Construction et stratégie d'exploitation des réseaux de confusion en lien avec le contexte applicatif de la compréhension de la parole

Bogdan Minescu

ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THÈSE

présentée à l'Université d'Avignon et des Pays de Vaucluse
pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : Informatique

École Doctorale 380 « Sciences et Agronomie »
Laboratoire d'Informatique (EA 931)

Construction et stratégie d'exploitation des réseaux de confusion en lien avec le contexte applicatif de la compréhension de la parole

par
Bogdan MINESCU

Soutenue publiquement le 11 Décembre 2008 devant un jury composé de :

M.	Paul Deléglise	Professeur, LIUM, Le Mans	Rapporteur
M.	Kamel Smaïli	Professeur, LORIA, Nancy	Rapporteur
M.	Frédéric Bechet	Maître de conférence, LIA, Avignon	Examinateur
M.	Jean-François Bonastre	Maître de conférence, LIA, Avignon	Examinateur
M.	Géraldine Damnati	Ingénieur, France Télécom R&D, Lannion	Examinateur
M.	Yannick Esteve	Maître de conférence, LIUM, Le Mans	Examinateur
M.	Renato De Mori	Professeur, LIA, Avignon	Directeur de thèse

Laboratoire d'Informatique d'Avignon

Résumé

Cette thèse s'intéresse aux réseaux de confusion comme représentation compacte et structurée des hypothèses multiples produites par un moteur de reconnaissance de parole et transmises à un module de post-traitement applicatif. Les réseaux de confusion (CN pour Confusion Networks) sont générés à partir des graphes de mots et structurent l'information sous la forme d'une séquence de classes contenant des hypothèses de mots en concurrence. Le cas d'usage étudié dans ces travaux est celui des hypothèses de reconnaissance transmises à un module de compréhension de la parole dans le cadre d'une application de dialogue déployée par France Telecom. Deux problématiques inhérentes à ce contexte applicatif sont soulevées.
De façon générale, un système de dialogue doit non seulement reconnaître un énoncé prononcé par un utilisateur, mais aussi l'interpréter afin de déduire sons sens. Du point de vue de l'utilisateur, les performances perçues sont plus proches de celles de la chaîne complète de compréhension que de celles de la reconnaissance vocale seule. Ce sont ces performances que nous cherchons à optimiser. Le cas plus particulier d'une application déployée implique de pouvoir traiter des données réelles et donc très variées. Un énoncé peut être plus ou moins bruité, dans le domaine ou hors-domaine, couvert par le modèle sémantique de l'application ou non, etc. Étant donnée cette grande variabilité, nous posons la question de savoir si le fait d'appliquer les mêmes traitements sur l'ensemble des données, comme c'est le cas dans les approches classiques, est une solution adaptée. Avec cette double perspective, cette thèse s'attache à la fois à enrichir l'algorithme de construction des CNs dans le but d'optimiser globalement le processus de compréhension et à proposer une stratégie adéquate d'utilisation des réseaux de confusion dans le contexte d'une application réelle.
Après une analyse des propriétés de deux approches de construction des CNs sur un corpus de données réelles, l'algorithme retenu est celui du "**pivot**". Nous en proposons une version modifiée et adaptée au contexte applicatif en introduisant notamment un traitement différencié des mots du graphe qui privilégie les mots porteurs de sens.
En réponse à la grande variabilité des énoncés à traiter dans une application déployée, nous proposons une stratégie de décision à plusieurs niveaux qui vise à mieux prendre en compte les spécificités des différents types d'énoncés. Nous montrons notamment qu'il est préférable de n'exploiter la richesse des sorties multiples que sur les énoncés réellement porteurs de sens. Cette stratégie permet à la fois d'optimiser les temps de calcul et d'améliorer globalement les performances du système.

Abstract

The work presented in this PhD deals with the confusion networks as a compact and structured representation of multiple aligned recognition hypotheses produced by a speech recognition system and used by different applications. The confusion networks (CN) are constructed from word graphs and structure information as a sequence of classes containing several competing word hypothesis. In this work we focus on the problem of robust understanding from spontaneous speech input in a dialogue application, using CNs as structured representation of recognition hypotheses for the spoken language understanding module. We use France Telecom spoken dialogue system for customer care. Two issues inherent to this context are tackled.
A dialogue system does not only have to recognize what a user says but also to understand the meaning of his request and to act upon it. From the user's point of view, system performance is more accurately represented by the performance of the understanding process than by speech recognition performance only. Our work aims at improving the performance of the understanding process. Using a real application implies being able to process real heterogeneous data. An utterance can be more or less noisy, in the domain or out of the domain of the application, covered or not by the semantic model of the application, etc. A question raised by the variability of the data is whether applying the same processes to the entire data set, as done in classical approaches, is a suitable solution. This work follows a double perspective : to improve the CN construction algorithm with the intention of optimizing the understanding process and to propose an adequate strategy for the use of CN in a real application.
Following a detailed analysis of two CN construction algorithms on a test set collected using the France Telecom customer care service, we decided to use the "pivot" algorithm for our work. We present a modified and adapted version of this algorithm. The new algorithm introduces different processing techniques for the words which are important for the understanding process.
As for the variability of the real data the application has to process, we present a new multiple level decision strategy aiming at applying different processing techniques for different utterance categories. We show that it is preferable to process multiple recognition hypotheses only on utterances having a valid interpretation. This strategy optimises computation time and yields better global performance.

Table des matières

Abréviations utilisées

CER	*Concept Error Rate*
CN	*Confusion Networks*
DCT	*Discrete Cosine Transform*
DET	*Detection Error Tradeoff*
DM	*Dialog Manager*
FA	*Fausse Alarme*
FR	*Faux Rejet*
FSM	*Finite State Machine*
HMM	*Hidden Markov Model*
IER	*Interpretation Error Rate*
MAP	*Maximum a posteriori*
MFCC	*Mel Frequency Cepstrum Coefficients*
NCE	*Normalized Cross Entropy*
OOV	*Out of Vocabulary*
PER	*Position-independent Error Rate*
SER	*Sentence Error Rate*
SLU	*Spoken Language Understanding*
SPR	*Speech Repair*
SRAP	*Système de Reconnaissance Automatique de la Parole*
SVM	*Support Vector Machine*
WER	*Word Error Rate*

Introduction

Pouvoir communiquer de manière orale avec une machine est un rêve qui est né avec l'apparition des premiers ordinateurs personnels dans les années 60. La communauté scientifique travaille depuis sur le développement des interfaces qui permettent une interaction la plus naturelle possible. Pour cela l'utilisateur doit pouvoir s'exprimer de manière libre et sans contrainte particulière. Les systèmes de dialogue homme-machine ont eu un regain d'intérêt ces dernières années notamment dû aux avancées dans le domaine de la compréhension de la parole spontanée (langage naturel). Car ces systèmes doivent non seulement reconnaître un énoncé prononcé par un utilisateur, mais aussi l'interpréter afin de déduire son sens. Cette phase est très importante car l'interprétation de l'énoncé déterminera la réponse donnée à l'utilisateur par le système. La tâche de compréhension est d'autant plus complexe qu'elle doit traiter des entrées bruitées, le bruit provenant d'une part des erreurs de reconnaissance et, d'autre part, des disfluences caractéristiques de la parole spontanée (les répétitions, les hésitations, les reprises, les erreurs grammaticales, etc.).

Contexte applicatif

De nos jours, de plus en plus d'applications de dialogue sont déployées pour le grand public. Ces applications fournissent généralement un service (information sur les horaires de train/avion, recherche des restaurants, service client, etc.) sur un domaine délimité pour lequel certaines restrictions, comme un vocabulaire limité ou une sémantique restreinte, sont imposées. Ces restrictions sont notamment dues à l'état de l'art des techniques de reconnaissance vocale et de compréhension de la parole spontanée.
Les travaux de cette thèse se placent dans le contexte d'une telle application de dialogue. Le contexte applicatif est défini par le service 3000, une application acceptant la parole spontanée, déployée par France Telecom, qui permet aux utilisateurs d'obtenir divers renseignements concernant leur ligne fixe, de souscrire ou d'accéder à des service dédiés.
L'utilisation de données réelles, collectées à partir du service 3000, soulève un certain nombre de problèmes. Une analyse de ces données, détaillée dans le chapitre 4, permet de distinguer deux types d'énoncés que le système doit traiter mais qui ne sont pas des requêtes valides (i.e. pas dans le domaine de l'application). Tout d'abord, l'utilisation

du service dans des conditions réelles implique que l'environnement de l'utilisateur est plus ou moins bruyant. Malgré l'utilisation d'un module de détection Bruit/Parole, le système est amené à traiter des énoncés ne contenant que du bruit. L'utilisateur peut aussi avoir certaines réactions qui font que les requêtes adressées au système sont considérées hors du domaine de l'application. Par exemple, l'utilisateur se parle à lui même ou à une tierce personne, s'énerve ou peut même insulter le système. Nous appelons ces énoncés *commentaires* (ou parole hors-domaine).
Un premier problème qui se pose est lié à la détection des ces énoncés. Des sous-modèles spécifiques sont utilisés à cet effet. Nous montrons également que la présence de ces énoncés et leur annotation pose un problème lors de l'évaluation des performances de reconnaissance. Une méthode de normalisation est proposée au chapitre 4.
Du point de vue fonctionnel du système de dialogue, ces deux types d'énoncés, que l'on qualifie de non-valides, doivent être rejetés car ils ne sont pas couverts par le domaine de l'application et leur interprétation peut mener le dialogue dans une mauvaise direction.

Objet de la thèse

Une grande partie des systèmes de dialogue utilise la meilleure solution fournie par le module de reconnaissance pour interpréter le message de l'utilisateur. L'utilisation d'un espace d'hypothèses plus grand en sortie de reconnaissance, notamment le graphe de mots, permet de maintenir plus long temps actives des hypothèses multiples et d'améliorer l'interprétation en retardant les décisions. Toutefois, le traitement de ces graphes est très coûteux en termes de temps de calcul et de ressources mémoire du fait de leur taille et de leur forte redondance. En 2000, une nouvelle structure appelée réseau de confusion (CN) a été proposée dans (Mangu et al., 1999). Cette structure, obtenue à partir des graphes de mots grâce à un nouveau critère de minimisation du taux d'erreur mot, peut être vue comme une succession de "classes" contenant chacune une ou plusieurs hypothèses de mots en compétition. Les CNs sont des structures très compactes et très peu redondantes ce qui les rend potentiellement bien adaptées vis à vis des modules applicatifs auxquels ils sont fournis en entrée. Par ailleurs, les probabilités *a posteriori* associées aux mots sur les transitions des CNs constituent une information pertinente pour l'estimation d'une mesure de confiance.
L'analyse détaillée de l'algorithme proposé dans (Mangu et al., 1999) montre une complexité élevée et l'application de l'algorithme nécessite un élagage drastique du graphe de mots au préalable pour être exploité. Malgré cela, l'algorithme reste difficilement utilisable dans un contexte temps-réel. Nous avons alors choisi d'utiliser l'algorithme proposé dans (Hakkani-Tur et Riccardi, 2003), appelé algorithme du "**pivot**", qui a non seulement une complexité fortement réduite, mais aussi des performances équivalentes en termes de taux d'erreur mot à celles de l'algorithme précédent.
Du point de vue de l'utilisateur, le processus de reconnaissance vocale et le processus d'interprétation sont transparents. Ainsi, pour une erreur du système, ce ne sont pas directement les erreurs de reconnaissance que l'utilisateur perçoit mais bien leur impact sur l'étape d'interprétation. L'objectif des travaux de cette thèse est d'étudier si le

recours aux CNs en tant que structure intermédiaire entre les étapes de reconnaissance et d'interprétation permet d'améliorer globalement le processus de compréhension du point de vue de l'utilisateur. Pour cela nous suivons deux pistes : d'une part réaliser une meilleure intégration des caractéristiques d'un système de dialogue en langage naturel dans la génération des CNs, et d'autre part, trouver des techniques et des stratégies permettant une meilleure prise en compte, dans le processus de reconnaissance et d'interprétation des énoncés, des spécificités liées à la grande variabilité des données réelles utilisées

En ce sens, cette thèse s'attache, à la fois, à enrichir l'algorithme de construction des CNs et à proposer une stratégie adéquate d'utilisation des CNs dans un contexte d'application réelle.

Construction des CNs Le processus d'interprétation se déroule en deux étapes successives. Une première étape consiste en une analyse des mots en concepts. Un concept est défini par un ensemble de mots ou de séquences de mots ayant le même sens. Une analyse sémantique des séquences de concepts permet ensuite le calcul d'une interprétation. Seule une partie des mots du vocabulaire sont porteurs de sens pour l'application et correspondent à des concepts. Les mots restants ne sont pas importants du point de vue de l'interprétation mais sont néanmoins pris en compte dans la phase de reconnaissance.

Dans sa formulation théorique, l'algorithme de génération des CNs correspond à un critère de minimisation du taux d'erreur mot, alors que dans une application de dialogue on cherche à minimiser l'erreur au niveau interprétation. Dans cette étude, nous proposons d'adapter l'algorithme du "**pivot**" afin d'optimiser plus généralement les performances du processus complet d'interprétation. Dans cette optique, l'algorithme proposé traite en priorité les mots porteurs de sens pour l'application. Si les CNs présentent une structure compacte et peu redondante en terme de nombre de transitions, leur complexité en terme de nombre de chemins peut dépasser largement celle des graphes dont ils sont extraits. Cette explosion combinatoire peut compromettre leur utilisation dans des opérations de composition avec des automates à états fini par exemple, et toute autre type d'opérations classiques sur les automates. Nous avons donc cherché à réduire cette complexité et proposons un algorithme de post traitement des CNs. Toujours dans l'optique de concentrer les hypothèses sur les informations pertinentes pour les modules applicatifs, cet algorithme de *parsing* des CNs vise à réduire la taille des CNs tout en ne gardant que l'information porteuse de sens pour une application en aval.

Exploitation des CNs Comme nous l'avons expliqué, le traitement de données réelles implique de devoir prendre en compte certains types d'énoncés qui, du point de vue du système, sont à rejeter. Les différentes études que nous avons réalisées ont montré que l'utilisation des CNs pour la détection et le rejet de ces énoncés n'est pas adaptée. De plus la génération des CNs pour ces énoncés est très coûteuse en termes de temps de calcul car les graphes correspondant sont très bruités et donc très "volumineux". Pour pallier cela nous proposons de rejeter ces énoncés dès la première passe de reconnaissance. Les CNs sont générés sur les énoncés considérés valides. Concernant le processus d'interprétation, il est, dans un premier temps, appliqué sur la meilleure solution du CN pour tous les CNs générés.

Au delà de cette approche séquentielle, nous proposons également une stratégie qui permet l'utilisation de la recherche intégrée sur le graphe de mots et sur les CNs dans le processus d'interprétation [1]. Cette stratégie permet d'introduire une étape supplémentaire de rejet des énoncés valides mais qui ne sont pas couverts par le modèle sémantique de l'application (aucune interprétation n'est trouvée). La probabilité *a posteriori* des mots est utilisée en tant que mesure de confiance pour améliorer les performances globales des deux stratégies.

Extension à d'autres domaines Si cette thèse se déroule principalement dans le cadre de la compréhension de la parole pour un système de dialogue, nous montrons que l'algorithme de génération des CNs proposé ainsi que l'algorithme de *parsing* sont facilement adaptables pour un autre domaine d'application. Des travaux préliminaires ont été conduits en traduction automatique de la parole. Ils sont décrits en annexe. Les méthodes proposées ont pour point commun de chercher à rendre plus robuste la reconnaissance de mots ou expressions porteurs d'information et pourrait dans ce sens s'appliquer à d'autres contextes applicatifs.

Organisation du document

Ce document est divisé en trois parties. La première partie, constituée de deux chapitres présente le contexte et le principe d'une application de reconnaissance de la parole continue ainsi qu'un état de l'art sur l'utilisation des mesures de confiance dans ce contexte. Une première contribution est aussi présentée :

- Le premier chapitre détaille le principe d'une application de reconnaissance de la parole continue et précise le role des modèles acoustiques et de langage. Le modèle *n-gramme* à base de classes ainsi que l'imbrication de plusieurs modèles de langage est présenté. Les différentes structures utilisées pour regrouper les solutions multiples en sortie du décodeur sont également présentées.
- Le second chapitre présente un état de l'art des mesures de confiance basées sur des paramètres prédictifs ainsi que des méthodes de combinaison de plusieurs paramètres. L'estimation de la probabilité *a posteriori* sur différentes structures (*N best*, graphes de mots, CNs) ainsi que son utilisation en tant que mesure de confiance sont également présentées. Le chapitre 2 contient également une contribution algorithmique relative à l'estimation de la probabilité *a posteriori* dans un graphe de mots. Cette méthode introduit une normalisation par transition dans le calcul de la probabilité *a posteriori* et est adaptée d'après une méthode de normalisation par trame proposé dans (Rabiner, 1989).

La seconde partie est divisée en trois chapitres et présente le contexte applicatif et les problématiques abordées dans cette thèse.

1. Le processus d'interprétation ne traite plus, de manière séquentielle, la meilleure séquence de mots pour obtenir une interprétation, mais un espace d'hypothèse plus grand, comme le graphe de mots. Ainsi, à travers des compositions avec des Automates à États Finis, représentant les différents modèles utilisés par l'analyse sémantique, les graphes de mots sont transformés successivement afin d'obtenir une interprétation. De cette façon la prise de décision d'une meilleure hypothèse ne se fait plus au niveau mot mais au niveau interprétation. La recherche intégrée est expliquée plus en détail dans la section 3.3.3.

- Le chapitre 3 introduit l'application de dialogue utilisée (le service 3000) et décrit le fonctionnement général de chaque module du système de dialogue sous-jacent. Le fonctionnement du module de compréhension de la parole est ensuite détaillé ainsi que son implémentation dans le service 3000. Une métrique d'évaluation au niveau interprétation, le taux d'erreur interprétation *IER*, est également introduite.
- Le chapitre 4 détaille les différentes problématiques posées par l'utilisation des données réelles de parole continue. Les modèles utilisés par le service 3000 ainsi que les corpus utilisés dans cette thèse sont présentés. Une méthode de normalisation des données pour l'évaluation du *WER* est également proposée.
- Le chapitre 5 introduit le principe de génération des CNs et présente un état de l'art des algorithmes de génération existant. Une comparaison des performances des deux principaux algorithmes est également réalisée et nous motivons le choix de l'algorithme du "**pivot**" utilisé pour la suite de ces travaux.

La troisième partie est divisée en deux chapitres et présente les principales contributions de cette thèse :

- Le chapitre 6 présente une analyse détaillée des performances de l'algorithme de génération choisi, appelé algorithme du "**pivot**", sur les différentes catégories d'énoncés. Les modifications apportées à cet algorithme dans le but de construire des CNs qui optimisent plus généralement les performances du processus complet d'interprétation sont détaillées. Le gain de performance obtenu suite à l'utilisation des probabilités *a posteriori* en tant que mesure de confiance est présenté. Les procédures d'élagage et de *parsing* des CNs sont également détaillées dans ce chapitre.
- Le chapitre 7 présente une stratégie de décision basée sur l'utilisation de la meilleure solution de la première passe de reconnaissance pour le rejet des énoncés non-valides et de la meilleure solution des CNs pour l'interprétation des messages valides. Une deuxième stratégie basée sur l'utilisation de la recherche intégrée sur les graphes de mots et sur les CNs dans le processus d'interprétation est également présentée dans ce chapitre.

Un bilan clôt ce document en évoquant les contributions de la thèse ainsi que les perspectives concernant les différents travaux.

Première partie

Contexte général - la reconnaissance de la parole continue spontanée

Chapitre 1

Reconnaissance de la parole

Sommaire

Un système de reconnaissance de la parole vise à transformer un signal acoustique reçu en entrée en une séquence de mots la plus proche possible de celle prononcée par l'utilisateur.
Soit $X = x_1, ..., x_T$ une séquence d'observations acoustiques représentant le signal de parole ; le système de reconnaissance de la parole recherche la séquence de mots la plus vraisemblable par rapport au signal acoustique en entrée. Il s'agit donc de trouver la séquence de mots $\hat{W}$ qui maximise la probabilité *a posteriori* $P(W|X)$ de la séquence de mots sachant le signal acoustique. Ceci revient à résoudre l'équation suivante :

$$\hat{W} = \underset{W}{argmax} P(W|X) \tag{1.1}$$

L'utilisation de cette formule est très difficile à cause de l'estimation de la probabilité $P(W|X)$. Cette difficulté réside dans la grande variabilité dans l'ensemble de départ des observations acoustiques. Il est plus facile d'estimer la probabilité d'avoir une certaine séquence d'observations acoustiques X sachant une séquence de mots W. La formule de Bayes permet de décomposer le terme $P(W|X)$:

$$\hat{W} = \underset{W}{argmax} \frac{P(W) \cdot P(X|W)}{P(X)} \tag{1.2}$$

Le problème est réduit à un problème d'optimisation par rapport à la séquence de mots W. La probabilité de la séquence d'observations acoustiques $P(X)$ ne dépend pas de la séquence de mots W ce qui ramène le problème d'optimisation à :

$$\hat{W} = \underset{W}{argmax} P(W) \cdot P(X|W) \tag{1.3}$$

Un système de reconnaissance de la parole a pour but de trouver la séquence de mots la plus vraisemblable par rapport au message prononcé par le locuteur. Pour ce faire, le système utilise différents modules et modèles pour analyser et décoder le signal acoustique reçu.

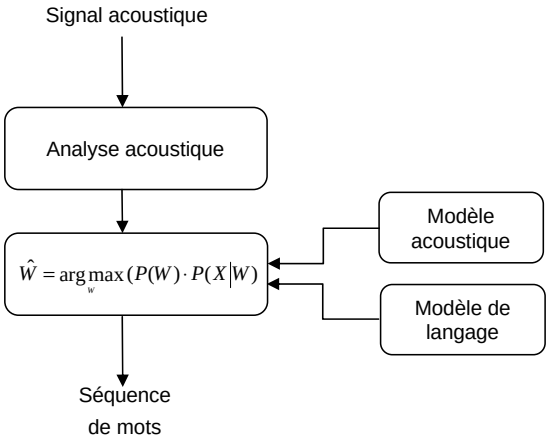

FIGURE *1.1* – Système de reconnaissance de la parole

Un système de reconnaissance de la parole (cf. figure 1.1) est composé d'un module d'analyse acoustique, présenté dans la section 1.1, suivi d'un décodeur de parole qui, à l'aide du modèle acoustique, présenté dans la section 1.2, et du modèle de langage, présenté dans la section 1.3, doit trouver une solution pour le problème d'optimisation de l'équation 1.3. La problématique de la combinaison des modèles de langage et acoustiques est traitée dans la section 1.4. L'espace de recherche utilisé par le décodeur et les différentes sorties possibles sont présentés dans la section 1.5 et les méthodes d'évaluation d'un système de reconnaissance sont détaillées dans la section 1.6.

1.1 Analyse acoustique

Le module d'analyse acoustique transforme le signal de parole en une séquence de vecteurs de coefficients qui est fournie en entrée du décodeur de parole. Les vecteurs de coefficients sont censés éliminer toute information qui n'est pas importante pour la reconnaissance comme les caractéristiques du locuteur (homme, femme), la réponse fréquentielle du canal ou toute sorte de bruit.
Le signal de parole est divisé en fenêtres temporelles (trames) et chaque portion du signal est alors analysée ; un vecteur de coefficients est le résultat de l'analyse de chaque portion. Afin de compenser dans une certaine mesure la forte pente spectrale du signal de parole dans le cas des voyelles et pour augmenter l'énergie du signal dans les hautes fréquences (spécialement pour les consonnes), le signal passe tout d'abord par un filtre de pré-accentuation de premier ordre de type passe-haut. Pour diviser les signal de parole en trames nous utilisons des fenêtres de Hanning (des fenêtres de Hamming peuvent aussi être utilisées). La longueur de la fenêtre se situe habituellement dans un intervalle de 10-32 ms et les fenêtres successives ont un facteur de recouvrement de 40-60%. Si ce recouvrement n'existait pas on perdrait les informations contenues aux bords des fenêtres. Dans nos travaux, la longueur de la fenêtre de Hanning utilisée est de 32 ms et le recouvrement de 16ms (50%) ce qui donne un vecteur de 256 points avec un recouvrement de 128 points (pour un signal échantillonné à 8kHz).
Il existe plusieurs méthodes pour analyser un signal de parole. Dans les année 70 la méthode la plus populaire était l'analyse linéaire prédictive (*linear predictive analysis*) (Makhoul, 1975) et les coefficients cepstraux LPCC (*Linear Prediction Cepstrum Coefficient*) associés (Rabiner et Juang, 1993). A partir du milieu des années 80, la représentation standard du signal de parole utilisée repose sur les *Mel-Frequency Cepstral Coefficients* (MFCC) (Davis et Mermelstein, 1980). Dans nos travaux nous utilisons les MFCC et le processus de calcul de ces coefficients est décrit ci-dessous.
Le spectre du signal de parole est calculé pour chaque trame à l'aide de la FFT (la densité spectrale est égale au carré du module de la transformé de Fourrier) ; ensuite le spectre est filtré à l'aide d'une série de filtres triangulaires passe-bande espacés de manière égale sur l'échelle de Mel. La liaison entre cette échelle et l'échelle linéaire utilisé habituellement (Hz) est donnée par l'équation : $f_{Mel} = 2595 log_{10}(1 + f_{Hz}/700)$. Dans l'échelle linéaire (Hz), les filtres sont espacés de manière linéaire jusqu'à 1kHz et ensuite la fréquence centrale et la bande passante augmentent de façon logarithmique. Le logarithme de l'énergie du signal dans chaque filtre est ensuite calculé et suite à une transformé en cosinus discret (DCT) de ces logarithmes on obtient les MFCC. La formule de calcul des MFCC à partir des logarithmes de l'énergie du signal E_k est la suivante (Davis et Mermelstein, 1980) :

$$MFCC_i = \sum_{k=1}^{N} E_k cos(i\pi \frac{k - \frac{1}{2}}{N}) \tag{1.4}$$

où N est le nombre de filtres (habituellement entre 20 et 25, nous utilisons 24 filtres) et i varie de $1, \ldots, M$, avec M étant le nombre de coefficients calculés.
La DCT a la propriété de rendre les coefficients quasiment non-corrélés. Il est alors possible de ne retenir que quelques termes de la transformé en cosinus pour représenter la

variation des logarithmes de l'énergie. Le résultat est une représentation compacte du signal de parole où seulement 8 valeurs suffisent ; le coefficient d'ordre 0 qui est lié à l'énergie de la trame est souvent remplacé par une autre mesure d'énergie. Les paramètres ainsi obtenus forment ce qu'on appel le vecteur statique.
Du fait du recouvrement des trames, les vecteur de coefficients sont sensiblement corrélés alors que la modélisation acoustique utilisée part de la prémisse que les vecteurs sont indépendants. Afin de prendre en compte les corrélations temporelles qui peuvent exister entre les coefficient cepstraux et rendre ainsi mieux compte de leur dynamique on rajoute les estimés de la dérivé première et seconde (les coefficients dynamiques) du vecteur statique. Cette idée a été proposé pour la première fois dans (Furui, 1986). Les dérivés sont calculées par régression sur quelques trames adjacentes.
Le vecteur de coefficients qui décrit chaque trame du signal de parole est donc formé du vecteur statique (MFCC et l'énergie) et de leurs dérivés premières et secondes. Ainsi, chaque trame du signal est représentée par un vecteur de 27 coefficients.

1.2 Modélisation acoustique

Pour une personne, prononcer le même mot plusieurs fois ne produit pas forcément le même résultat acoustique. La durée du signal, mais aussi la variabilité de la prononciation ne sont pas forcément les mêmes. Ces différences sont encore plus accentuées entre différentes personnes. Une modélisation statistique des différentes prononciations a été proposée au milieu des années 1970. Elle n'a vraiment été adoptée que dans la décennie suivante. Les modèles de Markov cachés (*Hidden Markov Models* - HMM) sont maintenant largement utilisés dans la modélisation acoustique parce qu'ils peuvent aisément prendre en compte des durées et de prononciations différentes pour un même mot. Leur utilisation s'est étendu) d'autres domaines, par exemple la reconnaissance des séquences du génome ou de l'écriture manuscrite. Par ailleurs, les modèles de Markov cachés et leurs applications ont été largement décrits, par exemple dans (Jouvet, 1988; Rabiner, 1989).
La modélisation acoustique d'une séquence de mots W est obtenue par une décomposition successive de celle-ci. Elle est ainsi séparée en mots qui sont à leur tour séparés en unités phonétiques (phonèmes, triphones, diphones, allophones). Chaque unité phonétique est représentée par un HMM (Rabiner et Juang, 1993), et la reconnaissance d'un signal de parole consiste à choisir la séquence de modèles qui attribue la vraisemblance la plus forte au signal observé.
Un HMM est un automate à N états $s_1, \ldots, s_N$ inter-connectés par de transitions, avec une probabilité attaché à chaque transition. Un HMM peut être vu comme un modèle où, à chaque trame, un symbole (une observation) est produit (émis) avec une probabilité donnée. Pendant le décodage, le processus émet un symbole avec une probabilité donnée. L'émission des observations peut être attachée soit aux états soit aux transitions. Ces deux points de vue sont équivalents (Jelinek, 1997), mais dans la littérature l'émission est souvent attachée aux états. Dans le cadre de nos travaux, l'émission est attachée aux transitions ce qui est plus adapté à l'utilisation des FSN (*Finite State Network*) dans le processus de reconnaissance (les entrées sont les vecteurs de coefficients

et les sorties, des séquences de mots).
La topologie principalement employée dans la littérature est un modèle gauche-droit d'ordre 1, dit de Bakis. Ceci implique donc que la probabilité de passer dans un état j dépend uniquement de l'état précédent. Le modèle est aussi supposé stationnaire, les probabilités de transition d'un état à l'autre ne dependent pas du temps. On a :

$$P(s_{t+1} = j | s_t = i) = a_{ij} \tag{1.5}$$

avec a_{ij} qui vérifie la relation suivante :

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{1.6}$$

On définit aussi la probabilité d'occupation initiale des états $P(s_1 = i) = \pi_i$ qui vérifie la relation :

$$\sum_{i=1}^{N} \pi_i = 1 \tag{1.7}$$

Comme nous l'avons mentionné, l'émission des observations peut être attachée soit aux états soit aux transitions. Lorsque le vecteur d'observations est à valeurs discrètes, on utilise une vraie valeur de probabilité pour l'émission. S'il s'agit des vecteurs à valeurs continues, comme dans note cas, une densité de probabilité est utilisée qui exprime la vraisemblance entre l'observation émise et le vecteur de coefficients. La densité de probabilité le plus souvent utilisé (dans la reconnaissance vocale), employée aussi dans notre système, est un mélange de densités gaussiennes. Le mélange peut être constitué d'une fonction (mono gaussienne) et jusqu'à un nombre maximal de gaussiennes. Plus le nombre est grand, meilleure (plus détaillée) sera la modélisation, mais le problème de l'estimation d'un grand nombre de paramètres à partir d'un volume de données restreint limite le nombre de gaussiennes utilisées afin d'obtenir une bonne estimation. La densité de probabilité d'émission d'une observation dans l'état i s'écrit :

$$b_i(x_t) = \mathcal{N}(x_t, \mu_i, \Sigma_i) \tag{1.8}$$

Un HMM est donc caractérisé par le vecteur constitué des probabilités initiales π_i, la matrice des probabilités de transition a_{ij} et la matrice des densités de probabilité d'émission $b_i(x_t)$. Les séquences de transition ne sont pas accessibles, seulement les observations émises le sont ; pour cela les modèles sont dits "cachés".

