)
`Vi, Vip`	groupe du verbe à l'infinitif, avec préposition
`Vr, Vrp`	groupe du verbe au participe présent, gérondif
`Vs`	groupe du verbe au participe passé
`PPs, PPx`	pronom personnel sujet, autres fonctions
`PRs, PRx`	pronom relatif sujet (qui), pronoms relatifs autres fonctions
`ED, EDp`	expression de date, avec préposition
`EH, EHp`	expression horaire, avec préposition
`X`	*chunk* inconnu

Tableau 5.3 : Échantillon du jeu d'étiquettes des *chunks* de Sibylle

Ensuite, pour chacune de ces classes de chunks, la grammaire définit l'ensemble des séquences de tags qui peuvent composer le chunk. Par exemple, la classe du

groupe nominal `N` contient les séquences « `adj_nom`, `adv_adj_nom`, `det_adj_nom`, `det_adv_adj_nom`, etc. ». Cette liste de séquences est générée automatiquement à partir d'une méta-grammaire définissant les séquences sous forme d'expressions régulières. Les séquences de l'exemple précédent sont ainsi définies par une seule règle :

```
?DET ?ADV ADJ NOM:*
```

Dans cette règle le « ? » marque l'optionalité et les étiquettes en majuscule sont des méta-tags décrivant une liste de tags (par exemple `NOM` décrit `nomcom` et `nomprp`). Le caractère « : » permet également d'introduire des informations supplémentaires comme la tête du *chunk* marquée par l'étoile. Au total la méta-grammaire contient plus de deux cents règles qui engendrent autour de 200 000 séquences (pour tous les *chunks*).

Cette grammaire n'est cependant pas exhaustive. En effet, il est toujours possible de construire une séquence qui n'appartiendra pas à une liste finie par un procédé de récursivité : « une très petite maison, une très très petite maison, une très très… ». Ces cas sont cependant très rares et le taux de *chunks* non définis très faible (il est par ailleurs possible d'ajouter manuellement des séquences de *chunks* à la grammaire).

La grammaire contient également trois autres informations : la fréquence d'utilisation de chaque séquence (apprise sur corpus), la gestion des accords à l'intérieur du *chunk* et le rôle du mot pour la prédiction. La séquence suivante extraite du *chunk* Np contient les trois rôles définis (marqués par les signes `%` `*` et `+`) :

```
Np[pre:% det adj nom:* adv adj:+]
```

L'étoile marque la tête du *chunk*, le signe plus indique que le mot doit être prédit avec la prédiction *intra-chunk*. Le signe pourcent marque la préposition pour la prédiction *inter-chunks*. Sur cet exemple, on peut constater que l'adjectif anté-posé au nom et l'adverbe ne bénéficient pas de la prédiction *intra-chunk*. En effet, le premier adjectif qualifie le nom à venir et l'adverbe apparaît avant l'adjectif qu'il modifie.

Pour la gestion des accords, de manière implicite tous les mots d'un *chunk* doivent s'accorder entre eux en genre et en nombre (sauf bien sûr les mots invariables comme les adverbes ou les prépositions). Dans certains cas comme le pronom personnel conjoint dans le groupe verbal « je Vc[les donne] », le pronom est marqué d'un trait spécial pour éviter l'accord en nombre avec le verbe. Pour permettre la gestion des accords entre les *chunks* (gestion que nous verrons par la suite) les *chunks* portent les marques flexionnelles du genre, du nombre et de la personne :

```
N:3mp[les/le/detartdef/.p gros/gros/adjcom/m.
```

Ceci conclut la partie présentation de SibyMot. Les parties suivantes vont décrire plus formellement la prédiction dans SibyMot. La description commence par la première étape de la prédiction : l'analyse syntaxique de surface.

5.2 L'analyseur syntaxique

L'analyseur syntaxique de SibyMot réalise une analyse syntaxique de surface (*shallow parsing*) de manière « classique » [Pierrel 2000] en deux passes. En première passe, l'analyseur attribue à chaque mot une étiquette grammaticale et en deuxième passe, le *chunker* segmente la phrase. Ces étapes successives sont présentées sur un exemple de phrase par la Figure 5.3 et détaillées dans les paragraphes suivants. C'est cette structure délivrée par l'analyseur qui sera par la suite utilisée par la prédiction.

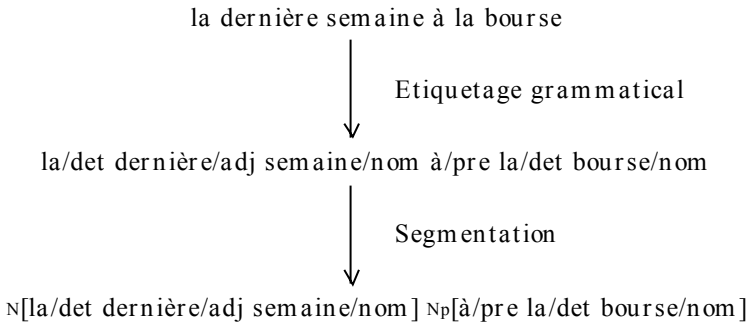


Figure 5.3 : Les étapes successives de l'analyseur

L'étiquetage grammatical est de manière habituelle vu comme un processus en trois étapes [Pierrel 2000]. la première concerne la segmentation de la chaîne de caractères en mots (*token* en anglais). Celle-ci est suivie d'un étiquetage *a priori,* (*lexical lookup*), chargé de relever l'ensemble des étiquettes morpho-syntaxiques possibles des mots. Enfin la désambiguïsation sélectionne l'étiquette pertinente en fonction du contexte.

5.2.1 Segmentation en mots

Avec l'outil informatique, un texte est identifié à un flux de caractères que le processus de *tokenisation* est chargé de segmenter en unités élémentaires, les *tokens*, construits à partir de classes de caractères (lettres, séparateurs, ponctuations, etc.). Le problème, pour les applications en TAL, est que ces unités ne correspondent pas toujours à la définition du mot comme unité linguistique (par exemple le mot *parce que*). De plus, la séquence peut être ambiguë comme le montre l'exemple de la séquence *avant que,* qui peut soit correspondre à l'adverbe *avant* suivi de la conjonction *que* (*tu aurais pu me dire avant que le magasin était fermé*), soit à la locution *avant que* (*avant que je parte…*).

À ce stade de l'analyse il est impossible de décider de la segmentation correcte, le choix nécessitant des connaissances syntaxiques, voire sémantiques ou pragmatiques. Une solution consiste à différer la décision en transmettant l'ambiguïté aux niveaux supérieurs. Par exemple, dans le cadre de l'action Grace [Rajman 1996] un formalisme de gestion des formes composées a été défini pour représenter les ambiguïtés de segmentation au niveau mot.

Dans SibyMot, le lexique contient de nombreuses formes composées comme les locutions adverbiales *en effet*, *sans doute*, etc. Cependant nous nous plaçons dans le cadre d'une saisie contrôlée : un mot composé est reconnu comme tel lorsqu'il a été sélectionné dans la liste des mots proposée à l'utilisateur.

Cette solution est partiellement insuffisante, en particulier si l'utilisateur tape l'expression en plusieurs mots. C'est pourquoi, à l'avenir, nous envisageons d'intégrer un modèle à séquences variables comme un de ceux présenté au chapitre précédent (cf. 4.3.5).

5.2.2 Étiquetage *a priori*

La phase d'étiquetage *a priori* consiste par un accès au lexique afin de relever toutes les étiquettes grammaticales possibles d'un mot. Dans le cas de notre analyseur, l'information du lemme est également ajoutée. La séquence de mot $W = w_1, \ldots, w_N$ est ainsi transformée en une séquence de liste de couples :

$$w_1, \ldots, w_N \rightarrow TL_{w1}, \ldots, TL_{wN}$$

où TL_{wi} désigne la liste des couples (tag, lemme) possibles d'un mot. $TL_{wi} = (t_{i1}, l_{i1}), \ldots, (t_{iM}, l_{iM})$ avec i_M le nombre d'ambiguïtés du mot w_i.

Une difficulté de l'étiquetage provient des mots inconnus. Il peut s'agir soit d'un mot incorrectement orthographié, soit d'un mot non référencé dans le lexique. Dans ce cas, les analyseurs ont recours à un oracle (*guesser*) qui doit déterminer la (ou les) catégorie(s) possibles du mot. Une catégorie un peu particulière est celle des noms propres, qui représente généralement à elle seule plus de 50 % des mots inconnus. En effet, à partir de la casse, il est possible de déterminer de manière presque systématique un nom propre, lorsque la première lettre est en majuscule (hors premier mot de la phrase). Pour les autres mots, les *guesser* se fondent sur des indices morphologiques, généralement la fin des mots, pour attribuer les catégories. Un analyseur comme celui de [Chanod 1995] est constitué de règles reflétant les suffixes productifs (*er* pour les verbes, *ible* pour les adjectifs, …). Des méthodes probabilistes utilisées sur les n dernières lettres des mots sont également applicables [Schmid 1995].

Dans SibyMot, nous avons cherché à minimiser le problème des mots inconnus par la taille du lexique (plus de 60 000 lemmes dont 20 000 noms propres). Lorsque le mot est inconnu, le *guesser* de SibyMot commence par attribuer toutes les étiquettes des classes ouvertes (un peu plus de dix étiquettes), avec un lemme spécial <mot inconnu>. Une liste de règles morphologiques (et de casse) est ensuite appliquée pour restreindre le nombre d'ambiguïtés. Exemple : l'étiquette `vercomppr` – verbe au participe présent – est retirée si le mot ne se termine pas par *ant*. Le choix final de l'étiquette la plus appropriée est levé, en contexte, lors de la dernière étape de désambiguïsation.

5.2.3 Désambiguïsation des étiquettes grammaticales

L'étape suivante de l'étiquetage concerne la levée des ambiguïtés des étiquettes syntaxiques. Dans le cadre de la modélisation probabiliste, l'étiquetage revient à aligner deux séquences, une de mots $W = w_1, \ldots, w_N$ et l'autre de *tags* $T = t_1, \ldots, T_N$, et à rechercher la séquence d'étiquettes T qui maximise la probabilité conditionnelle d'association à la séquence de mots W.

Dans SibyMot, nous avons utilisé le modèle n-POS, avec n = 3. SibyMot travaillant sur les lemmes, le modèle n-POS a été adapté pour estimer non plus la probabilité d'apparition d'un mot mais celle d'un lemme. L'estimation ainsi redéfinie devient :

$$P(l_i \mid l_{i-2}, l_{i-1}) = \sum_{t_i \in Tl_i} P(l_i \mid t_i) \times P(t_i \mid t_{i-2}, t_{i-1}) \qquad (5.1)$$

où t_i est l'étiquette associée au lemme l_i, et T_{li} l'ensemble de toutes les étiquettes grammaticales du lemme.

L'estimation du paramètre $P(l_i \mid t_i)$ est réalisée à l'aide de l'estimateur bayésien qui permet de traiter de manière implicite le cas des mots non observés sur corpus (une fréquence minimale de 1 est attribuée à chaque mot) :

$$P(l_i \mid t_i) = \frac{C(l_i, t_i) + 1}{C(t_i) + N(t_i)} \qquad (5.2)$$

où $C(l_i,t_i)$ est la fréquence du lemme l_i dans sa classe t_i et $N(t_i)$ le nombre de lemmes de la classe t_i. Chaque classe ouverte contient un lemme *mot inconnu* de fréquence nulle, ce qui affecte aux mots inconnus une probabilité minimale. Pour l'estimation de $P(t_i \mid t_{i-2}, t_{i-1})$, l'estimation classique par maximum de vraisemblance est utilisée :

$$P(t_i \mid t_{i-2},\ t_{i-1}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})} \qquad (5.3)$$

Ensuite, pour permettre une estimation fiable dans le cas des séquences rares, nous avons utilisé la méthode d'interpolation linéaire. L'estimation porte ainsi sur 3 modèles, tri-POS, bi-POS et uni-POS :

$$P(l_i \mid h_i) = \lambda_1\ P_{1\text{-}POS}(l_i \mid h_i) + \lambda_2\ P_{2\text{-}POS}(l_i \mid h_i) + \lambda_3\ P_{3\text{-}POS}(l_i \mid h_i) \qquad (5.4)$$

où $\lambda_1, \lambda_2, \lambda_3$ sont les paramètres d'interpolation, $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Ces paramètres sont appris sur corpus avec l'algorithme EM (comme tous les autres paramètres d'interpolation de SibyMot).

Enfin, l'algorithme de Viterbi a été implanté pour déterminer la meilleure séquence de tags. À la fin du processus d'étiquetage, la séquence de mots est donc enrichie de la séquence de couples (tag, lemme) :

$$w_1, \ldots, w_N$$
$$\rightarrow$$
$$(t_1,\ l_1), \ldots, (t_N,\ l_N)$$

5.2.4 Segmentation de la phrase

5.2.4.1 Principe

La dernière étape de notre analyseur est chargée de découper la phrase en *chunks*. Pour réaliser cette segmentation, nous proposons une solution originale qui s'inspire du modèle n-POS. Dans la modélisation adoptée, la phrase est vue non plus comme une séquence de mots, mais comme une séquence de *chunks* $C = c_1, \ldots, c_N$. Chaque *chunk* c_j contient un ou plusieurs mots représentés par leur étiquette grammaticale. On note s_j la séquence des étiquettes syntaxiques du *chunk* c_j (la lettre j est utilisée pour numéroter les *chunks*, la lettre i étant conservée pour la numérotation des mots).

Par analogie au modèle n-POS, la liste des parties du discours est identifiée à la liste des différentes classes de *chunk* (*Nc*, *Vi*, etc.) et l'ensemble des éléments d'une classe est constitué par les séquences de *tags* appartenant à cette classe (par exemple, *det_nom*, *det_nom_adj*, … pour le groupe nominal). La liste des séquences est directement donnée par la grammaire des *chunks*. Ainsi, l'estimation du modèle n-POS :

$$P(w_i \mid h_i) = \sum_{t_i \in Tw_i} P(w_i \mid t_i) \times P(t_i \mid t_{i-n+1}, \ldots, t_{i-1})$$

devient :

$$P(s_j \mid h_j) = \sum_{s_j \in Cs_j} P(s_j \mid c_j) \times P(c_j \mid c_{j-n+1},\ldots, c_{j-1}) \qquad (5.5)$$

où $P(s_j \mid c_j)$ est la probabilité de la séquence de tags s_j dans le *chunk* c_j et $P(c_j \mid c_{j-n+1},\ldots, c_{j-1})$ la probabilité du *chunk* c_j connaissant les n-1 derniers *chunk*s. Dans SibyMot nous avons fixé la valeur de n à 3. De même que dans l'étiqueteur, nous avons utilisé l'interpolation linéaire comme méthode de lissage, et l'algorithme de Viterbi pour le calcul de la meilleure séquence.

Pour illustrer le lien entre les deux modèles, prenons l'exemple de la séquence de mots « le candidat à l'élection présidentielle ». Dans le modèle n-POS, un des termes de l'estimation porte sur la probabilité du mot dans sa classe. Ainsi la séquence « le *candidat* à l'élection présidentielle… » sera estimée comme plus probable que « le *prétendant* à l'élection présidentielle… » car la probabilité du mot « candidat » est supérieure à celle de « prétendant » dans la classe des noms. Dans le modèle proposé, la probabilité mise en parallèle porte sur la probabilité de la séquence des étiquettes. La séquence « [le candidat] … » sera estimée comme plus probable que « [le sympathique candidat] … » car la probabilité de « déterminant nom » est supérieure à « déterminant adjectif nom » dans un groupe nominal.