La phase d'apprentissage permet d'estimer les paramètres du modèle acoustique grâce à un corpus d'apprentissage. Il s'agit d'estimer les probabilités des transitions ainsi que les paramètres des densités d'observation associées aux états (les vecteurs de moyennes et les matrices de covariance d'un ensemble de gaussiennes). Généralement, les algorithmes EM (*Expectation-Maximization*), introduit dans (Dempster et al., 1977), et Baum-Welch (Rabiner et Juang, 1993), sont utilisés pour réaliser l'apprentissage des HMM. L'apprentissage des modèles acoustiques nécessite un volume de données assez important, la transcription du signal de parole étant aussi nécessaire. La première étape

consiste en la phonétisation de la transcription. L'étape suivante consiste en l'application de l'algorithme d'apprentissage. Ceci peut se faire de manière supervisée ou non. Pour un algorithme supervisé, les mots (ou les phonèmes) sont alignés sur le signal de parole en leur faisant correspondre un nombre de trames. Ainsi, lors de l'apprentissage, on se sert de cette information afin de contraindre la recherche du chemin optimal dans ce sous-modèle. Dans le cas non-supervisé tout le modèle est accessible. Dans la pratique, l'apprentissage des chaînes de Markov est presque toujours supervisé.
Afin d'assurer la robustesse des modèles acoustiques, les corpus d'apprentissage doivent contenir une quantité importante d'énoncés. Il est aussi important que les énoncés proviennent de plusieurs locuteurs afin de mieux modéliser la variabilité inter-locuteur. Ceci permet de construire des systèmes de reconnaissance indépendants du locuteur.

1.3 Modélisation statistique du langage

Les modèles de langage sont utilisés dans un système de reconnaissance de la parole pour guider le décodage acoustique. Trouver un bon modèle de langage influence considérablement les performances d'un système de reconnaissance vocale. Un tel modèle doit être discriminant pour favoriser certaines hypothèses par rapport à d'autres, tout en n'étant pas trop restrictif pour bien généraliser aux nouvelles données rencontrées lors d'un décodage.
Il existe deux approches différentes dans la construction d'un modèle de langage : les grammaires formelles et les modèles de langage stochastiques. Étant donné que les modèles de langage ne font pas l'objet de ces travaux nous n'allons pas détailler les grammaires formelles décrites dans (Chomsky, 1957, 1965; Pullum et Gazdar, 1982). Les modèles de langage stochastiques sont les plus répandus dans les applications de reconnaissance vocale, dont celle utilisée pour réaliser ces travaux. Nous allons faire une courte description de ce type de modèle de langage.
Le but d'un modèle de langage est d'attribuer une probabilité $P(W)$ à un séquence de mots $W = w_0, ..., w_N$. Sachant que les symboles de début et de fin de phrase, respectivement notés $< s >$ et $< /s >$, sont inclus dans la séquence de mots, la probabilité $P(W)$ peut se décomposer de la manière suivante :

$$P(W) = P(w_0, ..., w_N) = \prod_{i=1}^{N} P(w_i | w_0, ..., w_{i-1}) \tag{1.9}$$

On défini l'historique h_i du mot w_i comme étant l'ensemble des mots le précédant dans l'énoncé, $h_i = w_0, ..., w_{i-1}$.

$$P(W) = \prod_{i=1}^{N} P(w_i | h_i) \tag{1.10}$$

Le modèle doit être capable d'estimer les différentes probabilités $P(w_i|h_i)$. Même pour une longueur moyenne des historiques et pour un lexique de taille moyenne le calcul de ces probabilités conditionnelles devient problématique. Le nombre d'historiques possibles est trop important pour avoir des estimations fiables des différentes probabilités.

Une réduction de la taille des historiques est donc nécessaire afin de pouvoir estimer correctement les probabilités $P(w_i|h_i)$.

1.3.1 Approximation par modèle n-gramme

Dans les modèles *n-gramme* l'historique est réduit aux $n-1$ derniers mots précédant w_i dans l'énoncé. La probabilité du modèle de langage $P(W)$ devient :

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-n+1}, ..., w_{i-1}) \tag{1.11}$$

Dans la pratique, n est rarement choisi supérieur à 3 pour une première passe dans le système de reconnaissance de la parole :

- pour $n = 2$, le modèle de langage est un *bigramme* et la probabilité d'un mot ne dépend que du mot qui le précède :

$$P(w_i|w_0, ..., w_{i-1}) \approx P(w_i|w_{i-1}) \tag{1.12}$$

- pour $n = 3$, le modèle de langage est un *trigramme* et la probabilité d'un mot ne dépend que des deux mots qui le précèdent :

$$P(w_i|w_0, ..., w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1}) \tag{1.13}$$

Le modèle de langages *n-gramme* est un modèle stochastique qui n'utilise aucune connaissance d'ordre sémantique ou syntaxique. Malgré sa simplicité de modélisation, ce modèle est très répandu dans les systèmes de reconnaissance vocale car il est assez discriminant et facilement utilisable avec un faible coût en termes de calcul lors du processus de décodage. En effet, un modèle *n-gramme* peut facilement se mettre sous la forme d'un modèle à états finis stochastique et s'intégrer aisément avec les HMMs du modèle acoustique (également des modèles stochastiques à états finis) pour former un FSN (*Finite State Network*). Ce type de réseau est utilisé par notre système de reconnaissance.

1.3.2 Modèle à base de classes

Une des étapes les plus importantes dans la construction d'un modèle de langage est l'apprentissage. Cette étape nécessite une grande quantité de données d'apprentissage qui ne peuvent pas être exhaustives en ce qui concerne la variabilité et la complexité d'un langage. Une variante des modèles de langage de type *n-grammes* sont les modèles de langages à base de classes de mots (Damnati, 2000; Kobus, 2006).
Certains mots peuvent avoir un comportement similaire et il est donc possible de les regrouper en classes de mots. Ceci présente deux avantages majeurs :

- le nombre d'événements à modéliser est réduit ce qui améliore le rapport entre le nombre de paramètres à estimer et la taille des données d'apprentissage.
- une généralisation est réalisée sur les événements possibles dans le *n-gramme*. Un événement non vu au niveau mot dans le corpus d'apprentissage peut être plus facilement modélisé au niveau des classes.

Dans ce contexte un mot w_i appartient à une classe c_i. L'historique n'est plus une séquence de mots, mais une séquence de classes. Le modèle est alors construit de la façon suivante à partir des $n-1$ classes précédentes :

$$P(w_i|c_{i-n+1}\dots c_{i-1}) = P(w_i|c_i)P(c_i|c_{i-n+1}\dots c_{i-1}) \tag{1.14}$$

où $P(w_i|c_i)$ est la probabilité d'appartenance du mot w_i à la classe c_i et $P(c_i|c_{i-n+1}\dots c_{i-1})$ est la probabilité de la classe c_i connaissant le n-1 classes précédentes.
Pour un modèles *bi*-classes (n=2), la probabilité de la séquence de mots W se calcule de la manière suivante :

$$P(W) = \prod_{i=1}^{N} P(w_i|c_i)P(c_i|c_{i-1}) \tag{1.15}$$

Le modèle de langage utilisé dans nos travaux est un bi-classes dont la majorité des classes est formé d'un seul mot.

1.3.3 Imbrication des modèles

Dans un système de reconnaissance on peut parfois avoir besoin de modéliser différemment certains segments dans un énoncé, comme par exemple un numéro de téléphone. On peut utiliser différent sous-modèles de langage (*n-grammes*, grammaires) qui sont imbriqué dans le modèle général.
On prend l'exemple d'un modèle bigramme P^G qui contient un certain nombre de classes de mots et un sous-modèle de langage de type bigramme P^S. Le sous-modèle de langage est inclus dans le modèle global est son comportement est assimilé à celui d'une classe de mot notée c_S. La probabilité de la séquence de mots w_1, w_2, w_3, w_4 où la séquence w_2, w_3 est gérée par le sous-modèle se calcule de la façon suivante :

$$\begin{aligned} P^{G\ avec\ S}_{w_1,w_2,w_3,w_4} = \quad & P^G(w_1|start) \cdot P^G(c_S|w_1) \cdot \\ & P^S(w_2|start) \cdot P^S(w_3|w_2) \cdot \\ & P^S(end|w_3) \cdot P^G(w_4|c_S) \end{aligned} \tag{1.16}$$

1.3.4 Lissage

Lors de l'étape d'apprentissage [1] on estime les valeurs des paramètres d'un modèle du langage. Ces paramètres sont estimés à partir du nombre d'occurrences des événements dans le corpus d'apprentissage. Or les données dont on dispose pour l'apprentissage ne sont pas exhaustives, ainsi il existe un grand nombre d'événements jamais observés. Afin d'éviter que certains événements se voient attribuer une probabilité nulle, un lissage des probabilités est nécessaire. Le principe du lissage est justement d'attribuer une probabilité non nulle aux événements jamais observés lors de l'apprentissage en prélevant une quantité de la masse de probabilités de l'ensemble des événements observés pour la redistribuer aux événements non vus. Ce processus est guidé par une

1. La technique d'*absolute discounting* (Ney et Essen, 1991) est utilisée dans nos travaux.

contrainte de normalisation qui implique que, pour un historique h, les probabilités $P(w|h)$ somment à 1 pour tous les mots w du lexique. Les principales techniques de lissage et leur performances sont décrites dans (Chen et Goodman, 1996).

1.4 Combinaison des modèles acoustiques et des modèles de langage

Comme le suggère la formule 1.3, les scores donnés par le modèle acoustique et le modèle de langage se combinent par une simple opération de multiplication. Dans la pratique, cela n'est pas aussi simple car l'ordre de grandeur des scores est très différent. En effet, la vraisemblance d'une densité de probabilité donnée par le modèle acoustique est beaucoup plus petite que la probabilité donnée par le modèle de langage : $P(X|W) << P(W)$. Les multiplier directement reviendrait à négliger l'influence d'un modèle par rapport à l'autre.
La solution la plus couramment utilisée est d'introduire un facteur de pondération au modèle de langage, appelé *fudge factor*. La formule 1.3 devient :

$$\hat{W} \approx argmax_W P(W)^f \cdot P(X|W) \tag{1.17}$$

La valeur optimale de ce facteur est déterminée empiriquement sur un corpus de développement. La valeur choisie est celle qui optimise les performances du système de reconnaissance. En général, f>1.
Afin de déterminer la séquence W qui maximise l'équation 1.17, le système doit procéder à des multiplications successives des nombres compris entre 0 et 1. La capacité limitée d'un ordinateur de représenter un nombre proche de 0 est donc rapidement atteinte. C'est pourquoi, dans la pratique, les systèmes de reconnaissance de la parole ne manipulent pas directement les probabilités mais leurs logarithmes. Ainsi, le passage aux logarithmes entraîne l'utilisation des additions à la place des multiplications et permet de bénéficier de la propriété des logarithmes qui changent très lentement d'ordre de grandeur. L'équation 1.17 devient :

$$\hat{W} \approx argmax_W [f \cdot logP(W) + logP(X|W)] \tag{1.18}$$

1.5 Espace de recherche et sorties de reconnaissance

Un système de reconnaissance de la parole a pour but de générer, à partir d'un signal de parole et des connaissances *a priori* (lexique, modèle acoustique, modèle de langage,...), un ensemble d'hypothèses de séquences de mots. L'algorithme de décodage le plus souvent utilisé dans les systèmes de reconnaissance de la parole est basé sur le critère du maximum de vraisemblance (*Maximum A Posteriori*–MAP) qui détermine la séquence de mots qui maximise l'equation 1.3. Pour déterminer l'ensemble des hypothèses de reconnaissance, le décodeur explore un espace de recherche qui contient

les informations nécessaires à la génération des hypothèses : les unités acoustiques associées à leur scores acoustiques (donnés par le modèle acoustique), les informations temporelles, les informations liées au modèle de langage, etc..
Dans la majorité des systèmes, la taille de l'espace de recherche est très importante ce qui ralenti considérablement la recherche de la séquence de mots de probabilité maximale. Les hypothèses sont construites et évaluées au fur et à mesure du décodage ce qui permet de mettre en oeuvre une méthode efficace d'élagage (*pruning*). Celle-ci élimine les hypothèses les moins probables par rapport à la meilleure hypothèse à un instant donné. Ce type de recherche est aussi appelée recherche en faisceau (*beam search*) (Ney et Ortmanns, 2000; Ortmanns et Ney, 1997).
L'utilisation des modèles de langage de type *n-gramme* avec un N assez grand peut aussi ralentir considérablement la recherche de la séquence de mots de probabilité maximale. La solution la plus répandue, malgré sa simplicité de modélisation, est d'utiliser un modèle *n-gramme* plus réduit comme par exemple un bigramme (N=2) ou un trigramme (N=3).
Dans la pratique, l'utilisation des techniques comme le (*beam search*) et des bigrammes ou trigrammes ne garanti pas l'obtention de la solution la plus probable, mais dans la majorité des cas, la solution trouvée est un bon compromis entre la durée de traitement et la perte de précision. Un modèle de langage plus complexe peut être utilisé en deuxième passe sur l'ensemble des solutions trouvées. En effet, un système de reconnaissance peut produire une seule solution (la séquence de mots de probabilité maximale), appelée aussi *1-best*, mais aussi un ensemble de solutions possibles structurées sous la forme :

- d'une liste ordonnée de N meilleures solutions.
- d'une structure compacte appelée graphe de mots. Le but de cette structure est de fournir des alternatives pour les parties du signal acoustique pour lesquelles l'ambiguïté de la prononciation est forte (Ney et Ortmans, 1997).

1.5.1 Liste de N meilleures solutions

Appelée aussi liste N *best*, elle contient les N meilleures solutions du décodeur ordonnées en fonction de leur probabilité calculée au sens MAP. Comme la *1-best*, les solutions de la liste ne contiennent aucune information temporelle en ce qui concerne les instants de début ou de fin des mots de chaque séquence. On ne connaît pas non plus les vraisemblances acoustiques de chaque mot.
Sur une liste de N *best* on peut calculer des mesures de confiance (voir 2.3.2), faire du *rescoring* afin de réordonner les solutions de la liste. L'espace d'hypothèses que représentent les solutions de la liste peut être utilisé comme espace de recherche par des modules applicatifs en aval. On pourrait penser que plus N est grand, meilleures seront les performances de la liste de N *best*. Outre le fait que dans (Wessel et al., 2000) les auteurs ont montré que les performances de mesures de confiance sont dégradées à partir d'une certaine valeur de N ($N == 1000$), l'utilisation d'une liste avec un N trop grand n'est pas très efficace en termes de temps de calcul et des ressources mémoire nécessaires. Il convient donc de choisir une valeur de N suffisamment grande pour que

l'espace d'hypothèses soit assez riche, mais cette valeur ne doit pas générer un temps de calcul et des besoins de ressources trop importants.

1.5.2 Graphes de mots

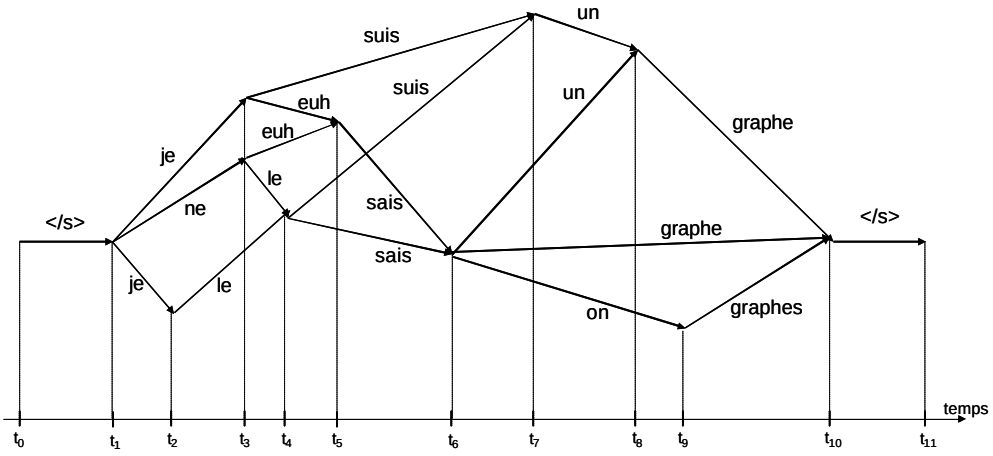

FIGURE *1.2* – Exemple simple de graphe de mots

La génération du graphe de mots se fait suite à un décodage de type Viterbi, synchrone dans le temps. A un instant donné, l'algorithme garde les hypothèses qui n'ont pas été éliminées par le *beam search*. La taille d'un graphe de mots dépend donc de la taille du faisceau, qui est un paramètre important dans la construction du graphe de mots.
La structure d'un graphe de mots est celle d'un graphe acyclique direct. La figure 1.2 montre un exemple simplifié. Deux éléments distincts composent le graphe de mots :

- les nœuds du graphe, aussi appelés états, sont caractérisés par l'instant temporel auquel ils ont été créés, appelé aussi trame de création d'un état.
- les arcs du graphe, aussi appelés transitions, sont caractérisés par un état de départ et un état de fin. Chaque transition porte une hypothèse de mot avec son score acoustique, qui mesure la vraisemblance entre les morceaux du signal acoustique correspondant à la transition et le modèle acoustique.

Les états du graphe sont numérotés en partant de 0 pour le premier état. Ainsi, du fait de sa structure acyclique, les états de fin des transitions se voit attribués un numéro plus grand que les états de début.

1.5.3 Réseaux de confusion

En 2000, une nouvelle structure, appelée réseau de confusion (CN - *Confusion Network*), a été proposée dans (Mangu et al., 2000) pour structurer l'ensemble des hypothèses en sortie de décodeur. Contrairement au graphe de mots, cette structure ne peut

pas être générée directement par le décodeur, mais seulement à partir d'un graphe de mots. Les algorithmes de génération sont détaillés au chapitre 5.

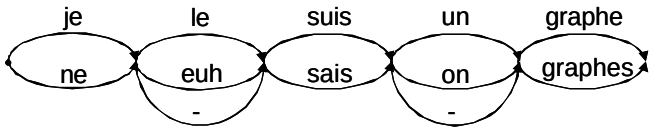

FIGURE *1.3* – Exemple de réseau de confusion. Le symbole "-" marque une omission

La figure 1.3 montre un exemple de réseau de confusion. On observe une structure similaire au graphe de mots par le fait que les éléments qui la constituent sont des transitions et des états. De plus, le CN garde la propriété d'être acyclique car on ne peut le parcourir que de gauche à droite sans qu'on puisse repasser deux fois par le même état. Comme pour les graphes de mots, les états sont numérotés et l'état de début d'une transition est plus petit que l'état de fin d'une transition. De plus, chaque état est caractérisé par un numéro de trame.
Comme le montre la figure 1.3, un réseau de confusion peut être assimilé a une séquence de "classes". Une classe est définie comme l'espace entre deux états consécutifs et contient une ou plusieurs transitions. Chaque transition est caractérisée par un état de début et un état de fin et se voit attachée une hypothèse de mot et sa probabilité *a posteriori*[2]. Une propriété importante d'une classe réside dans le fait que toutes les transitions de la classe partent du même état et finissent dans un même état (ce sont les deux états qui définissent la classe). Sur l'intervalle temporel représenté par la classe, les transitions sont donc en concurrence.
Du fait de sa structure, un chemin complet dans un CN (qui part de l'état 0 et arrive à l'état de fin) passe par tous les états du réseau et traverse ainsi toutes les classes en passant par une des transitions de chaque classe. En passant par une classe, le chemin peut soit passer par une transition portant un mot soit par une transition portant une "omission". Dans chaque classe on peut trouver au maximum une transition portant l'omission avec une probabilité *a posteriori* associée. Elle permet de traverser la classe sans passer par un mot. Les problématiques liées à cette transition, sa présence dans certaines classes et le calcul de la probabilité *a posteriori* associée sont discutées au chapitre 5.
Pour résumer, les éléments qui constituent un réseau de confusion sont :

- Les états caractérisés par un instant temporel. Les états sont numérotés en partant du premier état, à qui on attribue la valeur 0. Tout chemin complet du CN passe par tous les états en les traversant dans un ordre croissant de leur numéro.
- Les classes sont définies comme l'espace entre deux états consécutifs qui représentent les états de début et de fin de la classe. Tout chemin complet du CN traverse chaque classe une seule fois.
- Les transitions sont regroupées en classes, chacune pouvant en contenir une ou plusieurs transitions. Les états de début et de fin des transitions d'une classe sont identiques aux états de début et de fin de la classe. Chaque transition porte une hypothèse de mot avec la probabilité *a posteriori* associée. Un chemin complet du

2. Le calcul de cette probabilité est détaillé au chap 5

CN ne peut contenir qu'une seule transition de chaque classe. Dans une classe, il peut exister une transition portant une omission, avec une probabilité *a posteriori* associée, qui permet à un chemin complet de traverser une classe sans passer par un mot.

1.6 Évaluation des systèmes de reconnaissance de la parole

1.6.1 Taux d'erreur mot et *word accuracy*

Une fois la solution du système de reconnaissance obtenue, elle doit être évaluée afin de mesurer les performances du système. La mesure la plus souvent employée pour évaluer un système de reconnaissance de la parole et le taux d'erreur mot (*WER - Word Error Rate*). Ceci implique un comptage du nombre de mots mal reconnus par le système réalisé par une comparaison entre la meilleure solution du système (la séquence de mots reconnue) et la transcription manuelle du signal de parole (la séquence de mots réellement prononcée). La transcription manuelle du signal de parole est généralement appelée *transcription de référence* ou tout simplement *référence*.
Pour calculer le *WER*, les deux séquences de mots sont alignées et on mesure la distance de Levensthein, appelée aussi distance d'édition. Elle est égale au nombre minimal de mots qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre. Cette distance est d'autant plus grande que le nombre de différences entre les deux chaînes est grand.
Trois types d'erreurs possibles interviennent dans le calcul du *WER*.

- Les *omissions* sont les mots de la référence qui n'ont pas été reconnus par le système. Ils ne se retrouvent donc pas dans la solution fournie.
- Les *insertions* sont des mots reconnus par le système qui ont été insérés dans la solution en plus des mots de la référence.
- Les *substitutions* sont les mots qui ont été reconnus à la place d'autres mots de la référence.

Le *WER* est calculé en sommant les trois types d'erreurs et en normalisant par le nombre total de mots dans la référence :

$$WER = \frac{Nombre\ d'omissions\ +\ Nombre\ d'insertions\ +\ Nombre\ de\ substitutions}{Nombre\ de\ mots\ dans\ la\ référence} \tag{1.19}$$

Le *word accuracy* est défini comme étant égal à 1 moins la valeur du taux d'erreur mot.

1.6.2 Précision et rappel

D'autres métriques peuvent être utilisées pour mesurer les performances d'un système, comme par exemple la précision et le rappel, empruntés au domaine de la recherche d'informations.

La précision mesure la capacité du système de rejeter les hypothèses de mots incorrectes :

$$Précision = \frac{Nombre\ de\ mots\ correctement\ reconnus}{Nombre\ de\ mots\ reconnus} \tag{1.20}$$

Le rappel mesure la capacité du système à accepter les hypothèses de mots correctes :

$$Rappel = \frac{Nombre\ de\ mots\ correctement\ reconnus}{Nombre\ de\ mots\ dans\ la\ référence} \tag{1.21}$$

1.6.3 Taux d'erreur mot Oracle

Les métriques présentées se calculent seulement lorsque le résultat du système de reconnaissance est sous la forme d'une séquence d'hypothèses de mots. Mais certains systèmes sont aussi capables de produire en sortie des graphes de mots. Il est toujours possible d'extraire une séquence de mots du graphe (qui forme un chemin complet), généralement la meilleure solution, au sens de MAP, et de calculer le *WER*, la précision ou le rappel. Néanmoins, en raison des techniques utilisées lors de la construction du graphe, tel que le *beam search*, cette solution n'est pas forcément la meilleure solution que le système peut obtenir. Il est donc nécessaire d'utiliser une métrique qui permet de mesurer les performances du graphe de mots dans sa globalité, en prenant en compte l'ensemble des hypothèses présentes dans le graphe.
L'oracle est obtenu en recherchant dans le graphe de mots la séquence de mots la plus proche de la référence du point de vue de la distance d'édition. Le taux d'erreur mot Oracle (*WER* Oracle) est alors le meilleur taux d'erreur mot qu'on pourrait obtenir si on savait retrouver l'oracle de manière automatique. Il peut être vu comme la limite théorique du *WER* qui pourrait être obtenu en choisissant idéalement les séquences de mots dans le graphe. Ce taux peut servir à mesurer le potentiel d'un graphe de mots en termes de *WER*, mais aussi à comparer des graphes de mots obtenus avec différents systèmes de reconnaissance ou avec des paramètres différents d'un même système.
De la même manière que pour le *word accuracy* on peut définir un *oracle accuracy* qui est égal à 1 moins la valeur du taux d'erreur oracle.

1.7 Conclusions

Dans ce chapitre nous avons tout d'abord présenté les différents modules qui composent un système de reconnaissance de la parole et leur fonctionnement. Nous avons ainsi décrit le module d'analyse acoustique du signal qui a pour but de transformer le signal acoustique en une suite de vecteurs de coefficients utilisés ensuite dans le processus de décodage. Ce processus est réalisé par le décodeur de parole qui utilise des modèles acoustiques et de langage que nous avons également décrits. La problématique liée à la combinaison des deux modèles a aussi été discutée. Nous avons ensuite abordé les problématiques liées à l'espace de recherche utilisé par le décodeur. Une description de l'espace d'hypothèses en sortie de décodeur et les différentes structures utilisées ont

été abordées par la suite. La dernière partie du chapitre a présenté les différentes métriques utilisées pour l'évaluation des performances d'un système de reconnaissance de la parole.

Chapitre 2

Mesures de confiance

Sommaire

Les performances d'un système ASR peuvent être souvent altérées par différents paramètres comme un environnement bruité, la variabilité entre locuteurs, les disfluences inhérentes à la parole spontanée, etc. Il est donc nécessaire pour un tel système de pouvoir, de manière automatique, évaluer la fiabilité des solutions données par le système. En d'autres termes, la *mesure de confiance* associée à une hypothèse h du système ASR ($MC(h)$) peut être assimilée à la probabilité que l'hypothèse soit correcte. Celle-ci doit être comprise dans l'intervalle $[0, 1]$ et, idéalement, une valeur de 0 pour la mesure de confiance correspond à une hypothèse incorrecte, et une valeur de 1 à une hypothèse correcte.
De nombreux domaines du traitement de la parole utilisent les mesures de confiance (Lee, 2001). On les retrouve par exemple dans la reconnaissance de la parole (Wessel et Ney, 2005; Cox et Dasmahapatra, 2002; Soong et al., 2004; Ketabdar et al., 2006), dans les système de dialogue (San-Segundo et al., 2001; Raymond et al., 2004; Raymond, 2005),

dans l'identification des langues (Metze et al., 2000) ou dans la reconnaissance du locuteur (Preti et al., 2007). Dans ces domaines, les mesures de confiance peuvent être appliquées à plusieurs niveaux : au niveau du phonème (principalement dans la reconnaissance de la parole), du mot, de la phrase, des concepts (unité sémantique permettant d'exprimer le sens d'une séquence de un ou plusieurs mots de façon conceptuelle, principalement utilisée dans les systèmes de dialogue (Kobus, 2006)) ou bien au niveau de la phrase. Dans la suite de ce chapitre on se placera au niveau du mot en ce qui concerne l'utilisation des mesures de confiance, sachant que l'utilisation des mesures de confiance dans le cadre des travaux présentés ultérieurement dans cette thèse se fait également au niveau du mot.
L'article (Jiang, 2005) propose une classification des mesures de confiance en trois catégories distinctes :

1. Une grande majorité des travaux utilise les paramètres prédictifs dans le calcul des mesures de confiance. Un paramètre peut être appelé paramètre prédictif si la distribution de probabilité des mots reconnus comme étant corrects est différente de la distribution de probabilité des mots incorrects. Ces paramètres sont généralement collectés pendant le décodage et sont ensuite combinés afin d'obtenir une seule mesure indiquant le degré de véracité du mot. Ils incluent des paramètres acoustiques ainsi que des paramètres provenant du modèle de langage ou du comportement de l'algorithme de recherche.
2. La probabilité *a posteriori* d'un mot est souvent utilisée en tant que mesure de confiance étant donné le fait qu'elle représente une mesure absolue de la fiabilité d'une décision. La probabilité *a posteriori* est une estimation de la vraisemblance entre le mot w et la suite de vecteurs d'observations acoustiques X. Comme nous le verrons par la suite elle est assez difficile à calculer (voir 2.3), d'où les différentes méthodes proposées afin d'obtenir la meilleure approximation possible. Ces méthodes sont des méthodes simples, utilisant des modèles de type *filler*, aux approches plus complexes basées sur les graphes de mots.
3. Si les deux premières catégories présentent des mesures de confiance au niveau mot, de nombreux travaux ont été menés sur l'utilisation des mesures de confiance au niveau énoncé afin de vérifier le contenu d'une hypothèse (*utterance verification*). Une fois le décodage effectué, le système produit une hypothèse W dont on évalue la fiabilité à travers une mesure de confiance. L'estimation de la mesure de confiance est formulée ici comme un test statistique pour vérifier si l'hypothèse est correcte ou incorrecte.

Les travaux de cette thèse se concentrent sur l'utilisation des mesures de confiance au niveau mot et plus précisément sur l'utilisation de la probabilité *a posteriori* du mot comme mesure de confiance. Dans ce chapitre nous présentons, tout d'abord, différentes techniques d'évaluation des mesures de confiance dans la section 2.1. Nous détaillons ensuite les mesures de confiance au niveau mot basées sur les paramètres prédictifs dans la section 2.2 et sur la probabilité *a posteriori* dans la section 2.3. La dernière partie 2.4, décrit l'algorithme *Forward-Backward* que nous avons adapté pour le calcul des probabilités *a posteriori* sur les graphes de mots. Une nouvelle méthode de normalisation des variables de l'algorithme *Forward-Backward* est aussi décrite. Étant donné

que les mesures de confiance au niveau de la phrase ne font pas l'objet des travaux de cette thèse, nous ne détaillons pas ce point.

2.1 Evaluation des mesures de confiance

Il existe différentes métriques (Siu et Gish, 1999) pour évaluer les mesures de confiance. Quelques unes des plus simples sont détaillées ici, avec la précision qu'une partie de ces métriques, comme la Detection Error Tradeoff ou l'entropie croisée, seront également utilisées pour l'évaluation des travaux effectués.

2.1.1 Detection Error Tradeoff

La courbe *Detection Error Tradeoff* (DET) permet d'évaluer la capacité d'une mesure de confiance à accepter ou rejeter une hypothèse en faisant varier la valeur du seuil fixé pour cette mesure. En faisant varier la valeur du seuil α sur la mesure de confiance, la décision est prise de la manière suivante :

$$\text{hypothèse} = \begin{cases} \text{acceptée} & \text{si MC(hypothèse)} \geq \alpha \\ \text{rejetée} & \text{sinon} \end{cases} \tag{2.1}$$

L'équation 2.1 peut donc conduire à deux types d'erreurs :

- Une erreur de fausse acceptation. Appelée aussi *Fausse Alarme (FA)*, cette erreur survient dans le cas où une hypothèse est acceptée comme étant correcte alors qu'elle est incorrecte.
- Une erreur de rejet à tort. Appelée aussi *Faux Rejet (FR)*, cette erreur survient dans le cas où une hypothèse est considérée comme incorrecte alors qu'elle est correcte.

Au vu de ces deux types d'erreurs, il est possible de calculer deux taux sur un corpus d'évaluation :

- Le taux de fausse alarme :

$$FA = \frac{\text{Nombre d'hypothèses acceptées à tort}}{\text{Nombre total d'hypothèses}} \tag{2.2}$$

- Le taux de faux rejet :

$$FR = \frac{\text{Nombre d'hypothèses rejetées à tort}}{\text{Nombre total d'hypothèses}} \tag{2.3}$$

Si on considère les distributions des mesures de confiance calculées sur l'ensemble des hypothèses correctes et des hypothèses incorrectes dans la figure 2.1 on peut visualiser les taux de faux rejet (FR) et de fausses alarmes (FA) en fonction du seuil α. Quand le seuil se déplace vers la droite (sa valeur augmente), le taux de FA diminue alors que le taux de FR augmente. Si le seuil se déplace vers la gauche (sa valeur diminue), les tendances sont inversées, avec un taux de FA qui augmente et un taux de FR qui diminue. Pour chaque valeur du seuil α, un couple (FA, FR) peut être calculé. Ce couple de valeurs détermine ce qu'on appelle un point de fonctionnement du système.

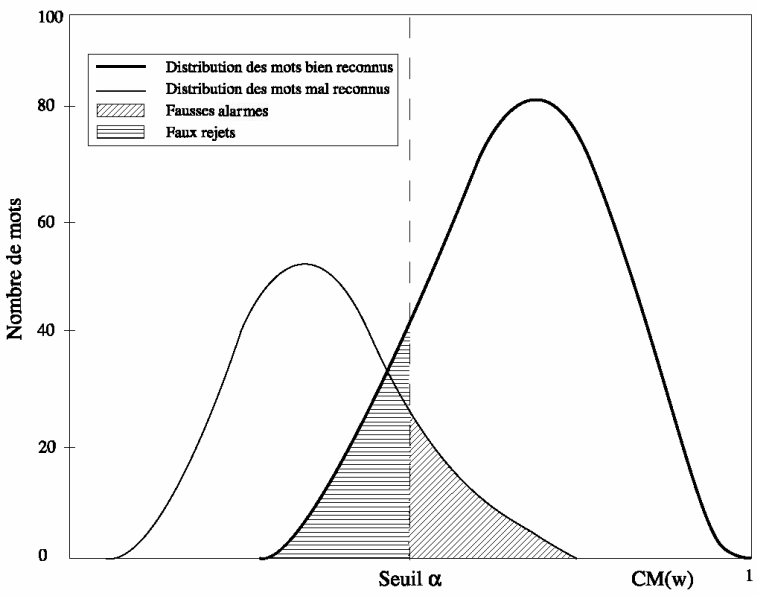

FIGURE 2.1 – Distribution des mesures de confiance sur les hypothèses de reconnaissance

La courbe DET, voir figure 2.2, décrit le taux de faux rejets (FR) en fonction du taux de fausses alarmes (FA) et permet de visualiser les différents points de fonctionnement du système qui correspondent à une valeur du seuil α variant de 0 à 1. Cette courbe, dans sa globalité, est plus adaptée dans la comparaison des performances de plusieurs systèmes que la comparaison d'un seul point de fonctionnement pour une valeur de seuil donnée. Néanmoins, un point de la courbe DET souvent utilisé dans la comparaison des systèmes est le point où les deux taux d'erreur sont égaux (EER : *Equal Error Rate*). Il se trouve à l'intersection de la courbe DET avec la droite d'égale erreur. Pour un système, plus ce point est proche de l'origine plus les mesures de confiance sont discriminantes.
Il existe aussi une autre variante de cette courbe, appelée courbe ROC (*Receiver Operating Characteristic* ou *Relative Operating Characteristic*) (Martin et al., 1997) qui, en général, décrit le taux de bonnes détections en fonction de taux de fausses alarmes.

2.1.2 Confidence Accuracy et Confidence Error Rate

Un autre moyen d'évaluer les mesures de confiance réside dans le calcul du taux d'erreur de confiance, le *Confidence Error Rate* (CER), et de son contraire, la *Confidence Accuracy* (CA). Ces deux métriques se calculent de la manière suivante :

$$CA = \frac{\text{Nombre d'étiquettes correctement assignées}}{\text{Nombre total d'étiquettes}} \tag{2.4}$$

$$CER = \frac{\text{Nombre d'étiquettes incorrectement assignées}}{\text{Nombre total d'étiquettes}} \tag{2.5}$$

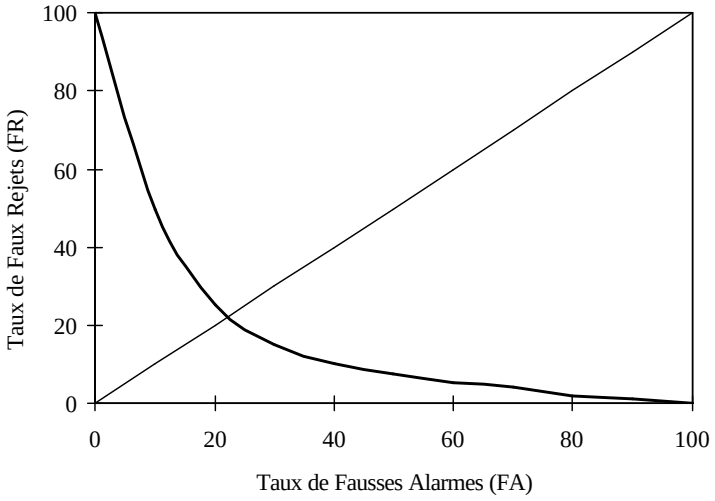

FIGURE 2.2 – Exemple de courbe DET

Le dénominateur des deux équations (*Nombre total d'étiquettes*) est le nombre total de mots reconnus par le système de reconnaissance. Chaque mot se voit désigner une étiquette, qui est en fait un attribut qui peut prendre deux valeurs, "correct" ou "incorrect". On dit qu'une étiquette est correctement assignée si à un mot reconnu comme étant correct on a assigné l'étiquette "correct" ou à un mot reconnu à tort on a assigné l'étiquette "incorrect". Le contraire signifie que l'étiquette a été incorrectement assignée. La décision d'assigner l'étiquette "correct"/"incorrect" à un mot dépend de la valeur du seuil appliqué sur la mesure de confiance du mot. Le seuil doit être optimisé sur un corpus de développement de telle façon à ce que la valeur du *Confidence Accuracy* soit la plus grande possible et inversement la valeur du *Confidence Error Rate* soit la plus petite possible. Suite à ce calibrage, la valeur du seuil obtenue peut être utilisée sur un corpus de test afin d'évaluer les performances de la mesure de confiance.

2.1.3 Entropie croisée

Les mesures de confiance doivent permettre d'estimer avec fiabilité la probabilité qu'une hypothèse soit correcte étant donné un jeu de mesures de confiance associé. Afin de choisir le jeu de mesures de confiance le plus adapté plusieurs méthodes peuvent être utilisées afin d'en évaluer leur pertinence. Une de ces méthodes consiste à mesurer la diminution relative de l'entropie croisée engendrée par la mesure de confiance.
Soit un corpus de test $\mathcal{C}$ constitué de N hypothèses de mot ; l'entropie croisée H se calcule de la manière suivante :

$$H = -\frac{1}{N}\sum_{i=1}^{N}(\delta_i \log p_i + (1-\delta_i)\log(1-p_i)) \tag{2.6}$$

δ_i est un facteur égal à 1 si l'hypothèse du mot w_i est correct et 0 sinon. La probabilité p_i représente la probabilité que l'hypothèse du mot w_i soit correcte. Sans utiliser la mesure de confiance, cette probabilité peut être évaluée comme étant la probabilité moyenne qu'un mot reconnu soit correct, ce qui est égal à la précision ($Prec$) sur le corpus de test :

$$Prec = \frac{\sum_{i=1}^{N} \delta_i}{N} \tag{2.7}$$

On peut définir ainsi l'entropie croisée initiale du corpus de test, notée H_{init} :

$$H_{init} = -Prec \log(Prec) - (1 - Prec) \log(1 - Prec) \tag{2.8}$$

Étant donné un jeu de mesures de confiance $MC = mc_1, \ldots, mc_n (n >= 1)$, la probabilité p_i devient $p_i = P(Cor|MC(w_i))$, la probabilité $P(Cor|MC(w_i))$ étant la probabilité que l'hypothèse de mot w_i soit correcte étant donné le jeu de mesures de confiance associé $MC(w_i)$. L'entropie croisée du corpus de test se calcule alors :

$$H_{MC} = -\frac{1}{N} \sum_{i=1}^{N} (\delta_i \log P(Cor|MC(w_i)) + (1 - \delta_i) \log(1 - P(Cor|MC(w_i)))) \tag{2.9}$$

Une mesure de confiance est d'autant plus efficace que l'information additionnelle apportée à l'hypothèse de mot est grande. L'évaluation se fait en calculant la diminution relative induite sur l'entropie croisée d'un corpus de test donnée, noté ΔH, par le jeu de mesures de confiance choisi :

$$\Delta H = \frac{H_{init} - H_{MC}}{H_{init}} \cdot 100 \tag{2.10}$$

Pour une mesure de confiance purement aléatoire il n'y a aucune information additionnelle et donc l'entropie du corpus de test reste inchangée. Par conséquence, la diminution relative est nulle. En revanche, pour une mesure de confiance "idéale", la prise de décision sur l'exactitude d'une hypothèse est parfaite, et donc $H_{MC} = 0$. La valeur de la diminution relative ΔH varie alors de 0 à 100%. Plus ΔH est élevée, plus l'information additionnelle apportée est importante et la mesure de confiance prédictive.

2.2 Paramètres prédictifs

Un paramètre prédictif idéal doit fournir une information suffisante afin de pouvoir départager les mots reconnus qui sont corrects de ceux reconnus à tort de telle manière que le recouvrement des distributions des deux classes soit le plus petit possible.

2.2.1 Paramètres acoustiques

Un paramètre simple évoqué dans (Jiang, 2005) est le score acoustique par trame. Celui-ci est obtenu en normalisant la vraisemblance acoustique du mot par le nombre

de trames acoustiques du mot. Toutefois, ce paramètre ne permet pas à lui seul d'obtenir une estimation précise de la pertinence du mot.

Une autre approche utilisant le score acoustique vise à isoler l'information acoustique de celle provenant du modèle de langage en utilisant, en parallèle avec le moteur de reconnaissance, une boucle de phonèmes non contrainte par un modèle de langage. Cette boucle permet à n'importe quel phonème d'en suivre un autre avec une probabilité égale et le décodage ne se base que sur la vraisemblance acoustique entre le signal et le modèle acoustique. L'utilisation d'une boucle de phonèmes n'est pas contraignante en termes de complexité de calcul. Celle-ci est très réduite par rapport à un moteur de reconnaissance intégrant un modèle de langage. Le ratio entre le score acoustique du mot reconnu et le score des phonèmes obtenus par la boucle de phonèmes sur le même intervalle temporel est proposé comme mesure de confiance dans (Young et al., 1997).

Une autre façon de calculer les mesures de confiance est proposée dans (Cox et Dasmahapatra, 2002) qui utilise des techniques de corrélation entre phonèmes. Ainsi, une mesure de confiance proposée est estimée en calculant une matrice de confusion croisée entre la transcription en phonèmes du mot reconnu, donnée par le dictionnaire, et la séquence de phonèmes reconnus par le décodeur. Les valeurs de la matrice sont estimées en réalisant un alignement entre les deux séquences et en comptant le nombre de phonèmes correctement alignés. Étant donné que cette méthode ne tient pas compte du fait que le mot ou la séquence de phonèmes soient corrects ou non, les auteurs proposent d'utiliser un corpus de développement afin de construire deux matrices de confusion croisées entre la séquence de phonèmes et les mots corrects d'un coté et les mots incorrects de l'autre coté. Ces deux matrices sont ensuite utilisées pour estimer la probabilité qu'un mot soit correct ou incorrect étant donné l'alignement des séquences de phonèmes dans les deux matrices. La mesure de confiance est estimée comme étant le rapport de vraisemblance entre ces deux probabilités.

Un autre paramètre évoqué dans (Jiang, 2005; Wessel et al., 2001) est la stabilité acoustique. La raison qui pousse au calcul de ce paramètre consiste dans le fait qu'un mot a une probabilité élevée d'être correct s'il est présent dans la même position, donnée par l'alignement de Levensthein, dans une majorité des phrases décodées par le SRAP utilisant différentes techniques ou paramètres. Généralement, il s'agit soit de faire une comparaison sur les N meilleures hypothèses issues du décodage soit de réaliser plusieurs décodages en variant différents paramètres. Dans une grande majorité des cas on effectue plusieurs décodages en variant la valeur du *fudge* afin d'obtenir la meilleure hypothèse du SRAP pour chaque décodage. La stabilité acoustique est définie comme le rapport entre le nombre de fois où le mot apparaît dans les hypothèses alternatives et le nombre total d'hypothèses. Dans (Wessel et al., 2001) la stabilité acoustique obtient de bonnes performances sur la majorité de corpus utilisés (ARISE, Verbmobil, Broadcast News, NAB). De très bonnes performances sont obtenues sur le corpus ARISE, qui peuvent s'expliquer par une longueur moyenne des énoncés très faible. Toutefois, la différence entre les performances de la stabilité acoustique et celles de la probabilité *a posteriori* calculée sur les graphes de mots (voir 2.3) est assez significative.

Discussions

Le but principal de ces techniques utilisées pour le calcul des mesures de confiance est d'essayer d'isoler le modèle de langage et le modèle acoustique afin de pouvoir calculer des mesures de confiance basées uniquement sur la modélisation acoustique des mots. En effet, dans (Palmer et Ostendorf, 2001) et (Cox et Dasmahapatra, 2002) on a observé que la probabilité conditionnelle qu'un mot soit correct ou incorrect dépend du fait que le mot précédent est correct ou incorrect. Ceci suggère un découplage entre les deux modèles afin d'éliminer les effets corrélatifs du modèle de langage. Toutefois, un découplage total des deux modèles n'est pas possible. Les deux approches (Young et al., 1997; Cox et Dasmahapatra, 2002), présentées ci-dessus, proposent d'utiliser une boucle de phonèmes en parallèle avec le SRAP. Les performances des mesures de confiance observées dans (Cox et Dasmahapatra, 2002) sur un corpus extrait du *Wall Street Journal database* ne sont pas très convaincantes et sont très loin des performances obtenues avec la stabilité acoustique.

2.2.2 Paramètres linguistiques

De la même manière que le modèle acoustique, le modèle de langage peut lui aussi constituer une source dans le calcul des paramètres prédictifs. On peut par exemple utiliser des paramètres comme le score du modèle de langage ou le comportement du repli (le degré du modèle de repli [1]) directement comme mesure de confiance(San-Segundo et al., 2001). La technique du repli consiste à utiliser un modèle de langage plus général lorsqu'un modèle spécifique ne détient pas les informations nécessaires. Plus précisément, le système peut faire appel à un $(n-1)$-gramme si une séquence des n mots n'a pas été vue dans le corpus d'apprentissage pour un n-gramme. Une mesure de confiance est ainsi attribuée en fonction du degré du repli. Dans (Uhrik et Ward, 1997), les auteurs proposent d'attribuer un score de confiance arbitraire au mot considéré, en fonction du degré de repli pour les deux mots précédents ou encore pour le mot précédent et le mot suivant.

Un autre type de paramètre est celui lié au *parsing* des hypothèses de reconnaissance. Par exemple, dans (Zhang et Rudnicky, 2001), en partant de la prémisse que les hypothèses de reconnaissance correctement reconnues sont plus grammaticales que celles incorrectement reconnues, on utilise un analyseur sémantique (*parser*) afin d'extraire la mesure de confiance. Cet analyseur produit un arbre contenant des groupes de mots résultants du parsing. Ceci est une source d'information pour le calcul d'une mesure de confiance au niveau mot en considérant deux paramètres : le degré du repli du groupe de mots (en partant d'un *quatre-grammes*) et le mode de parsing qui indique si un mot a été analysé comme faisant partie d'un groupe et donne également la position du mot dans le groupe (milieu ou extrémités) et le deuxième. Les auteurs obtiennent

1. Le modèle de repli (*backing-off*), introduit par (Katz, 1987), utilise un modèle de langage plus général. Pour un modèle *n-gramme*, si un certain *n-gramme* (h, w) n'a jamais été observé dans le corpus d'apprentissage, on descend d'un degré et le modèle de niveau inférieur *(n-1)-gramme* est utilisé.

ainsi des performances qui dépassent celles d'autres mesures de confiance utilisées : le score acoustique normalisé, la probabilité *a posteriori* dans un graphe de mots calculée en utilisant soit le score du modèle de langage soit le score du modèle acoustique, la stabilité acoustique (sur une liste de N meilleures hypothèses, etc.). Les performances de la mesure de confiance ainsi proposée restent toutefois inférieures aux performances obtenues avec la probabilité *a posteriori* de mots dans un graphe (calculée en utilisant le modèle de langage et le modèle acoustique).

Dans (Palmer et Ostendorf, 2001), des connaissances sémantiques, comme l'appartenance d'un mot à une classe sémantique, appelée aussi *entité nommée*, comme une "localisation, organisation ou personne", ont été utilisées afin d'estimer la mesure de confiance d'un mot reconnu.

Une mesure de confiance peut être calculée non seulement pour les mots d'une phrase mais aussi pour la phrase entière. Par exemple, dans (Pao et al., 1998), les connaissances sémantiques ont été utilisées pour le calcul des mesures de confiance en définissant des classes sémantiques. Chaque classe reçoit un certain poids et une distance sémantique est calculée sur les N meilleures hypothèses à l'aide de l'algorithme d'alignement de Levensthein. Par exemple, pour une classe qui regroupe des villes dans une application qui donne les prévisions météo, la substitution d'un mot de cette classe par un autre mot de la classe produit une distance sémantique très grande. Ainsi, les auteurs utilisent les poids des classes sémantiques de la meilleure solution ainsi que les distances sémantiques de trois premières solutions comme paramètres pour calculer une mesure de confiance à l'aide d'arbres de décision. Dans (San-Segundo et al., 2001), des paramètres prédictifs du modèle de langage, comme ceux cités au dessus, mais aussi des paramètres liés à l'analyseur sémantique en concepts utilisé par les auteurs, sont utilisés pour estimer une mesure de confiance au niveau de la phrase. Dans (Estève et al., 2003), l'estimation de la mesure de confiance au niveau de la phrase se base sur le fait que les événements non vus dans le corpus d'apprentissage lors de la construction du modèle de langage sont mal modélisés. Ainsi, pour une phrase donnée, la mesure de confiance est estimée en calculant le rapport entre le nombre de n-grammes présents dans la phrase et ayant été observés dans le corpus d'apprentissage du modèle de langage et le nombre total des n-grammes dans la phrase.

Discussions

On observe une polyvalence des paramètres linguistiques en ce qui concerne l'estimation de mesures de confiance tant au niveau mot qu'au niveau phrase. A la différence des paramètres acoustiques, certains paramètres linguistiques comme le repli ou les *entités nommées* peuvent être utilisés aussi bien au niveau mot qu'au niveau phrase. Par exemple, dans (Uhrik et Ward, 1997), le repli donne de très bonnes performances au niveau phrase mais des performances moindres au niveau mot. Ceci s'explique par une confusion pour savoir exactement quel mot est incorrect dans le cas des insertions et omissions des mots. Une adaptation est proposée afin de définir des groupes de mots qui peuvent contenir des erreurs potentielles ce qui augmente considérablement les

performances. Les *entités nommées* constituent elles aussi une technique qui peut être employé pour le calcul des mesures de confiance au niveau mot (Palmer et Ostendorf, 2001) et au niveau phrase (Pao et al., 1998).
Une observation intéressante est faite dans (San-Segundo et al., 2001), qui utilise le repli ainsi que le score de modèle de langage comme paramètres linguistiques et les compare à des paramètres acoustiques, comme la stabilité acoustique (pour une liste de N-meilleures hypothèses) ou le score acoustique normalisé par trame. Sur un corpus collecté à l'aide du système *CU Communicator* et utilisant un *multi-layer perceptron* pour combiner les différents paramètres, les auteurs observent de meilleures performances pour les paramètres linguistiques comparés aux paramètres acoustiques permettant une meilleure détection des mots incorrects. Il est à noter toutefois que les meilleurs résultats sont obtenus en combinant les deux types de paramètres. Des résultats similaires ont été observés dans (Zhang et Rudnicky, 2001) avec la précision que la probabilité *a posteriori*, qui peut être vue comme une combinaison des paramètres acoustiques (le score acoustique) et des paramètres de langage (le score de modèle de langage), dépasse les performances des paramètres linguistiques ou acoustiques. Cette observation va dans le sens de nombreuses publications qui s'accordent à dire que la probabilité *a posteriori* calculée dans un graphe de mots peut constituer une meilleure mesure de confiance que les paramètres prédictifs.

2.2.3 Autres paramètres

Il existe aussi d'autres paramètres prédictifs utilisés dans l'estimation de la mesure de confiance calculés à partir des différents éléments du système de reconnaissance et qui ne peuvent pas être classés dans les deux catégories présentées précédemment.
Des paramètres comme la durée du mot ou du phonème, l'état du HMM dans lequel on se trouve peuvent être utilisés dans l'estimation de la mesure de confiance (Vergyri, 2000).
L'utilisation de la liste des N-meilleures hypothèses peut être aussi une source importante d'information (Guo et al., 2004; Gillick et al., 1997; Hazen et al., 2002). On peut estimer une mesure de confiance à partir de paramètres comme les scores des premières hypothèses, la différence de score entre la meilleure hypothèse et les suivantes, le nombre de fois qu'un mot apparaît dans les différentes hypothèses, etc. Ainsi, plus un mot est présent dans les hypothèses plus il a des chances d'être correct.
Les graphes de mots peuvent être aussi utilisés dans l'estimation de la mesure de confiance. S'ils sont utilisés principalement pour le calcul de la probabilité *a posteriori*, voir la section 2.3, il existe aussi d'autres paramètres tel que la densité de l'hypothèse, c'est-à-dire le nombre de transitions en concurrence dans l'intervalle temporel du mot w (Kemp et Schaaf, 1997; Wessel et al., 1999). Comme pour le cas précédent, plus il existe d'hypothèses du même mot dans un intervalle de temps, plus le mot a des chances d'être correct.

2.2.4 Combinaison de plusieurs paramètres prédictifs

Comme nous l'avons précisé, le paramètre prédictif "*idéal*" doit fournir une information suffisante afin de pouvoir départager les mots reconnus qui sont corrects de ceux reconnus à tort. Pour ce faire le recouvrement entre les distributions des classes correct/incorrect doit être le plus petit possible. Or ce n'est pas le cas des paramètres présentés car, comme le montrent de nombreuses études (Kemp et Schaaf, 1997; Schaaf et Kemp, 1997), le recouvrement des deux distributions est assez important même pour les meilleurs paramètres.
Afin d'essayer de contourner ce problème, la combinaison de plusieurs de ces paramètres semble être un moyen efficace pour obtenir de meilleures performances. Un classifieur permet de regrouper les différents paramètres prédictifs afin d'obtenir une seule mesure de confiance comprise entre 0 et 1 pour un mot w.

La **régression logistique** permet de combiner plusieurs paramètres prédictifs afin d'obtenir une mesure de confiance (Charlet et al., 2001). La probabilité qu'une hypothèse soit correcte étant donné les valeurs respectives des paramètres prédictifs acoustiques et linguistique utilisés est donnée par :

$$P(COR|MC_{acc}, MC_{lang}) = \frac{1}{1 + e^{-(a_0 + a_1 \cdot MC_{acc} + a_2 \cdot MC_{lang})}} \tag{2.11}$$

Les paramètres a_0, a_1 et a_2 sont estimés de façon à minimiser l'entropie croisée (voir 2.1.3 sur un corpus de développement.
La régression logistique peut être utilisée aussi pour évaluer une seule mesure de confiance. Cette mesure de confiance doit permettre d'estimer la probabilité qu'une hypothèses soit correcte étant donné la valeur de la mesure de confiance. Cette probabilité, noté $P(COR|MC)$, peut être approximée à l'aide de la régression logistique en n'utilisant que deux paramètres de calibration :

$$P(COR|MC) = \frac{1}{1 + e^{-(a_0 + a_1 \cdot MC)}} \tag{2.12}$$

Les paramètres a_0 et a_1 sont estimés de façon à minimiser l'entropie croisée sur un corpus de développement.

Une autre méthode qui permet d'obtenir une mesure de confiance pour une hypothèse w est **l'interpolation linéaire**.

$$MC(w) = \sum_{n=1}^{N} c_n MC_n(w), \text{ avec } \sum_{n=1}^{N} c_n = 1 \tag{2.13}$$

Dans (Guo et al., 2004), les coefficients c_n ont été appris de manière à minimiser le taux d'erreur sur un corpus de type *Switchboard*. L'optimisation de ces coefficients a été faite en utilisant une procédure de validation croisée. Cette procédure consiste en un découpage du corpus en plusieurs parties, trois dans ce cas, et l'utilisation de deux corpus pour le calibrage des coefficients et du troisième pour le calcul du taux d'erreur. La

procedure tente ainsi de trouver le vecteur de coefficients qui obtient les meilleures performances sur les trois corpus de tests. Les auteurs combinent une mesure de confiance liée à la notion d'information mutuelle inter-mots (pour un mot elle se calcule comme la moyenne de l'information mutuelle du mot et des autres mots de la phrase) et une mesure de confiance utilisée plus fréquemment, la probabilité *a posteriori*, dont nous parlons au 2.3. Ces deux mesures de confiance étant indépendantes elles peuvent être combinées, l'interpolation linéaire permettant ainsi d'obtenir une amélioration des performances en termes de taux d'égale erreur (voir 2.1.1) de près de 10% par rapport à l'utilisation des deux mesures séparément.

Les **arbres de décision** (Kemp et Schaaf, 1997; Zhang et Rudnicky, 2001; Fu et Du, 2005; Kobus, 2006) sont souvent utilisés dans la reconnaissance de parole afin de prendre une décision binaire *correct/incorrect* sur les hypothèses de reconnaissance. Pour calculer la mesure de confiance d'un mot on y associe un vecteur de paramètres prédictifs. Les critères de décision associés à chaque nœud et les valeurs des paramètres déterminent, grâce à une décision binaire, le chemin à parcourir dans l'arbre de décision pour arriver à une feuille de l'arbre. Chaque feuille est associée à une probabilité que l'hypothèse soit correcte. Du fait de leur construction, les arbres de décision permettent d'avoir une vision sur les interactions des différents paramètres ainsi que sur la contribution de chacun d'entre eux.

Les **réseaux de neurones** (Kemp et Schaaf, 1997; San-Segundo et al., 2001; Charlet et al., 2001) prennent en entrée un vecteur constitué d'un certain nombre de paramètres prédictifs et permettent d'obtenir en sortie une mesure de confiance pour un mot w. En comparaison avec des méthodes de combinaison des paramètres comme l'interpolation linéaire ou les arbres de décision, les réseaux de neurones sont les classifieurs qui donnent les meilleures combinaisons des paramètres.

Une autre technique utilisée pour la combinaison des paramètres est la classification par **SVM** (*support vector machine*) (Zhang et Rudnicky, 2001). Il s'agit de délimiter au mieux deux nuages de points représentant les deux classes *correct/incorrect*. Pour ce faire les SVM utilisent des *fonctions kernel* qui doivent être évaluées afin d'obtenir une mesure de confiance (pour plus de détails voir (Burges, 1998), qui réalise une introduction très explicite de cette approche). Certaines fonctions apportent une amélioration des performances par rapport aux arbres de décision et aux réseaux de neurones. Si pour les réseaux de neurones le calibrage des paramètres utilisés est aisé, il n'en est pas de même pour les SVM. Les *fonctions kernel* ne sont pas très robustes et le moindre changement des paramètres peut produire un résultat sensiblement différent (Zhang et Rudnicky, 2001).

Il existe également d'autres classifieurs comme les modèles linéaires généralisés (*Generalized Linear Model*) présentés dans (Gillick et al., 1997; Siu et al., 1997; Siu et Gish, 1999) ou les méthodes de boosting (Moreno et al., 2001).