5.2.4.2 Estimation des séquences de *tags*

La formulation proposée en (5.5) serait suffisante si la phrase complète était donnée à l'analyseur. Pour pouvoir analyser une phrase en cours de saisie, et en particulier les derniers mots tapés, le modèle doit permettre l'estimation de la probabilité d'un début de *chunk*. Par exemple, après « le sympathique », l'analyse doit retourner « [le/det sympathique/adj » identifié comme début de groupe nominal.

La solution consiste ici à construire une représentation arborée des séquences de *tags* de chaque *chunk,* en factorisant leur fréquence. Un caractère « fin de séquence » est ajouté pour différencier les séquences complètes des séquences préfixes. La figure suivante illustre cette construction sur quelques séquences du groupe nominal.

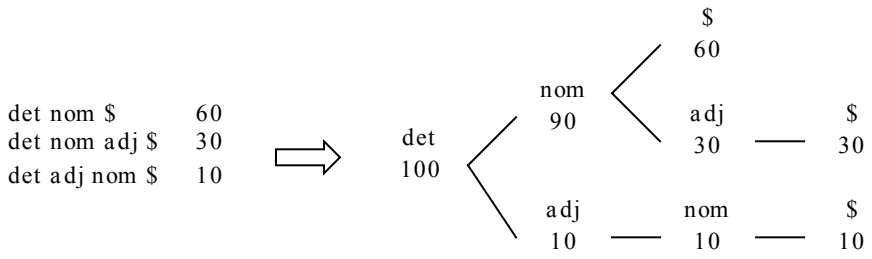


Figure 5.4 : Exemple de grammaire arborée

Avec cette construction, chaque séquence s_j est estimée à partir de la fréquence associée à son nœud Nd_{sj} dans l'arbre et de la fréquence du chunk c_j (donnée par la fréquence du nœud racine de l'arbre) :

$$P(s_j \mid c_j) = \frac{C(Nd_{sj})}{C(c_j)} \tag{5.6}$$

5.2.4.3 Estimation des séquences inconnues

Bien que la grammaire définit un très grand nombre de séquences (plus de 200 000), le problème des séquences inconnues peut se poser, y compris si l'utilisateur saisit une séquence incorrecte (deux prépositions par exemple). Pour remédier à ce problème, une solution relativement simple a été adoptée, inspirée de [Deligne 1996]. Une catégorie de *chunk* supplémentaire a été ajoutée « *X* ». Cette classe est formée par les séquences de longueur 1 qui correspondent à chacune des étiquettes du lexique. Les fréquences sont établies à 1, les probabilités qu'un *chunk* soit étiqueté *X* sont donc très faibles, seulement si la séquence n'est pas reconnue ailleurs. La phrase suivante donne un exemple d'étiquetage *X* :

[je] [voudrais] [un billet] $_X$[pour] [pour Paris]

5.2.5 Conclusion sur l'analyseur

L'analyseur que nous avons présenté est un analyseur syntaxique de surface en deux passes : une étape d'étiquetage puis de segmentation de la phrase. Il met en œuvre des techniques issues de l'approche stochastique du langage (modèle n-POS), tout en intégrant des contraintes symboliques par l'intermédiaire d'une grammaire des *chunk*s. Il délivre en sortie une séquence de quadruplets (mot, tag, lemme, flexion) pour chaque mot de la phrase en cours de saisie. C'est cette représentation qui est utilisée pour la prédiction. Les performances de l'analyseur seront présentées par la suite dans la partie sur l'évaluation.

5.3 La prédiction

La prédiction dans le système SibyMot est un processus en cinq étapes qui réalise une estimation pour chacun des mots du lexique (Figure 5.5). Les quatre premières étapes vont successivement prédire les segmentations possibles à partir de la segmentation courante, puis prédire les relations, les accords et enfin prédire les lemmes. L'étape finale combine les probabilités des accords et des lemmes calculées précédemment pour délivrer la probabilité des mots.

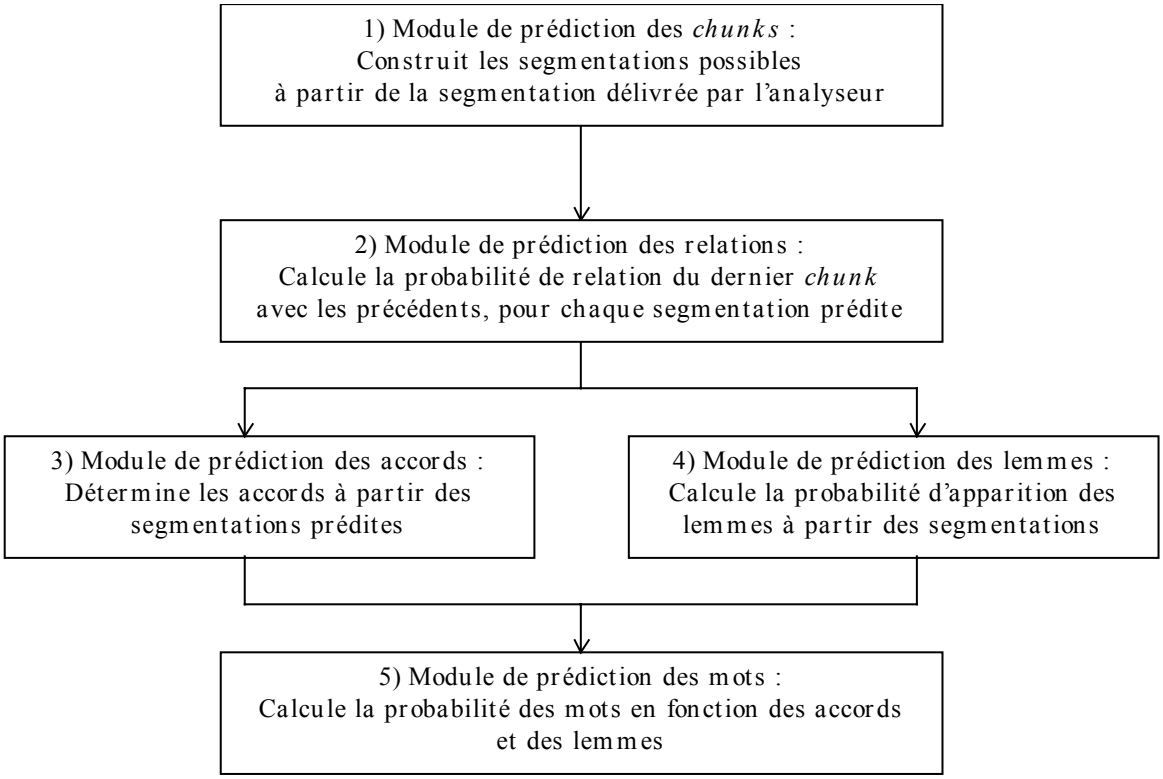


Figure 5.5 : Décomposition de la prédiction dans SibyMot

5.3.1 Prédiction des segmentations de la phrase

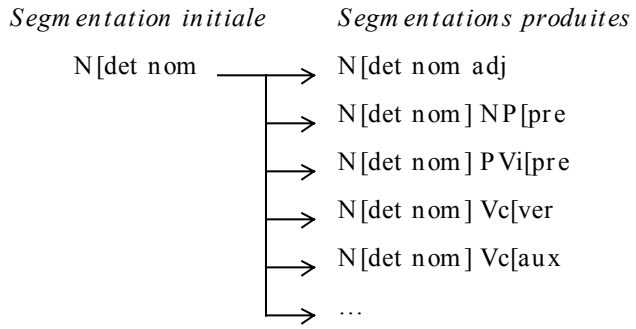
Le rôle de la première étape de la prédiction est de construire l'ensemble des segmentations possibles pour le mot suivant et de fournir, pour chacune de ces segmentations, une estimation de sa probabilité d'apparition. Le processus de construction est réalisé à l'aide de la grammaire des *chunks*, tandis que l'estimation utilise la formulation vue lors de l'étape de segmentation de l'analyseur.

5.3.1.1 Construction des segmentations

Pour illustrer le processus de construction, prenons l'exemple de la saisie « l'année ». Après l'analyse « N[l'/det année/nom », on peut avoir :

- un adjectif rattaché au groupe nominal :
  N[l' année *dernière*/adj
- une préposition commençant un groupe prépositionnel :
  N[l' année] Np[*du*/pre (… dragon)
- une préposition commençant un groupe verbal à l'infinitif :
  N[l' année] Vip[*à*/pre (… venir)
- un verbe conjugué :
  N[l' année] Vc[*commence*/vrb
- un groupe verbal commencé par un auxiliaire :
  N[l' année] Vc[*a*/aux (… commencé)
- …

c'est à dire, si l'on ne considère que les étiquettes attribuées aux *chunks* et aux mots :



La construction détermine donc l'ensemble des triplets (*nouveau chunk ?*, *étiquette du chunk*, *étiquette du mot*). Le premier paramètre *nouveau chunk ?* indique si le mot à prédire poursuit le chunk en cours ou s'il commence un nouveau chunk. Le deuxième paramètre spécifie l'étiquette attribuée au nouveau chunk et le troisième la partie du discours envisagée pour le mot. Cette liste est établie à partir de la grammaire selon l'algorithme suivant :

```
// Construction des segmentations qui poursuivent le chunk en cours
si la grammaire accepte des successeurs à cj[tj1, …, tjN alors
       pour chaque étiquette tu successeur de cj[tj1, …, tjN faire
              ajouterSegmentation (nouveauChunk=faux, cj, tu)

// Construction des segmentations des nouveaux chunks
si la grammaire accepte que la séquence cj[tj1, …, tjN se termine alors
       pour chaque chunk cv de la grammaire faire
              pour chaque étiquette tu commençant cv faire
                     ajouterSegmentation (nouveauChunk=vrai, cv, tu)
```

L'algorithme ajoute d'abord les segmentations qui poursuivent le *chunk* en cours (« N[det nom adj » sur l'exemple précédent) puis envisage toutes les segmentations commençant un nouveau *chunk*. Compte tenu de la grammaire, entre 100 et 200 segmentations, en moyenne, sont ainsi produites.

5.3.1.2 Estimation de la probabilité des segmentations

Les segmentations produites sont ensuite estimées en se fondant sur le même principe que lors de l'analyse. Deux cas sont à différencier selon que le *chunk* continue ou non. Si S_{uv} est la segmentation à estimer :

$$P(S_{uv} \mid h) = \begin{cases} [1 - P(\$ \mid c_v \,\&\, t_{v,1} \ldots t_{v,Nv})] \times P(t_u \mid c_v \,\&\, t_{v,1} \ldots t_{v,Nv}), \text{ si le } chunk \text{ continue} \\ P(\$ \mid c_{v-1} \,\&\, t_{v-1,1} \ldots t_{v-1,Nv-1}) \times P(t_u \mid c_v \,\&\, \varnothing) \times P(c_v \mid c_{v-2}, c_{v-1}), \text{ sinon} \end{cases} \quad (5.7)$$

où :

- S_{uv} est la segmentation à estimer et identifiée par le *tag* t_u et le *chunk* c_v du mot à prédire.
- c_v désigne le dernier *chunk* et c_{v-1} l'avant dernier *chunk*. Dans le cas où le *chunk* continue le *chunk* c_v désigne donc le *chunk* en cours. Dans le cas d'un nouveau *chunk* c_v désigne le nouveau *chunk*.
- $t_{v,1} \ldots t_{v,N}$ est la séquence de *tags* du *chunk* c_v. La notation $t_{x,y}$ identifie un *tag* par son numéro de *chunk* x et son index y dans le *chunk*. Nj désigne le nombre de *tags* dans le *chunk* j
- $P(\$ \mid c_{v-1} \,\&\, t_{v-1,1} \ldots t_{v-1,Nv-1})$ est la probabilité que le *chunk* c_v se termine après la séquence de *tags* $t_{v,1} \ldots t_{v,N}$. \$ marque un pseudo *tag* « fin de *chunk* »
- $P(t_u \mid c_v \,\&\, t_{v,1} \ldots t_{v,Nv})$ est la probabilité d'apparition du *tag* t_u dans le *chunk* c_v après la séquence $t_{v,1} \ldots t_{v,N}$
- $P(t_u \mid c_v \,\&\, \varnothing)$ est la probabilité d'apparition du *tag* t_u dans le *chunk* c_v en position initiale. $\varnothing$ marque un pseudo *tag* « début de *chunk* »
- $P(c_v \mid c_{v-2}, c_{v-1})$ est la probabilité d'apparition du *chunk* c_v après la séquence des deux derniers *chunks* c_{v-} et c_v

Les estimations (5.7) doivent être interprétées de la manière suivante. Lorsqu'un *chunk* continue, la probabilité de la segmentation $P(S_{uv} \mid h)$ est le produit de la probabilité $[1 - P(\$ \mid c_v \,\&\, t_{v,1} \ldots t_{v,Nv})]$ que le *chunk* ne se termine pas, par la probabilité $P(t_u \mid c_v \,\&\, t_{v,1} \ldots t_{v,Nv})$ que le *tag* t_u apparaisse après la suite de *tags* $t_{v,1} \ldots t_{v,N}$ du *chunk* c_v. Dans l'autre cas, lorsqu'un *chunk* est nouveau, la probabilité de la segmentation est le produit de la probabilité $P(\$ \mid c_{v-1} \,\&\, t_{v-1,1} \ldots t_{v-1,Nv-1})$ que le *chunk* en cours se termine, par la probabilité $P(t_u \mid c_v \,\&\, \varnothing)$ que le *tag* t_u commence le nouveau *chunk* c_v, par la probabilité $P(c_v \mid c_{v-2}, c_{v-1})$ que le *chunk* c_v succède à la séquence c_{v-}, c_v.

Cette formulation vérifie que la somme des probabilités de toutes les segmentations produites est égale à 1 : $\sum S_{uv} = 1$.

5.3.1.3 Prédiction multi-analyses

La prédiction des segmentations telle que nous venons de la définir pose cependant un problème lorsque le (ou les) dernier(s) mot(s) de la phrase tapée peut correspondre à plusieurs segmentations. Le cas le plus fréquent est celui de la préposition. Lorsque l'utilisateur vient de taper une préposition « il faut pour », plusieurs segmentations sont envisageables (groupe nominal prépositionnel, groupe verbal à l'infinitif, etc.). Dans ce cas, l'analyse privilégie généralement un début de groupe nominal prépositionnel « il faut N[pour/pre », ce qui exclut lors de la prédiction toute étiquette grammaticale ne faisant pas partie du groupe nominal.

Pour résoudre ce problème, la prédiction utilise en entrée les *n* meilleures segmentations déterminées par l'analyseur. Chacune de ces segmentations est pondérée par sa probabilité issue de l'analyseur. Les probabilités sont ensuite normalisées pour obtenir une somme égale à 1. Au final, chaque segmentation prédite doit tenir compte de la probabilité de la segmentation issue de l'analyseur.

5.3.2 Prédiction des relations

L'étape suivante de la prédiction concerne la mise en relation du dernier chunk avec les chunks qui précèdent. Cette relation servira par la suite dans la gestion des accords et la prédiction des lemmes.