Comme nous l'avons vu, les SVM représentent un très bon classifieur avec des per-

formances supérieures aux arbres de décision et aux réseaux de neurones. Néanmoins, ils restent difficile à manier du fait que leur performances dépendent de la *fonction kernel* choisie mais aussi de l'optimisation des paramètres de cette fonction, opération qui peut s'avérer difficile. D'un autre coté, les réseaux de neurones et les arbres de décision utilisant des paramètres prédictifs de langage et acoustiques produisent de moins bonnes performances comparé à la probabilité *a posteriori* sur un graphe de mots (Zhang et Rudnicky, 2001).

2.3 Probabilité *a posteriori*

Comme nous l'avons montré au chapitre 1, un SRAP utilise le critère de *maximum a posteriori* pour trouver la séquence de mots $\hat{W}$ qui maximise la probabilité *a posteriori* $P(W|X)$ étant donné le signal acoustique X (voir l'équation 1.1). En théorie, la probabilité *a posteriori* est une très bonne mesure de confiance, mais cette formule est très difficile à utiliser du fait de la grande variabilité dans l'ensemble de départ des observations acoustiques. Pour cela, en pratique, un SRAP essaie de trouver la séquence de mots $\hat{W}$ qui maximise le produit $P(W) \cdot P(X|W)$ obtenu à l'aide de la formule de Bayes (voir l'équation 1.3). On observe que le terme $P(X)$ a été omis car il est indépendant de la séquence W. Cette méthode de calcul des scores acoustiques explique pourquoi ces scores sont inadaptés en tant que mesure de confiance.
Donc, pour calculer la probabilité *a posteriori* il suffit de normaliser par $P(X)$. En théorie, ce terme ce calcul de la manière suivante :

$$P(X) = \sum_{hyp} P(hyp) \cdot P(X|hyp) \tag{2.14}$$

où hyp représente une hypothèse possible du signal X, et la somme doit se faire sur toutes les hypothèses possibles, ce qui inclut toute combinaison possible de mots, phonèmes, bruits et autres événement acoustique. Il est évident que sans une contrainte particulière, $P(X)$ est très difficile à estimer de manière exacte. En pratique certaines contraintes doivent être imposées ainsi que des méthodes approximatives d'estimation du $P(X)$.
Dans une première catégorie on rencontre les modèles de type *filler* ("*remplissage*") présentés dans (Cox et Rose, 1996; Kampari et Hazen, 2000; Young, 1994). Ces approches peuvent obtenir des performances assez raisonnables. Dans une autre catégorie on rencontre les probabilités basées sur des listes de N-meilleures solutions et ensuite les probabilités basées sur les graphes de mots, qui incluent aussi les réseaux de confusion, calculés suite à des étapes de post traitement. La première catégorie, correspondant aux approches basées sur de modèles de type *filler*, ne fait pas l'objet de travaux dans cette thèse et ne sera détaillée. En revanche, dans cette partie, nous allons détailler les probabilités *a posteriori*, leur calcul et leur application, basées sur les graphes de mots, les listes de N-meilleures hypothèses et les réseaux de confusion.

2.3.1 Approximation par graphes de mots

Un système de reconnaissance automatique de la parole peut produire en sortie un graphe de mots. Comme le montrent les auteurs dans (Wessel et al., 1998, 1999, 2000, 2001; Kemp et Schaaf, 1997; Metze et al., 2000; Goel et al., 2001; Soong et al., 2004) il est possible d'approximer la probabilité *a posteriori* sur un espace plus restreint de solutions qui est le graphe de mots. Lors du décodage, la mémoire utilisée par le système de reconnaissance ainsi que la rapidité d'exécution de l'algorithme sont des facteurs importants à prendre en considération. Pour cela, comme il a été expliqué au chapitre précédent, des techniques d'élagage et de *beam search* sont employées lors de la génération du graphe de mots. De ce fait, le graphe de mots ne peut pas contenir toutes les hypothèses possibles. Néanmoins, l'approximation du terme $P(X)$ sur le graphe de mots reste correcte d'autant plus que le graphe de mots ne contient que les hypothèses les plus probables qui sont dominantes dans le calcul de $P(X)$.
Un graphe de mots G pour un vecteur d'observations acoustiques X est donc constitué de transitions auxquelles sont associées un mot w, son score acoustique $S(w)$ un instant de début et un instant de fin. Dans tout graphe de mots, il existe deux états particuliers : le premier est l'état de début du graphe qui correspond à l'instant de début de l'observation acoustique et l'état de fin qui correspond à l'instant de fin de l'observation acoustique. Tout chemin dans le graphe qui relie ces deux états s'appelle un chemin complet et représente une hypothèse de l'observation acoustique. Si on considère le chemin complet $C = T(w_1, d_1, f_1), T(w_2, d_2, f_2), \ldots, T(w_n, d_n, f_n)$ formé de n transitions, la probabilité de la séquence de mots représentée par ce chemin se calcule de la manière suivante :

$$P(C|G) = \prod_{i=1}^{n} S(w_i)_{d_i}^{f_i} \cdot P(w_i|h_i) \tag{2.15}$$

où h_i représente l'historique du mot w_i pour le chemin C, $P(w_i|h_i)$ représente la probabilité du modèle de langage calculée avec un modèle de type *n-gramme*et $S(w_i)_{d_i}^{f_i}$ représente le score acoustique associé à la transition T ayant pour instant de début d_i et de fin f_i et portant le mot w_i.
En utilisant les notations ci-dessus, pour une transition T du graphe de mots G, la probabilité *a posteriori* $P(T|G)$ se calcule comme étant le rapport entre la probabilité de tous le chemins dans G passant par T et la probabilité de tous le chemins dans G :

$$P(T|G) = \frac{\sum_{C \in G, T \subset C} P(C|G)}{\sum_{C \in G} P(C|G)} \tag{2.16}$$

où $C \in G$ représente un chemin complet dans le graphe G et $T \subset C$ montre que le chemin C contient la transition T. La probabilité *a posteriori* $P(T|G)$ peut être calculée de manière très efficace en utilisant l'algorithme *Forward-Backward* (Baum, 1972). Nous détaillons cet algorithme dans la partie 2.4.
La probabilité *a posteriori* d'une transition T peut être utilisée directement en tant que mesure de confiance pour le mot w porté par la transition. Il a été néanmoins montré (Wessel et al., 2001; Wessel et Ney, 2001) que cet usage est particulièrement inadapté et que les performances sont très limitées. La principale raison pour cela est le fait que dans un graphe de mots, à part la transition $T(w, d, f)$, il existe un nombre assez élevé

de transitions portant le mot w mais ayant des temps de début d et de fin f légèrement différents. La mesure de confiance du mot w est donc sous-estimée si on utilise seulement la probabilité *a posteriori* de la transition T. Il est alors important de prendre en compte les autres transitions portant le mot w et ayant un recouvrement temporel non-nul avec la transition T. Dans (Wessel et al., 2001) trois méthodes différentes sont proposées pour prendre en compte toutes les transitions $T(w, d \pm \varepsilon, f \pm \varepsilon)$ dans le calcul de la mesure de confiance pour le mot w.
Pour la première méthode le calcul de la mesure de confiance, appelée C_{sec}, pour la transition $T(w, d, f)$ s'effectue en sommant les probabilités *a posteriori* de toutes les transitions portant le même mot et ayant un recouvrement temporel non-nul avec la transition T (les transitions portant le même mot n'ont pas forcément un recouvrement temporel non-nul entre elles). Toutefois, avec cette méthode, la somme des probabilités *a posteriori* C_{sec} des différents mots pour une trame donnée entre la trame 0 et la trame de fin du graphe de mots n'est plus égale à 1. Malgré de meilleures performances pour C_{sec} par rapport à l'utilisation des probabilités *a posteriori* calculées avec l'algorithme *Forward-Backward*, le fait de ne pas normaliser cette mesure de confiance peut avoir des influences négatives sur la mesure de confiance. La normalisation devrait se faire de manière à ce que la somme des probabilités *a posteriori* C_{sec} des différents mots pour une trame donnée soit égale à 1. La deuxième méthode proposée essaie de contourner ce problème de normalisation. Le calcul de la probabilité *a posteriori* de la transition T est restreint à la sommation des probabilités *a posteriori* des transitions portant la même hypothèse de mot et dont les supports temporels, incluant celui de la transition T, doivent avoir une trame commune. De cette manière la somme de ces probabilités cumulées pour différentes hypothèses de mot à un instant de temps donné (compris entre 0 et l'instant de fin de l'énoncé) est égale à 1. Ainsi, la nouvelle mesure de confiance pour la transition T, appelée C_{med}, est calculée en sommant sur toutes les transitions portant le mot w qui ont un chevauchement temporel avec la trame médiane de la transition T. Les résultats montrent des performances comparables à C_{sec} et l'absence de normalisation dans le calcul de celle-ci semble ne pas avoir d'influence sur les performances. La troisième méthode se propose de déterminer si le choix de la trame d'intersection à une influence sur la mesure de confiance. Le calcul pour la mesure de confiance, appelée C_{max}, se fait en sommant les probabilités *a posteriori* non seulement pour la trame médiane, comme expliqué pour C_{med}, mais pour toutes les trames du support temporel de la transition T et en choisissant la valeur maximale de ces sommes comme étant la valeur de C_{max}. Cette nouvelle mesure de confiance engendre des performances légèrement meilleures que les deux précédentes. L'évaluation des performances des trois mesures de confiance décrites a été effectuée sur plusieurs corpus de test (ARISE (dialogues homme-machine via un service téléphonique d'information sur les horaires de train), Verbmobil (parole spontanée), Broadcast News (journaux télévisé et radio), NAB (articles lus à partir de différents journaux)) en évaluant le taux d'erreur de confiance (CER). Le taux de référence a été défini comme le rapport entre la somme de nombre d'insertions et de substitutions divisée par le nombre total de mots reconnus.

La probabilité *a posteriori* peut être utilisée aussi dans le domaine de l'apprentissage non-supervisé des modèles acoustiques. Dans (Wessel et Ney, 2001), l'utilisation des

probabilités *a posteriori* en tant que mesure de confiance permet de réduire considérablement la taille du corpus d'apprentissage (les transcriptions manuelles) avec une dégradation moindre des performances du système en termes de taux d'erreur mot.

2.3.2 Approximation par liste de N meilleures hypothèses

Comme pour les graphes de mots, la probabilité *a posteriori* d'un mot w peut également être calculée sur une liste de N meilleures hypothèses (liste N *best*) (Wessel et al., 1999, 2001; Rueber, 1997). Chaque énoncé d'une liste de N *best* est seulement une séquence de mots sans aucune information temporelle sur les temps de début et de fin de chaque mot. Comme décrit dans la section précédente (voir 2.3.1), la relaxation des frontières des hypothèses de mots dans un graphe est très importante pour le calcul d'une mesure de confiance fiable. Un des principaux avantages d'une liste N *best* est alors l'absence de l'information temporelle sur les hypothèses de mots de chaque énoncé. La mesure de confiance d'une hypothèse de mot peut être calculée en se basant seulement sur la position des mots dans les énoncés. Néanmoins, la notion de la position du mot dans la phrase n'est pas très bien définie. Tout d'abord les phrases de la liste N *best* n'ont pas forcement la même longueur. Même dans le cas contraire, les mots ne peuvent pas être comparés directement à cause des erreurs d'insertion et d'omission. Pour ces raisons, les phrases de la liste doivent être alignées en utilisant l'algorithme de Levensthein (Levensthein, 1966). L'algorithme vise à minimiser la somme des insertions, omissions et substitutions dans la comparaison entre deux phrases.
Pour chaque mot w_m se trouvant à la position m dans la phrase w_1^M, l'algorithme peut lui faire correspondre un mot v dans n'importe quelle autre phrase $v_1^{M_1}, \ldots, v_1^{M_N}$ de la liste. On écrit cela de la manière suivante : $v = L_m(w_1^M, v_1^{M_n})$. La fonction $L_m(w_1^M, v_1^{M_n})$ donne le mot v dans la phrase $v_1^{M_n}$ qui correspond au mot w_m de la phrase w_1^M selon l'alignement de Levensthein. Utilisant ces notations, la probabilité *a posteriori* du mot w_m dans la phrase de référence w_1^M se calcule de la manière suivante :

$$p(w_m|X, w_1^M, L) = \frac{\sum_{n=1}^{N} p(X|v_1^{M_n})^{\alpha} p(v_1^{M_n})^{\beta} \delta(w_m, L_m(w_1^M, v_1^{M_n}))}{\sum_{n=1}^{N} p(X|v_1^{M_n})^{\alpha} p(v_1^{M_n})^{\beta}} \tag{2.17}$$

où la fonction de Kronecker δ est égale à 1 si les deux arguments sont identiques et à 0 autrement. α et β sont des facteurs d'échelle qui doivent être optimisés sur un corpus de développement.
Dans (Wessel et al., 2001), les auteurs ont montré que les performances des probabilités *a posteriori* calculées sur une liste N *best* peuvent être similaires ou moins bonnes que celles des probabilités *a posteriori* calculées sur des graphes de mots en fonction de la tâche choisie. Les performances sont mesurées en comparant le taux d'erreur de confiance. Sur les corpus ARISE et NAB, les performances des deux types de probabilités *a posteriori* sont équivalentes alors que sur Verbmobil et Broadcast News les probabilités *a posteriori* sur les graphes de mots, et plus particulièrement C_{max} (voir section précédente), ont des performances très supérieures aux probabilités *a posteriori* calculées sur la liste N *best*. Plusieurs explications ont été avancées à ce sujet. Ainsi, sur ARISE, l'équivalence des performance peut être attribuée à une longueur moyenne de l'énoncé

très courte, d'environ 3.4 mots par énoncé. Même si les corpus NAB et Broadcast News ont des caractéristique similaires (taille du lexique équivalente, longueur moyenne des énoncé très grande) les performances sont loin d'être similaires. Une explication peut résider dans le fait que le premier corpus est enregistré dans des conditions audio nettement supérieures et les énoncés sont lus. Ainsi, du fait d'une meilleure qualité des modèles acoustique et de langage la distribution de probabilité est plus discriminante entre les différentes hypothèses de mot. Ceci mène à l'utilisation de moins d'énoncés dans la liste de *N best* pour le calcul de la mesure de confiance. C'est aussi la combinaison entre la taille de la liste *N best* et les modèles acoustique et de langage qui peut expliquer les moins bonnes performances pour les corpus Verbmobil et Broadcast News.
Dans (Wessel et al., 1999, 2001) il a aussi été montré que la probabilité *a posteriori*, qu'elle soit basée sur les graphes de mots ou sur une liste de *N best*, engendre des performances nettement supérieures, en termes de taux d'erreur de confiance, par rapports aux mesures de confiance, comme la stabilité acoustique ou la densité de l'hypothèse, décrites au 2.2.
Pour une liste *N best*, on serait tenté de dire que plus elle est grande meilleures sont les performances. Ceci n'est pas forcement vrai, ce qui est démontré aussi par une observation intéressante faite par les auteurs dans (Wessel et al., 2001) qui montre la dégradation des performances sur la liste *N best* pour une $N = 1000$ (les autres valeurs de N sont 100, 200 et 300). De plus, l'utilisation des listes *N best* très grandes n'est pas très efficace en termes de complexité et temps de calcul et des ressources nécessaires. Une analyse détaillé des listes *N best* (Wessel et al., 2001) montre aussi que des mots dont les supports temporels sont éloignés sont parfois mis en correspondance. L'algorithme de Levensthein mène parfois à des alignements de mots "*peu raisonnables*" ce qui peut conduire à des problèmes supplémentaires dans le calcul de la probabilité *a posteriori*. L'information sur le temps de début et de fin des mots contenue dans les graphes de mots peut donc s'avérer importante et nécessaire pour une meilleure estimation de la probabilité *a posteriori*.

2.3.3 Approximation par réseaux de confusion

Comme nous l'avons expliqué au 2.3.1, l'utilisation directe de la probabilité *a posteriori* des transitions portant des mots dans un graphe de mots en tant que mesure de confiance du mot peut poser des problèmes de fiabilité. Comme montré ceci est du à l'alignement temporel des différentes hypothèses du même mot et pour cela différentes méthodes qui essaient de calculer une meilleure mesure de confiance sur les mots du graphe ont été présentées dans la littérature et certaines détaillées au 2.3.1. Les réseaux de confusion, de part leur construction, évitent ce problème d'alignement car dans une classe de mots, qui est associée en fait à un intervalle temporel, il ne peut y avoir qu'un seul arc portant un mot w et donc une seule probabilité *a posteriori* du mot dans cet interval du temps. Ce problème est néanmoins présent lors de la construction des CNs, mais cette problématique sera détaillée au chapitre 5. Dans cette partie nous allons seulement décrire l'utilisation des réseaux de confusion, et plus précisément des probabilités *a posteriori* des mots, dans le calcul de mesures de confiance en présentant

quelques exemples décrits dans la littérature.
Dans (Evermann et Woodland, 2000a) par exemple, les auteurs réalisent une comparaison entre les performances des probabilités *a posteriori* dans les graphes de mots et dans les réseaux de confusion. Pour les graphes de mots, une technique de *rescoring* est utilisée. Dans une première étape les probabilités *a posteriori* de transitions du graphe sont calculées à l'aide de l'algorithme *Forward-Backward* et ensuite ces probabilités sont combinées avec les scores acoustiques des transitions. Un simple algorithme A^* est utilisé pour calculer la meilleure solution. Les résultats de cette solution en termes de WER et de NCE(entropie croisée normalisée, voir 2.1.3) sont comparés aux performances de la *consensus hypothesis* sur un corpus de Broadcast News et un corpus contenant des conversations téléphoniques. Les réseaux de confusion se révèlent être plus robustes que la technique de *rescoring* utilisée sur les graphes de mots en produisant des améliorations similaires en termes de WER et SER sur les deux corpus. Si en termes de WER les tests sont assez concluants, en termes de NCE, les résultats montrent que les deux approches (les probabilités *a posteriori* C_{sec} (Wessel et al., 2001) basées sur les graphes de mots et les probabilités *a posteriori* des mots dans les réseaux de confusion) donnent des résultats comparables. Les modèles acoustiques utilisés (quinphone ou triphone) et la taille des graphes de mots font pencher la balance dans un sens ou dans un autre. L'utilisation d'un arbre de décision pour le calcul des scores de confiance améliore les résultats pour les deux approches.
Dans (Xue et Zhao, 2006) on propose de comparer les performances en termes de taux d'erreur de confiance sur les annotations (voir 2.1.2) de trois classifieurs : SVM, arbre de décision et *random forests* (forêts aléatoires). La nouveauté de ces travaux réside dans l'utilisation de nouveaux paramètres issus des CNs dans les *random forests*. Les *random forests* (Breiman, 2001) sont constitués d'un large nombre d'arbres de décision. Afin de classifier un objet, celui-ci est traité par chaque arbre de la forêt, la classification avec le plus grand nombre de votes étant choisie. Une caractéristique importante de ce classificateur est celle de pouvoir estimer quels paramètres sont les plus importants dans le processus de classification.
Dans ces travaux, les *random forests* utilisent des paramètres issus du décodeur de parole (les scores acoustiques et de langage, le score *a posteriori* local d'un mot) et des paramètres issus des CNs (la probabilité *a posteriori* des mots, la probabilité *a posteriori* conditionnelle étant donné le contexte, l'entropie pour les CNs). La probabilité *a posteriori* conditionnelle est calculée étant donné un contexte de type bigramme ($P(w_i|w_{i-1}, X)$) ou de type trigramme ($P(w_i|w_{i-1}, w_{i-2}, X)$). L'entropie pour les CNs mesure les différences entre les probabilités *a posteriori* des mots de la même classe et elle est calculée comme la somme sur tous les mots d'une classe du produit entre la probabilité *a posteriori* du mot et son logarithme. Les évaluations sont réalisées sur des énoncés collectés à partir d'une application de télé-médecine. Les résultats montrent, d'un coté, de meilleures performances des *random forests* par rapport aux SVM et aux arbres de décision, et d'un autre coté l'importance de l'entropie et des probabilités *a posteriori* dans les *random forests*. Le fait d'enlever seulement un des deux paramètres engendre des baisses de performances en terme d'erreur d'annotation. Du fait d'un certain degré de corrélation entre les deux paramètres, l'absence de l'entropie et de la probabilité *a posteriori* en même temps engendre la plus grande baisse des performances par rapport à

l'absence d'autres paramètres.
La combinaison des systèmes est une technique qui est devenue très populaire ces dernières années car elle permet l'obtention d'une solution ayant de meilleures performances que n'importe quelle solution des systèmes de départ. Dans (Evermann et Woodland, 2000b), les auteurs proposent une amélioration de la technique ROVER. A la base, celle-ci utilise des séquences de mots de plusieurs systèmes et une technique de programmation dynamique pour réaliser un alignement des séquences. Grace à cet alignement une décision est prise entre les mots alignés soit par un simple vote, soit en tenant compte aussi des scores de confiance. La limite de la technique ROVER réside dans l'utilisation des séquences de mots et dans la procedure de décision. Ainsi, seule une hypothèse de mot choisie par un des systèmes peut faire partie de la solution finale. L'amélioration proposée par les auteurs permet d'utiliser l'alignement des CNs produits par les systèmes afin de décider sur les hypothèses de mots. Cette technique permet d'obtenir des améliorations en termes de WER ainsi que d'entropie croisée normalisée (*Normal Crossed Entropy - NCE*). Les évaluations ont été effectuées sur des corpus de type Switchboard et CallHome.
Comme nous l'avons décrit, les probabilités *a posteriori* des mots dans un réseau de confusion peuvent être utilisées de manière efficace en tant que mesure de confiance. Une légère tendance des probabilités *a posteriori* des mots de surestimer la probabilité que le mot soit correct à été observée dans (Evermann et Woodland, 2000b). Cet effet a été plus prononcé sur des systèmes ayant des taux d'erreur mot plus grands et sur des graphes de mots de petite taille. Des techniques pour compenser cette surestimation sont proposées comme les arbres de décision ou les réseaux de neurones qui calculent les scores de confiance.
Dans les travaux présentés nous utilisons également les probabilités *a posteriori* de mots dans un réseau de confusion pour le calcul des scores de confiance. Le calcul de la probabilité que le mot soit correct étant donné sa probabilité *a posteriori* sera effectué en utilisant la régression logistique. Les coefficients nécessaires au calcul du score de confiance (voir la section 2.2.4) sont appris sur un corpus de développement. Les scores de confiance ainsi calculés sont ensuite utilisés dans le filtrage de la meilleure solution des CNs en appliquant un seuil sur la valeur des scores de confiance. La séquence de mots ainsi obtenue est ensuite utilisée dans la construction des différentes stratégies d'interprétation qui améliorent les performances du système de dialogue utilisé (voir section 3.1). Les évaluations de l'utilisation de ces scores de confiance sont présentées au dernier chapitre.

2.4 Calcul de la probabilité *a posteriori* sur les graphes de mots

Le calcul de la probabilité *a posteriori* d'une transition dans un graphe de mots est réalisé à l'aide de l'algorithme *Forward-Backward* (Baum, 1972). Cet algorithme a été décrit de manière détaillé dans (Rabiner, 1989; Rabiner et Juang, 1993; Jelinek, 1997) afin de répondre à certaines questions soulevées par l'utilisation des HMM. Toutefois, dans notre cas, l'émission des observations est attachée au transitions et non aux nœuds comme c'est le cas pour les HMM dans (Rabiner, 1989). Nous avons donc du adapter

les formules de calcul décrites dans (Rabiner, 1989) pour le calcul de la probabilité *a posteriori* sur un graphe de mots. Le détail de ces formules est présenté dans la section A.1 de l'annexe A.
Le calcul de la probabilité *a posteriori* de la transition t, se fait en deux temps à travers une procédure "*forward*" (qui définit la variable $\alpha(t)$) qui réalise une passe avant dans le graphe sur les chemins partiels jusqu'à la transition t et une procédure *backward* (qui définit la variable $\beta(t)$) qui réalise une passe arrière sur les chemins partiels partant de la transition t jusqu'à l'état de fin de graphe. Toutefois, le calcul des variables définies par ces procédures pose des problèmes numériques de type *underflow* car on multiplie des variables inférieures à 1. Ces problèmes sont liés à la capacité limitée des ordinateurs à représenter un nombre positif, très proche de 0, avec beaucoup de décimales après la virgule.
Une solution couramment employée dans la littérature est de ne pas prendre en compte les transitions pour lesquelles le calcul de la probabilité *a posteriori* est impossible. Un élagage de ces transitions est alors obligatoire. De plus, cet élagage du à un problème d'*underflow* sur une des variables $\alpha(t)$ ou $\beta(t)$ (les variables sont considérées comme égales à 0, et donc la probabilité *a posteriori* de la transition est égale à 0) rend le calcul des probabilités *a posteriori* approximatif. Par exemple, la variable $\alpha(t) = 0$ implique un élagage de la transition t. Le calcul de $\alpha(t')$, avec t' un successeur de t dans le graphe (l'instant de fin de t est égal à l'instant de début de t'), dépend de $\alpha(t)$ (voir la section A.1). L'élagage de la transition t implique alors que l'ensemble des chemins partiels arrivant dans t est éliminé du calcul de la probabilité *a posteriori* sur t'. Cette méthode est approximative et force un élagage des transitions pour des raisons de calcul numérique. Et cela sans qu'on puisse mesurer son influence sur les performances de graphes de mots.
Une deuxième solution est proposée dans (Rabiner, 1989) et consiste en l'introduction des coefficients de normalisation des variables α et β. Nous avons choisi cette deuxième méthode car l'introduction de ces coefficients permet un calcul exact de la probabilité *a posteriori* en éliminant les problèmes d'*underflow* et sans qu'aucun élagage soit nécessaire. Néanmoins, dans (Rabiner, 1989) cette méthode est décrite dans le contexte de l'utilisation de l'algorithme *Forward-Backward* sur les HMM. Par conséquence, les coefficients sont définis pour une normalisation au niveau de la trame. Leur définition ne peut pas s'appliquer directement aux graphes de mots car les variables α et β sont calculées au niveau de la transition qui s'étend sur plusieurs trames. Nous proposons donc une nouvelle normalisation des variables qui couvre l'ensemble des trames du support temporel de la transition. Nous avons adapté les formules de calcul de α et β pour intégrer cette nouvelle normalisation. Ces formules sont détaillées dans la section A.2 de l'annexe A.

2.5 Conclusions

Dans ce chapitre nous avons tout d'abord présenté les différentes métriques utilisées pour l'évaluation des mesures de confiance. Nous avons ensuite présenté deux catégories de paramètres utilisés dans le calcul des mesures de confiance : les paramètres

prédictifs et la probabilité *a posteriori*. Dans le cas des paramètres prédictifs nous avons détaillé les différents types de paramètres (acoustiques, linguistiques) ainsi que les techniques les plus courantes de combinaison de paramètres. Pour la probabilité *a posteriori* nous avons présenté trois méthodes d'estimation à partir des graphes de mots, liste de *N best* et réseaux de confusion.
Les travaux de cette thèse n'essaient pas de faire une comparaison des différentes mesures de confiance ou d'étudier les performances des différents classifieurs sur les réseaux de confusion. Comme nous l'avons évoqué à plusieurs reprises, la probabilité *a posteriori* peut constituer une très bonne mesure de confiance. De plus, il a été montré aussi que la probabilité *a posteriori* des mots dans un réseau de confusion constitue une meilleure mesure de confiance que la probabilité *a posteriori* des mots dans un graphe (Evermann et Woodland, 2000a). Nous avons donc décidé d'utiliser la probabilité *a posteriori* des mots dans les réseaux de confusion en tant que mesure de confiance. Afin de faciliter les calculs, nous avons choisi la régression logistique pour calculer la probabilité conditionnelle qu'un mot soit correct étant donné sa probabilité *a posteriori*. La régression logistique s'est avérée être assez performante et la phase de calibrage de paramètres facile à mettre en œuvre.
Nous avons également détaillé l'algorithme *Forward-Backward*. Nous avons adapté les formules pour être utilisées dans le calcul de la probabilité *a posteriori* d'une transition dans un graphe de mot. L'implémentation de l'algorithme *Forward-Backward* pose des problèmes d'*underflow*. Une méthode de normalisation par trame des variables de l'algorithme a été proposée dans (Rabiner, 1989). Nous avons modifié cette méthode afin de définir une nouvelle normalisation par transition. Les formules qui intègrent cette nouvelle normalisation ont aussi étaient décrites. Nous avons choisi d'utiliser cette méthode de normalisation car elle permet un calcul exact de la probabilité *a posteriori* et ne nécessite aucun élagage des transitions.

Deuxième partie

Contexte applicatif et problématique

Chapitre 3

Description de l'interaction vocale

Sommaire

Un système de dialogue oral est un système informatique qui assure le dialogue vocal avec un utilisateur humain en vue de fournir un service. L'utilisateur doit pourvoir interagir avec le système de la manière la plus naturelle possible. La communication étant orale le système doit être capable de comprendre non seulement les paroles de l'utilisateur mais plus important leur sens de manière à produire une réponse adéquate et ceci de manière orale.
Les travaux de cette thèse se placent dans le contexte de compréhension de la parole continue à travers l'utilisation d'une application téléphonique de dialogue oral en langage naturel appelée **service 3000**, décrite dans la section 3.1. Ensuite nous décrivons les différents composants d'un système de dialogue dans la section 3.2. Le module de compréhension de la parole est détaillé dans la section 3.3 tel qu'il a été implémenté dans le service 3000. Dans la section 3.4 nous présentons une métrique d'évaluation au niveau interprétation, le taux d'erreur d'interprétation (*IER*) . Une courte description du gestionnaire du dialogue est aussi donnée dans la section 3.5.

3.1 Le service 3000

Le **service 3000** est le premier service déployé à France Télécom acceptant la parole spontanée non contrainte. Il a été mis en service en Octobre 2005. Ce service permet aux clients de France Télécom d'obtenir des renseignements, de souscrire à environ 30 services liés à leur ligne téléphonique, ou bien d'accéder à des services dédiés comme la consultation de la consommation, le paiement de la facture ou l'activation d'un transfert d'appel.
Etant donné que le service 3000 fonctionne dans des conditions réelles, les utilisateurs peuvent appeler de n'importe où et leur environnement peut être plus ou moins bruyant. Pour cela, le service 3000 utilise un module de détection Bruit/Parole. Ce module est placé avant le SRAP et a pour but d'éviter les activations intempestives du SRAP dues aux bruits environnants. Donc si le signal reçu par ce module ne contient que du bruit, ce module est censé le détecter et le rejeter. De cette façon le module de reconnaissance n'est censé recevoir qu'un signal contenant de la parole. En réalité, le calibrage du module de détection ne peut pas être parfait, et des signaux ne contenant que du bruit seront envoyés au SRAP. Comme nous l'expliquons dans les parties suivantes, les signaux bruités sont une caractéristique à prendre en considération lorsqu'on travaille avec des corpus réels. Le calibrage du module de détection Bruit/Parole n'est pas concerné par les travaux de cette thèse et ne sera pas abordé.