Dans la théorie des grammaires formelles introduites par Chomsky, la structure syntaxique de la phrase peut être représentée sous forme hiérarchique. Cette représentation décrit la façon dont sont regroupés les mots selon la notion de constituants. La figure suivante donne un exemple d'une telle représentation.

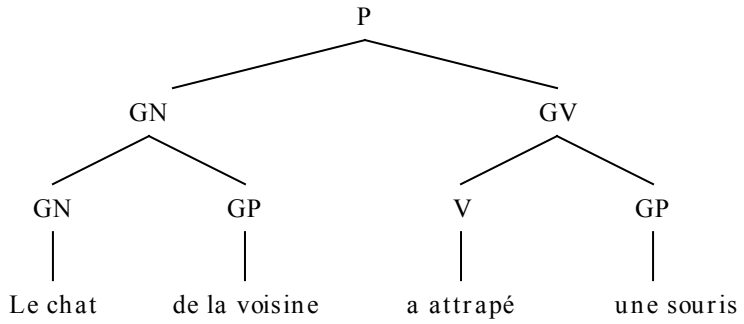


Figure 5.6 : Exemple de représentation hiérarchique en constituants

Dans le système SibyMot, le formalisme adopté s'inspire des grammaires de dépendance [Tesniere 1959], [Sleator 1991]. La notion de dépendance lie deux mots ou groupes de mots dans une relation de dominant-dominé (Figure 5.7).

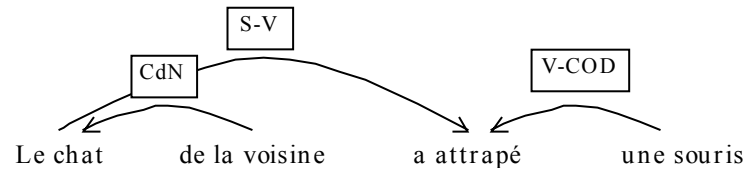


Figure 5.7 : Exemple de représentation en dépendances

Dans SibyMot, la notion de relation est utilisée entre les *chunks*, d'une part, pour gérer certains accords comme l'accord sujet-verbe et, d'autre part, pour la prédiction des lemmes à partir des têtes lexicales des *chunks* mis en relation. En prédiction, seules les relations « arrières » sont utilisables, nous inversons donc les relations « avant » comme la relation sujet-verbe (lorsqu'elle est dans cet ordre). De plus, nous ne cherchons pas à établir le type de relation entre les deux *chunks*. Enfin, par convention, une relation entre deux *chunks* est notée par la distance r qui les sépare. Lorsqu'un *chunk* est en relation avec aucun autre (en début de phrase, ou une incise par exemple), la distance est considérée nulle (r = 0). Sur l'exemple précédent, les relations deviennent :

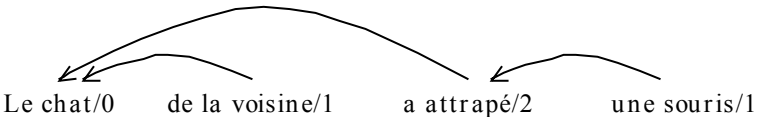


Figure 5.8 : Exemple de relations dans SibyMot

Ainsi, on peut observer que la relation entre le verbe et le sujet est inversée et que le premier mot de la phrase est noté sans relation.

Si S_{uv} est une segmentation prédite, on appelle « segmentation-relation » S_{uvr} la segmentation où le dernier *chunk* est en relation rel_r avec le *chunk* à distance r. La probabilité de la segmentation-relation S_{uvr} est estimée comme suit :

$$P(S_{uvr} \mid h) = P_{rel}(rel_r \mid c_{v-2}, c_{v-1}, c_v) \times P(S_{uv} \mid h) \qquad (5.8)$$

où :

- $P(S_{uvr} \mid h)$ désigne la probabilité de la segmentation-relation S_{uvr} selon le contexte h.
- $P_{rel}(rel_r \mid c_{v-2}, c_{v-1}, c_v)$ est la probabilité que le *chunk* c_v soit en relation rel_r avec le *chunk* c_{v-r}. r désigne la distance entre les deux *chunks*, r = 0, 1, 2.

– $P(S_{uv}|h)$ est la probabilité de la segmentation S_{uv} (formule 5.7).

La probabilité $P_{rel}(rel_r | c_{v-2}, c_{v-1}, c_v)$ est estimée uniquement à partir de l'étiquette attribuée au dernier *chunk* c_v et des étiquettes des n-1 derniers *chunks* c_{v-2}, c_{v-1}. La probabilité est définie comme suit :

$$P_{rel}(rel_r | c_{v-2}, c_{v-1}, c_v) = \lambda_1 \cdot P(rel_r | c_{v-2}, c_{v-1}, c_v) + \lambda_2 \cdot P(rel_r | c_{v-r}, c_v) \quad (5.9)$$

avec

$$P(rel_r \mid c_{v-2}, c_{v-1}, c_v) = \frac{C(c_{v-2}, c_{v-1}, c_v, rel_r)}{C(c_{v-2}, c_{v-1}, c_v)} \quad (5.10)$$

$$P(rel_r | c_{v-r}, c_v) = \frac{C(c_{v-r}, c_v)}{\sum_{r=0,1,2} C(c_{v-r}, c_v)} \quad (5.11)$$

$\lambda_1$et λ_2 sont les paramètres d'interpolation $\lambda_1+\lambda_2 = 1$.

5.3.3 Prédiction des flexions

L'étape suivante du processus de prédiction concerne la prédiction des flexions. Cette étape est chargée de déterminer, pour chaque segmentation-relation S_{uvr}, une estimation de chaque flexion que le tag t_u peut avoir. Par exemple, pour les parties du discours relatives aux noms et aux adjectifs, une estimation est calculée pour les quatres cas masculin-singulier, masculin-pluriel, féminin-singulier et féminin-pluriel. Cette estimation repose d'une part sur la relation *r* pour permettre un éventuel accord inter-chunks, par exemple entre un verbe et son sujet. D'autre part, elle s'appuie sur les contraintes intra-chunk pour gérer les accords locaux comme l'accord entre déterminant-nom.

5.3.3.1 Prédiction inter-chunks

La prédiction inter-chunks a pour but de transmettre les accords potentiels entre les chunks. Comme l'analyseur ne détermine pas les fonctions syntaxiques des chunks, la transmission des accords est considérée comme un processus probabiliste. Dans SibyMot, la probabilité P_{acc} estime la probabilité que deux chunks identifiés par leur étiquette soit accordés ou non. Par exemple, l'accord sujet-verbe est rendu possible si une des segmentations relations S_{uvr} établit une relation entre le chunk du verbe c_v et un chunk c_{v-r} qui correspond à son sujet. La probabilité d'accord entre les chunks N (groupe nominal) et Vc (groupe du verbe conjugué) sera donc forte (proche de 1) tandis que pour d'autres couples, Np et Np par exemple (deux groupes prépositionnels), la probabilité d'accord sera non significative (mais non nulle puisque

statistiquement les deux chunks peuvent avoir le même cas). Pour chaque segmentation-relation S_{uvr}, deux cas sont donc envisagés, accord ou non, et la probabilité d'une « segmentation-relation-accord » S_{uvra} est estimée comme suit :

$$P(S_{uvra} \mid h) = P_{acc}(acc_a \mid S_{uvr}) \times P(S_{uvr} \mid h) \quad (5.12)$$

où :

- $P(S_{uvra} \mid h)$ désigne la probabilité de la segmentation-relation-accord S_{uvra} selon le contexte h.
- $P_{acc}(acc_a \mid S_{uvr})$ est la probabilité que le *chunk* c_v soit accordé ou non avec le chunk $c_{v\text{-}r}$. La valeur acc_a est un booléen qui indique s'il y a ou non accord.
- $P(S_{uvr} \mid h)$ est la probabilité de la segmentation-relation S_{uvr} (formule 5.8)

La probabilité $P_{acc}(acc_a \mid S_{uvr})$ est estimée comme suit :

$$P_{acc}(acc_a \mid S_{uvr}) = \frac{C(c_{v-r}, c_v, acc_a)}{C(c_{v-r}, c_v)} \quad (5.13)$$

Pour les segmentations S_{uvra} où il y a accord, le genre, le nombre et la personne du *chunk* $c_{v\text{-}r}$ est transmis au *chunk* c_v. Par exemple, après un pronom personnel « je » marqué « .s1 » (1ère personne du singulier, genre indéfini), la segmentation S_{uvra} prédisant un *chunk* Vc avec accord héritera de cette flexion. À l'inverse, pour les segmentations prédisant de nouveaux *chunks* sans accord, la flexion reste de type « indéfini ».

5.3.3.2 Prédiction intra-chunk

À ce stade, chaque segmentation S_{uvra} identifie un mot par son tag tu dans un chunk cv. Ce dernier porte une flexion reflétant à la fois un éventuel accord avec un chunk précédent et les flexions des éventuels mots qui précèdent dans le chunk. Par exemple : « N:mp[les/le/det/.p petits/petit/adj/mp ». La prédiction des flexions est réalisée selon le tag t_u (qui identifie les différentes flexions) et la flexion portée par son chunk c_v :

$$P_{flx}(f_{u,i} \mid S_{uvra}) = P(f_{u,i} \mid c_v.flx) \quad (5.14)$$

où :

- $f_{u,i}$ désigne une flexion qui dépend du tag tu donné au mot. Par exemple ms, fs, mp, fp pour les noms et les adjectifs

– $c_v.flx$ désigne la flexion du chunk portant les informations de genre, nombre et personne. Cette flexion peut être ambiguë (genre indéfini par exemple).

La probabilité $P(f_{u,i} \mid f_j)$ d'une flexion $f_{u,i}$ en fonction d'une flexion f_j de type genre, nombre, personne et apprise sur corpus de la manière suivante :

$$P(f_{u,i} \mid f_j) = \frac{C(f_j, f_{u,i})}{C(f_j)} \tag{5.15}$$

5.3.4 Prédiction des lemmes

L'étape suivante du processus de prédiction concerne la prédiction des lemmes[10]. Comme nous l'avons indiqué dans les principes du modèle SibyMot, un lemme est prédit différemment selon sa position dans le *chunk*. Dans SibyMot, nous avons identifiés trois types de prédiction, que nous rappelons ici :

5. Une prédiction *inter-chunks*. Ce type de prédiction est appliqué aux mots identifiés comme tête de *chunk*. La prédiction *inter-chunks* permet d'utiliser le pouvoir prédictif des n-1 dernières têtes de *chunk*.
6. Une prédiction *intra-chunks*. Cette prédiction s'applique aux mots lexicaux d'un *chunk* qui suivent la tête.
7. Une prédiction simple pour les autres mots, principalement les mots grammaticaux.

Le type de prédiction appliqué à un mot est fixé par la grammaire. Nous redonnons l'exemple de la séquence « *pre adj nom adv adj* » du *chunk Np* qui illustre tous les cas :

```
Np[pre:% det adj nom:* adv adj:+]
```

Le signe * associée au nom indique la tête du *chunk* et donc une prédiction *inter-chunks*. Celle-ci utilisera la préposition du *chunk* marquée par le signe `%`. Le dernier adjectif est marqué par un signe + spécifiant une prédiction *intra-chunk*, cet adjectif sera prédit en fonction de la tête.

Pour la prédiction du lemme l_i d'une segmentation-relation S_{uvr}, les estimations sont réalisées comme suit :

[10] Les lemmes ne portant pas de marque flexionnelle, cette étape pourrait être bien sûr réalisée en parallèle à l'étape précédente de prédiction des flexions.

$$P_{lem}(l_i \mid S_{uvr}) = \begin{cases} \lambda_{inter,1}\, P_{inter}(l_i \mid S_{uvr}) + \lambda_{inter,2}\, P_{n\text{-}lem}(l_i \mid S_{uvr}) & \text{cas } \textit{inter-chunks} \\ \lambda_{intra,1}\, P_{intra}(l_i \mid S_{uvr}) + \lambda_{intra,2}\, P_{n\text{-}lem}(l_i \mid S_{uvr}) & \text{cas } \textit{intra-chunk} \\ P_{n\text{-}lem}(l_i \mid S_{uvr}) & \text{sinon} \end{cases} \quad (5.16)$$

avec :

$$P_{inter}(l_i \mid S_{uvr}) = P(l_i \mid hd_{v\text{-}r},\ prp_v,\ t_u) \quad (5.17)$$

$$P_{intra}(l_i \mid S_{uvr}) = P(l_i \mid hd_v, t_u) \quad (5.18)$$

$$\begin{aligned} P_{n\text{-}lem}(l_i \mid S_{uvr}) = {} & \lambda_3\, P(l_i \mid l_{i-2}\ \&\ t_{i-2},\ l_{i-1}\ \&\ t_{i-1}, t_u) \\ & + \lambda_2\, P(l_i \mid l_{i-1}\ \&\ t_{i-1}, t_u) + \lambda_2\, P(l_i \mid t_u) \end{aligned} \quad (5.19)$$

La probabilité *inter-chunks* d'un lemme $P(l_i \mid hd_{v\text{-}r},\ prp_v,\ t_u)$ exprime le fait que sa probabilité d'apparition dépend de la tête du *chunk* $hd_{v\text{-}r}$ avec lequel il est en relation et de l'éventuelle préposition de son *chunk* prp_v.

La probabilité *intra-chunk* d'un lemme $P(l_i \mid hd_v, t_u)$ est estimée en fonction de la tête de son *chunk* hd_v.

La probabilité $P_{n\text{-}lem}(l_i \mid S_{uvr})$ est une probabilité « 3-lemme » : probabilité d'un lemme après les deux derniers lemmes (et *tags*) son étiquette grammaticale t_u (avec repli sur 2-lemme et 1-lemme).

5.3.5 Prédiction des mots

La prédiction de mots est l'étape finale de la prédiction. Elle délivre en sortie la probabilité de chaque mot du lexique. Le calcul de la probabilité est réalisé à partir des probabilités des lemmes et des accords précédemment calculées, et ce, pour chacune des segmentations S_{uvra} prédites :

$$P(w_i \mid h) = \sum_{(uvra)} \sum_{(j,k) \in WLi} P_{lem}(l_j \mid S_{uvr}) \times P_{flx}(f_{u,k} \mid S_{uvra}) \times P(S_{uvra} \mid h) \quad (5.20)$$

Cette équation exprime le fait que la probabilité d'apparition d'un mot est calculée en additionnant la probabilité de toutes les formes fléchies auxquelles il correspond et ce, pour chacune des segmentations S_{uvra} prédites. Une forme fléchie est définie par son lemme (dans une catégorie grammaticale) et sa flexion. La matrice *WL* établit pour chaque forme de mot la liste des formes fléchies correspondantes (exemple « les » : {lemme:le, tag:det, flexion:mp}, { lemme:le, tag:det, flexion:fp }, etc.).

5.4 Apprentissage

À l'heure actuelle, il n'existe pas pour le français de corpus annoté manuellement suffisamment vaste et qui soit disponible. On peut regretter par exemple que le corpus annoté dans le cadre de l'action GRACE ne soit pas mis à disposition. Les principales ressources que nous avons utilisées sont le lexique de l'ABU (Association des Bibliophiles Universels), le lexique « Lexique », le corpus du journal « Le Monde » (années 95 à 99). L'étiqueteur Cordial (cf. *infra*) nous a également permis de réaliser l'apprentissage des données de notre étiqueteur grammatical. Les paramètres des niveaux supérieurs (segmentation, relation, etc.) a été réalisé par apprentissage non supervisé.

5.4.1 Constitution du lexique

Le lexique de SibyMot a été créé principalement à partir des deux lexiques de l'ABU et « Lexique ».

5.4.1.1 Le Lexique de l'ABU

Le lexique de l'ABU[11] contient 300 000 entrées lexicales (formes fléchies). Chaque entrée est définie par son mot, son lemme, sa catégorie grammaticale et ses flexions. Il y a 11 catégories grammaticales. Le Tableau 5.4 donne les huit premières entrées de ce lexique.

a	a	Nom:Mas+InvPL
à	à	Pre
a	avoir	Ver:IPre+SG+P3
abaca	abaca	Nom:Mas+SG
abacule	abacule	Nom:Mas+SG
abaissa	abaisser	Ver:IPSim+SG+P3
abaissable	abaissable	Adj:InvGen+SG
abaissai	abaisser	Ver:IPSim+SG+P1

Tableau 5.4 : Extrait du lexique de l'ABU

5.4.1.2 « Lexique »

La base de données « Lexique »[12] contient 55 000 lemmes (129 000 formes fléchies) calculées à partir de deux sources : les pages Internet françaises indexées par le moteur de recherche *FastSearch* et un recueil de textes publiés après 1950, gérés par l'ATILF (Analyses et Traitements Informatiques du Lexique Français). La base de données des mots est disponible sous deux formes : la base « Graphème » indexée par forme (Tableau 5.5) et la base « Lemmes » indexé par lemme (Tableau 5.6).

[11] Le lexique de l'ABU est accessible en ligne à l'adresse http://www.abu.cnam.fr/DICO.
[12] « Lexique » est disponible à l'adresse http://www.lexique.org.

forme	catégorie	flexion	lemmes
dans	NOM;PRE	m(p)	dan;dans
dansa	VER:ind:ps	3s	danser
dansai	VER:ind:ps	1s	danser
dansaient	VER:ind:impf	3p	danser

Tableau 5.5 : Extrait de la base « Graphème » de « Lexique »

lemme	formes fléchies associées	catégories	genre	nombre
danois	danois;danoise;danoises	ADJ;NOM	m;f	s;(p)
dans	dans	NOM;PRE	m	(p)
dansant	dansant;dansante; dansantes;dansants	ADJ;VER:ppre	f;m	3p;(p)
danse	danse;danses	NOM;VER:imp:pr;ind:pr	f	s;1s;3s;2s
danser	dansa;dansai;dansaient;	ADJ;NOM;VER:cond…	f;m	1s;3s;…

Tableau 5.6 : Extrait de la base « Lemme » de « Lexique »

Cette base de données contient d'autres informations comme la transcription phonétique et des fréquences d'usage.

L'organisation des entrées de ce lexique peut cependant porter à confusion. Par exemple au lemme « danse » (Tableau 5.6) sont associées les deux formes fléchies du nom (« danse » au singulier et « danses » au pluriel) mais les catégories données contiennent également le verbe dont le lemme est défini à l'entrée suivante. De plus, toujours pour « danse », le champ du nombre ne spécifie pas de pluriel.

Pour exploiter cette base nous avons donc dû effectuer des analyses croisées avec le lexique de l'ABU et nous n'avons extrait que les lemmes et formes fléchies pour lesquels il n'y avait pas d'ambiguïtés.

5.4.1.3 Le lexique de SibyMot

Le lexique de SibyMot est organisé par catégorie grammaticale et ensuite par lemme (Tableau 5.7). Cette implantation est adaptée à notre modèle qui utilise principalement les notions de catégorie grammaticale et de lemme. Une indexation par forme permet de retrouver rapidement la liste des couples (tag, lemme) associés à chaque forme.

lemme	formes fléchies ms, fs, mp, fp	occ.
le	le, la, les, les	98632
l'	l', l', ,	21363

Tableau 5.7 : Extrait du lexique de SibyMot, catégorie detartdef

Pour les trois catégories `adjcom`, `nomcom`, `vercomcnj` les entrées sont légèrement différentes. La liste des formes fléchies de chaque lemme est représentée par un couple (racine, « déclinaison ») qui permet une optimisation importante en terme de place mémoire (dans le cas extrême des verbes l'espace alloué est ainsi divisé par 20) (Tableau 5.8).

Num	1s-ip	2s-ip	3s-ip	1p-ip	2p-ip	3p-ip	1s-ii	
3	e	es	e	ons	ez	ent	ais	…
4	ce	ces	ce	çons	cez	cent	çais	…
5	e	es	e	eons	ez	ent	eais	…

lemme	racine	decl.	occ
dire	d	113	1030
rester	rest	3	834
sembler	sembl	3	658

Tableau 5.8 : Organisation du lexique des verbes conjugués

Par exemple, le verbe « rester » a pour racine « rest » et comme numéro de déclinaison « 3 ». Ainsi pour obtenir le verbe « rester » à la 1ère personne du singulier de l'indicatif présent (noté 1s-ip), on ajoute à la racine « rest » la terminaison « e ».

Pour les lexiques principaux (adjectifs, adverbes, noms communs et verbes conjugués) les entrées sont issues de l'intersection des lexiques de l'ABU et de « Lexique ». Par exemple, pour les adjectifs, ces deux lexiques contiennent autour de 14 000 lemmes. L'union de ces deux lexiques donne 20 000 lemmes et l'intersection 10 000. Ce choix garantit une certaine fiabilité des entrées de notre lexique. Au total le lexique de SibyMot contient près de 50 000 lemmes.

5.4.2 Paramètres de l'étiqueteur grammatical

L'apprentissage des paramètres de l'étiqueteur grammatical a été réalisé sur un extrait du corpus du journal « Le Monde » avec l'étiqueteur Cordial[13].

Le corpus « Le Monde » n'est pas directement exploitable tel qu'il est fourni. Il contient de nombreuses balises dans le texte spécifiant des mises en forme (saut de page, de colonne, retour ligne, etc.) ou des champs (auteur, date, etc.). De plus le format des textes est variable (80 colonnes, mots coupés ou non en fin de ligne, etc.). Par ailleurs, ces balises et ces formats ne sont pas spécifiés et évoluent entre l'année 1995 et 1999. Nous avons donc réalisé les pré-traitements nécessaires pour rendre ce corpus exploitable. En fin de traitement, certains articles ont été filtrés (résultats sportifs, listes de résultats électoraux ou de nominations), puis le corpus a été segmenté par jour pour permettre le traitement par Cordial.

L'analyseur Cordial permet l'étiquetage de textes même si ceux-ci ne sont pas parfaitement formatés. Il utilise un jeu d'étiquettes proche de celui spécifié dans l'action GRACE (Tableau 5.9). De plus, l'analyseur Cordial reconnaît les locutions comme « en effet », « afin que », ce qui nous a permis de compléter le lexique de SibyMot.

[13] L'analyseur Cordial est distribué par la société « Synapse Développement »

n° mot	n° phrase	mot	lemme	ambiguités	étiquette Cordial	étiquette GRACE
1	1	AVEC	avec	A2	PREP	Sp
2	1	l'	le	A4	DETDFS	Da-ms-d
3	1	année	année		NCFS	Ncfs
4	1	1995	1995	A2	ADJNUM	Mc..

Tableau 5.9 : Étiquetage par Cordial

Une fois l'étiquetage réalisé par Cordial, une dernière étape est nécessaire pour traduire les étiquettes GRACE dans notre jeu d'étiquettes. En particulier nos étiquettes ne portent pas d'indices flexionnels (mais l'information des flexions est conservée pour l'apprentissage des paramètres de la gestion des accords). Après une dernière transformation sur la mise en forme, on obtient un texte de la forme :

```
avec/avec/precom/ l'/l'/detartdef/fs année/année/nomcom/fs
1995/1995/numann/.. ,/,/pctfbl/ une/un/detartind/fs
nouvelle/nouveau/adjcom/fs institution/institution/nomcom/fs
```

où chaque mot est représenté par quatre champs : mot/lemme/étiquette/flexion.

C'est à partir de ce texte annoté que l'apprentissage des paramètres de l'étiqueteur grammatical sont réalisés : nombre d'occurrences des séquences n-tags, nombre d'occurrences des lemmes dans leur classe, paramètres d'interpolation. Le corpus final utilisé correspond au mois de janvier 1995, soit près de deux millions de mots.

5.4.3 Paramètres du segmenteur

Pour la segmentation des énoncés, les paramètres à apprendre sur corpus sont les fréquences des séquences de tags de la grammaire (par exemple la fréquence de la séquence « det nom » du *chunk* « N ») et les fréquences n-*chunks* (fréquence des séquences formées par les étiquettes de chunks). Le corpus d'apprentissage doit donc être segmenté.

Nous avons envisagé d'utiliser Cordial qui fournit également une analyse de la phrase en syntagmes. Cependant, nous avons rejeté cette possibilité pour trois raisons. D'abord, sur les tests effectués, l'analyse de Cordial en syntagmes nous est apparue de qualité moyenne. Ensuite la transformation des syntagmes Cordial dans notre système de *chunks* semblaient difficile à réaliser. Enfin, dans les fichiers produits par notre version de Cordial, certaines informations sur les syntagmes sont manquantes (ce qui n'est pas le cas lorsque l'on utilise l'application fenêtrée de Cordial). Par exemple, dans le cas des compléments d'une infinitive, l'infinitive et tous ses compléments sont marqués par le même syntagme.

L'apprentissage a donc été réalisé de manière non supervisée de la manière suivante. Dans un premier temps, toutes les fréquences (des séquences de la grammaire et n-chunks) sont initialisées à 1, puis le texte est segmenté avec ces

paramètres. À partir de la segmentation produite, un premier apprentissage est réalisé. La boucle segmentation-apprentissage est ensuite réitérée jusqu'à atteindre un état relativement stable où les segmentations ne sont que très peu modifiées d'une étape à l'autre. La dernière étape est finalement conservée.

Cette méthode d'apprentissage est peu satisfaisante. Cependant les résultats obtenus sont très convenables (comme nous le verrons dans l'évaluation). Une raison tient dans notre définition « maximale » des *chunks* et dans notre méthode d'estimation d'une séquence de *chunks*. En effet, lors de la première segmentation, pour une séquence comme « det nom adj », la probabilité de la séquence N[det nom adj] est inférieure à celle N[det nom] A[adj] car cette dernière combine la probabilité de deux *chunks*. Notre *segmenteur* privilégie donc les *chunks* longs, ce que nous souhaitons.

L'erreur la plus fréquemment observée avec cette méthode concerne la segmentation des adverbes. En effet, l'étiquette grammaticale de l'adverbe est très ambiguë pour le segmenteur :

1. il Vc[semble/vercomcnj] A[très/adv content/adj]
2. il Vc[semble/vercomcnj parfaitement/adv] A[content/adj]
3. il Vc[semble/vercomcnj] F[ici/adv] A[content/adj]

Dans ces trois phrases, la séquence de tags est la même, mais les segmentations sont différentes. Dans la première, l'adverbe de quantité « très » modifie l'adjectif qu'il précède. Dans le deuxième cas, nous rattachons l'adverbe de manière au verbe. Dans la troisième phrase, l'adverbe « ici » est un circonstanciel de lieu.

La phrase suivante donne un exemple de segmentation produite. Les mots d'un même chunk sont notés entre <>, les informations du chunk (étiquette et flexion) sont marquées à la fin du chunk après la balise # :

```
<avec/avec/precom//% l'/l'/detartdef/fs année/année/nomcom/fs/*
1995/1995/numann/../+#Np:f3s> <,/,/pctfbl/#Ys> <une/un/detartind/fs
nouvelle/nouveau/adjcom/fs
institution/institution/nomcom/fs//*#N:f3s>
```

5.4.4 Paramètres du prédicteur

Dans notre modèle, l'étape suivant la prédiction des segmentations concerne la prédiction des relations. Une fois de plus nous avons dû réaliser un apprentissage non supervisé pour ces paramètres.

Dans le cadre de cet apprentissage non supervisé, nous avons cherché à établir un maximum de dépendances avec les deux derniers *chunks*. En particulier, nous ne cherchons ni à déterminer des relations de distance supérieure à celle utilisée par le

modèle (2), ni à trouver de relation nulle. L'objectif revient donc à déterminer des deux *chunks* qui précèdent, quand c'est possible, celui dont la relation est la plus probable. Nous avons pour cela utilisé l'information mutuelle. Cette mesure est souvent employée pour déterminer si la co-occurrence de deux mots est significative (les mots sont liés) ou si elle est due au hasard. Cette mesure a par exemple servi de critère pour l'extraction de séquence de mots [Jelinek 1990]. L'information mutuelle $J(w_i, w_j)$ entre deux mots w_i et w_j est mesurée par :

$$J(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i) \times P(w_j)} \tag{5.21}$$

Ainsi pour chaque *chunk* avec une tête lexicale du corpus précédemment segmenté, nous avons mesuré l'information mutuelle entre la tête hd_v du *chunk* en cours et les deux têtes précédentes hd_{v-1} hd_{v-2} : $J(hd_v, hd_{v-1})$ et $J(hd_v, hd_{v-2})$. La relation est établie avec le *chunk* qui maximise l'information mutuelle. Tout comme l'apprentissage non supervisé pour les segmentations, le processus est réitéré plusieurs jusqu'à une certaine stabilisation.

Contrairement à l'apprentissage non-supervisé précédent, cette fois-si les mises en relation déterminées sont éloignées de ce que l'on pourrait attendre. Cependant elles demeurent cohérentes d'un point de vue probabiliste.

La phrase suivante donne la mise en relation produite. L'information de relation est ajoutée au chunk par l'indication de la distance (en gras) :

```
<avec/avec/precom//% l'/l'/detartdef/fs année/année/nomcom/fs/*
1995/1995/numann/../+#Np:f3s:0> <,/,/pctfbl/#Yf::0>
<une/un/detartind/fs nouvelle/nouveau/adjcom/fs
institution/institution/nomcom/fs//*#N:f3s:2>
```

Une fois ces relations établies, le corpus contient toutes les annotations nécessaires à l'ensemble des paramètres d'apprentissage tels que définis dans la partie précédente. En particulier, il s'agit de déterminer pour la prédiction des relations les fréquences $C(c_{v-2}, c_{v-1}, c_v, rel_r)$, pour la gestion des accords les fréquences $C(c_{v-r}, c_v, acc_a)$, ou encore, pour la prédiction des lemmes les fréquences $C(hd_{v-r}, prp_v, hd_v)$.

L'apprentissage tel qu'il est réalisé n'est pas totalement satisfaisant et un corpus annoté serait souhaitable. Nous reviendrons sur l'apprentissage des paramètres lors de l'évaluation et de la conclusion.

5.5 Évaluation

Cette dernière partie du chapitre est consacrée à l'évaluation de SibyMot. Nous avons deux objectifs principaux pour cette évaluation. D'abord il s'agit de montrer l'intérêt de la prédiction n-*chunks* et de valider certaines hypothèses comme l'analyse de plusieurs segmentations en parallèle. De manière plus générale, nous avons évalué la plupart des étapes de la prédiction. Le deuxième objectif est, au final, d'évaluer les capacités prédictives du modèle.