3.2 Systèmes de dialogue oral

La majorité des systèmes de dialogue travaillent de manière séquentielle grâce à un enchaînement de modules spécialisés qui les composent comme montré dans la figure 3.1. En général, il s'agit de quatre types de modules :

- `LE MODULE DE RECONNAISSANCE DE LA PAROLE` : est en charge de la réalisation de la transcription du signal de parole produit par l'utilisateur. Cette représentation textuelle est ensuite fournie au module suivant pour extraire le sens du message de l'utilisateur.
- `LE MODULE DE COMPREHENSION` : il se base sur la transcription réalisée par le SRAP afin de trouver un sens aux paroles de l'utilisateur. Le processus d'interprétation est réalisé en deux étapes successives et met en œuvre différentes techniques qui permettent de passer des mots au sens. Dans un premier temps, le module réalise un passage des mots vers des concepts et ensuite, à partir des concepts et des connaissances sur le dialogue en cours, construit une représentation sémantique qui sera exploitée par le module suivant, qui est le gestionnaire de dialogue. Le module de compréhension est présenté plus en détail dans la partie 3.3.
- `LE GESTIONNAIRE DE DIALOGUE` : il est chargé d'assurer le bon déroulement du dialogue. Il se base sur la représentation sémantique du sens du message de l'utilisateur (fournie par le module de compréhension) mais aussi sur l'historique du dialogue pour prendre les décisions sur les actions à entreprendre. L'action

peut être d'interroger une base de données si l'utilisateur à émis une requête, d'orienter l'utilisateur vers un autre service, de demander des précisions supplémentaires sur une requête émise ou bien de demander à l'utilisateur de répéter si le système ne l'a pas compris la première fois.

– LE MODULE DE SYNTHÈSE DE PAROLE : le synthétiseur de parole doit transformer la réponse textuelle du gestionnaire de dialogue en un signal de parole afin que le système puisse converser de manière orale et naturelle avec l'utilisateur. Cette partie ne rentre pas dans les considérations de cette thèse, pour plus d'informations voir par exemple (Sorin et De Mori, 1998).

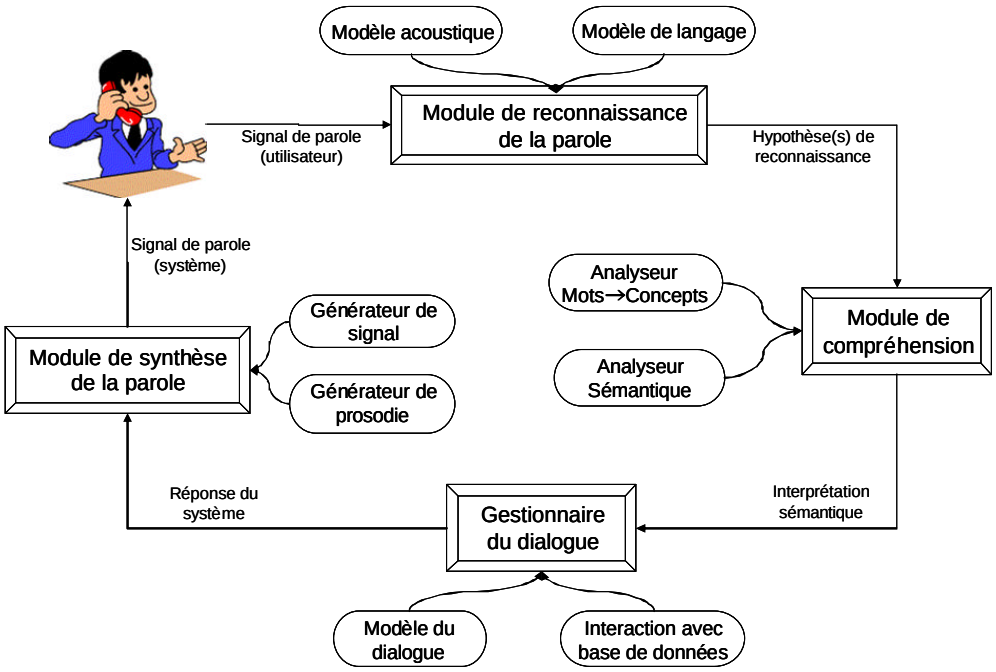

FIGURE *3.1* – Diagramme des modules d'un système de dialogue

3.3 Compréhension de la parole

Le but d'un module de compréhension est d'extraire le sens d'un signal de parole afin de pouvoir interagir avec l'utilisateur du système. Actuellement, seuls les systèmes ayant un domaine limité peuvent être construits. Ceci est du au fait que seule une telle limitation autorise la construction des modèles spécifiques et une description sémantique complète permettant la création d'un système de dialogue robuste et utilisable. Pour ce faire il existe des méthodes manuelles (Minker et Bennacef, 1996) ainsi que des méthodes statistiques (Pieraccini et Levin, 1993; Riccardi et Gorin, 1998). Ces dernières permettent de réduire fortement l'intervention humaine dans le développement du module de compréhension.
En pratique, le but est de transformer le signal de parole en une structure sémantique qui puisse être utilisée par le gestionnaire de dialogue (DM) pour décider de l'action

à entreprendre. Cette analyse complexe doit être effectuée en dépit des difficultés inhérentes à un système de dialogue en langage naturel. On retrouve ainsi les difficultés posées par le langage naturel, parlé de façon spontanée, qui ne respecte pas forcement la grammaire de l'écrit et qui comporte des hésitations, des répétitions, etc. Toutes ces particularités peuvent être groupées sous le nom de disfluences, nom qui sera utilisé dans la suite de la rédaction pour désigner les caractéristiques du langage naturel spontané. Un autre aspect qui peut augmenter la complexité de cette analyse sémantique résulte des erreurs de reconnaissance de la parole effectuées par le SRAP. Pour cela la conception d'un analyseur sémantique robuste est nécessaire au bon fonctionnement d'un système de dialogue. Bien qu'une interprétation puisse être obtenue à partir d'une analyse syntaxique (Roark, 2002) l'extraction sémantique se limite en général aux éléments porteurs de sens du message tout en ignorant les parties redondantes ou non-essentielles à l'application (des parties qui ne sont pas couvertes par le domaine restreint de l'application).
Le fonctionnement du module de compréhension du service 3000 est basé sur ce principe de n'utiliser que les éléments porteurs de sens dans l'extraction du sens du message de parole. L'extraction de l'interprétation se fait en deux étapes successives à l'aide de deux ensembles distincts de règles manuelles qui ont été écrites de façon à couvrir au mieux le domaine de l'application. Cette extraction en deux étapes fait intervenir un objet intermédiaire entre les mots et l'interprétation elle-même, appelé **concept**. Nous présentons ci-dessous les deux étapes nécessaires pour obtenir l'interprétation du message : la première consiste en une analyse des mots vers les concepts, détaillé au 3.3.1 et la deuxième en une analyse sémantique pour obtenir une interprétation à partir des concepts. Cette étape est détaillée au 3.3.2.

3.3.1 Analyseur Mots $\longrightarrow$ Concepts

Le service 3000 est un système de dialogue en langage naturel multi-utilisateur. Un des paramètres importants pour ce système est la multitude des utilisateurs, et donc une grande diversité dans la façon d'exprimer la même idée. Il est important pour le module de compréhension de pouvoir prendre en compte ces variations naturelles de la manière d'exprimer une idée. L'exemple suivant montre justement cette variété. Il s'agit ici de demander au système un transfert des appels vers un deuxième numéro pré-enregistré.

> "*transférer mes appels vers le deuxième numéro*"
> "*je désire transférer mes appels vers le deuxième numéro*"
> "*je vous appelle pour transférer mes appels vers mon deuxième numéro*"
> "*je pars en vacances pour quelques semaines et je voudrais transférer mes appels vers mon deuxième numéro*"

L'idée de départ est assez simple, mais selon l'utilisateur la manière de verbaliser cette idée varie beaucoup. On a d'un coté l'utilisateur qui "va droit au but" dans le premier exemple mais aussi un utilisateur, dans le dernier exemple, qui réalise un requête très détaillée sans qu'elle soit pertinente pour le système dans son intégralité.

L'introduction des concepts permet d'avoir une vision conceptuelle du langage et d'unifier le plus possible les variations du langage. Cette vision conceptuelle du langage permet aussi de ne modéliser que les idées qui appartiennent au domaine de l'application. De cette manière les parties non-essentielles à l'application peuvent être écartées.
Le passage d'une séquence de mots à une séquence de concepts se réalise à l'aide d'une liste de règles qui contient des associations permettant le passage de un ou plusieurs mots à un concept. Lorsqu'un mot ou une séquence de mots correspond à un concept on dit que le concept est allumé par le mot ou la séquence de mots. Pour reprendre l'exemple précédent, la liste utilisée contiendrait les règles suivantes :

transférer $\longrightarrow$ [Transférer]
appels $\longrightarrow$ [Appel]
vers $\longrightarrow$ [Pour]
deuxième $\longrightarrow$ [Deuxième]
numéro $\longrightarrow$ [Numéro]

et le résultat serait le même pour les quatre messages :"*[Transférer] [Appel] [Pour] [Deuxième] [Numéro]*". On peut observer que les règles d'allumage de concepts ont permis d'éliminer les parties non-essentielles des messages comme par exemple "*je pars en vacances pour quelques semaines*".
La liste utilisée par le service 3000 contient **1200 règles** d'allumage de concepts pour un total d'environ **400 concepts**. On observe un nombre de concepts largement inférieur au nombre de règles ce qui illustre bien les variations de langage chez les utilisateurs. Cette liste contient donc des associations entre des séquences de mots allant jusqu'à 4 mots et des concepts. Plusieurs de ces séquences peuvent allumer un même concept mais aucune séquence ne peut allumer plusieurs concepts en même temps. Voici quelques exemples de règles d'allumage de concepts :

"comment connaître" $\longrightarrow$ [Connaître]
"je ne suis pas" $\longrightarrow$ [SuisPlus]
"mot de passe" $\longrightarrow$ [CodeConfidentiel]
"choix" $\longrightarrow$ [Choix]

Le principe du module de compréhension est de n'utiliser que les mots ou les séquences de mots porteurs de sens dans le processus d'interprétation. Mais l'utilisateur ne s'exprime pas en utilisant seulement ces mots ou séquences de mots. Ainsi, le lexique d'une application doit aussi contenir des mots non porteurs de sens pour l'application mais qui sont employés par l'utilisateur. Nous faisons la distinction entre deux types de mots :

- Les mots **non-vides** sont les mots porteurs de sens qui rentrent dans la composition d'au moins une règle d'allumage de concept.
- Les mots **vides** sont les mots qui ne sont pas porteurs de sens pour l'application. Un mot vide est un mot qui ne rentre dans la composition d'aucune règle d'allumage de concept.

Cette catégorisation des mots du lexique est utilisée par la suite pour améliorer les performances des algorithmes de génération de réseaux de confusion.
En pratique, pour transformer une séquence de mots en séquence de concepts le module de compréhension cherche les concepts en partant des séquences les plus longues

pour finir sur les concepts allumés par un seul mot. Prenons la séquence de mots "$w_1, w_2, \ldots w_6$" et une liste qui contient les règles suivantes : "$w_1 \longrightarrow c_1; w_1 w_2 \longrightarrow c_2; w_3 w_5 \longrightarrow c_3; w_3 \longrightarrow c_4; w_6 \longrightarrow c_5$" où c_i représente des concepts. Le module commence un parsing de la séquence de mots par le mot w_1. Il peut allumer tout seul le concept c_1 mais il fait aussi partie d'une séquence plus longue. Il regarde ensuite si la séquence $w_1 w_2$ allume un concept (ici le concept c_2) ou fait partie d'une séquence encore plus longue (ce n'est pas le cas dans cet exemple). Le module décide donc que la séquence $w_1 w_2$ allume le concept c_2. Le parsing se poursuit de la même manière avec le mot w_3 qui va allumer le concept c_4 (w_3 fait partie d'une séquence plus longue mais celle-ci n'est pas présente dans notre séquence). A la fin du parsing, la séquence de concepts obtenue sera "$c_2 c_4 c_5$".

3.3.2 Analyseur sémantique

Dans cette étape le module de compréhension essaie de trouver un sens au message de parole en utilisant la séquence de concepts obtenue à l'étape précédente et une liste ordonnée de règles. Le résultat de cette étape, que nous allons appeler *interprétation*, se présente sous la forme d'une composition de paires attribut-valeurs ($Attribut(Valeur1, Valeur2, \ldots)$), comme par exemple *Gest(Désactiver,TransfertAppel)*.
Pour obtenir cette interprétation, le module de compréhension utilise une liste ordonnée de règles d'interprétation. Une règle d'interprétation est en fait une combinaison logique de plusieurs concepts. Il existe trois opérateurs logiques qui peuvent rentrer dans la composition de ces règles : le OU ("|"), le ET ("&") et le ET sans ordre ("#"). Ce dernier opérateur fonctionne comme un ET sauf que les concepts à gauche et à droite de l'opérateur peuvent se trouver dans n'importe quel ordre dans la séquence à interpréter. Dans l'exemple suivant (I_i est une interprétation), pour la première règle c_2 doit se trouver après c_1 dans la séquence de concepts, alors que pour la deuxième règle l'ordre dans la séquence de concepts ne compte pas. On peut aussi bien avoir c_1 suivi de c_2 ou l'inverse, la règle sera activée dans les deux cas.

$$c_1 \,\&\, c_2 \longrightarrow I_1$$
$$c_1 \,\#\, c_2 \longrightarrow I_2$$

On dit que la liste est ordonnée car les règles sont présentées et traitées selon leur priorité. On attribue une priorité à chaque règle, car contrairement à la première étape (le passage de mots aux concepts) une séquence de concepts peut activer plusieurs règles d'interprétation. Ceci est du au fait que l'étape d'interprétation n'est pas sensible aux insertions de concepts. Si on reprend l'exemple précédent, on pourrait croire que seule la séquence $c_1 c_2$ peut activer la première règle. Mais ce n'est pas le cas, car toute séquence de type $c_1 c_3 \ldots c_N c_2$ pourrait activer cette règle. Une telle séquence n'active pas seulement cette règle mais aussi celles qui contiennent d'autres concepts présents dans la séquence. D'où la nécessité d'avoir une liste ordonnée de règles. Dès qu'une règle est activée par la séquence de concepts, l'analyseur choisi l'interprétation correspondante est arrête l'analyse.
Prenons, par exemple, la séquence de concepts "Message Express Consulter Abonne-

ment" et la liste de règles suivante (les règles sont données dans l'ordre de leur priorité) :

(Abonnement # (Message & Express))⟶ Abon(MessageExpress)
(Message & Express)⟶ Serv(MessageExpress)
(Abonnement # ((Message | Boite) # (Express | Perso)))⟶
Abon(DescripOK(MessageExpress))

On peut observer que le concept *Consulter* ne rentre dans aucune des règles. Ceci n'empêche pas le fait que la séquence active la première règle, car *Message* et *Express* sont dans le bon ordre (ça aurait été le cas même si un ou plusieurs concepts été insérés entre les deux) et le concept *Abonnement* est présent dans la séquence. Le # (ET sans ordre) permet l'activation de la règle même si l'ordre dans la séquence de concepts et dans la règle n'est pas le même. De même, les deux règles qui suivent sont aussi activées par la séquence de concepts, mais seule la première est validée en raison de sa priorité plus élevée et l'interprétation choisie est *Abon(MessageExpress)*. L'interprétation choisie est ensuite transmise au gestionnaire de dialogue afin de pouvoir décider de la suite de dialogue avec l'utilisateur.
La liste utilisée pour cette étape est formée d'approximativement 3200 règles d'interprétation qui produisent 2030 interprétations distinctes. Le nombre d'interprétations est plus petit que celui des règles car plusieurs règles peuvent activer la même interprétation :

(Ouvrir & Ligne)⟶ {Serv(DixQuatorze)}
(Modifier & Nom)⟶ {Serv(DixQuatorze)}

Cette situation montre bien les variations dans la manière d'exprimer la même idée par des utilisateurs différents.
La figure 3.2 est un résumé de la modélisation utilisé dans le service 3000, que ce soit au niveau SRAP (détaillé au chapitre 4) ou au niveau du module de compréhension.

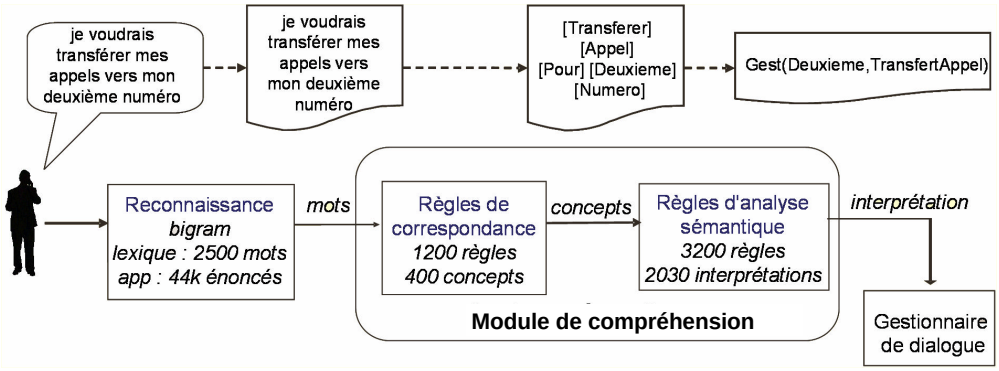

FIGURE 3.2 – Le service 3000

3.3.3 Projet LUNA

Le projet européen LUNA (Spoken **L**anguage **UN**derstanding in Multilingu**A**l Communication Systems) aborde le problème de la compréhension en temps-réel du lan-

gage naturel dans le contexte des services de télécommunications. L'objectif principal du projet LUNA est la création d'un toolkit robuste pour la compréhension de la parole spontanée et naturelle dans le contexte des services de dialogues multilingues capables de réaliser une conversation homme-machine avec un bon degré de satisfaction de la part de l'utilisateur.
Du point de vue technologique, les objectifs du LUNA sont de proposer de nouvelles méthodes, algorithmes et outils pour une mise en œuvre plus rapide d'un système robuste de compréhension de la parole pour les services téléphoniques multi-langues. Pour ce faire, le projet LUNA se concentre sur cinq objectifs importants :

- Modélisation de langage pour la compréhension de la parole
- Modélisation sémantique pour la compréhension de la parole
- Apprentissage non-supervisé
- Problèmes de robustesse pour les systèmes de compréhension de la parole
- Portabilité des langues pour les composants des systèmes de compréhension de la parole

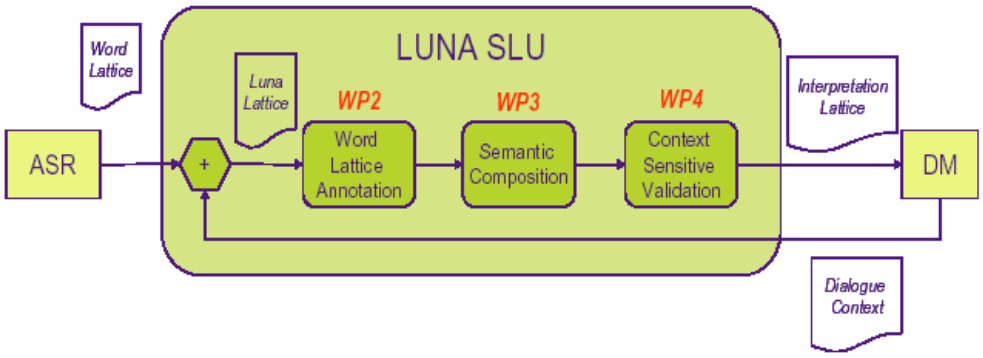

FIGURE 3.3 – Diagramme des modules du système de dialogue LUNA

Le module de compréhension de la parole présenté au 3.2 utilise une seule séquence de mots fournie par le SRAP pour produire une interprétation du signal de parole. Mais un SRAP peut aussi produire des graphes de mots qui sont, certainement, plus complexes à traiter, mais contiennent plus d'informations qu'une simple séquence de mots. Le projet LUNA est axé sur l'utilisation et l'exploitation des graphes de mots dans son système de dialogue. Ainsi, comme montré dans la figure 3.3, le module de compréhension ne traite plus une seule séquence de mots mais, bien un graphe de mots afin d'obtenir une liste d'interprétations à partir de laquelle le système extrait la meilleure interprétation selon différents critères. Cette interprétation sera ensuite fournie au gestionnaire du dialogue. Une partie des travaux expérimentaux de cette thèse (Minescu et al., 2007) est effectuée dans le contexte du projet LUNA.
L'implementation des deux étapes du module de compréhension n'est pas la même selon que l'on utilise une séquence de mots ou un graphe de mots. Ainsi, pour le traitement des graphes de mots, le module de compréhension de la parole utilise les Automates à États Finis (*Finite State Machine - FSM*)(Mohri, 1996, 1997), implémentés à l'aide des outils d'AT&T FSM Library (Pereira et Riley, 1997; Mohri et al., 1998, 2002). Ensuite les différentes propriétés de composition et projection des automates à états finis sont utilisées afin d'obtenir une interprétation. Dans un premier temps, un graphe

de concepts est obtenu à partir d'un graphe de mots à travers une composition avec un transducteur représentant les règles d'allumage de concepts (Raymond et al., 2006; Damnati et al., 2007). Ensuite, les règles d'interprétation sont aussi mises sous la forme d'un transducteur ce qui permet, à travers une composition avec le graphe de concepts, d'obtenir un numéro d'identification de la règle d'interprétation (qui correspond à la position de la règle dans la liste ordonnée des règles d'interprétation). Nous appelons ce processus de calcul de l'interprétation, à partir des graphes de mot, une recherche intégrée sur les graphes de mots. Pour une séquence de mots, l'implementation est équivalente à celle décrite au 3.3.
Voir http ://www.research.att.com/sw/tools/fsm/ pour plus de détails sur le fonctionnement des FSM. La construction et le fonctionnement du module de compréhension implémentés à l'aide des outils d'AT&T sont détaillés dans (Raymond, 2005).

3.4 Evaluation au niveau interprétation

Du point de vue de l'utilisateur, l'hypothèse de reconnaissance ainsi que la séquence de concepts correspondante sont transparentes et les éventuelles erreurs ne sont pas perçues directement par celui-ci. L'interprétation est celle qui guide la réponse du gestionnaire de dialogue et ce sont les erreurs au niveau interprétation qui reflètent le mieux les performances du système perçues par l'utilisateur. L'évaluation au niveau mot, comme le *WER*, ne reflète pas les performances du système d'un point de vue de l'utilisateur. Afin d'avoir une évaluation qui reflète au mieux ce point de vue, nous évaluons le taux d'erreur interprétation (*Interpretation Error Rate - IER*).
Comme nous l'avons montré au 3.3.2, une interprétation se présente sous la forme d'une composition de paires attribut-valeurs ($Attribut(Valeur1, Valeur2, \ldots)$), comme par exemple *Gest(Désactiver,TransfertAppel)*. Une interprétation est considérée comme étant correcte si tous les éléments qui la composent sont corrects. Par exemple :

$$Gest(\text{Désactiver}, TransfertAppel) \neq Tarif(TransfertAppel)$$

parce que l'attribut est différent et, en plus, les valeurs ne sont par les même. Cette erreur compte alors pour une substitution car on analyse l'interprétation dans son ensemble, comme s'il s'agissait d'une séquence de caractères. On distingue ainsi trois types d'erreurs au niveau interprétation :

- *Fausse Alarme (FA)* quand une hypothèse d'interprétation valide est trouvée pour un énoncé dont la référence ne donne lieu à aucune interprétation valide.
- *Substitution (Sub)* quand une hypothèse d'interprétation est différente de l'interprétation de référence.
- *Faux Rejet (FR)* quand aucune hypothèse d'interprétation n'est trouvée alors que la référence produit une interprétation valide.

L'interprétation de référence d'un énoncé est calculée à partir de la transcription de référence de l'énoncés au niveau mot, qui est ensuite soumise au processus d'interprétation réalisé par le module de compréhension. Pour une séquence de mots, l'analyse sémantique peut soit produire une interprétation valide (activée par une des règles de l'analyse sémantique), soit ne produire aucune interprétation, auquel cas, on considère

que le module de compréhension génère un rejet de la séquence de mots (l'interprétation "Rejet" est activée si aucune des règles d'interprétation n'a été activée par la séquence de mots).
Le taux d'erreur d'interprétation se calcule comme suit :

$$IER = \frac{FA + FR + Sub}{\text{Nombre d'énoncés interprétables.}} \tag{3.1}$$

Le nombre d'énoncés interprétables représente le nombre d'énoncés pour lesquels le processus d'interprétation trouve une interprétation valide pour la référence.

3.5 Gestionnaire de dialogue

Le module de gestion de dialogue ne fait pas l'objet des travaux de cette thèse. Néanmoins, dans cette partie, nous présentons de manière très succincte son fonctionnement afin de donner une vision la plus complète possible sur le fonctionnement du service 3000.
Le gestionnaire de dialogue (DM), qui constitue le troisième module d'un système de dialogue, est représenté sous la forme d'un automate à états finis. Il peut emprunter plusieurs états entre deux tours de parole et les états qui engendrent un message du système sont appelés phases et sont au nombre de 137. Pour chaque phase est défini l'ensemble des interprétations qui permettent d'emprunter une transition dans l'automate des états de dialogue. Ainsi, la notion d'interprétation attendue pour une phase est donnée par la connaissance de cet ensemble. La reconnaissance d'une interprétation non-attendue entraîne une réaction d'incompréhension de la part du système.

3.6 Conclusions

Dans ce chapitre nous avons tout d'abord présenté le service 3000, application qui constitue le cadre applicatif principal de ces travaux. Nous avons ensuite présenté les différents modules d'un système de dialogue, en détaillant tout particulièrement le module de compréhension de la parole. Au niveau du service 3000, ce module est implémenté à travers deux étapes successives qui assurent les passages de mots en concepts suivi de l'obtention d'une interprétation. Nous avons aussi décrit le contexte applicatif du projet européen LUNA qui est axé sur l'exploitation des graphes de mots au lieu des séquences de mots pour l'obtention d'une liste d'interprétations. Une partie des travaux de cette thèse sont effectués dans ce contexte. Nous avons également présenté une métrique d'évaluation au niveau interprétation, le taux d'erreur d'interprétation *IER*. Une brève description du gestionnaire du dialogue a aussi été donnée.

Chapitre 4

Cadre expérimental

Sommaire

Ce chapitre introduit le cadre expérimental défini par l'utilisation du service 3000 ainsi que les différentes problématiques liées à l'utilisation des corpus réels. Le chapitre est divisé en cinq sections. Dans la première section 4.1, nous faisons une analyse des énoncés utilisateur et nous posons les problématiques liées au comportement du système sur différents types d'énoncés. Dans la section 4.2, nous présentons les modèles de langage et acoustique utilisés par le SRAP ainsi que le lexique de l'application. Une description des données expérimentales (corpus d'apprentissage, de développement et de test) est réalisée dans la section 4.3. La section 4.4 présente les problèmes liées à l'évaluation au niveau mot des corpus réels et décrit plusieurs méthodes de normalisation des données. La dernière section 4.5 est consacrée à l'évaluation au niveau interprétation sur le corpus de test.

4.1 Caractérisation des énoncés utilisateur

Pour le service 3000 on distingue deux types de dialogues. Certains utilisateurs appellent afin d'activer des services auxquels ils ont déjà souscrit, comme le nombre de minutes utilisées ou le transfert de leur numéro de téléphone. Pour ce type de demande l'utilisateur est, ensuite, transféré vers un service vocal dédié à ces tâches spécifiques. Dans ce cas, le service 3000 peut être vu comme un service de routage qui redirige,

de manière efficace, les utilisateurs, en exploitant éventuellement leur profil utilisateur pour des informations complémentaires. Ces utilisateurs sont assez familiers avec le système et l'utilisent régulièrement. Ils sont plus prédisposés à utiliser des phrases courtes ou seulement des mots clés et l'interaction avec le système est rapide (entre deux et trois tours de parole avant d'être redirigés vers le service demandé). Ce type de dialogue représente 80% des appels sur le service 3000. Pour les 20% restants, l'interaction avec l'utilisateur est entièrement gérée par le service 3000 lui même. Ces dialogues proviennent généralement des utilisateurs demandant des informations sur certains services ou des utilisateurs qui cherchent à s'abonner à un nouveau service. Pour ce type de dialogue, la longueur moyenne de la phrase est plus élevée ainsi que le nombre moyen de tours de parole. Les utilisateurs sont moins familiers avec l'application et le nombre moyen de disfluences ainsi que de mots hors vocabulaire (les mots qui ne sont pas couverts par le lexique de l'application) augmente.
Un autre aspect important pour ce deuxième type de dialogue est le taux assez élevé de commentaires de la part des utilisateurs. Un commentaire est une phrase prononcée par l'utilisateur qui est considérée comme étant hors du domaine de l'application. On distingue plusieurs catégories de commentaires. L'utilisateur peut se parler à lui même ("*oh non, j'en ai marre de ce truc*"), peut être surpris ("*qu'est-ce que je dois dire maintenant ?*"), peut être énervé (*j'ai déjà dit ça*") ou peut même insulter le système. On distingue aussi un type de commentaire que nous appelons "*aparté*", lorsque l'utilisateur parle à une tierce personne se trouvant à coté de lui ("*va ranger ta chambre*"). Les commentaires peuvent constituer une phrase entière mais il existe aussi des commentaires qui peuvent se retrouver avec de l'information utile dans le même énoncé.
Le problème se pose lorsque le système essaie de décoder et d'interpréter ces phrases. Comme elles sont hors du domaine de l'application, une partie des mots utilisés sont des mots hors vocabulaire, mais plus important, le module de compréhension n'est pas capable de les interpréter. Dans le meilleur des cas, une phrase qui est un commentaire génère une incompréhension de la part du système qui demandera à l'utilisateur de répéter. Mais il peut y avoir aussi le cas opposé, où une telle phrase génère une interprétation (les mots reconnus peuvent allumer des concepts auxquels correspond une règle d'interprétation) ce qui peut conduire le DM dans une mauvaise direction. Ce dernier cas oblige l'utilisateur à revenir en arrière et à répéter sa demande et donne l'impression que le système a commis une erreur, ce qui n'est pas le cas. Il existe aussi la possibilité qu'une phrase contenant des commentaires mais aussi de l'information utile ne puisse pas être interprétée par le système qui génère une réponse d'incompréhension. Dans tous ces cas, un autre problème majeur outre le comportement du système est le traitement de la phrase prononcée par l'utilisateur. Nous allons montrer que ces traitements peuvent être très complexes et coûteux en termes de temps de calcul et de ressources utilisées et nous allons proposer, dans le chapitre 7, différentes stratégies afin de contourner ce problème.
Afin de mieux analyser un système de compréhension de la parole, il est important de ne pas se contenter d'évaluer les performances du système dans son ensemble mais plutôt d'observer son comportement sur les différents types de messages auxquels il est confronté. La première grande distinction concerne les messages qui contiennent une interprétation valide et ceux qui n'en contiennent pas. Les messages qui ne contiennent

pas d'interprétation valide sont considérés, du point de vue du système, comme étant à rejeter. Une deuxième grande distinction concerne la présence des commentaires. Puisque les commentaires ne sont pas couverts par le domaine de l'application nous allons appeler les séquences de mots qui forment les commentaires, de la parole *Hors-Domaine*, et par conséquence tous ce qui n'est pas un commentaire est appelé parole *Dans-le-Domaine*. La parole *Hors-Domaine*, de par sa définition, ne peut pas donner lieu à une interprétation valide et fait donc partie des énoncés à rejeter. Suite à ces observations nous pouvons distinguer trois catégories de messages, dont les deux premières sont des messages à rejeter :

- **C1-Non-Parole**- Les énoncés non-parole. Tous les énoncés ne contenant que du bruit, admis à tort par le module de détection Bruit/Parole. Ce sont des énoncés qui ne donnent lieu à aucune interprétation valide et sont donc à rejeter.
- **C2-Hors-Domaine**- La parole *Hors-Domaine*. Tous les énoncés qui contiennent des commentaires.
- **C3-Dans-le-Domaine**- La parole *Dans-le-Domaine*. Tous les énoncés qui ne rentrent pas dans le deux premières catégories.