Dans les expériences qui suivent, le corpus utilisé pour l'apprentissage des paramètres du modèle est celui défini dans la partie précédente, ce qui correspond à un mois du Monde, soit près de deux millions de mots. En ce qui concerne le corpus de test, nous avons extrait de manière aléatoire 20 articles des cinq années à notre disposition (95 à 99), soit 100 articles qui représentent plus de 50 000 mots. Ces textes ont été étiquetés et segmentés manuellement.

L'évaluation a porté successivement sur le lexique, l'étiqueteur grammatical, le *segmenteur*, le *prédicteur* et enfin sur les capacités prédictives.

5.5.1 Converture du lexique

La qualité du lexique est importante car non seulement les mots inconnus ne pourront bénéficier de la prédiction mais en plus ces mots inconnus risquent de dégrader les performances de l'analyse et de la prédiction. Le tableau suivant relève le nombre de mots inconnus trouvés sur le corpus de test. Nous avons également mentionné une catégorie spéciale « les mots connus mal assignés » qui sont des mots connus mais aucune des étiquettes grammaticales du lexique ne correspond à l'étiquette correcte (par exemple le nom « arrérages » a été reconnu seulement comme le verbe « arrérager » conjugué).

	Occ	%
Mots inconnus	1200	2,4 %
Mots connus mal assignés	61	0,1 %
Mots connus	49 628	97,5 %
Total des mots	50 889	100,0 %

Tableau 5.10 : Évaluation des mots inconnus

Le nombre de mots mal assignés est très faible (0,1 %), on peut donc considérer que les performances ne seront pas dégradées de manière significatives par ces mots.

Le nombre de mots inconnus (2,4 %) peut être considéré comme relativement faible. Le Tableau 5.11 détaille leur répartition dans les différentes catégories (par

rapport au corpus de test). Lors de l'évaluation de l'étiqueteur nous préciserons leur taux d'étiquetage correct séparément.

Catégorie	Mots	%
adjectifs	55	4,6 %
adverbes	11	0,9 %
locutions adverbiales	27	2,3 %
noms communs	197	16,4 %
noms propres	735	61,3 %
expressions numériques	75	6,3 %
verbes	36	3,0 %
locutions prépositionnelles	42	3,5 %
autres locutions	8	0,7 %
autres	14	1,2 %
	1200	100,0 %

Tableau 5.11 : Répartition des mots inconnus par catégorie

Nous retrouvons ici un résultat habituel à savoir un très fort taux de noms propres inconnus (61,3 %). Si l'on associe aux 61,3 % des noms propres les 6,3 % des expressions numériques (par exemple « 6,13 »), qui ne peuvent être toutes recensées, soit 67,6 %, il reste seulement un tiers (32, 4 %) de mots communs qui ne sont pas dans notre lexique. Reporté au taux de mots inconnus (2,4%, Tableau 5.10), nous avons donc que 0,8 % de mots communs inconnus du lexique, ce qui souligne la qualité de notre lexique.

Une étude plus fine sur ces mots communs inconnus montre que plus de 80 % des occurrences de ces mots correspond à des mots qui n'apparaissent qu'une fois. Ceci est un résultat important pour l'apprentissage des mots communs inconnus (non réalisé par SibyMot). En effet, il semble plus judicieux d'établir un compte des mots inconnus et de ne l'ajouter au lexique qu'à partir d'un certain nombre d'observations dans un ou plusieurs textes.

À l'inverse, le nombre de noms propres est élevé et plus de 40 mots apparaissent plus de deux fois, l'apprentissage des noms propres inconnus (au moins les noms propres qui commencent avec des majuscules et qui sont non composés) est à envisager pour SibyMot.

5.5.2 Évaluation de l'étiqueteur grammatical

Nous rappelons que nos étiquettes assignées aux mots ne donnent pas d'indications flexionnelles (nombre, genre, personne, temps, mode) pour faciliter la tâche d'étiquetage. Nous avons donc réalisé une première évaluation sur l'intérêt d'un tel étiquetage en mesurant le nombre d'ambiguïtés par mot, d'une part, si l'on compte toutes les formes fléchies et, d'autre part, si l'on ne compte que le nombre de lemmes (Tableau 5.12).

Ambiguïté moyenne en nombre de formes fléchies	3,1
Ambiguïté moyenne en nombre de lemmes	1,9

Tableau 5.12 : Ambiguïté moyenne par mot

On peut constater que l'ambiguïté est fortement réduite puisque en moyenne chaque mot correspond à un peu plus de trois formes fléchies et seulement deux lemmes. On peut donc supposer que la tâche d'étiquetage est facilitée : d'un côté il y a moins d'ambiguïtés et de l'autre le nombre de paramètres à estimer est moins important donc, *a priori*, plus fiables. Nous rappelons par ailleurs qu'en prédiction les flexions (qui ne sont donc pas marquées par les étiquettes) sont prédites aussi mais de manière séparée.

	Mots	%
Étiquetage correct	49 812	97,9 %
Étiquetage incorrect	1 077	2,1 %
Total	50 889	100,0 %
Mots inconnus correctement étiquetés	1 129	94,1 %
Mots inconnus incorrectement étiquetés	71	5,9 %
Total	1 200	100,0 %

Tableau 5.13 : Évaluation de l'étiquetage

Les résultats montrent un bon taux d'étiquetage correct (97,9 %). Notons cependant que ce bon taux est lié en partie au jeu d'étiquettes réduit et sans marque flexionnelle. On peut observer également un très bon taux d'étiquetage pour les mots inconnus (94,1 %). L'étape de segmentation qui suit l'étape d'étiquetage grammatical s'appuie donc sur un étiquetage robuste.

La dernière évaluation sur l'étiqueteur a porté sur le nombre de remises en cause : au fur et à mesure que la phrase est saisie, l'étiquetage d'un mot peut être remis en cause avec les nouveaux mots saisis, ce qui signifie que l'étiquette précédemment définie et donnée au *segmenteur* était erronée. Sur le corpus de test, ce taux a été évalué à 2,3 % des mots. Ce taux est relativement faible mais si on l'additionne au

taux d'étiquetage incorrect à 2,1 % (en supposant qu'il n'y ait pas intersection), près d'un mot sur vingt (4,4 %) a une étiquette erronée avant l'étape de segmentation. Une étude plus poussée semble donc nécessaire pour éventuellement proposer à la segmentation non pas une mais plusieurs étiquettes possibles.

5.5.3 Évaluation de la segmentation

Pour la segmentation, nous avons utilisé les trois métriques habituellement proposées pour ce genre de tâche :

$$rappel = \frac{nombre\ de\ chunks\ corrects}{nombre\ de\ chunks\ du\ corpus\ de\ test}$$

$$précision = \frac{nombre\ de\ chunks\ corrects}{nombre\ de\ chunks\ produits}$$

$$F_\beta = \frac{(\beta^2 + 1).précision.rappel}{\beta^2.précision + rappel} \text{ avec } \beta = 1$$

	Étiquettes correctes données	Étiquettes de l'analyseur
Nombre de *chunks* du corpus de test	22 560	22 560
Nombre de *chunks* produits	22 566	22 721
Nombre de *chunks* corrects	21 620	21 193
% Rappel	95,8 %	93,9 %
% Précision	95,8 %	93,3 %
% $F_{\beta=1}$	95,8 %	93,6 %

Tableau 5.14 : Évaluation de la segmentation

Le Tableau 5.14 présente deux évaluations. Ces évaluations ne tiennent pas compte des *chunks* « ponctuations » que nous ajoutons dans notre segmentation. Dans un premier temps, nous avons évalué le *segmenteur* seul, en lui donnant en entrée les mots correctement étiquetés. Le nombre de *chunks* produits étant proche de celui dans le corpus de test, les taux de précision et de rappel sont identiques à 95,8 %. La deuxième évaluation utilise les étiquettes fournies par l'étiqueteur grammatical, globalement le taux baisse de 2 % ($F_{\beta=1}$ = 93,6 %), ce qui correspond approximativement au taux d'erreur d'étiquetage. Nous avons donc une légère dégradation des performances tout en restant à un taux acceptable. Tout comme l'étiqueteur, notre *segmenteur* obtient donc des taux satisfaisants.

Nous avons également mesuré le taux de remise en cause de la même manière que pour l'étiqueteur, ce taux (qui était de 2,3 % pour l'étiqueteur) est de 15,4 %. Ce

pourcentage élevé justifie que plusieurs propositions de segmentations soient données en entrée de la prédiction (prédiction multi-analyses 5.3.1.3).

5.5.4 Évaluation de la prédiction

Pour l'évaluation du module de prédiction nous avons surtout cherché à évaluer l'intérêt de la prédiction n-*chunks*. Nous rappelons que dans notre modèle, pour un élément qui n'a pas de prédiction particulière (i.e. différente de la prédiction *inter-chunks* et *intra-chunk)* son estimation est réalisée par une prédiction n-lemmes (hors composantes de repli) :

$$P_{n\text{-}lem}(l_i \mid S_{uvr}) = P(l_i \mid l_{i\text{-}2} \;\&\; t_{i\text{-}2},\; l_{i\text{-}1} \;\&\; t_{i\text{-}1}, t_u) \qquad (5.22)$$

Tandis que la prédiction *inter-chunks* porte sur l'éventuelle préposition et la tête du *chunk* en relation (hors composantes de repli) :

$$P_{inter}(l_i \mid S_{uvr}) = P(l_i \mid hd_{v\text{-}r},\; prp_v,\; t_u) \qquad (5.23)$$

Nous avons donc évalué dans le corpus de test, pour chaque tête de *chunk* lexicale, d'un côté la probabilité 3-lemmes (comme si le mot ne bénéficiait pas de prédiction *inter-chunks*) et les probabilités *inter-chunks* sur les 2 têtes qui précèdent. Selon un processus d'élection nous incrémentons de 1 le meilleur type de prédiction (3-lemme, tête$_{-1}$, tête$_{-2}$), dans le cas où toutes les probabilités sont à 0, nous incrémentons un type « repli ». Les résultats sont donnés par le Tableau 5.15.

	Occ	% tête lex	% total
repli	8 415	59,3 %	37,3 %
3-lemmes	4 058	28,6 %	18, %
tête$_{-1}$	1 405	9,9 %	6,2 %
tête$_{-2}$	312	2,2 %	1,4 %
chunks avec tête lexicale	14 190	100,0 %	62,9 %
chunks sans tête lexicale	8 370		37,1 %
Total	22 560		100,0 %

Tableau 5.15 : Évaluation de la prédiction n-*chunks*

Nous rappelons que normalement la probabilité $P_{inter}(l_i \mid S_{uvr})$ n-*chunks* est interpolée avec une probabilité n-lemme (la probabilité 3-lemmes étant elle-même interpolée avec un 2-lemme et un 1-lemme). Ici nous n'avons donc pas calculé les composantes de repli.

Dans ce tableau les occurrences sont en nombre de *chunks*, on peut constater que sur les 22 560 *chunks* du corpus, 37,1 % sont sans tête lexicale (*chunks* de ponctuation et *chunks* de mot grammaticaux comme les pronoms) et donc 62,9 % avec des têtes lexicales. La dernière colonne exprime le pourcentage des occurrences par rapport au nombre total de *chunks* (22 560) et la colonne intermédiaire le pourcentage en fonction du nombre de têtes lexicales (14 190).

Sur ces 14 190 têtes de *chunks*, 8 415 (59, 3%) ont obtenu une probabilité nulle par le 3-lemme, avec la tête$_{-1}$ et la tête$_{-2}$. Ce taux est élevé puisqu'il signifie que dans près de 60 % on utilise un repli sur un 2-lemme ou 1-lemme. Ceci suggère peut-être que notre corpus, malgré sa taille (2 millions de mots) est insuffisant. Enfin, sur les 40,7 % restants, dans 28,6 % des cas le 3-lemme est meilleur et dans 11,1 % la prédiction *inter-chunks* est meilleure.

On peut donc constater que la composante 3-lemme obtient globalement de meilleurs résultats dans 28,6 % des cas, elle reste donc essentielle dans le processus de prédiction. D'un autre côté, dans 11,1 % des cas la prédiction n-*chunks* apporte une amélioration. Sachant que ces prédictions sont combinées avec une interpolation, on s'attend donc à un comportement global meilleur dans les capacités prédictives.

5.5.5 Évaluation des capacités prédictives

Enfin, nous avons testé les capacités de notre système SibyMot pour l'économie de saisie. Le corpus de test de 50 889 mots contient 2 175 phrases (23 mots par phrase) et 242 049 caractères (4,8 caractères par mot), hors caractères espaces. Dans le mode de saisie proposé, le caractère espace est automatiquement inséré à la suite du mot prédit.

Pour cette évaluation, la taille de la liste est fixée à cinq mots. Nous avons également testé le modèle n-gramme avec un paramètre de n = 1 à 3. Les n-grammes ont été entraînés sur le même corpus d'apprentissage que SibyMot et les paramètres d'interpolation ont été estimés de la même manière que ceux de SibyMot. Les résultats de l'évaluation sont donnés par le Tableau 5.16.

	% économisés
1-gramme	43,9 %
2-gramme	51,2 %
3-gramme	55,8 %
SibyMot	57,1 %

Tableau 5.16 : Évaluation des prédicteurs

Le modèle 1-gramme établit la probabilité d'apparition d'un mot hors contexte, il correspond aux propositions que ferait un logiciel du commerce tels que nous les avons présentés au chapitre II. Les prédictions de SibyMot sont donc nettement meilleures : 57,1 % contre 43,9 % (+ 13,2 %). L'économie de saisie obtenue par SibyMot est également plus importante que le bi-gramme (+ 5,9 %) et plus efficace que le modèle 3-gramme (+ 1,3 %). Cette dernière comparaison montre que notre modèle avec des connaissances syntaxiques obtient de meilleurs résultats qu'un modèle n-gramme. Compte tenu que nous ne disposions d'aucune donnée d'apprentissage étiquetée manuellement, il est fort probable que notre modèle a encore une certaine marge de progression.

Enfin nous conclurons en comparant SibyMot à d'autres systèmes d'aide. Le système PAL, sur de l'anglais, économise jusqu'à 50 % des saisies. HandiAS, avec une liste de 3 mots permet d'économiser plus de 43 % des saisies par rapport à 35 % au modèle fréquentiel. Enfin VITIPI permet, sans liste de mot, d'économiser de 26 à 37 %. Comme on peut le constater, si les conditions d'expérimentation sont différentes, notre modèle SibyMot semble se situer dans le haut de l'échelle des systèmes d'aide à la communication. On peut donc considérer que l'objectif de notre modèle est atteint.

Chapitre 6

Sibylle

6.1 Introduction

La phase de validation par les utilisateurs est une étape nécessaire de l'évaluation [Preece 1994]. Les résultats théoriques ne se traduisent en effet pas toujours par des gains effectifs. Dans le cas de Sibylle, cette évaluation est même indispensable à cause de l'aspect dynamique du clavier simulé susceptible d'accroître la charge cognitive. De manière plus générale, l'application nous permet d'évaluer les modules linguistiques (pertinence des prédictions en fonction de la saisie effective) et les solutions ergonomiques mises en œuvre.

Une partie de la conception de l'application Sibylle a été réalisée en concertation avec le centre de Kerpape. En particulier, nous avons pu comparer les divers logiciels disponibles à Kerpape, et connaître leur utilisation par plusieurs personnes handicapées et dans différents contextes (communication courante, école de Kerpape, ergothérapie) [Scapin 1990].

Nous avons réalisé une application opérationnelle et conviviale et ce pour plusieurs raisons. D'abord, il convient de rappeler que quelque soit l'outil de communication, la composition de texte demande aux personnes handicapées un effort. Ces personnes sont généralement habituées à un système de communication et l'utilisation d'une autre aide demande une phase d'apprentissage. Cet effort supplémentaire sera d'autant mieux accepté que l'interface est conviviale et intègre des commodités et des fonctionnalités dont l'utilisateur a l'habitude. Ensuite, il existe de nombreux logiciels dans le commerce et certaines personnes handicapées utilisent couramment l'ordinateur. Elles ont donc une certaine exigence quant aux interfaces proposées. Enfin, l'expérience de Kerpape a montré qu'une interface réalisée juste dans le cadre d'une évaluation peut révéler des fautes d'ergonomie qui vont fortement pénaliser l'évaluation.

6.2 Présentation de l'interface

L'interface présentée (Figure 6.1) est celle actuellement utilisée au centre de Kerpape.

Figure 6.1 : Interface de Sibylle et quelques définitions de touche

L'interface est composée d'une zone d'édition de texte et du clavier simulé.

Le clavier simulé est composé de pavés de touches. L'objectif de cette organisation est toujours d'optimiser le nombre de défilements en balayage automatique. Le principe est ici de regrouper les touches de même thème (lettres, chiffres, touches de déplacement, fonctions générales). Le basculement d'un pavé à l'autre est réalisé par des touches de « saut de clavier ». En particulier, la touche pour accéder à la liste de mots à partir du clavier de lettre est définie sur la deuxième position en haut à gauche du clavier simulé.

Selon le même principe de « saut », l'accès aux ponctuations (en bas du clavier de lettres) est réalisable par une touche de raccourci située en haut à droite du clavier de

lettres. En validant cette touche, le curseur se positionne directement sur la première touche des ponctuations.

On peut également remarquer que le caractère « fin de mot » (le caractère « espace ») est symbolisé par le tiret bas « _ ». Dans la Figure 6.1 le caractère est en fin de clavier de lettres car un mot vient d'être sélectionné et donc un « espace » vient d'être inséré. De plus, dans le texte tapé, les caractères en grisé sont les caractères prédits par SibyMot.

Nous avons également implanté un « clic long ». Il permet d'offrir un degré de liberté supplémentaire pour les personnes handicapées qui sont à même de contrôler leur geste. Le « clic long » est réalisé en maintenant enfoncé plus longtemps le bouton de validation (son activation et la durée sont paramétrables). Lorsque la durée d'appui du « clic long » est atteinte, le curseur change de couleur. En défilement linéaire, la fonction associée au clic long est le retour en position initiale du curseur.

La liste suivante présente quelques unes des fonctionnalités de Sibylle :

- Adaptation à l'interface matérielle : le logiciel s'adapte à l'interface matérielle d'entrée. Les deux modes d'interaction proposés sont le mode « souris », et le mode « balayage automatique ». En mode « souris », le logiciel propose un suivi de curseur (mise en relief de la touche sous le curseur souris). Pour le balayage automatique il est possible d'utiliser le défilement linéaire ou ligne/colonne. Les trois ordres alphabétique, fréquentiel et dynamique (vus au chapitre II) sont implantés.
- Configuration de l'interface matérielle : le geste dirigeant l'interface matérielle est souvent mal contrôlé. Il est possible de corriger certaines erreurs à l'aide de paramètres. Ces paramètres sont : la durée minimale d'enfoncement des touches (pour éviter les appuis intempestifs), la durée minimale entre deux clics (évite des successions de clics non désirées), la vitesse de défilement du balayage. Actuellement ces paramètres doivent être réglés manuellement par un assistant (sauf la vitesse de défilement, réglable par la personne handicapée). Notons cependant qu'il existe des jeux de tests pour régler semi-automatiquement ces paramètres [Noirhomme 2000] ; Le système EDITH [Pino 2000] propose une vitesse de balayage adaptative.
- Configuration de la police de caractère : les troubles de la vue sont fréquents (en particulier chez les personnes âgées), la police de caractère des lettres du clavier est donc configurable (taille, style, casse et couleur).

De manière générale, l'interface est largement paramétrable (polices, couleurs, taille et position des claviers). Il est également possible de modifier les claviers, les touches, par l'intermédiaire de fichiers de configuration

6.3 Exemple de saisie

Nous allons maintenant présenter un exemple de saisie sur une phrase issue des textes relevés lors de l'expérience à Kerpape : « cher François , le ski … ». Les résultats de la saisie sont donnés dans le Tableau 6.1. La première colonne rappelle le contexte, la deuxième colonne donne les prédictions de SibyLettre et la troisième les prédictions de SibyMot. La liste de mot est configurée pour afficher cinq mots.

Contexte	Lettres	Mots
cher François , le ski	d l p a e c s **m**…	, de . du et
cher François , le ski m	a e i o…	magique mais mondial militaire **me**
cher François , le ski me	d l p a e c s **m**…	a le est la les
cher François , le ski me m	d l p a e c s **m**…	met **manque** montre mêle marque

Tableau 6.1 : Exemple de saisie dans Sibylle

Après le contexte « … le ski », la prédiction de mot délivre les cinq mots « , de . du et ». Le mot souhaité « me » n'apparaît pas, l'utilisateur sélectionne donc la lettre « m » dans le clavier simulé. Cette lettre se situe en 8ème position. Une fois la lettre « m » tapée, les deux prédictions sont actualisées. Cette fois-ci le mot « me » est affiché en 5ème position dans la liste des mots, l'utilisateur le sélectionne, le caractère espace est automatiquement inséré. Pour la saisie du mot suivant « manque » les mots prédits sont « a le est la les » (les deux auxiliaires à la troisième personne de l'indicatif présent ainsi que des pronoms personnels). Le mot n'apparaissant pas, la lettre « m » est une nouvelle fois sélectionnée dans le clavier de lettres. Les deux prédictions sont actualisées et le mot « manque » apparaît.

Les figures suivantes montre les différentes étapes de la saisie du mot « me » dans les mêmes conditions.

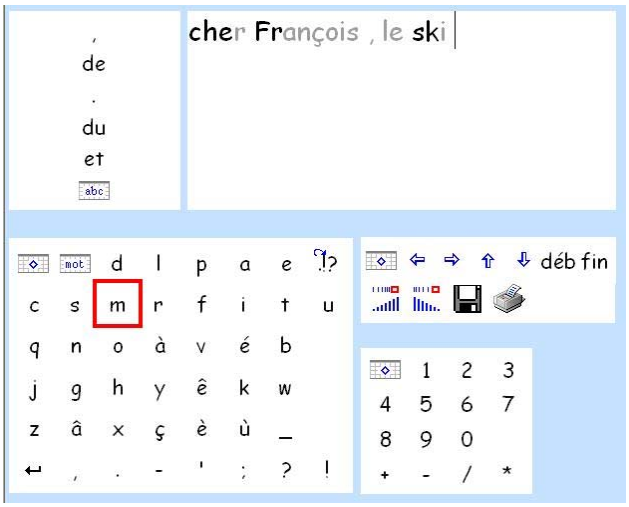


Figure 6.2 : Sélection de la lettre « m » dans le clavier simulé

Au début de la saisie du mot « me » la liste de mots affiche « , de . du et » (Figure 6.2). Le mot n'apparaissant pas l'utilisateur doit taper la lettre « m ». Le curseur parcourt la liste des lettres sur le clavier simulé jusqu'à atteindre la lettre « m » (Figure 6.2).

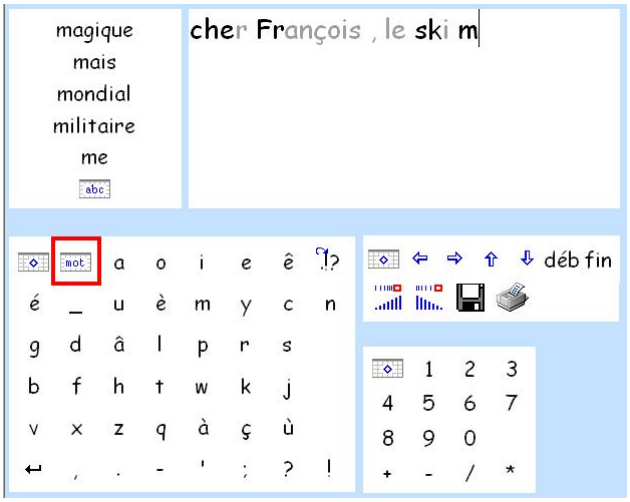


Figure 6.3 : Sélection de la touche d'accès aux mots

Après la sélection de la lettre « m », le clavier simulé et la liste de mots ont été actualisés (Figure 6.3) et le curseur remis en position initiale, en haut à gauche du clavier de lettre. Cette fois le mot « me » est affiché dans la liste. Pour accéder à cette liste, l'utilisateur doit d'abord sélectionner la touche d'accès à la liste des mots. Cette touche figure en deuxième position sur le clavier simulé (Figure 6.3).

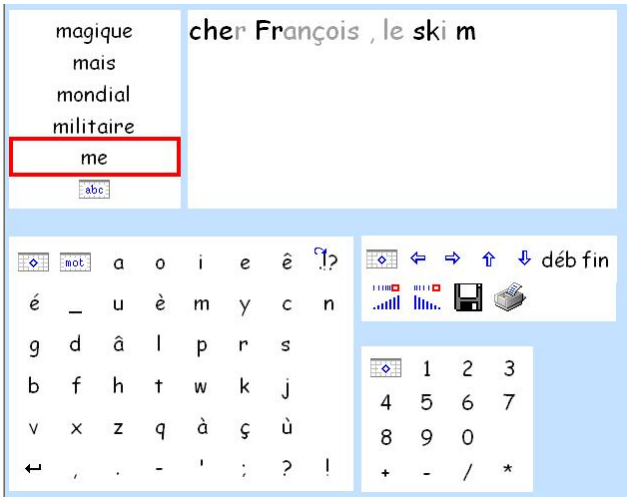


Figure 6.4 : Sélection du mot « me » dans la liste des mots

Une fois l'accès à la liste de mot réalisé, le défilement reprend du haut vers le bas. Le mot « me » est sélectionné (Figure 6.4) et un caractère « fin de mot » (un espace) est ajouté. Notons que sur cet exemple, malgré la petite taille du mot « me », la saisie par la liste de mots apporte un gain. En effet, celle-ci nécessite 1 défilement + 1 clic pour accéder à la liste et encore 4 défilements + 1 clic (soit 5 défilement et 2 clics) ; la saisie de la lettre « e » et de l'espace aurait nécessité : 5 défilements + 1 clic (pou le « e »), une réorganisation du clavier puis 4 autres défilements + 1 clic pour l'espace (en 3e position après « me »), soit 9 défilements et 2 clics (et une réorganisation du clavier de lettres).

Cette interface telle qu'elle vient d'être présentée est actuellement testée au centre de rééducation et de réadaptation fonctionnelles de Kerpape par quatre personnes IMC.

Chapitre 7

Conclusion

Dans ce mémoire de thèse, nous avons présenté un système linguistique d'aide à la communication pour personne handicapée. Notre contribution se situe autant dans le domaine du handicap que dans celui du Traitement Automatique des Langues. Le système propose deux aides, l'une pour accélérer la vitesse de sélection des lettres sur le clavier simulé, l'autre pour économiser le nombre de saisies. Cette dernière intègre une prédiction de mot fondée sur un modèle de langage original que nous avons conçu et appelé n-chunks.

Concernant la vitesse de sélection sur clavier simulé, nous avons d'abord montré par une étude formelle que le système habituellement proposé par les logiciels du commerce, le défilement ligne/colonne sur un clavier standard (ordre azerty ou alphabétique), est clairement sous-optimal. L'étude menée a ainsi montré que le nombre de défilements pour atteindre une touche est autour de 6-7 en moyenne, alors que celui-ci pourrait être abaissé à une valeur inférieure à 4,5.

Pour améliorer encore la vitesse de sélection, la solution que nous proposons est d'intégrer une prédiction de lettre fondée sur le modèle statistique n-gramme. Nous avons commencé par étudier l'intérêt d'une telle prédiction. Avec le modèle n-gramme, les expériences ont montré que la lettre souhaitée apparaît en moyenne au troisième rang dans la liste des propositions. En présentant dans le clavier simulé les prédictions après chaque saisie, il est ainsi possible d'abaisser le nombre de défilements moyen à 3 en défilement linéaire.

Notre contribution réside également dans l'étude approfondie des capacités du modèle n-gramme pour la prédiction de lettre. Nous avons ainsi trouvé que la limite de ce modèle se situe, au mieux, à un rang moyen de 2,7. Dans nos expérimentations, la valeur maximale étudiée n = 5 obtient un rang moyen de 2,9 ce qui est donc proche de la valeur limite théorique, tandis que pour les valeurs n < 4 les performances sont nettement dégradées.

Pour valider ces résultats théoriques, nous avons ensuite réalisé une application avec un clavier simulé dynamique intégrant la prédiction de lettre. Cette application a été testée au centre de rééducation fonctionnelle de Kerpape avec des enfants IMC dans un contexte scolaire. L'expérience a été concluante et l'aide apportée par SibyLettre a été appréciée des utilisateurs. Le résultat principal de cette expérience est que les personnes utilisant SibyLettre composent plus de textes, ce qui montre que la saisie est rendue moins pénible avec notre système. SibyLettre est toujours utilisé à Kerpape et nous avons recueilli plus d'une centaine de textes. Il est intéressant de noter que certaines personnes ayant quitté Kerpape ont demandé à conserver cette aide.

Une difficulté de SibyLettre est la saisie de la première lettre. Les utilisateurs à Kerpape ont relevé ce que nous avions montré théoriquement : la première lettre des mots est nettement moins bien prédite (rang moyen de la lettre à 7 contre 3 globalement). Dans notre modélisation, nous avons fait l'hypothèse d'un contexte vide à chaque début de mot. Une première idée d'amélioration serait donc de remettre en cause cette hypothèse et d'étudier l'influence des caractères précédant un début de mot sur la prédiction. Une autre idée est donnée par les premières lettres délivrées par la prédiction en début de mot : « d, l » qui correspondent aux mots les plus fréquents « de », « le » et leurs dérivés. Il serait donc intéressant d'utiliser les prédictions de SibyMot pour améliorer la prédiction de lettre. Ceci constitue une première perspective pour SibyLettre : intégrer les prédictions de SibyMot et les combiner aux prédictions du n-gramme de lettre pour faire face aux mots inconnus.

La deuxième perspective pour nous proposons pour SibyLettre est l'adaptation à l'utilisateur. Nous souhaitons étudier à partir des textes recueillis si la prédiction peut être encore améliorée en actualisant les paramètres de SibyLettre avec les saisies de l'utilisateur.

La part la plus importante de ce travail de thèse réside cependant dans la conception du modèle de langage utilisé par la prédiction de mot. Dans le cadre de la modélisation probabiliste du langage et à l'image du Structured Language Model de Jelinek, nous proposons un modèle intégrant des connaissances syntaxiques. Le point fort de notre modèle et son originalité est de se fonder sur la notion de *chunk* utilisée en TAL robuste. En particulier, nous avons montré que pour la prédiction des têtes de *chunk* la prédiction était améliorée dans 11 % des cas. Nous avons également obtenu une économie de saisie de 57,1 % contre 43,9 % pour une prédiction utilisant uniquement les fréquences de mot hors contexte comme dans les logiciels du commerce.