4.2 Modèles utilisés par le service 3000

Le système de reconnaissance de la parole (SRAP) du service 3000 utilise un lexique de 2548 mots. Comme nous l'avons expliqué au 3.3, le lexique est formé de deux types de mots : les mots **non-vides**, qui sont en proportion de 44% dans le lexique et les mots **vides** qui constituent le reste du lexique.
Comme expliqué au 3.1, le SRAP est précédé par un module de détection Bruit/Parole. Ce module à pour but d'éviter les activations intempestives du SRAP dues aux bruits environnants. Afin de détecter et rejeter les énoncés ne contenant que du bruit qui n'ont pas été détectés par le module Bruit/Parole, le SRAP utilise un sous-modèle de détection (rejet) des bruits. Afin de détecter les mots hors-vocabulaire (*OOV - Out of Vocabulary*), un modèle phonétique contraint par la longueur (Hamimed et Damnati, 2002) est utilisé. Le SRAP utilise également un modèle phonétique avec des phonèmes hors-contexte afin de détecter les faux départs, que nous notons *SPR* (*Speech Repair*) dans ce document.
Le modèle de langage utilisé est un modèle de type bi-classes dont la majorité des classes sont constituées d'un seul mot (voir 1.3.2 pour la description de ce type de modèle). Afin de détecter les commentaires, un sous-modèle de langage (Damnati et al., 2007) est inclut dans le modèle principal. Pour construire ce sous-modèle, les commentaires ont été annotés manuellement sur un corpus d'apprentissage afin de pouvoir séparer les segments *Hors-Domaine* des segments *Dans-le-Domaine*. Les commentaires de type *aparté* (voir la section 4.1) peuvent être très diversifiés et ne sont pas modélisés par ce modèle de langage. Les autres commentaires qui sont adressés au système ont tendance à être assez redondants et donc plus facile à modéliser. Le corpus d'apprentissage pour le sous-modèle de détection de la parole *Hors-Domaine* contient 1712 séquences ayant de la parole *Hors-Domaine* et le lexique utilisé est de 765 mots.
Le sous-modèle, appelé LM^{HD} a été intégré au modèle général LM^{G}. Dans le corpus

d'apprentissage du modèle général (voir la section 4.3), les commentaires ont été annotés et remplacés par l'étiquette <*COMMENTAIRE*>. Cette étiquette est rajoutée ensuite dans le bigramme général et les probabilités $P(< COMMENTAIRE > |w)$ et $P(w| < COMMENTAIRE >)$ sont apprises en même temps que les autres probabilités du bigramme (suivant le principe des classes *a priori*). Pendant le décodage, les probabilités du modèle général et celles du sous-modèle LM^{HD} sont combinées (voir 1.3.3 pour plus de détails sur l'imbrication des modèles). Pour certaines évaluations, comme celles présentées dans les chapitres 5 et 6, le sous-modèle de détection de commentaires n'est pas intégré au modèle général. Les mots ou séquences de mots qui auraient été détectées par ce sous-modèle sont détectés alors comme tout autre mot par le SRAP en utilisant seulement le modèle de langage général. Ces mots ou séquences de mots ne sont donc pas idéntifiés comme un commentaire.

4.3 Description des données expérimentales

L'avantage d'utiliser un service déployé comme le 3000 est de pouvoir travailler sur des énoncés issus des utilisateurs qui se trouvent en situation réelle d'utilisation du système. Tous les corpus utilisés sont formés des énoncés issus des utilisateurs faisant appel au service et collectés sur différentes périodes de temps.

Données d'apprentissage

Les données d'apprentissage du modèle de langage se présentent sous la forme d'un corpus constitué de transcriptions de phrases prononcées par les utilisateurs de l'application. Le corpus d'apprentissage est constitué de 44000 énoncés.
Ce corpus a été utilisé pour l'apprentissage du bigramme mais aussi pour l'apprentissage du sous-modèle de détection des commentaires. Pour ce dernier, les corpus a été annoté manuellement pour différentier les commentaires du reste de l'énoncé.
Par la suite, le corpus d'apprentissage sera noté **App**.

Données de développement et de test

Plusieurs corpus constituent les données de développement et de test. Chaque énoncé constituant les corpus est associé à une phrase de référence qui correspond à la transcription manuelle des paroles prononcées par l'utilisateur. Dans cette référence les commentaires peuvent être ou ne pas être annotés. Outre la meilleure hypothèse de reconnaissance générée par le SRAP pour chaque énoncé, un graphe de mots est également généré. Les scores acoustiques associés aux mots du graphe sont calculés lors de cette première passe de décodage.
Plusieurs corpus sont utilisés dans le cadre de nos travaux :

- **Dev** est un corpus de développement, constitué de 1764 énoncé, ce qui représente 905 dialogues. Les phrases de référence associées à ces énoncés sont constituées de 4843 mots.
- **Test_I** est un premier corpus de test constitué de 2005 enregistrements. Les transcriptions de référence contiennent un total de 5595 mots.
- **Test_II** est un second corpus de test constitué de 6501 énoncés. Les transcriptions de référence contiennent un total de 13890 mots.

Le corpus **Dev** est utilisé principalement pour le calibrage des coefficients de la régression logistique (expliqué au 2.2.4). Le corpus **Test_I** est utilisé pour effectuer la comparaison des performances en termes de *WER* de deux algorithmes de génération de réseaux de confusion présentés au chapitre 5. Le corpus **Test_II** est utilisé dans la majorité des évaluations effectuées dans ces travaux.

Corpus	# Énonces	# Total de mots dans la référence	Longueur moyenne
Dev	1764	4843	2.7
Test_I	2005	5595	2.8
Test_II	6501	13890	2.1

TABLE *4.1* – *Description des données de développement et de test*

Le tableau 4.1 donne le détail de chaque corpus en termes de nombre total d'énoncés, nombre total de mots dans la référence et la longueur moyenne de chaque référence (le nombre moyen de mots par énoncé de référence). Le tableau 4.2 donne la taille des graphes de mots pour les deux corpus de test en termes de nombre de transitions par graphe et de nombre de transitions par mot de la référence (calculé comme une moyenne sur le nombre de transitions par mot de la référence calculé pour chaque enregistrement). Plusieurs transitions dans le graphe peuvent porter la même hypothèse de mot. On observe que la taille des graphes est assez grande par rapport au nombre moyen de mots dans la référence. Par ailleurs, pour **Test_II**, le nombre moyen d'hypothèses de mots différent présents dans le graphe est de 42 mots.

Corpus	# Graphes de mots	# Transitions / Graphe	# Transitions / Mot de la référence
Test_I	2005	26800	13600
Test_II	6501	17000	12000

TABLE 4.2 – *Statistiques des graphes de mots construits sur les corpus de test*

Les graphes de mots sur l'ensemble des corpus sont générés en utilisant le modèle de langage décrit au 1.3.3. Le sous-modèle de détection de commentaires n'est toutefois pas inclus dans le modèle général. Ceci est du au fait qu'il est difficile d'estimer la probabilité *a posteriori* sur les segments *Hors-Domaine*. En effet, si on utilise le sous-modèle de détection des commentaires, dans le graphe de mots les séquences de mots détectées lors du décodage comme étant des commentaires seraient remplacées par l'étiquette <*COMMENTAIRE*>. Toutefois, dans le graphe de mots, l'étiquette <*COMMENTAIRE*> n'a pas une vraisemblance acoustique associée et donc le calcul de sa probabilité *a posteriori* passe d'abord par un calcul des probabilités *a posteriori* de chaque mot ayant

été détecté comme faisant partie du commentaire. L'algorithme doit alors avoir accès à la séquence de mots détectée et aussi aux vraisemblances acoustiques de chaque mot. Même en ayant calculé ces probabilités, l'obtention de la probabilité *a posteriori* du commentaire n'est pas aisée ; une simple multiplication des probabilités *a posteriori* des mots n'est pas possible parce que on compterait plusieurs fois les mêmes chemins dans le calcul de la probabilité *a posteriori* du commentaire (la probabilité *a posteriori* d'une transition est la somme des probabilités *a posteriori* des chemins passant par cette transition normalisé par la somme de tous les chemins dans le graphe). La complexité de calcul étant trop importante nous avons décidé de générer les graphes de mots sans détection des commentaires.

4.4 Evaluation du WER sur un corpus réel

Du point du vu du système, potentiellement tout mot en entrée est un mot à reconnaître. Le calcul du *WER* se fait en évaluant la distance de Levensthein (voir 1.6) entre la référence et l'hypothèse de reconnaissance fournie par le système. Toutefois, comme nous l'avons décrit, lorsqu'on travaille avec un corpus constitué d'énoncés de parole continue collectés à partir d'une application déployée on doit tenir compte de multiples événements.
Prenons l'exemple d'un énoncé non-parole dans le tableau 4.3. Un tel énoncé est à rejeter, donc lors de la transcription manuelle l'étiquette <*REJET*> lui sera attribuée et la référence ne contiendra que cette étiquette. Lors du décodage, un certain nombre de ces énoncés sont reconnus par le système comme des énoncés non-parole (ne contenant que du bruit) grâce au sous-modèle de rejet des bruits et, par conséquence, vont se voir attribués l'étiquette <*REJET*>. Par contre il est aussi possible que le système fournisse une hypothèse vide (le décodage n'est pas passé par le sous-modèle de rejet des bruits et aucun mot n'a été reconnu) ou une hypothèse contenant l'étiquette *OOV*(le décodage est passé par le modèle de détection des mots hors-vocabulaire).
Dans le premier cas, lors du calcul du *WER*, aucune erreur ne sera comptée mais pour le deuxième cas, l'algorithme de calcul va compter une omission (l'étiquette <*REJET*> dans la référence a été omise dans l'hypothèse de reconnaissance) ou une substitution (l'étiquette <*REJET*> dans la référence a été substituée avec l'étiquette *OOV* dans l'hypothèse de reconnaissance) dans le dernier cas.

Type d'énoncé	Référence	Hypothèse de reconnaissance possibles	# Erreurs comptés dans calcul WER
Énonce non-parole	"<*REJET*>"	"<*REJET*>"	0
		""	1 omission
		"*OOV*"	1 substitution

TABLE 4.3 – *Exemple d'annotation dans le calcul du WER*

Il existe des situations plus complexes qui peuvent donner un nombre plus important d'erreurs. Par exemple, un *aparté* (un type d'énoncé *Hors-Domaine* difficile à modéliser de part sa diversité) est un énoncé à rejeter et la référence est constituée seulement de

l'étiquette <*REJET*>. Par contre, lors du décodage, le système peut détecter, entre autre, un enchaînement de mots hors-vocabulaire et l'hypothèse fournie sera constituée de plusieurs étiquettes *OOV*. Donc, lors du calcul du *WER*, une substitution et plusieurs insertions sont comptées.
La façon d'annoter tous ces événements influe directement sur le calcul du *WER*. L'annotation doit être tout d'abord homogène ; par exemple, l'étiquette attribuée à un énoncé non-parole dans la référence doit être la même que celle attribué par le système suite à une détection non-parole faite par le sous-modèle de rejet des bruits. Le problème qui se pose ensuite et de savoir si certains événements ayant des annotations différentes n'ont pas la même signification dans le contexte plus large du système de dialogue. Par exemple, l'hypothèse de reconnaissance vide qui génère une omission dans le tableau 4.3 peut être considérée comme un énoncé rejeté par le système de dialogue. Ceci implique que l'hypothèse de reconnaissance contient désormais l'étiquette <*REJET*> et aucune erreur n'est comptée dans le calcul du *WER*. On peut aussi donner la même signification à l'hypothèse de reconnaissance contenant l'étiquette *OOV*. On considère alors que le système rejette un énoncé pour lequel il ne détecte que des mots hors vocabulaire (voir des *SPR*).
Il est évident que pour réaliser une évaluation pertinente du *WER* sur un corpus réel, dans le contexte plus large du système de dialogue, nous avons besoin d'une annotation homogène et de réaliser une normalisation de la notion du rejet du aux différents événements comme les détections non-parole, les *OOV*, les *SPR*, les commentaires.

4.4.1 Méthodes de normalisation

Nous définissons quatre méthodes de normalisation des énoncés et nous montrons leur influence sur le calcul du *WER* à travers un exemple. Pour définir la première méthode, nous partons de la prémisse que tout mot en entrée est un mot à reconnaître et qu'en l'absence d'un mot aucune étiquette ne sera attribuée à un énoncé (par exemple, les énoncés non-parole ne se verront pas attribuer l'étiquette <*REJET*>). Dans la deuxième méthode, les énoncés non-parole et les énoncés ayant une référence vide sont considérés comme étant des énoncés à rejeter et leur référence est remplacée par l'étiquette <*REJET*>. Le SRAP attribue la même étiquette si l'hypothèse de reconnaissance est vide ou si un énoncé est détecté par le sous-modèle de rejet des bruits. La troisième méthode introduit la notion de rejet énoncé parole (les énoncés qui ne contiennent que des *OOV* ou des *SPR*). Les énoncés considérés comme étant des rejets énoncé parole se voient attribuer l'étiquette <*REJET*>. Dans ces trois premières méthodes la détection de commentaires n'est pas faite (le sous-modèle de langage n'est pas inclut dans le modèle général lors du décodage). De ce fait, l'annotation des commentaires dans la référence est ignorée et les mots ayant été annotés en commentaire sont traités comme tout autre mot de l'application. La dernière méthode est utilisée dans le cas où le système peut détecter les commentaires et, par conséquence, les séquences de mots annotées en commentaire dans la référence sont remplacées par l'étiquette <*COMMENTAIRE*>. La notion de rejet énoncé parole est alors étendue aux commentaires.
Afin de mieux illustrer l'effet de ces méthodes et l'importance de la normalisation de la référence et de l'hypothèse de reconnaissance dans le calcul du taux d'erreur mot

(*WER*), nous prenons l'exemple du tableau 4.4. Il est constitué de quatre énoncés réels qui résument les principaux problèmes rencontrés dans le calcul du WER sur un corpus réel.

Énonce	Transcription manuelle annotée	Hypothèse de reconnaissance fournie
(f)acture	*vide*	*rejet* [1]
énonce non-parole	*bruit*	*OOV*
faux départ	*SPR*	*OOV OOV*
oh merde oh là oh	[com :] oh merde oh là oh [com :]	[com :] ah merde oh là c' est trop [:com] [2]
oh là là payer ma facture	[com :] oh là là [:com] payer ma facture	oh payer ma facture

TABLE 4.4 – *Exemple d'énoncés réels*

Dans cet exemple, le premier énoncé contient un mot tronqué (le "f" est prononcé, mais le signal de parole est coupé par l'automate de détection bruit/parole). Pour le SRAP il est difficile de reconnaître ce mot à partir du signal tronqué et donc la transcription manuelle est vide (même si en écoutant l'énoncé tronqué, un humain est capable de reconnaître le mot prononcé). Les deux exemples suivants correspondent respectivement à un énoncé non-parole (du bruit) et à un faux départ (*SPR*). L'exemple suivant contient de la parole qui est en fait un commentaire, d'où les deux étiquettes ([com :], [:com]) dans la transcription annotée. On retrouve les mêmes étiquettes dans l'hypothèse de reconnaissance parce que le décodage est passé par le sous-modèle de détection des commentaires. La même séquence de mots est toutefois détectée par le SRAP sans utiliser le sous-modèle de détection des commentaires (dans ce cas les étiquettes ne sont pas rajoutées). Le dernier exemple contient de la parole *Hors-Domaine* et de la parole *Dans-le-Domaine*, d'où l'annotation dans la transcription manuelle d'une partie de l'énoncé. L'hypothèse de reconnaissance ne contient aucune annotation car aucun commentaire n'est détecté par le SRAP utilisant le sous-modèle de langage.
Les tableaux qui suivent pour chaque méthode montrent les différences dans le calcul du *WER* en fonction de la normalisation effectuée sur les énoncées de l'exemple du tableau 4.4. Chaque tableau donne la référence obtenue par la normalisation de l'annotation manuelle ainsi que de l'hypothèse de reconnaissance. La dernière partie de chaque tableau présente le détail du nombre d'erreurs et de mots corrects sur quatre colonnes. La première colonne "*C*", donne le nombre de mots corrects. Les trois colonnes suivantes, dans l'ordre "*I*" "*S*" "*O*", donnent respectivement le nombre d'insertions, de substitutions et d'omissions. La dernière ligne de chaque tableau donne la valeur du *WER*.

Méthode 1 : Les énoncés non-parole ne sont pas étiquetés et la référence correspondant à un tel énoncé est vide. Il en est de même pour l'hypothèse de reconnaissance qui sera vide lorsque le système passe par le sous-modèle de rejet des bruits.

1. Le SRAP est passé par le modèle de rejet des bruits.
2. La même séquence de mots est également détectée par le SRAP sans utiliser le sous-modèle des commentaires.

Référence de l'énoncé	Hypothèse de reconnaissance	Détails erreurs			
		C	I	S	O
		0	0	0	0
	OOV	0	1	0	0
SPR	*OOV OOV*	0	1	1	0
oh merde oh là oh	ah merde oh là c' est trop	3	2	2	0
oh là là payer ma facture	oh payer ma facture	4	0	0	2
WER=75%		7	4	3	2

TABLE *4.5* – *Calcul du WER :* ***Méthode1***

Dans ce premier tableau les énoncés non-parole ne sont pas étiquetés et la référence est vide. Il en est de même pour l'hypothèse de reconnaissance du premier énoncé de l'exemple. Malgré le fait que le SRAP est passé par le sous-modèle de rejet des bruits, l'hypothèse est vide car aucune étiquette n'est attribué.

Méthode 2 : Les énoncés non-parole sont étiquetés et la référence correspondant à un tel énoncé contient l'étiquette <*REJET*>. Il en est de même pour l'hypothèse de reconnaissance qui contiendra l'étiquette <*REJET*> lorsque le système passe par le sous-modèle de rejet des bruits. Une référence d'un énoncé ou une hypothèse de reconnaissance vide sont également considérées comme un rejet et l'étiquette <*REJET*> leur est attribuée.

Référence de l'énoncé	Hypothèse de reconnaissance	Détails erreurs			
		C	I	S	O
<*REJET*>	<*REJET*>	1	0	0	0
<*REJET*>	*OOV*	0	0	1	0
SPR	*OOV OOV*	0	1	1	0
oh merde oh là oh	ah merde oh là c' est trop	3	2	2	0
oh là là payer ma facture	oh payer ma facture	4	0	0	2
WER=64%		8	3	4	2

TABLE *4.6* – *Calcul du WER :* ***Méthode2***

Les références et les hypothèses de reconnaissance qui étaient vides dans la première méthode contiennent l'étiquette <*REJET*>. On observe que le nombre total d'erreurs est le même, car l'insertion dans le deuxième énoncé est devenue une substitution du fait de la modification de la référence. La valeur plus petite du *WER* s'explique par un nombre plus grand de mots dans la référence (on passe de 12 à 14 mots) qui a pour résultat un dénominateur plus grand pour un numérateur constant dans la formule de calcul du *WER*.

Méthode 3 : Rejet énoncé parole. En plus des conditions de la **Méthode2**, un énoncé qui ne contient que des mots hors-vocabulaire *OOV* ou des *SPR* est compté comme un énoncé à rejeter (rejet énoncé parole). La référence et l'hypothèse de reconnaissance correspondantes sont alors remplacées par l'étiquette <*REJET*>.

L'utilisation de la **Méthode3** introduit la notion de rejet énoncé parole. Le deuxième énoncé est considéré par le système comme étant un rejet et annoté en conséquence. Le troisième énoncé devient un énoncé à rejeter et il est aussi considéré par le système

Référence de l'énoncé	Hypothèse de reconnaissance	Détails erreurs			
		C	I	S	O
<*REJET*>	<*REJET*>	1	0	0	0
<*REJET*>	<*REJET*>	1	0	0	0
<*REJET*>	<*REJET*>	1	0	0	0
oh merde oh là oh	ah merde oh là c' est trop	3	2	2	0
oh là là payer ma facture	oh payer ma facture	4	0	0	2
WER=43%		10	2	2	2

TABLE *4.7* – *Calcul du WER :* ***Méthode3***

comme un rejet. Cette fois ci, la diminution de la valeur du WER n'est pas due à un nombre plus grand de mots dans la référence (il est le même que pour la précédente) mais bien à un nombre plus petit d'erreurs. Cette baisse du nombre d'erreurs est accompagnée par une augmentation du nombre de mots correctement reconnus.

Méthode 4 : Rejet énoncé parole et détection des commentaires. Les commentaires annotés dans la référence sont remplacées par l'étiquette <*COMMENTAIRE*>. Il en est de même pour les commentaires détectés par le SRAP lors du décodage. La notion de rejet énoncé parole définie à la **Méthode3**, est étendue aux commentaires (un énoncé qui ne contient que des *OOV*, des *SPR* ou des commentaires est compté comme un rejet et remplacé par l'étiquette <*REJET*>).

Référence de l'énoncé	Hypothèse de reconnaissance	Détails erreurs			
		C	I	S	O
<*REJET*>	<*REJET*>	1	0	0	0
<*REJET*>	<*REJET*>	1	0	0	0
<*REJET*>	<*REJET*>	1	0	0	0
<*REJET*>	<*REJET*>	1	0	0	0
<*COMMENTAIRE*>payer ma facture	oh payer ma facture	3	0	1	0
WER=12%		7	0	1	0

TABLE *4.8* – *Calcul du WER :* ***Méthode4***

Le fait d'avoir annoté le quatrième énoncé comme étant un commentaire et sa détection en tant que commentaire font que les séquences de mots sont remplacés par l'étiquette <*REJET*> dans la référence et dans l'hypothèse. En revanche, dans le dernier énoncé, seul la séquence de mots annotée est remplacée par l'étiquette <*COMMENTAIRE*>. L'énoncé n'est pas considéré comme un rejet énoncé parole car en plus du commentaire il contient d'autres mots. Pour cet énoncé le SRAP n'a détecté aucun commentaire.

Comme montré à travers ces exemples, le calcul du *WER* sur un corpus réel n'est pas aisé et une normalisation de la référence ainsi que de l'hypothèse de reconnaissance est nécessaire afin d'effectuer une évaluation pertinente et homogène dans le contexte du système de dialogue. La différence entre les valeurs du *WER* obtenues avec les deux premières méthodes alors que le nombre d'erreurs est le même montre l'importance d'utiliser la même méthode de calcul pour pouvoir comparer correctement plusieurs évaluations (le nombre de mots de la référence diffère pour les deux méthodes).
Dans les méthodes que nous avons présentées, seules les deux dernières sont utilisées

dans les évaluations présentées dans ce rapport. Ainsi, la **Méthode3** est utilisée dans les évaluations où la détection des commentaires n'est pas réalisée et la **Méthode4** dans le cas contraire.

4.4.2 Evaluation du taux d'erreur mot

Dans cette section nous présentons les évaluations effectuées en termes de *WER* sur la *1-best* des deux corpus de test utilisant les quatre méthodes de normalisation présentées. Le tableau 4.9 présente l'évaluation en termes de *WER* sur le corpus de test **Test_I**. L'évaluation du *WER* pour le corpus de test **Test_II** est présenté dans le tableau 4.10. Dans les deux tableaux indique aussi le taux de mots correctement reconnus ("C") ainsi que les taux d'insertion, de substitution et d'omission ("I", "S", "O").

Test_I	WER	C	I	S	O
Methode 1	45.4%	67.0%	12.5%	24.5%	8.4%
Methode 2	45.1%	66.4%	11.4%	25.4%	8.3%
Methode 3	45.1%	66.5%	11.6%	25.2%	8.3%
Methode 4	44.2%	72.1%	16.3%	20.3%	7.6%

TABLE *4.9* – *Evaluation de la* 1-best *sur le corpus* ***Test_I*** *à l'aide des quatre méthodes de normalisation*

Test_II	WER	C	I	S	O
Methode 1	47.7%	74.4%	22.2%	18.8%	6.7%
Methode 2	43.0%	73.1%	16.1%	21.4%	5.5%
Methode 3	42.0%	73.9%	15.9%	20.6%	5.5%
Methode 4	41.8%	76.0%	17.8%	19.4%	4.6%

TABLE *4.10* – *Evaluation de la* 1-best *sur le corpus* ***Test_II*** *à l'aide des quatre méthodes de normalisation*

Nous rappelons que les valeurs des *WER* sur les quatre méthodes ne peuvent pas être comparées directement entre elles car la méthode utilisée pour la normalisation de la référence n'est pas la même et donc le nombre de mots dans la référence diffère d'une méthode à l'autre. La dernière ligne de chaque tableau correspond à l'utilisation du sous-modèle de détection des commentaires dans la première passe de reconnaissance. Pour les évaluations présentées sur les trois premières lignes, le sous-modèle de détection des commentaires n'est pas utilisé et donc les commentaires ne sont pas détectés. En conséquence, les annotations des commentaires dans la référence sont ignorées et la référence est constituée de la séquence de mots qui été annotée comme étant un commentaire.

4.5 Evaluation du taux d'erreur d'interprétation

Dans cette section nous présentons une évaluation de la *1-best* des corpus de test au niveau interprétation en calculant l'*IER* (voir 3.4 pour la définition de cette métrique).

Contrairement à l'évaluation au niveau mot, au niveau interprétation on n'a pas besoin d'effectuer une normalisation des données. Ceci est du au fait que les problèmes de normalisation présentée au 4.4 (exceptée la Méthode 4 qui introduit la détection de commantaires) sont résolus directement par le processus d'interprétation. Par exemple un énoncé vide sera interprété comme étant un rejet, de même qu'un énoncé ne contenant que des *OOV* ou *SPR*. Ce dernier, lors de l'analyse en concepts, produit comme résultat une séquence vide qui est interprétée ensuite comme étant un rejet.
Pour chaque corpus de test nous présentons deux évaluations de la *1-best* : la première sans détection de commentaires, les annotations dans la référence sont donc ignorées ; la deuxième avec détection des commentaires, et donc les séquences de mots de la référence annotées comme étant des commentaires sont remplacées par l'étiquette <*COMMENTAIRE*> (elle n'allume aucun concept). Le tableau 4.11 présente l'évaluation sur le corpus **Test_I** et le tableau 4.12 l'évaluation sur le corpus **Test_II**. Les deux tableaux présentent aussi le taux de mots correct ainsi que les détails des erreurs : le taux de faux rejet (FR), de substitution (Sub) et de fausse alarme (FA).

Test_I	IER	C	FR	Sub	FA
sans détection des commentaires	22.0%	86.0%	10.8%	3.3%	7.9%
avec détection des commentaires	20.4%	85.7%	7.6%	6.7%	6.1%

TABLE *4.11* – *Evaluation de la* 1-best *sur le corpus* ***Test_I*** *sans et avec détection des commentaires*

Test_II	IER	C	FR	Sub	FA
sans détection des commentaires	22.7%	92.7%	1.6%	5.7%	15.4%
avec détection des commentaires	24.6%	92.8%	1.6%	5.5%	17.5%

TABLE *4.12* – *Evaluation de la* 1-best *sur le corpus* ***Test_II*** *sans et avec détection des commentaires*

Nous rappelons que sur un corpus de test les deux évaluations, avec et sans détection de commentaires ne peuvent pas être comparées directement du fait que la référence n'est pas la même. En effet, l'*IER* est calculé par rapport au nombre d'énoncés interprétables (la référence produit une interprétation valide). Comme nous l'avons expliqué au 4.1 les séquences mots annotées comme étant des commentaires peuvent contenir des mots qui allument des concepts et qui peuvent donner lieu à une interprétation. Ceci est le cas lorsqu'on ignore ces annotations comme pour l'évaluation sans détection de commentaires. Lorsque ces séquences sont remplacées par l'étiquette <*COMMENTAIRE*>, elles n'allument aucun concept et aucune interprétation n'est alors possible. Le nombre d'énoncés interprétables n'est donc pas le même sur les deux évaluations. Le tableau 4.13 montre le nombre d'énoncés interprétables pour les deux évaluations. On observe bien une différence qui provient du fait que pour 114 énoncés, les séquences de mots annotées comme étant des commentaires dans la référence produisent une interprétation si l'annotation est ignorée.