Notons également que notre système qui possède son propre analyseur peut être utilisé comme tel, pour des tâches d'étiquetage grammatical ou de segmentation. En particulier, dans ce cadre, nous participons à la campagne d'évaluation EASY.

Si les résultats obtenus sont déjà satisfaisants, nous souhaitons encore améliorer le modèle. Dans SibyMot, la prédiction est réalisée en plusieurs étapes et chacune de ces étapes doit être étudiée plus en détail. Ceci est particulièrement vrai pour les prédictions des relations et des flexions que nous n'avons pas évaluées. Nous devons également tester l'hypothèse d'indépendance entre la prédiction des lemmes et des flexions pour la prédiction de mot.

La deuxième perspective pour SibyMot est l'adaptation à l'utilisateur. Nous sommes conscient que le corpus d'apprentissage reflète mal les usages de la langue courante. En utilisant les mêmes techniques que celles utilisées pour l'apprentissage des paramètres du modèle, il est tout à fait envisageable de prendre en compte les saisies de l'utilisateur. Cependant, compte tenu de la taille du corpus d'apprentissage, l'influence sur la prédiction risque d'être faible si les occurrences sont directement ajoutées à celles issues de l'apprentissage. Nous proposons donc d'utiliser un espace de paramètres séparé pour chaque utilisateur. Le problème revient alors à combiner les estimations issues de deux modèles et à faire évoluer de manière adéquate les facteurs de pondération.

Concernant l'évaluation de SibyMot par des utilisateurs, celle-ci est en cours au centre de Kerpape. Les conditions sont les mêmes que lors de l'expérimentation de SibyLettre. Afin d'améliorer la qualité de la saisie, nous ajouterons rapidement les deux fonctionnalités suivantes : d'abord la mise en majuscule du premier mot des phrases et ensuite le respect des conventions typographiques concernant le caractère « espace » (par exemple pas d' « espace » entre un mot et les ponctuations comme la virgule ou le point).

Notons également que, comme nous l'avons présenté dans le chapitre sur l'état de l'art des systèmes d'aide à la communication, le problème de la saisie sur clavier n'est pas exclusive au domaine du handicap. Nous pourrions donc envisager d'utiliser nos systèmes prédictifs dans les appareils comme les assistants numériques.

Enfin, pour conclure ce mémoire, nous pouvons annoncer que le modèle prédictif SibyMot va être commercialisé en France par la société MicroVocal, société spécialisée dans les aides techniques pour personnes handicapées. Dans le cadre de cette collaboration, il est également prévu de réaliser une version de SibyMot pour l'anglais. Le passage de SibyMot à l'anglais est pour nous une perspective très intéressante. En particulier, nous pourrons vérifier si notre approche est générique. En effet, le passage à l'anglais devrait nécessiter un nombre limité de modifications : adaptation partielle des jeu d'étiquettes (en ce qui concerne les classes des mots grammaticaux), révision de la méta-grammaire (deux cents règles « simples » en français). De plus, à l'inverse du français, nous bénéficierons des nombreuses ressources disponibles en langue anglaise pour l'apprentissage des paramètres de notre modèle.

Bibliographie

[Abeillé 1993]
Abeillé A. Les nouvelles syntaxes – Grammaires d'unification et analyse du français. Armand Colin, 1993.

[Abeillé 2000]
Abeillé A., Clément L., Kinyon A. Building a treebank for French. In Proceedings of the Second International Conference on Language Ressources and Evaluation, LREC 2000, pp 87-94, Athènes, Grèce.

[Abeillé 2000b]
Abeillé A., Rambow O. Tree adjoining grammars: formal, linguistic and processing issues. Stanford, CSLI, 2000.

[Abney 1991]
Abney S. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny (Eds.), Principle based parsing. Kluwer Academic.

[Abney 1996]
Abney S. Partial Parsing. In Tutorials of ANCL'94. Stuttgart. http://www.sfs.nphil.uni-tuebingen.de/abney/Papers.html

[Abraham 2000]
Abraham M. Reconstruction de phrases oralisées à partir d'une écriture pictographique, Handicap 2000, pp 151-156, Paris, 2000.

[Allen 1997]
Allen, J. *Natural Language Understanding*, Chapter VII. Benjamins Cummings editions, 1997.

[Alm 1988]
Alm N. Towards a Conversation Aid for Severely Disabled Non-Speaking People. PhD Thesis, University of Dundee, Dundee, Scotland, 1988.

[Alm 1989]
Alm N, Arnott J, Newell A. Database Design for Storing and Accessing Personal Conversational Material. In Proceedings of the 12th Annual Conference of RESNA. New Orleans, Louisiana, USA, pp 147-148, 1989.

[Alm 1992]
Alm N., Arnott J., Newell A. Prediction and Conversational Momentum in an Augmentative Communication System. Communications of the ACM, 35(5):46-57, 1992.

[Antoine 2003]
Antoine J.-Y., Goulian J., Villaneau J. Quand le TAL robuste s'attaque au langage parlé – Analyse incrémentale pour la compréhension de la parole spontannée. TALN'2003, Batz-sur-Mer, France, juin 2003.

[Baayen 1995]
Baayen R. H., Piepenbrock R., Gulikers L. The CELEX lexical database (release 2), CD-ROM. Linguistic Data Consortium, Philadelphia, PA.

[Baker 1979]
Baker J. K. Trainable grammars for speech recognition. In Proceedings of the 9th meeting of the Acoustical Society of America, 1979.

[Bellegarda 1997]
Bellegarda J. R. A Latent Semantic Analysis Framework for Large-Span Language modeling. Eurospeech'97, pp 1451-1454, Rhodes, Greece, 1997.

[Bimbot 1997]
Bimbot F., El-Bèze M., Jardino M. An alternative scheme for perplexity estimation. In Proceeding of the International Conference on Acoustics, Speech and Signal Processing, Munich, Germany, 1997.

[Binder 1988]
Binder K., Heermann D. W. Monte Carlo Simulation in Statistical Physics. Springer-Verlag. 1988.

[Boissière 1990]
Boissière P. VITIPI un système auto-organisationnel pour faciliter le dialogue écrit homme-machine. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1990.

[Boissière 2000]
Boissière P. VITIPI : Un système d'aide à l'écriture basé sur un principe d'auto-apprentissage et adapté à tous les handicaps moteurs. Handicap 2000, pp 81-86, Paris, 2000.

[Boissière 2001]
Boissière P. Comment VITIPI un système d'assistance à l'écriture pour les personnes handicapées peut offrir des propriétés intéressantes pour le TALN ? 8e conférence sur le Traitement Automatique des Langues Naturelles, pp 183-192, Tours, 2001.

[Boite 2000]
Boite R. Traitement de la parole. Presses polytechniques et universitaires romandes, 2000.

[Booth 1990]
Booth L., Beattie W., Newell A. I Know what you mean. Special Children (41) pp 26-27. 1990.

[Brophy 1991]
Brophy M., Alm N., Newell A, Arnott J. The Effect of a Predictive Communication Aid Employing Speech Acts on the Conversation of Non-Vocal Speakers. Speech Therapy in Practice, 162, Juillet 1991.

[Broumley 1990]
Broumley L., Cairns A., Arnott J. A Case Study in Applying Artificial Intelligence in a Personalised Communication Aid. In The Proceedings of the 1st ECART Conference, 14.1, Novembre 1990.

[Brown 1992]
Brown P., De Souza P. V., Mercer R. L., Della Pietra V. J., Lai J. C. Class-Based n-gram Models of Natural Language. Computational Linguistics 18(4), pp 467-479. MIT Press, December 1992.

[Brugnara 1996]
Brugnara F., Federico M. Techniques for approximating a trigram language model. In Proceeding of International Conference on Spoken Language Processing. Philadelphia, PA, 1996.

[Cantegrit 2001]
Cantegrit B., Toulotte J.-M. Réflexions sur l'aide à la communication des personnes présentant un handicap moteur. TALN 2001, vol 2, pp 193-202, Tours, 2001.

[Carlberger 1997]
Carlberger J. Design and Implementation of a Probabilistic Word Prediction Program. 1997.

[Cerf-Danon 1991]
Cerf-Danon H., El-Bèze M. Three different probabilistic language models: Comparison and combination. In Proceeding of the International Conference on Acoustics, Speech and Signal Processing, pp 297-300. Toronto, Canada, 1991.

[Charniak 1993]
Charniak E. Statistical Language Learning. MIT Press, Cambridge, 1993.

[Chanod 1995]
Chanod J.-P., Tapanainen P. Creating a tagset, lexicon and guesser for a French tagger. ACL SIGDAT workshop on “From texts to Tags: Issues In Multilingual Language Analysis”, pp 58-64, University College, Dublin, Ireland, 1995.

[Chanod 2001]
Chanod J.-P. Robust Parsing and Beyand. Robustness in Language and Speech Technology, pp 187-204, J.-C. Junqua and G. van Noord editions, Dordrecht, Nederland, 2001.

[Chen 1997]
Chen S., Rosenfeld R. Topic adaptation for language modelling using unnormalized exponential models. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 681-684, Seattle, WA, 1997.

[Chomsky 1957]
Chomsky N. Syntactic Structures. Mouton, La Haye, 1957.

[Clarkson 1997]
Clarkson P., Robinson A. Language model adaptation using mixtures and an exponentially decaying cache. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 799-802, Munich, Germany, 1997.

[Clément 2001]
Clément L. Construction et exploitation d’un corpus syntaxiquement annoté pour le français. Thèse de doctorat en Linguistique, École Doctorale Sciences du Langage, Université Paris 7, 2001.

[Colmerauer 1975]
Colmerauer A. Les grammaires de métamorphose. Rapport interne, Université Aix-Marseille. Repris dans Metamorphosis Gramamars, Natural Language Communication with computers, L. Bolc editions, Springer Verlag, 1978.

[Deligne 1996]
Deligne S. Modèles de sequences de longueurs variables : Application au traitement du langage écrit et de la parole. Thèse de doctorat, École Nationale Supérieure des Télécommunications, 1996.

[Demasco 1992]
Demasco P. W., McCoy K. F. Generating text from compressed input: An intelligent interface for people with severe motor impairments. Communications of the ACM, 35(5), pp 68-78, 1992.

[Deerwester 1990]
Deerwester S. Indexing by Latent Semantic Analysis. J. Am. Soc. Inform. Science. Vol 41, pp 391-407, 1990.

[Easton 1994]
Easton J., Grist E., O'Connel C. A Newly Developed Biofeedback Aid: Does This Inhibit the festinant Speech of Clients with Parkinson's Disease ? In Proceedings of the Sixth Biennial Conference of the International Society for Augmentative and Alternative Communication, Maastricht, Pays-Bas, pp 499-500, 1994.

[El-Bèze 1990]
El-Bèze M., Derouault A. M. A Morphological model for large vocabulary speech recognition. In Proceeding of the International Conference on Acoustics, Speech and Signal Processing, pp 577-580, 1990.

[Ejerhed 1996]
Ejerhed E. Finite State Segmentation of Discourse into Clauses. Proceedings of ECAI Workshop on Extended Finite State Model of Language, pp 24-33, A. Kornai Ed., 1996.

[Ferrand 1997]
Ferrand T. Coordination motrice inter-individuelle et loi de Fitts : Fonction d'échange vitesse-précision pour le mouvement de pointage dyade. In Cognito, 7, pp 25-32, 1997.

[Fillmore 1968]
Fillmore C. J. The case for case. In E. Bach and R. Harms editions, Universals in Linguistic Theory, pp 1-90. New York: Holt, Rinehart and Winston, 1968.

[Fitts 1954]
Fitts P. M. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, 47(6), pp 381-391. 1954.

[Gazdar 1979]
Gazdar G. Pragmatics, implicature, presupposition and logical form. Academic Press, New York, 1979.

[Gazdar 1985]
Gazdar G., Klein E., Pollum G., Sag I. Generalized Phrase Structure Grammar. Harvard University Press, 1985.

[Gee 1983]
Gee J., Gorsjean F. Performances structures: psycholinguistic and linguistic appraisal. Cognitive Psychology, 15, pp 411-458. 1983.

[Gillet 1998]
Gillet J., Ward W. A language model combining trigrams and stochastic context-free grammars. In Proceeding of International Conference on Spoken Language Processing, Sydney, Australia, 1998.

[Good 1953]
Good, I. J. The population frequencies of species and the estimation of population parameters. *Biometrika*, pp. 237-264, 1953.

[Goosse 2000]
Goosse A. Grevisse, le bon usage, 13ème édition, 5ème tirage. Éditions Duculot, Paris, 2000.

[Goulian 2002]
Goulian J. Stratégie d'analyse détaillée pour la Compréhension Automatique robuste de la Parole. Thèse de doctorat, Université de Bretagne Sud, 2002.

[Gumperz 1982]
Gumperz J. Discourse Strategies. Cambridge University Press, London, 1982.

[Harbusch 2003]
Harbusch K., Kühn M. Towards an Adaptative Communication Aid with Text Input from Ambiguous Keyboards. In Proceedings of the Demo Sessions of EACL'03, Budapest, April 12-17, 2003.

[Hirose 2001]
Hirose K., Minematsu N., Hashimoto Y., Iwano K. Continuous Speech Recognition of Japanese Using Prosodic Word Boundaries Detected by Mora Transition Modeling of Fundamental Frequency Contours. In Proceedings ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding, pp 61-66. Red Bank. 2001.

[Hunnicutt 1986]
Hunnicutt S. Lexical Prediction for a Text-to-Speech System in Communication and Handicap: Aspects of Psychological Compensation and Technical Aids, E. Hjelmquist & L.-G. Nilsson editions, Elsevier Science Publishers.

[Hunnicutt 1993]
Hunnicutt S., Carlson R., Ribe L. A choice-based large-vocabulary predictive writing aid. In Proceedings of the Annual Conference of RESNA '93, Juin 1993.

[Iyer 1994]
Iyer R., Ostendorf M., Rohlicek J. R. Language modeling with sentence-level mixtures. In Proceedings of the ARPA Human Language Technology Workshop, pp 82-86. Plainsbaro, NJ, 1994.

[Jardino 1993]
Jardino M., Adda G. Language modelling for csr of large corpus using automatic classification of words. In Proceeding of the European Conference on Speech Communication and Technology, pp 1191-1194, September 1993.

[Jardino 2000]
Jardino M., Beaujard C. Rôle du contexte dans les modèles « n-classes » – Application et évaluation sur MASK et RAILTEL. In K. Chibout, J. Mariani, N. Masson, F. Néel editions, Ressources et évaluation en ingénierie des langues, pp 379-389, De Boeck Université, Duculot, 2000.

[Jelinek 1976]
Jelinek F. Continuous speech recognition by statistical models. Proceedings of the IEEE, 1976.

[Jelinek 1990]
Jelinek F. Self-organized language modeling for speech recognition. Reading in Speech Recognition, pp 450-506. Edited by Alex Waibel and Kai-Fu Lee, Morgan Kaufmann, San Mateo California, 1990.