4.6 Conclusions

Dans ce chapitre nous avons présenté le cadre expérimental des travaux de cette thèse. Nous avons tout d'abord réalisé une analyse des énoncés utilisateur et nous

Test_II	Nombre énoncés interprétables
sans détection des commentaires	4253
avec détection des commentaires	4139

TABLE *4.13* – *Nombre d'énoncés interprétables pour le corpus **Test_II** avec et sans détection de commentaires*

avons mis en évidence plusieurs catégories d'énoncés comme les énoncés non-parole (ne contenant que du bruit), la parole *Hors-Domaine* (les commentaires) et la parole *Dans-le-Domaine*. Nous avons aussi posé les problématiques liées au comportement du système de dialogue en fonction du type d'énoncé traité et nous avons introduit la notion d'énoncé à rejeter. Nous avons ensuite présenté le modèle de langage et le modèle acoustique utilisé par le SRAP. Un sous-modèle de langage pour la détection des commentaires a également été introduit. Une description des données expérimentales (corpus d'apprentissage, de développement et de test) a aussi était donnée.
Une problématique importante abordée dans ce chapitre est liée à l'évaluation au niveau mot des corpus réels. Nous avons ainsi présenté les différents problèmes posés par le traitement des données réelles et nous avons décrit quatre méthodes de normalisation de données qui permettent le calcul du *WER* sur un ensemble homogène. Une évaluation des corpus de test en termes de *WER* a aussi été présenté. Nous n'avons retenu que les deux dernières méthodes pour la suite des évaluations. L'évaluation des corpus de test en termes de *IER*a également été présentée.

Chapitre 5

Réseaux de confusion

Sommaire

En reconnaissance de la parole, l'approche standard (Bahl et al., 1983) utilisée pour le calcul de la meilleure solution est basée sur le critère de maximum *a posteriori* (MAP - *maximum a posteriori probability*). Ainsi, un SRAP produit comme résultat une séquence d'hypothèses de mots correspondant au chemin ayant la meilleure probabilité *a posteriori* étant donné les modèles acoustique et de langage. La théorie de Bayes (Duda et Hart, 1973) dit que la maximisation de la probabilité *a posteriori* de l'énoncé minimise le taux d'erreur phrase (SER - *Sentence Error Rate*, la probabilité d'avoir au moins une erreur dans la phrase). Or, la métrique employée généralement dans les évaluations d'un SRAP est le taux d'erreur mot (*WER*). On peut supposer de manière empirique que le

SER et le *WER* sont des métriques corrélées, et donc la minimisation du *SER* devrait conduire à la minimisation du *WER*. Des expériences (Stolcke et al., 1997) ont montré que la minimisation du *SER* ne garantit pas une minimisation du *WER*. Intuitivement on peut néanmoins envisager qu'en maximisant la probabilité *a posteriori* des mots, au lieu de la probabilité *a posteriori* des phrases, on devrait minimiser le *WER*.
La solution proposée dans (Mangu et al., 2000) est une approche de minimisation du *WER* basée sur l'utilisation des graphes de mots. Cette approche définit une nouvelle distance d'édition entre des hypothèses multiples basée sur la création d'un nouvel alignement pour toutes les hypothèses du graphe de mots, appelé réseau de confusion. Même si cette approche a tendance à surestimer le taux d'erreur mot dans certains cas, il a été montré (Mangu et al., 2000) que c'est une bonne approximation de *WER* standard.
Dans la section 5.1 nous présentons deux approches de minimisation du *WER* basées sur une liste de *N best* et sur des graphes de mots. Les différents concepts et notions théoriques utilisés par l'approche proposée dans (Mangu et al., 2000) sont présentés dans la section 5.2. Les différents algorithmes de génération des réseaux de confusion sont présentés dans la section 5.3. La dernière section 5.4 présente une comparaison de deux algorithmes de génération que nous avons implémentés. Une discussion des performances ainsi que du choix de l'algorithme utilisé dans la suite des travaux est aussi proposée dans cette section.

5.1 Minimisation du taux d'erreur mot

Pour obtenir le *WER* d'une hypothèse de reconnaissance W on doit calculer la distance d'édition entre W et une séquence de mots de référence R. Ceci est possible seulement si on connaît la séquence de mots de référence R. Si on ne connaît pas cette séquence de référence, ce qui est le cas en pratique pour un SRAP, on ne peut qu'estimer le *WER* sur un espace d'hypothèses. Pour cela on doit faire des hypothèses sur la séquence de mots de référence en choisissant, sur un espace donné, une séquence de mots comme étant correcte étant donné une probabilité associée. Si on considère $\mathcal{H}$ comme l'espace des hypothèses [1], l'espérance du taux d'erreur mot de l'hypothèse $W \in \mathcal{H}$ est donnée par la formule :

$$E[WER(W)|X] = \sum_{\tilde{R}} P(\tilde{R}|X) WER(W|\tilde{R}) \tag{5.1}$$

où $P(\tilde{R}|X)$ est la probabilité *a posteriori* associée à l'hypothèse $\tilde{R}$ et $WER(W|\tilde{R})$ est le taux d'erreur mot de l'hypothèse W connaissant la référence $\tilde{R}$. On obtient alors un nouveau problème d'optimisation par rapport à la séquence de mots W :

$$\hat{W} = \underset{W}{argmin} \sum_{\tilde{R}} P(\tilde{R}|X) WER(W|\tilde{R}) \tag{5.2}$$

L'équation 5.2 pose les bases de la minimisation du *WER* à partir des probabilités *a posteriori* des séquences de mots. Une implémentation algorithmique de cette équation

1. Dans ce cas, une hypothèse est une séquence de mots

implique deux étapes : une somme effectuée sur toutes les références potentielles $\tilde{R}$ suivie d'une minimisation sur l'hypothèse W. La première étape équivaut à calculer le *WER* pour toutes les paires $(W, \tilde{R})$ de l'espace d'hypothèses $\mathcal{H}$. Ceci implique un volume de calcul proportionnel au carré du nombre d'hypothèses dans $\mathcal{H}$. De plus, le calcul du *WER* implique un alignement du W et $\tilde{R}$, réalisé à l'aide de la programmation dynamique. Le temps de calcul du *WER* pour chaque paire d'hypothèses est donc proportionnel au carré de la longueur des hypothèses. La deuxième étape consiste à choisir l'hypothèse W avec la plus petite somme. Le volume de calcul élevé soulève la question de la faisabilité de la minimisation du WER dans un contexte de reconnaissance de la parole continue à très grand vocabulaire (LVCSR - *Large Vocabulary Continuous Speech Recognition*).

5.1.1 Approche par listes de *N best*

Des travaux (Stolcke et al., 1997; Goel et al., 1998) ont montré qu'on peut implémenter un algorithme de minimisation du *WER* en limitant l'espace des hypothèses à une liste de *N best*.

$$\hat{W} = \underset{W}{argmin} \sum_{i=1}^{N} P(W_i|X) WER(W|W_i) \tag{5.3}$$

où W et W_i sont des hypothèses de la liste. Les probabilités *a posteriori* des hypothèses sont estimées utilisant la liste de *N best*.
Pour chaque hypothèse dans la liste on calcule la somme des distances d'édition par rapport aux autres hypothèses de la liste pondérées par leur probabilité *a posteriori*. L'hypothèse choisie est celle à laquelle correspond la plus petite somme.
Pour un N très grand, l'algorithme nécessite N^2 alignements pour le calcul de la distance d'édition ce qui peut devenir coûteux pour une liste de plus de 1000 hypothèses. De plus, l'algorithme choisi systématiquement une hypothèse se trouvant dans les 10 meilleures de la liste *N best* initiale (observation empirique dans (Stolcke et al., 1997)). Pour réduire la complexité de l'algorithme, la minimisation du *WER* peut être réalisée sur une liste de K hypothèses plus petite ($K \ll N$ et une complexité de $O(KN)$). Les tests effectués (Stolcke et al., 1997) sur des corpus de type *Switchboard* et *Spanish Call-Home* montrent des performances faibles en termes de *WER*. Des tests sur un corpus de type *North American Business (NAB) Newstask*, avec un *WER* compris entre 10% et 30%, montrent que l'algorithme proposé choisit invariablement l'hypothèse ayant la meilleure probabilité *a posteriori*. Ceci s'explique par le fait que, dans ce type de corpus, une valeur petite du *WER* signifie que le nombre d'erreurs de mots dans une phrase est très petit (une ou deux erreurs). Ainsi, dans beaucoup de cas, une erreur mot correspond à une erreur phrase et inversement.
Il est donc possible de construire un algorithme de minimisation du *WER* basé sur des listes de *N best* et ayant une complexité de calcul relativement réduite. Il est aussi plus judicieux d'utiliser un algorithme de ce type sur des corpus ayant un *WER* élevé pour maximiser ses performances. Toutefois, cette approche n'est pas optimale car le nombre d'hypothèses est très restreint par rapport à l'espace de recherche du SRAP. Ceci a une influence importante sur l'estimation des probabilités *a posteriori* des énoncés de la liste,

mais aussi sur la minimisation du *WER* dans l'équation 5.3 du fait d'un nombre restreint de références.

5.1.2 Approche par graphes de mots

Un moyen de contourner le problème d'un espace d'hypothèses trop petit pour les listes de *N best* est l'utilisation de graphes de mots. Le passage aux graphes de mots implique néanmoins certains problèmes de calcul. Le nombre d'hypothèses dans un graphe de mots est très grand par rapport aux N hypothèses d'une liste *N best* et donc le calcul dans l'équation 5.3 pour choisir la meilleure hypothèse devient impossible. Un deuxième problème est posé par le calcul de la distance d'édition entre une hypothèse W et les autres hypothèses du graphe de mots. Ce calcul implique un alignement des mots de l'hypothèse W avec les mots de toutes les autres hypothèses. Cet alignement ne tient pas compte des supports temporels des mots. Or, dans un graphe de mots, l'alignement des chemins est basé sur leur support temporel. Une différence d'un seul mot dans un chemin du graphe peut alors avoir des conséquences globales sur l'alignement de ce chemin avec les autres chemins dans le graphe.
Afin de pouvoir construire un algorithme de minimisation du *WER* basé sur des graphes de mots, une nouvelle structure est proposée dans (Mangu et al., 2000) qui permet de réaliser un alignement des hypothèses multiples du graphe. Cette approche permet d'associer toutes les hypothèses du graphe de mots à un seul alignement et les erreurs entre n'importe qu'elle paire d'hypothèses sont calculées en fonction de cet alignement. Le *WER* se calcule alors en comptant localement le nombre d'erreurs (insertion, substitution, omission) entre deux hypothèses. L'estimation de l'espérance du *WER* de la séquence W ($E[WER(W)|X]$) peut alors être approximée par une somme sur les optimums locaux. La séquence $\hat{W}$ est choisie comme étant l'hypothèse qui minimise localement cette espérance.

5.2 Alignement des hypothèses multiples : concepts et définitions

La difficulté principale de l'approche de minimisation du *WER* à partir des graphes de mots est de trouver un alignement pour les hypothèses du graphe. De manière formelle, un alignement est une relation d'équivalence définie sur les hypothèses de mots du graphe (les transitions d'un graphe de mots) et un ordonnancement complet des classes d'équivalence qui soit cohérent avec celui du graphe de mots d'origine. Dans cette partie nous définissons ces notions et nous présentons le principe de construction de l'alignement.

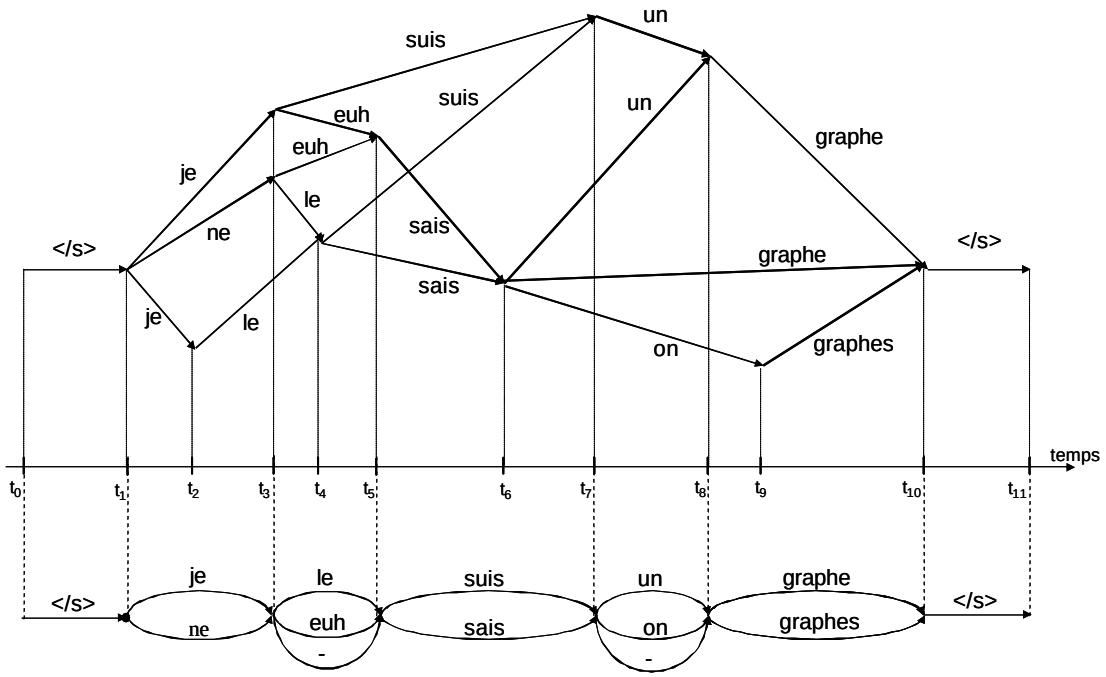

FIGURE *5.1* – Exemple de graphe de mots et le réseau de confusion correspondant. Le symbole "-" marque une omission

5.2.1 Classes d'équivalence

Une classe d'équivalence est constituée d'une ou plusieurs hypothèses de mots (transitions) du graphe ; les transitions ne doivent pas être en relation d'ordre. Par exemple, dans la figure 5.1, chaque transition peut constituer une classe d'équivalence, de même que les transitions portant les mots "**je**" et "**ne**" peuvent former une classe d'équivalence. En revanche, les transitions portant les mots "**un**" et "**graphe**" ne peuvent pas former une classe car il existe un chemin dans le graphe qui passe par les deux transitions.

5.2.2 Recouvrement temporel

Une notion importante utilisée pour regrouper des transitions dans une classe d'équivalence (ou des classes d'équivalence) est le **recouvrement temporel**. Il convient d'introduire cette notion afin de pouvoir quantifier la superposition entre les supports temporels de deux transitions dans un graphe de mots. La valeur du recouvrement temporel peut être calculée de différentes façons et nous précisons pour chaque algorithme décrit dans la section 5.3 la méthode de calcul utilisée.

5.2.3 Relation d'ordre

Intuitivement, sur un graphe de mot on peut établir un ordre entre les transitions se trouvant sur un même chemin. Par exemple, dans le graphe de mots représenté dans la figure 5.1, la transition portant le mot "**je**" et "avant" la transition portant le mot "**suis**", mais aussi "avant" la transition portant le mot "**graphe**". On peut affirmer ceci car dans le graphe de mots on retrouve le chemin "**je suis un graphe**" qui passe par les trois transitions.
Le graphe de mots définit une **ordre partiel** $\leq$ entre les transitions de la façon suivante : si E représente l'ensemble des transitions du graphe de mots, pour $e_1, e_2 \in E$, $e_1 \leq e_2$ si :

- $e_1 = e_2$ ou
- $Fstate(e_1) = Sstate(e_2)$ (où $Sstate$ et $Fstate$ sont les états de début et de fin d'un transition) ou
- $\exists f \in E$ tel que $e_1 \leq f$ et $f \leq e_2$.

D'une manière informelle on dit qu'il existe un chemin dans le graphe de mots qui passe par les deux transitions et que e_1 est "avant" e_2.
On peut étendre cette notion d'ordre aux classes d'équivalence. Si $\xi \subset 2^E$ est un ensemble de classes d'équivalence sur E, on peut définir l'ordre partiel $\preceq$ sur ξ de la manière suivante :

> Soient $C_1, C_2 \in \xi$:
> $C_1 \preceq C_2 \Longleftrightarrow \exists e_1 \in C_1$ et $e_2 \in C_2$ tel que $e_1 \leq e_2$

La relation d'ordre sur les classes d'équivalence $\preceq$ définie ainsi est consistante avec $\leq$ car elle préserve l'ordre temporel sur les mots des transitions dans le graphe. C'est une relation binaire sur ξ qui vérifie les trois propriétés suivantes :

- **réflexive** : elle met tout élément en relation avec lui même.
  $\forall C \in \xi, C \preceq C$
- **antisymétrique** : les éléments distincts ne sont jamais en relation mutuelle.
  $\forall C_1, C_2 \in \xi$ si $C_1 \preceq C_2 \bigwedge C_2 \preceq C_1$ alors $C_1 = C_2$
- **transitive** : deux éléments sont mis en relation dès qu'on peut transiter par un troisième.
  $\forall C_1, C_2, C_3 \in \xi$ si $C_1 \preceq C_3 \bigwedge C_3 \preceq C_2$ alors $C_1 \preceq C_2$

Dans la suite du document, on dit que deux transitions "sont en relation d'ordre" si $e_1 \leq e_2$ ou $e_2 \leq e_1$. On utilise la même expression pour les classes si $C_1 \preceq C_2$ ou $C_2 \preceq C_1$.

5.2.4 Principe de construction de l'alignement

Afin de générer l'alignement, on procède à un regroupement des classes d'équivalence suivant un certain nombre de critères et ceci jusqu'à l'obtention d'un ordre (linéaire) complet. Celui-ci est obtenu lorsque la condition suivante est satisfaite :

> $\forall e_1, e_2 \in E$ avec $e_1 \in C_1$ et $e_2 \in C_2$ on a :
> $C_1 \preceq C_2$ ou $C_2 \preceq C_1$

L'obtention d'un ordre complet en un nombre fini d'itérations est garantie par la définition de la relation d'ordre sur les classes d'équivalence. Étant donné cette relation, deux classes C_1 et C_2, qui ne sont pas en relation ($C_1 \not\preceq C_2$ et $C_2 \not\preceq C_1$), peuvent être regroupées afin de former une nouvelle classe d'équivalence dans le même espace ξ. Cet espace contiendra moins de classes qui ne sont pas en relation et on est assuré d'obtenir un ordre complet après un nombre fini d'itérations. Une preuve formelle est donnée dans (Mangu et al., 2000).
Avant de procéder au regroupement une étape d'initialisation des classes d'équivalence est nécessaire. Une méthode simple est, par exemple, de considérer chaque transition comme constituant une classe d'équivalence différente. D'autres manières d'initialiser les classes d'équivalence sont définies par les algorithmes décrit au 5.3.

5.2.5 Réseaux de confusion

Comme nous l'avons décrit ci-dessus, une fois l'ordre complet obtenu, l'alignement est une succession de classes d'équivalence. L'alignement ainsi obtenu est un réseau de confusion comme dans la figure 5.1. Les classes d'équivalence constituent donc les classes de mots du CN.
Chaque classe d'équivalence regroupe des transitions portant plusieurs hypothèses de mot ; plusieurs transitions pouvant porter la même hypothèse de mot. Dans une classe du CN, une hypothèse de mot est portée par une seule transition. Cette transition est le résultat du regroupement des transitions portant cette hypothèse de mot dans la classe d'équivalence. On calcule ainsi la probabilité *a posteriori* de l'hypothèse de mot du CN en sommant les probabilités *a posteriori* des transitions de la classe d'équivalence. D'une manière générale, la probabilité *a posteriori* d'une hypothèse de mot dans une classe du CN est égale à la somme de probabilités *a posteriori* des chemins du graphe qui contiennent l'hypothèse de mot dans une position donnée normalisée par la somme des probabilités *a posteriori* de tous le chemins du graphe.
On observe dans la figure 5.1 que tous les chemins du graphe passent par un des mots de la première classe du CN. La somme des probabilités *a posteriori* des mots dans cette classe ne peut alors dépasser 1. De manière générale, la somme des probabilités *a posteriori* des mots dans une classe est égale ou inférieure à 1. Mais cette somme peut être strictement inférieur à 1. Par exemple, pour la deuxième classe du CN, le chemin du graphe "**je suis un graphe**" ne passe par aucun des mots de la classe. La différence de probabilité pour atteindre une probabilité totale de 1 dans la classe représente la probabilité d'une omission de mot. Cette omission est alors introduite dans la classe sous la forme d'une transition portant le symbole "-" dont la probabilité *a posteriori* est égale à la probabilité nécessaire pour que la probabilité totale de la classe soit égale à 1. De manière générale, la probabilité de l'omission dans une classe est égale à la somme des probabilités *a posteriori* de tous les chemins du graphe qui ne contiennent aucun mot dans cette position.

Consensus hypothesis

Afin d'extraire la meilleure solution, on doit trouver la séquence de mots W qui minimise l'espérance du *WER*. Étant donné que les décisions prises dans chaque classe sont indépendantes, la minimisation de l'espérance du *WER* de la séquence W peut être approximée par une minimisation réalisée localement sur chaque classe en choisissant le mot correspondant. La meilleure solution des réseaux de confusion, appelée aussi *consensus hypothesis (CH)*, est obtenue en choisissant dans chaque classe la transition ayant la meilleure probabilité *a posteriori*. Cette transition peut porter un mot ou une omission, ce qui signifie qu'aucun mot ne sera présent dans la *consensus hypothesis* dans la position correspondante.

5.3 Description des principaux algorithmes

Comme nous l'avons décrit, la minimisation du taux d'erreur mot à partir d'un graphe de mots passe par la construction de structures plus compactes et plus adaptées à cette tache, les réseaux de confusion. Du fait de la structure des graphes, de leur grande taille et de leur redondance en ce qui concerne les hypothèses de mots (un nombre assez élevé de transitions portant la même hypothèse de mot mais avec des petites différences de support temporel), les post-traitements (comme la génération des réseaux de confusion) sur les graphes de mots peuvent s'avérer très complexes et très coûteux en termes de temps de calcul. Nous avons recensé dans la littérature trois algorithmes principaux pour la construction des CNs. Le premier est apparu en 1999 et a été présenté dans (Mangu et al., 1999) et décrit plus en détail par les mêmes auteurs dans (Mangu et al., 2000). Nous allons l'appeler l'algorithme de génération par regroupement des classes de transitions et il est détaillé dans la section 5.3.1. Un deuxième algorithme de génération de CNs a été décrit en 2003 dans (Hakkani-Tur et Riccardi, 2003). Une description plus détaillé de l'algorithme ainsi que différentes applications des CNs ont été présentés par les même auteurs dans (Hakkani-Tur et al., 2006). Nous allons l'appeler l'algorithme du *pivot* et une description détaillée est fournie dans la section 5.3.2. En 2006, un troisième algorithme de génération des CNs est présenté dans (Xue et Zhao, 2006) . Appelé algorithme de génération par regroupement en groupes d'états il est brièvement décrit dans la section 5.3.3. Pendant les travaux de cette thèse nous avons implémenté et testé les deux premiers algorithmes et nous les décrivons plus en détail. Nous présentons le dernier algorithme afin de donner une vision la plus complète possible de l'état de l'art de la génération des CNs, mais celui-ci n'a pas fait l'objet des travaux dans cette thèse. Nous faisons également une comparaison des performances dans la section 5.4 des deux premiers algorithmes et nous motivons par la même occasion le choix de l'algorithme utilisé dans la suite nos travaux.

5.3.1 Algorithme de génération par regroupement des classes de transitions

Ce premier algorithme présenté dans (Mangu et al., 1999) est un algorithme itératif constitué de deux étapes : le regroupement "*intra-mots*" et le regroupement "*inter-mots*". Dans la première étape, l'algorithme regroupe seulement les classes d'équivalence correspondant à la même hypothèse de mot et dans la deuxième étape, le regroupement se fait sur l'ensemble des classes d'équivalence issues de la première étape. Après cette première étape il est possible de calculer des probabilités *a posteriori* pour les mots mais c'est seulement au bout de la deuxième étape que les hypothèses de mots se trouvant en concurrence peuvent être identifiées.
Les classes d'équivalence sont initialisées de manière à ce qu'elles contiennent toutes les transitions portant le même mot et ayant exactement le même support temporel (trame de début et de fin).

Regroupement "*intra-mots*"

Le but de cette première étape est de regrouper toutes les transitions portant la même hypothèse de mot. Ceci est réalisé en regroupant des classes d'équivalence constituées des transitions portant la même hypothèse de mot qui ne sont pas en relation d'ordre (voir 5.2.3). La fonction de coût utilisée pour cette étape est une fonction de similarité entre deux classes d'équivalence :

$$SIM(C_1, C_2) = \max_{\substack{e_1 \in C_1 \\ e_2 \in C_2}} overlap(e_1, e_2) \cdot P(e_1) \cdot P(e_2) \tag{5.4}$$

où $overlap(e_1, e_2)$ est le recouvrement temporel entre les deux transitions. La valeur du recouvrement temporel est calculée comme étant le nombre de trames communes des support temporels des deux transitions divisé par l'union de leur longueur. Si on considère l'exemple dans la figure 5.2, la formule de calcul du recouvrement temporel est la suivante :

$$overlap(e_1, e_2) = \frac{t(3) - t(2)}{t(4) - t(1)} \tag{5.5}$$

où $t(i)$ est la trame de création de l'état i.

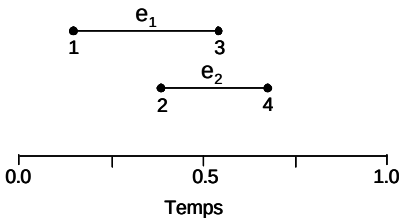

FIGURE 5.2 – Calcul du recouvrement temporel entre deux transitions.

Le recouvrement temporel est pondéré par les probabilités *a posteriori* des transitions

afin de rendre la mesure de similarité moins sensible aux hypothèses de mots peu probables (transitions ayant une faible probabilité). La valeur de la fonction de similarité entre deux classes d'équivalence est calculée en prenant tous les couples de transitions des deux classes et en calculant le produit entre leur recouvrement temporel et leur probabilité *a posteriori*, la meilleure valeur étant gardée.
A chaque itération, l'algorithme calcule la fonction de similarité entre toutes les paires de classes d'équivalence candidates (les classes qui ne sont pas en relation d'ordre et qui correspondent au même mot). Il regroupe ensuite pour chaque mot les 2 classes les plus similaires. A la fin de ce processus itératif on obtient un ensemble de classes d'équivalence, chaque classe étant constituée de transitions portant la même hypothèse de mot et ayant un recouvrement temporel non nul. Toutes les transitions portant la même hypothèse de mot ne se retrouvent pas forcément dans la même classe à cause des contraintes imposées par la relation d'ordre. Pour chaque classe on peut ainsi calculer la probabilité *a posteriori* du mot en sommant les probabilités *a posteriori* des transitions qui composent la classe.

Regroupement "*inter-mots*"

Dans cette étape, l'algorithme regroupe les classes d'équivalence portant des hypothèses de mots différentes. Les candidats à ce regroupement sont toutes les classes qui ne sont pas en relation d'ordre. L'algorithme s'arrête lorsqu'il n'y a plus de candidats (l'ordre total a été atteint).
La fonction de coût pour cette étape est une fonction de similarité basée sur la similarité phonétique entre deux mots :

$$SIM(C_1, C_2) = \underset{\substack{w_1 \in Mots(C_1) \\ w_2 \in Mots(C_2)}}{moy} sim(w_1, w_2) \cdot P_C(w_1) \cdot P_C(w_2) \tag{5.6}$$

où $P_C(w_1) = P(e \in C_1 : Mot(e) = w)$ et $Mot(e)$ est le mot porté par la transition e. La fonction de similarité phonétique $sim(w_1, w_2)$ se calcule sur une transcription phonétique[2] des mots. Sa valeur est égale à 1 moins la distance d'édition entre les transcription phonétiques de mots, sachant que la distance d'édition est normalisée par rapport à la somme des longueurs des transcriptions phonétiques. D'autres fonctions de similarité phonétique plus complexes peuvent être utilisées ; par exemple la fonction de similarité peut prendre en compte différents paramètres des phonèmes comme sa nature (voyelle/consonne) ou le voisement.

5.3.2 Algorithme du "*pivot*"

Un deuxième algorithme à été proposé en 20003 dans (Hakkani-Tur et Riccardi, 2003). Appelé aussi algorithme du "**pivot**", c'est un algorithme itératif qui réalise un regroupement guidé des classes d'équivalence. En effet, l'algorithme utilise une séquence

2. Dans (Mangu et al., 2000) on utilise la transcription phonétique la plus probable

de transitions, appelée *pivot*, comme point de départ et guide pour tout le processus de regroupement. Les classes d'équivalence sont initialisées de manière à ce que chaque classe contienne une seule transition du graphe. Intuitivement, on peut assimiler le graphe de mots à un nuage de points, où chaque point représente une classe d'équivalence. A chaque itération, l'algorithme insère des points du nuage dans le *pivot*, suivant certaines conditions, jusqu'à ce qu'il ne reste plus aucun point dans le nuage.

Le pivot

Le *pivot* est en fait un chemin complet du graphe de mots. Ce chemin peut être un chemin choisi de manière aléatoire. On peut aussi choisir le chemin le plus long, ou le meilleur chemin au sens du MAP. Ce chemin constitue le point de départ de l'algorithme. Les états de ce chemin héritent de l'information temporelle du graphe de mots. Ainsi, pour chaque état on garde l'instant temporel (la trame) du graphe de mots et on peut définir un support temporel entre deux états consécutifs.
Dans l'implémentation de l'algorithme nous utilisons le meilleur au sens MAP, un choix fait aussi dans (Hakkani-Tur et Riccardi, 2003).

Déroulement de l'algorithme

Nous présentons ci-dessous le déroulement de l'algorithme du "**pivot**" et nous détaillons ensuite chaque pas. Nous attribuons un nom à chaque étape (entre crochets) pour faciliter leur présentation dans la suite des travaux (notamment au chapitre 6).

1. Extraire la séquence *pivot*.

2. Pour toute transition T du graphe de mots parcouru dans un ordre topologique :

(1) *[Recherche de l'emplacement optimal]*
Trouver les deux états consécutifs $\hat{S}_s$ et $\hat{S}_f$ qui définissent le support temporel ayant le recouvrement temporel maximal, $overlap(T, \hat{S}_s, \hat{S}_f)$, avec T.

(2) *[Insertion transition]*
S'il n'existe aucune transition entre $\hat{S}_s$ et $\hat{S}_f$ qui soit en relation d'ordre avec T, insérer T entre les deux états.

(3) *[Création nouveau état et insertion transition]*
Sinon :

(a) Créer un nouveau état S_n.

(b) Changer l'état de fin de toutes les transitions qui partent de $\hat{S}_s$ à S_n [3].

(c) Insérer T entre S_n et $\hat{S}_f$.