[Jelinek 2000]
Chelba C., Jelinek F. Structured language modeling. Computer Speech and Language 14, pp 283-332. 2000. http://www.ideallibrary.com

[Jelinek 2002]
Jelinek F., Xu P., Chelba C. A Study on Richer Syntactic Dependencies for Structured Language Modeling. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL). Pp 191-198, Philadelphia, 2002.

[Joshi 1987]
Joshi A. Introduction to Tree Adjoining Grammar. In A. Manaster-Ramer editions, The Mathematics of Language, J. Benjamins, 1987.

[Kaplan 1982]
Kaplan R., Bresnan J. LFG: a formal system for grammatical representation. J. Bresnan editions, 1982.

[Khudanpur 2000]
Khudanpur S., Wu J. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. Computer Speech and Language (14), pp 355-372, Academic Press, 2000. http://www.ideallibrary.com

[Kneser 1991]
Kneser R., Ney H. Forming word classes by statistical clustering for statistical language modeling. In Proceeding of the First International Conference on Quantitative Linguistics, éd. Köhler, pp 221-226. Trier, Germany, 1991.

[Kneser 1995]
Kneser R., Ney H. Improved clustering technique for class-based statistical language modelling. In Proceeding of the European Conference On Speech Communication and Technologie, pp 973-976. Berlin, Germany, 1993.

[Kneser 1996]
Kneser R. Statistical language modelling using a variable context. In Proceeding of International Conference on Spoken Language Processing. Philadelphia, PA, 1996.

[Knuth 1968]
Knuth D. E. The Art of Computer Programming. Addison-Wesley Publishing Company. 1968.

[Kuhn 1988]
Kuhn R. Speech recognition and the frequency of recently used words: a modified markov model for natural language. In Proceeding of COLING, pp 348-350. Budapest, Hungary, 1998.

[Kuhn 1990]
Kuhn R., DeMori R. A cache-based natural language model for speech recognition. IEEE Transtactions Pattern and Machine Intelligence 12(6), pp 570-582, 1990.

[Kühn 2001]
Kühn M., Garbe J. Predictive and highly ambiguous typing for a severely speech and motion impaired user. In C. Stephanidis, editor, Universal Access in Human-Computer Interaction, pp 933-937. Lawrence Erlbaum, Mahwah, NJ, 2001.

[Kupiec 1989]
Kupiec J. Probabilistic models of short and long distance word dependencies in running text. In Proceeding of the DARPA Workshop on Speech and Natural Language, edited by Kaufmann, pp 290-295, 1989.

[Kushler 1998]
Kushler C. AAC using a reduced keyboard. In CSUN Center of Disabilities. Online proceedings of the Technology and Persons with Disabilities Conference 1998. California State University, Nortridge, CA, 1998. http://www.dinf.ne.jp/doc/english/Us_Eu/conf/csun_98/csun98_140.htm

[Kudoh 2000]
Kudoh T., Matsumoto Y. Use of Support Vector Learning for Chunk Identification. In Proceedings of CoNLL-2000 and LLL-2000, pp 142-144, Lisbon, Portugal, 2000.

[Kuno 1962]
Kuno S., Oettinger A. A multiple path syntactic analyser. Information processing, 1962.

[Langlois 1999]
Langlois D., Smaïli K. A new based distance language model for a dictation machine : Application to Maud. In Proceeding of the European Conference On Speech Communication and Technologie. Budapest, Hungary, 1999.

[Le Pévédic 1997]
Le Pévédic B. Prédiction morphosyntaxique évolutive dans un système d'aide à la saisie de textes pour des personnes handicapées physiques. Thèse de doctorat, École doctorale Sciences pour l'ingénieur de Nantes, 1997.

[Lesher 2000]
Lesher G. W., Moulton B. J. A method for optimizing single-finger keyboards. In Proceedings of the RESNA 2000 Annual Conference, pp 91-93, 2000.

[Levinson 1974]
Levinson R. A Plato Reader. Boston : Houghton-Mifflin, 1974.

[Lin 1965]
Lin S. Computer solutions of the travelling salesman problem. Bell Systems Technical Journal, 44, 2245-2269. 1965.

[Mac Kenzie 1995]
Mac Kenzie I. S., Soukoref W. Theoretical upper and lower bounds on typing speeds using a stylus and keyboard. Behaviour & Information Technology, pp 375-379. 1995.

[Mac Kenzie 1999]
Mac Kenzie I. S., Zang S. X. The design and evaluation of a high-performance soft keyboard. In Proceedings of CHI'99: ACM Conference on Human Factors in Computing Systems (14) pp 25-31, 1999.

[Marcus 1995]
Marcus M., Santorini B., Marcinkiewicz M. Building a large annotated corpus of English: the Penn TreeBank. Computational Linguistics, 19, pp 313-330. 1995.

[Maurel 2001a]
Maurel D., Rossi N., Thibault N. HandiAS : un système multilingue pour l'aide à la communication des personnes handicapées. TALN 2001, vol 2, pp 203-212, Tours, 2001.

[Maurel 2001b]
Maurel D., Le Pévédic B. The syntactic prediction with Token Automata: Application to HandiAS system. Theoretical Computer Science, vol. 267, pp 121-129, 2001.

[Mc Coy 1995]
Mc Coy K. F., Demasco P. Some applications of natural language processing to the field of augmentative and alternative communication. In Proceedings of the IJCAI'95 Workshop on Developing AI Applications for Disabled People, pp 97-112, Montreal, Canada, 1995.

[Ménézo 1996]
Ménézo J., Genthial D., Courtin J., Dujardin D. La méthode des structures – Principes et mise en oeuvre dans CELINE, un système mulit-agent de détection et correction des erreurs lexicales et syntaxiques. Conférence TALN, Marseille, 1996.

[Ménier 2001]
Ménier G., Poirier F. Système adaptatif de prédiction de texte. TALN 2001, vol 2, pp 213-222, Tours, 2001.

[Mood 1974]
Mood A. M., Graybill F. A., Boes D. C. Introduction to the Theory of Statistics. Mc Graw Hill, Singapore, 1974.

[Morris 1975]
Morris R., Cherry L. Computer detection of typographic errors. IEEE Transactions on professional communication (18), 1975.

[Morris 1993]
Morris C., Booth L., Ricketts I., Alm N., Newell A. Evaluation of a Syntax-Driven Word Predictor for Children with Language Impairments. In Proceedings of the Annual Conference of RESNA '93, pp 423-425. 1993.

[Murray 1991]
Murray I., Arnott J., Alm N., Newell A. Communication System for the Disabled with Emotional Synthetic Speech Produced by Rule. In Proceedings of Eurospeech 91, 2nd European Conference on Speech Communications and Technology, pp 311-314. 1991.

[Ney 1994]
Ney H. Essen U., Kneser R. On structuring probabilistic dependences in stochastic language modelling. Computer Speech and Language, pp 1-38, vol. 8, 1994.

[Niesler 1996]
Niesler T. R., Woodland P. C. A variable-length category-based n-gram language model. In Proceedings of the International Conference on Audio, Speech and Signal Processing, 1996.

[Noirhomme-Fraiture 2000]
Noirhomme-Fraiture, M., Charrière C., Vanderdonckt J. A Laboratory of Ergonomic Analyses for Children Suffering form Cerebral Palsy, in Tools for Working with Guidelines. TFWWG 2000, pp 35-49, 2000.

[Pasero 1994]
Pasero R., Sabatier P. Composition de phrase assistée : Principes, Outils et Applications. Actes des Journées Traitement Automatique du Langage Naturel (TALN 94), pp 51-74, Marseille, 1994.

[Pastor 1998]
Pastor J., San-Segundo R., Pardo J.M. An asymmetric stochastic language model based on multi-tagged words. In Proceedings of International Conference on Spoken Language Processing, 1998.

[Pereira 1980]
Pereira F., Warren D. Definite clause grammars for natural language analysis: a survey of the formalism and a comparison with Augmented Transition Networks. Artificial Intelligence, 13:3, 1980.

[Pierrel 2000]
Pierrel J. M. Ingénierie des Langues. Éditions Hermès, octobre 2000.

[Pino 2000]
Pino, P. EDITH : Adaptation automatique du temps de défilement aux caractéristiques et intentions de l'utilisateur, Handicap 2000, pp 125-130, Paris, 2000.

[Poirier 1994]
Poirier F. Vers une communication naturelle homme-machine – Apports de la reconnaissance des formes et des méthodes connexionnistes. Rapport d'habilitation à diriger des recherches, Université Paris XI, informatique, 1994.

[Pollard 1994]
Pollard C., Sag I. Head-driven phrase structure grammar. CSLI series, University of Chicago Press, 1994.

[Preece 1994]
Preece J., Rogers Y., Sharp H., Benyon D., Holland S., Carey T. Human-Computer Interaction. Addison-Wesley ed, Wokingham, 1994.

[Premack 1974]
Premack, D., Premack A. Teaching visual language to apes and language-deficient persons. In R. Schiefelbusch and L. Lloyd editors, Language perspectives : Acquisition, retardation and interventions, pp 347-375. Baltimore : University Park Press, 1974.

[Rajman 1996]
Rajman M. Format de description lexicale pour le français. Partie 1 : Concepts généraux, réf. GRACE GTR-2-1.2, 1996.

[Rajman 1997]
Rajman M., Lecomte J., Paroubek P. Format de description lexicale pour le français. Partie 2 : Description morpho-syntaxique, réf. GRACE GTR-3-2.1, 1997.

[Reichman 1985]
Reichman R. Getting Computers to Talk Like You and Me – Discourse Context, Focus and Semantics (An ATN Model). MIT Press, Cambridge, Mass., 1985.

[Ricco 2001]
Ricco X., Dutoit T. Vers un logiciel multilingue et gratuit pour l'aide aux personnes handicapées de la parole : HOOK (une interface du projet W). TALN 2001, vol 2, pp 223-232, Tours.

[Richardet 1998]
Richardet, N. Composition de phrase assistée – Un système d'aide à la communication pour handicapés. Thèse de doctorat, Université de la Méditerranée, 1998.

[Rivenc 1979]
Rivenc P. Le français fondamental vingt-cinq ans après. Le français du monde (148), 1979.

[Rosenfeld 1992]
Rosenfeld R, Huang X. Improvement in stochastic language modeling. In Proceeding of the DARPA Workshop on Speech and Natural Language, pp 107-111. San Mateo, CA, February 1992.

[Rosenfeld 1994]
Rosenfeld R. Adaptive Statistical Language Modeling: A Maximum Entropy Approach. Pittsburgh, PA 15213, PhD Thesis, School of Computer Science Carnegie Mellon University, April 1994 .

[Roukos 1996]
Roukos S. Survey of the state of the Art in Human Language Technology (Chap. 1.6), Cambridge University Press, 1996. http://cslu.cse.ogi.edu/HLTsurvey/chlnode8.html

[Sabah 1988]
Sabah G. L'intelligence artificielle et le langage. Hermès, Paris, 1988.

[Scapin 1990]
Scapin, D. L. Des critères ergonomiques pour l'évaluation et la conception d'interfaces. Actes du XXVIème Congrès de la SELF, Montréal, 1990.

[Schadle 1999]
Schadle I., Antoine J.-Y., Memmi D. Connectionist language models for speech understanding : the problem of word order variations. Eurospeech'99, Budapest, Hungary, 1999.

[Schmid 1995]
Schmid H. Improvements in Part-Of-Speech Tagging with an Application to German. From texts to tags: Issues in multilingual language analysis. Proceedings of the EACL Sigdat Workshop, Dublin.

[Schukat 1995]
Schukat-Talamazzini E.G., Hendrych R., Kompe R., Niemann H. Permugram language models. In Proceeding of the European Conference On Speech Communication and Technologie, pp 1773-1776. Madrid, Spain, 1995.

[Shank 1977]
Shank R. Scripts, Plans, Goals and Understanding. New Jersey, Lawrence Erlbaum, 1977.

[Sleator 1991]
Sleator D., Temperley D. Parsing English with a Link Grammar. CMU-CS-91-196, États-Unis, 1991.

[Spargins 1965]
Spargins J. A note on the iterative application of bayes' rule. IEEE Transactions Information Theory (IT-11), pp 544-549. 1965.

[Tesnière 1959]
Tesnière L. Éléments de syntaxe structurale. Klincksiek, Paris, 1959.

[TextWare 1998]
TextWare Solutions. The Fitaly one-finger keyboard. 1998. http://www.twsolutions.com/domperignon/domperignon2.htm.

[Tillmann 1996]
Tillmann C., Ney H. Selection Criteria for Word Trigger Pairs in Language Modeling. In Grammatical Inference: Learning Syntax from Sentences. Third International Colloquium, ICGI'96 (Lecture Notes in Artificial Intelligence 1147), édité par Miclet et de la Higuera, pp 98-106. Montpellier, France, 1996.

[Tjong Kim Sang 2000]
Tjong Kim Sang E. F., Buchholz S. Introduction to the CoNLL-2000 Shared Task: Chunking. In Proceedings of CoNLL-2000 and LLL-2000, pp 127-132, Lisbon, Portugal, 2000.

[Toulotte 1986]
Toulotte J.-M. Development of a communication system for handicapped persons – Generation of system. Proceedings of the eighth Annual Conference of the IEEE Engineering in Medicine and Biology Society, pp 1829-1832, Dallas, Texas, November 1986.

[Vaillant 1997]
Vaillant P. PVI : Système de traduction d'icônes en langue – Interaction entre modalités sémiotiques : de l'icône à la langue. Thèse de doctorat en Sciences Cognitives, Université Paris-XI, Orsay, 1997.

[Vergne 2000]
Vergne J. Étude et modélisation de la syntaxe des langues à l'aide de l'ordinateur – Analyse syntaxique automatique non combinatoire. Thèse d'habilitation, Université de Caen, 2000.

[Witschel 1993]
Witschel P. Constructing linguistic oriented language models for large vocabulary speech recognition. European Conference on Speech Communication and Technology, pp 1199-1202, 1993.

[Witten 1991]
Witten, I. H., Bell, T. C. The zero-frequency problem: estimating the probabilities of novel events in adaptative text compression. *IEEE Trans. on Information Theory*, pp 1085-1094, July 1991.

[Woods 1970]
Woods W. Transition network grammars for natural language analysis. Communications of the ACM, 13(10):591-606, 1970.

[Wright 1994]
Wright J., Jones G., Lloyd-Thomas H. A Robust Language Model Incorporating a Substring Parser and Extended N-Grams. In Proceedings of ICASSP-94. Adelaide, Australie, pp 361-364. 1994.

[Zangari 1994]
Zangari C., Lloyd L., Vicker B. Augmentative and alternative communication: A historical perspective. Augmentative and Alternative Communication, 10 (1):105-160, Mars 1994.

[Zhai 2000]
Zhai S., Hunter M., Smith B. A. The Metropolis Keyboard – An Exploration of Quantitative Techniques for Virtual Keyboard Design. In Proceedings of ACM Symposium on User Interface Software and Technology (UIST 2000), pp 119-128, San Diego, California, November 2000.

[Zhang 1998]
Zhang S. X. A high performance soft keyboard for mobile systems, 1998, The University of Guelph. p 99.

[Zitouni 2000]
Zitouni I. Modélisation du langage pour les systèmes de reconnaissance de la parole destinés aux grands vocabulaires : application à MAUD. Thèse de Doctorat, Université Henry Poincarré, Nancy, 2000.