Dans cette description, les états S_i sont des états du réseau de confusion et non pas des états du graphe de mots.
Le graphe de mots doit tout d'abord être ordonné de manière topologique (les états du

3. La création de l'état S_n est détaillée dans la suite du document. La transition T ne peut que venir "après" les transitions existante entre $\hat{S}_s$ et $\hat{S}_f$ du fait de parcours topologique du graphe

graphe sont ordonnés de telle manière que toute transition allant de l'état i à l'état j satisfait la condition $i < j$). Ensuite, le parcours du graphe de mots se fait lui aussi dans un ordre topologique, partant de l'état le plus petit (l'état du début de graphe) jusqu'au dernier état du graphe.

(1) Pour chaque transition T on doit chercher deux états consécutifs $\hat{S}_s$ et $\hat{S}_f$ qui définissent un support temporel ayant un recouvrement temporel maximal avec T. La valeur du recouvrement temporel est calculée comme étant le nombre de trames communes entre les supports temporels :

$$\begin{aligned}(\hat{S}_s, \hat{S}_f) &= \underset{S_s,S_f}{argmax}\ overlap(T, [S_s, S_f]) \\ &= \underset{S_s,S_f}{argmax}\ (min(t(S_f), t(Fstate(T))) - max(t(S_s), t(Sstate(T))) \end{aligned} \quad (5.7)$$

(2) On vérifie s'il existe une transition entre les deux états qui est en relation d'ordre avec T avec la précision que cette relation d'ordre ne peut être qu'une relation de précédence (la transition se trouvant entre les deux états ne peut que précéder T sur un chemin du graphe) du au parcours topologique du graphe de mots. Si cette transition n'existe pas, l'insertion du T entre les deux états peut se faire de deux manière différentes. S'il existe déjà une transition portant la même hypothèse de mot que T on ne fait que rajouter la probabilité *a posteriori* du T à celle de la transition existante. Sinon, on crée une transition entre les états $\hat{S}_s$ et $\hat{S}_f$ portant la même hypothèse de mot que T et ayant la même probabilité *a posteriori* (le pas 2.2 dans l'algorithme).

(3) Si toutefois il existe une transition entre $\hat{S}_s$ et $\hat{S}_f$ qui précède T sur un chemin du graphe, un nouvel état S_n est inséré dans l'alignement entre ces deux états (le pas 2.2.a). Un instant temporel est attribué à ce nouvel état $(t(S_n) \in (t(\hat{S}_s), t(\hat{S}_f))$. Les transitions qui partent de l'état $\hat{S}_s$ auront désormais comme état de fin le nouvel état créé S_n (le pas 2.2.b). On insère ensuite T entre l'état S_n et $\hat{S}_f$ (le pas 3.3.c). Dans (Hakkani-Tur et al., 2006) l'instant $t(S_n)$ est calculé comme étant la moyenne entre les instant temporels des états $\hat{S}_s$ et $\hat{S}_f$.

5.3.3 Algorithme de génération par regroupement en groupes d'états

L'algorithme de génération des réseaux de confusion présenté en 2006 dans (Xue et Zhao, 2006) part du principe que chaque état du CN est en fait un ensemble d'états du graphe de mots initial. Ainsi, l'algorithme se base sur le regroupement des états du graphe de mots en groupes d'états dans la première phase suivi de l'insertion des transitions entre les groupes d'états.
La construction des groupes d'états est très importante. Dans (Xue et Zhao, 2006) on propose de diviser les états du graphe de mots en un nombre fini de groupes d'états de manière à ce que les états de début et de fin de chaque transition se retrouvent dans deux groupes d'états consécutifs. L'algorithme de génération des CNs par regroupement en groupes d'états se base ensuite sur une série de suppositions. $N = \{n_0, n_1, \ldots\}$

représente l'ensemble des états et $\theta = \{T_0, T_1, \ldots\}$ représentes l'ensemble des transitions du graphe de mots, avec $t(n_i)$ désignant l'instant (trame) attribué à l'état n_i et $T_{u \to v}$ une transition du graphe avec les états de début et de fin qui sont respectivement u et v. $NS = \{N_0, N_1, \ldots\}$ représente l'ensemble des groupes des états dans le CN et $TS_{N_i \to N_j}$ est un ensemble de transitions ayant leurs états de début dans N_i et les états de fin dans N_j. Les propriétés suivantes sont supposées être vraies sur les réseaux de confusion :

a. $\forall n_i \in N_i$ et $n_j \in N_j$, $t(n_i) < t(n_j) \Rightarrow i \leq j$

b. $\forall n_i \in N_i$ et $n_j \in N_j$, $t(n_i) = t(n_j) \Rightarrow i = j$

c. $\forall T_{u \to v} \in \theta$, $u \in N_i$ *et* $v \in N_j$, si $T_{u \to v}$ appartient à l'ensemble de transitions $TS_{N_m \to N_n} \Rightarrow i \leq m < n \leq j$, avec $n = m + 1$. En effet, si les états de la transition ne sont pas dans deux groupes consécutifs, la transition doit être alignée sur deux groupes d'états consécutifs.

L'algorithme proposé dans (Xue et Zhao, 2006) est un algorithme itératif qui se déroule en trois étapes :

1. Trier les états du graphe de mots initial en fonction de l'instant $t(n_i)$ associé.
2. Inclure l'état n_0 dans le groupe d'états N_0.
3. Pour chaque état n_i parcouru dans l'ordre topologique ($i = 1, 2, \ldots$) :
   a) Si $n_{i-1} \in N_j$ et il n'existe aucune transition entre les états du groupe d'états N_j et n_i, alors n_i est inséré dans N_j. Sinon, n_i est inséré dans N_{j+1}.
   b) Pour chaque transition $T_{u \to n_i} \in \theta$, avec $u \in N_s$ et $n_i \in N_f$ si $f = s + 1$ alors la transition est insérée directement dans l'ensemble $TS_{N_s \to N_f}$. Sinon, la transition est insérée dans l'ensemble TS_{N_{n-1}, N_n}, avec $s + 1 \leq n \leq f$. L'ensemble de transitions est calculé en fonction de la probabilité *a posteriori* de la transition et du recouvrement temporel entre la transition et les différents ensembles possible : $n = \underset{s+1 \leq k \leq f}{argmax} SIM(TS_{N_{k-1}, N_k}, T)$.

$$SIM(TS_{N_{k-1}, N_k}, T) = \frac{1}{|TS_{N_{k-1}, N_k}|} \times \sum_{l \in TS_{N_{k-1}, N_k}} sim(w(l), w(T)) \cdot overlap(TS_{N_{k-1}, N_k}, T) \quad (5.8)$$

avec la fonction de similarité phonétique $sim(,)$ et la fonction de recouvrement temporel $overlap(,)$ calculées de la même manière que pour l'algorithme de regroupement par classes de transitions présenté au 5.3.1. La longueur $|TS_{N_{k-1}, N_k}|$ du groupe de transitions est égale à $T_{min}(N_k) - T_{max}(N_{k-1})$ où $T_{max}(N_i) = max\{t(n_i) : n_i \in N_i\}$ et $T_{min}(N_i) = min\{t(n_i) : n_i \in N_i\}$.

La complexité de calcul de cet algorithme $O(N)$ est linéaire avec le nombre de transitions dans le graphe de mots.

Lors de la génération d'un réseau de confusion, les transitions qui sont insérées dans les différents groupes peuvent être très longues en comparaison des transitions déjà présentes dans le groupe. Pour cette raison il n'est peut être pas raisonnable d'insérer

la transition dans ce groupe mais de faire en sorte qu'elle puisse être alignée avec deux groupes consécutifs comme dans la figure 5.3. Pour réaliser ceci, à l'étape 4.b, la transition pourra être insérée soit dans le groupe TS_{N_{n-1},N_n} soit dans TS_{N_{n-2},N_n}. Le calcul de la fonction SIM qui permet l'insertion de la transition reste inchangé.

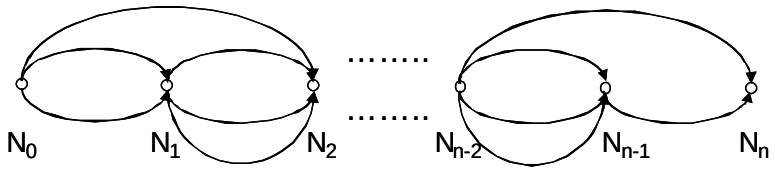

FIGURE 5.3 – Exemple du réseau de confusion modifié

Les tests effectués sur un corpus *Switchboard 2001 HUB-5* montrent des temps d'exécution très inférieurs à l'algorithme de regroupement par classes de transitions malgré un élagage systématique pour cet algorithme (seul 5% des transitions du graphe sont utilisées) et des graphes de mots ayant une taille raisonnable (2-10k transitions). En termes de *WER*, les deux algorithmes de génération de CNs obtiennent de meilleures performances que la meilleure solution du SRAP avec un très léger avantage pour l'algorithme de génération par regroupement en classes d'états.
Comme nous l'avons précisé, cet algorithme n'a pas fait l'objet de travaux dans cette thèse. Nous ne détaillons pas plus son fonctionnement mais nous avons souhaité le présenter néanmoins pour donner une vue complète sur l'état de l'art.

5.4 Discussions

Dans cette partie nous faisons tout d'abord une analyse critique de l'algorithme de regroupement par classes de transitions présenté dans la section 5.4.1. Nous présentons ensuite une comparaison entre les performances de cet algorithme et l'algorithme du "**pivot**" et nous motivons par la même occasion le choix de l'algorithme du "**pivot**" pour la génération des réseaux de confusion.

5.4.1 Algorithme de génération par regroupement des classes de transitions

L'algorithme de génération des réseaux de confusion proposé dans (Mangu et al., 2000) est un algorithme de type "*greedy*" (à chaque itération il prend la décision qui correspond à la meilleure du moment sans prendre en compte les conséquences sur la suite des décisions). Lorsqu'il regroupe deux classes, de nouvelles contraintes sont rajoutées à l'ordre partiel entre les classes restantes. En conséquence, certaines classes qui auraient pu être regroupées avant ne sont plus des candidats valides pour le processus de regroupement. Il est donc important d'utiliser des fonctions de similarité robustes qui soient capables de réaliser les regroupements dans le bon ordre et surtout de privilégier les hypothèses de mots ayant des probabilités *a posteriori* élevées.
L'analyse de cet algorithme porte sur deux aspects : d'un coté, le nombre de transitions

dans les graphes de mots qui influe directement sur la complexité de l'algorithme et qui nécessite l'introduction d'une étape d'élagage et de l'autre coté, l'aspect "*greedy*" de l'algorithme qui fait que les fonctions de similarité et la manière de regrouper les classes d'équivalence sont très importantes.

Élagage des transitions

Une grande majorité des graphes de mots contient des transitions ayant des probabilités *a posteriori* très petites. Ces transitions n'ont pas une influence très grande dans le calcul des probabilités *a posteriori* des hypothèses de mots dans le CN mais elles peuvent avoir une influence négative sur l'alignement et le regroupement des classes d'équivalence. Ceci peut arriver car l'algorithme de génération cherche à respecter à tout moment les relations d'ordre entre les transitions du graphe peu importe leur probabilité. Cette situation est illustrée dans la figure 5.4. Dans cet exemple qui montre un "morceau" d'un graphe de mots, les mots "*je*" et "*ne*" qui sont très proches du point de vue phonétique et ont un recouvrement temporel non nul devraient donc être regroupés ensemble dans la même classe d'équivalence. Du fait de la relation d'ordre, l'existence d'au moins un chemin dans le graphe contenant les deux hypothèses de mots (ce qui est le cas dans notre exemple), peu importe la valeur de la probabilité *a posteriori*, empêchera le regroupement.

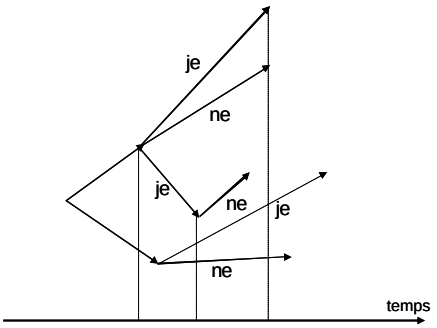

FIGURE 5.4 – Exemple d'un "morceau" d'un graphe de mots

Afin d'éviter ces situations, les auteurs ont introduit une étape d'élagage (*pruning*) des transitions des graphes de mots ayant une probabilité *a posteriori* en dessous d'un seuil établi de façon empirique. Ainsi, l'étape d'initialisation des classes d'équivalence et les deux étapes de regroupement ne traitent que les transitions ayant survécu à l'étape d'élagage.
Les testes menés dans (Mangu et al., 2000) sur un corpus de type *Switchboard* (le nombre de transitions du graphe par rapport au nombre de mots de la référence est de 1350 et le *WER* de la meilleure hypothèse MAP est de 38.5%) ont montré que l'utilisation de seulement 2% des transitions du graphe permet l'obtention d'une *consensus hypothesis* ayant le même *WER* que l'hypothèse du SRAP et que 5% des transitions permettent d'obtenir une amélioration du *WER* de 1.2%. Les tests ont aussi montré que pour une valeur de seuil d'élagage de 10% les performances en termes de *WER* restent constantes.

Toutefois aucune valeur du *WER* pour des CNs construits sans élagage des transitions n'est donnée. On ne sait donc pas si l'utilisation de toutes les transitions du graphe de mots influence de manière négative les performances des CNs.
Les tests montrent aussi que pour un seuil d'élagage de 5% (on utilise seulement 5% des transitions du graphe) le *WER* oracle sur CN [4] est équivalent à celui obtenu sur les graphes de mots [5] (10%). Pour des valeurs plus grandes du seuil le *WER* oracle est nettement amélioré (3.5% de réduction absolue pour un seuil d'élagage de 16%). L'utilisation d'une valeur du seuil d'élagage plus grande peut donc être justifiée si des algorithmes de post-traitement sont appliqués sur les CNs.
Un autre aspect très important lié au seuil d'élagage, et donc au nombre de transitions utilisées pour la génération des CNs, est le temps de calcul. Avec une complexité de calcul de $O(N^3)$ (N est le nombre de transitions dans le graphe), le temps de calcul peut vite devenir prohibitif pour une application temps réel dans laquelle les graphes de mots ont une taille importante. Il est important de pouvoir utiliser un seuil d'élagage qui réalise un bon compromis entre les performances des CNs en terme de *WER* et le temps de calcul nécessaire pour leur génération.
A la différence des graphes de mots utilisés dans (Mangu et al., 2000), qui ont un nombre moyen de transitions par graphe de 10000, les graphes de mots générés sur le corpus de test **Test_I**, que nous utilisons pour l'évaluation de l'algorithme, ont un nombre moyen de transitions par graphe de 26800. On retrouve un facteur 2.7 entre les deux valeurs. Les observations, faites dans (Mangu et al., 2000), en ce qui concerne l'élagage rendent d'autant plus nécessaire cette étape préliminaire d'élagage des transitions du graphe sur **Test_I**.

Fonctions de similarité

Un autre aspect important de cet algorithme est constitué par les fonctions de similarité 5.4 et 5.6 qui guident le regroupement des classes d'équivalence. Si pour l'étape "*intra-mots*" la fonction de similarité ne pose pas vraiment de problème, en revanche, pour l'étape "*inter-mots*", on peut se demander si la fonction de similarité phonétique joue ou non un role important dans le regroupement des classes d'équivalence. Ceci d'autant plus que les mots ont différentes variantes de prononciations qui ne peuvent pas être prises en compte dans le calcul de la distance d'édition entre deux transcriptions phonétiques. Les tests réalisés dans (Mangu et al., 2000) ont montré que le fait d'enlever cette similarité phonétique n'a aucune influence sur la valeur du *WER*. On peut conclure que la topologie du graphe de mots est une contrainte suffisante tant que les classes d'équivalence ayant une probabilité *a posteriori* grande sont regroupées en premier. La nouvelle fonction de coût devient :

$$SIM(C_1, C_2) = \underset{\substack{w_1 \in Mots(C_1) \\ w_2 \in Mots(C_2)}}{moy} P_C(w_1) \cdot P_C(w_2) \tag{5.9}$$

4. Le taux d'erreur oracle qui est défini pour les réseaux de confusion de la même manière que pour les graphes de mots.
5. Le taux d'erreur oracle est calculé avant élagage.

L'importance des probabilités *a posteriori* dans le regroupement des classes d'équivalence est aussi démontrée par un *WER* plus élevé lorsque les deux fonctions de similarité ne sont pas pondérées par ces probabilités. Dans ce cas, c'est l'augmentation de nombre d'omissions qui influe le plus sur le *WER*.
Comme nous l'avons expliqué, le bon ordre de regroupement des classes, particulièrement dans l'étape "*inter-mots*" de l'algorithme, est très important car tout regroupement crée de nouvelles relations d'ordre entre les classes restantes. Il est donc important de privilégier le regroupement des classes ayant la meilleure probabilité *a posteriori* mais aussi de tenir compte de leur alignement afin d'éviter de regrouper des classes correspondant à des intervalles de temps très écartés dans le temps. Le but étant de mettre en concurrence des hypothèses de mots dans un même intervalle temporel, il est logique de privilégier le regroupement des hypothèses de mots ayant le meilleur recouvrement temporel. Pour cela, nous proposons de modifier la fonction de similarité de l'étape "*inter-mots*" afin d'introduire la valeur du recouvrement temporel. La fonction devient :

$$SIM(C_1, C_2) = \underset{\substack{w_1 \in Mots(C_1) \\ w_2 \in Mots(C_2)}}{moy} overlap(ST(w_1), ST(w_2)) \cdot P_C(w_1) \cdot P_C(w_2) \tag{5.10}$$

où $overlap(ST(w_1), ST(w_2))$ est le recouvrement temporel entre les supports temporels des mots w_1 et w_2. Ces mots sont issus de l'étape "*intra-mots*" et chaque mot correspond à une classe d'équivalence. Comme nous l'avons expliqué, après cette étape on peut calculer la probabilité *a posteriori* des mots en sommant les probabilités *a posteriori* des transitions qui forment chaque classe d'équivalence. Afin d'utiliser la nouvelle formule de calcul de la fonction de similarité 5.10 pour l'étape "*inter-mots*" nous devons définir un support temporel $ST(w)$ pour chaque mot issu de la première étape. Les instants de début et de fin qui caractérisent le support temporel sont calculés comme étant la moyenne des instants de début et de fin des transitions du graphe qui composent la classe d'équivalence correspondant au mot w. Nous avons aussi essayé une autre façon de définir le support temporel en le rendant équivalent à celui de la transition ayant la meilleure probabilité *a posteriori* de la classe d'équivalence.
Les expériences [6] réalisées ont montré que l'utilisation de la fonction de similarité proposée dans l'équation 5.10 n'améliore pas les performances en terme de *WER*, qui reste constant, par rapport à l'implémentation de l'algorithme utilisant la fonction de similarité donnée par l'équation 5.9. Les deux définitions du support temporel $ST(w)$ produisent les mêmes performances.

5.4.2 Algorithme du "pivot"

L'algorithme du "**pivot**" a une complexité de $O(N \times k)$, où N est le nombre de transitions dans le graphe de mots et k est le nombre de classes finales dans le CN, qui en règle générale est beaucoup plus petit que N. Par exemple, si le meilleur chemin est

6. Les tests ont été réalisés sur un corpus de parole continue de 1600 énoncés contenant 6400 mots. Le nombre moyen de transitions par graphe est de 5000.

utilisé comme *pivot*, k est égal à la longueur du meilleur chemin plus le nombre d'états inséré dans l'alignement. Du point du vue de la complexité de calcul, cet algorithme est plus rapide que l'algorithme proposé dans (Mangu et al., 2000) qui a une complexité de $O(N^3)$.
Dans (Hakkani-Tur et al., 2006) on retrouve une comparaison entre les performances de l'algorithme du "**pivot**" et l'algorithme proposé dans (Mangu et al., 2000). Le corpus utilisé a été collecté à partir d'un service téléphonique de dialogue homme-machine *How may I help you ?* de AT&T (Gorin et al., 1997) et la comparaison des performances est réalisée sur une tache de classification binaire dans laquelle le système essaie de rejeter les mots mal reconnus en fonction d'un seuil sur le score de confiance des mots. Le score de confiance utilisé dans ce cas est la probabilité *a posteriori* des mots. Les performances des CNs sont supérieures par rapport aux performances du meilleur chemin du graphe. En revanche, les courbes des fausses alarmes par rapport aux faux rejets obtenues avec l'algorithme du "**pivot**" et avec l'algorithme de regroupement par classes de transitions sont identiques, mais les temps du calcul de l'algorithme du "**pivot**" sont nettement inférieurs (sans avoir eu recours à un élagage a priori des transitions du graphe de mots).

5.4.3 Choix de l'algorithme de génération des CNs

Dans cette partie nous faisons une comparaison entre les performances de la meilleure solution ASR et la meilleure solution des CNs générés à la fois par l'algorithme de regroupement par classes de transitions et par l'algorithme du "**pivot**".
Les tests ont été effectués sur un sous-corpus du corpus de test **Test_I**. Ce sous-corpus est composé de 1999 enregistrements pour un total de 5519 mots. Nous n'avons pas pu réaliser les tests sur l'intégralité du corpus de test **Test_I** pour des raisons liées à l'implémentation de l'algorithme de regroupement de transitions et à son temps d'exécution. Le tableau 5.1 montre les résultats obtenus sur ce sous-corpus en termes de *WER*. Le nombre de mots corrects ("C") et le détail des erreurs ("I" - insertions, "S" - substitutions, "O" - omissions) sont donnés dans ce tableau .

	WER	C	I	S	O
1-best ASR	44.4%	66.9%	11.3%	24.8%	8.3%
CN par regroupement des classes (seuil=5%)	44.2%	66.6%	10.8%	23.3%	10.1%
CN par regroupement des classes (seuil=30%)	43.0%	67.1%	10.1%	22.3%	10.6%
Algorithme du "**pivot**"	44.0%	66.5%	10.5%	23.8%	9.7%

TABLE 5.1 – *Comparaison des performances des algorithmes de génération des CNs*

La ligne "1-best ASR" donne les performances de la meilleure solution ASR. Les deux lignes suivantes correspondent à l'algorithme de regroupement par classes de transitions [7] avec deux valeurs différentes pour le seuil d'élagage des transitions. La dernière ligne de ce tableau correspond à l'algorithme du "**pivot**".

7. Nous avons utilisé la fonction de similarité décrite dans l'équation 5.9 pour l'étape "*inter-mots*"

Comme le montre le tableau 5.1, avec l'algorithme de génération des CNs par regroupement des classes et un seuil de 5% on obtient une légère amélioration du *WER* par rapport à la *1-best*. L'algorithme du "**pivot**" engendre une légère amélioration du *WER* par rapport à la meilleure solution ASR et par rapport à l'algorithme de base de génération par regroupement des classes de transitions. Une différence importante entre les deux algorithmes réside dans le nombre de transitions utilisées pour générer les CNs et dans les temps d'exécution. En effet, dans le cas de l'algorithme de regroupement par classes de transitions on a besoin de réaliser un élagage de transitions avant, en raison de la complexité de l'algorithme. Ceci n'est pas le cas pour l'algorithme du "**pivot**" qui utilise toutes les transitions du graphe. De plus, le temps d'exécution de l'algorithme du "**pivot**" est nettement inférieur.
Pour l'algorithme de regroupement par classes de transitions, l'utilisation d'un seuil d'élagage plus élevé engendre une amélioration du *WER* de 1.2% en absolu par rapport à l'utilisation d'un seuil de 5%. On observe une amélioration du *WER* en utilisant une valeur du seuil plus élevée, contrairement aux observation faites dans (Mangu et al., 2000). Cette amélioration dépasse aussi les performances de l'algorithme du "**pivot**". Toutefois, l'augmentation du seuil engendre une augmentation significative du temps d'exécution de l'algorithme.
Dans une application temps réel, tel que le service 3000, il n'est pas envisageable d'utiliser un algorithme de génération des CNs avec un temps d'exécution très long. Les performances en temps de calcul de l'algorithme de regroupement par classes de transitions le rend inadapté à une utilisation temps réel . En revanche, l'algorithme du "**pivot**" obtient des performances en terme de *WER* comparable et avec un temps d'exécution qui est nettement inférieur. Pour ces raisons nous avons décidé d'utiliser l'algorithme du "**pivot**" comme algorithme pour la génération des CNs dans nos travaux.

5.5 Conclusions

Dans ce chapitre nous avons présenté une approche de minimisation du *WER* basée sur les graphes de mots. Cette approche, proposée dans (Mangu et al., 2000) consiste en la construction d'un seul alignement pour toutes les hypothèses du graphe de mots. Nous avons détaillé les principes de construction de cet alignement, appelé réseau de confusion. Nous avons également présenté trois algorithmes de génération des réseaux de confusion. Une comparaison des performances de deux algorithmes implémentés est également présentée, et nous motivons aussi le choix de l'utilisation de l'algorithme du "**pivot**" dans la suite de nos travaux.

Troisième partie

Contributions et expérimentations

Chapitre 6

Optimisation de la construction des réseaux de confusion

Sommaire

Comme nous l'avons décrit dans le chapitre 3, dans un système de dialogue en langage naturel, le processus de reconnaissance vocale ainsi que l'analyse en concepts de l'hypothèse de reconnaissance sont transparents pour l'utilisateur. Les éventuelles erreurs à ces niveaux ne sont pas perçues directement par celui-ci. L'interprétation est ce qui guide la réponse du gestionnaire de dialogue et ce sont les erreurs au niveau interprétation qui reflètent le mieux les performances du système perçues par l'utilisateur.
Dans sa formulation théorique, l'algorithme de génération des CNs correspond à un critère de minimisation du taux d'erreur mot, alors que dans une application de dialogue on cherche à minimiser l'erreur au niveau interprétation. Nous proposons d'adapter l'algorithme du "**pivot**" dans le but de construire des CNs qui optimisent plus généralement les performances du processus complet d'interprétation, et ceci afin d'améliorer les performances du système perçues par l'utilisateur. Ces modifications introduisent un traitement différencié des mots du graphe et privilégient le traitement des mots porteurs de sens pour l'interprétation, favorisant ainsi une minimisation du taux d'erreur interprétation.
Le chapitre est organisé en six sections. Une analyse du comportement de l'algorithme

du "**pivot**" ainsi que des résultats obtenus sur le corpus **Test_II** sont présenté au 6.1. Dans la section 6.2 nous proposons une nouvelle approche heuristique pour le calcul des relations d'ordre entre les transitions. Les modifications apportées à l'algorithme du "**pivot**" sont présentées au 6.3. Dans la section 6.4 nous présentons l'influence des mesures de confiance sur les performances des CNs. Une étape d'élagage des classes finales des CNs est décrite au 6.5. Dans la dernière section 6.6 nous présentons une étape de post-traitement des CNs qui consiste en un algorithme de *parsing* basé sur les règles d'allumage des concepts.

6.1 Analyse de l'algorithme du "pivot"

Afin de pouvoir proposer des modifications de l'algorithme du "**pivot**" nous devons tout d'abord analyser le comportement de l'algorithme et la structure des CNs sur un corpus de données réelles. Pour réaliser cette analyse et pour présenter les performances des différentes modifications que nous détaillerons dans les sections suivantes, nous utilisons le corpus de test **Test_II**, composé de 6501 énoncés réels collectés à partir de l'application de dialogue en langage naturel 3000 (voir 4.3 pour une description plus détaillée). La **Methode 3** [1] est utilisée pour la normalisation lors de l'évaluation du *WER*, du *WER* oracle ou des métriques comme la précision et le rappel.

	WER Oracle	C	I	S	O
Graphes de mots	18.9%	84.8%	3.6%	10.6%	4.6%
CNs	6.2%	93.8%	0.1%	0.8%	5.4%

TABLE *6.1* – *Performances de l'algorithme du "**pivot**" en termes de* WER *oracle*

Le tableau 6.1 détaille les performances en terme de *WER* oracle de l'algorithme du "**pivot**". Les quatre dernières colones donnent le détail des erreurs pour les deux métriques (C - Mots corrects, I - Insertions, S - Substitutions, O - Omissions). On observe une nette amélioration des performances avec les CNs qui donnent un *WER* oracle trois fois inférieur à celui obtenu sur les graphes de mots. Ces performances sont dues au fait que la structure des CNs permet la création de nouveaux chemins qui n'existent pas dans le graphe. Le nombre très réduit d'insertions et de substitutions pour la solution oracle des CNs est une conséquence de la structure des réseaux de confusion. Lorsqu'on recherche la séquence la plus proche de la référence, les transitions portant l'omission, présentes dans un grand nombre de classes, permettent la construction d'un chemin entre deux mots de la solution oracle se trouvant dans deux classes non-adjacentes sans passer par d'autres mots. En effet, à la différence d'un graphe où un chemin entre deux mots non-adjacents passe obligatoirement par d'autres mots, dans un CN ce même chemin peut ne pas passer par d'autres mots si les classes parcourues contiennent toutes une transition portant l'omission. Le tableau 6.2 donne le détail du *WER* pour la *1-best*

1. Cette méthode est définie dans la section 4.4.1. Les énoncés non-parole ainsi que les énoncés ne contenant que des *OOV* et *SPR* sont considérés comme des énoncés à rejeter et sont annotés comme tel dans la référence et dans l'hypothèse de reconnaissance.

		WER [2]	C	I	S	O
Énonces non-parole	1-best	7.0%	6.5%	4.0%	3.1%	0.0%
1333 én.	*consensus hypothesis*	11.1%	3.7%	5.3%	5.8%	0.0%
Énonces parole	1-best	35.0%	67.4%	11.9%	17.5%	5.5%
5168 én.	*consensus hypothesis*	37.9%	65.6%	13.0%	18.6%	6.3%
Total	1-best	42.0%	73.9%	15.9%	20.6%	5.5%
6501 én.	*consensus hypothesis*	49.0%	69.4%	18.3%	24.4%	6.3%

TABLE 6.2 – WER *de la* 1-best *et de la* consensus hypothesis *sur les énoncés non-parole et les énoncés parole*

de la première passe de reconnaissance et la *consensus hypothesis*. Nous avons divisé le corpus en deux catégories : les énoncés non-parole (ne contenant que du bruit) et les énonces parole. Le *WER* sur chaque catégorie est calculé comme une contribution sur le *WER* de l'ensemble du corpus. Ainsi, le *WER* total de 42% = *WER* énoncés non-parole 7% + *WER* énoncés parole 35%.
On observe que, sur l'ensemble du corpus de test, les performances de la *consensus hypothesis* sont inférieures par rapport à la *1-best* avec une augmentation sur les trois types d'erreurs. On retrouve la même tendance sur chacune des deux catégories.
Les énoncés non-parole représentent 20% du total du corpus. 16% des erreurs sur la *1-best* sont produites par les énoncés de cette catégorie, alors que ce pourcentage est de 22% pour la *consensus hypothesis*. En effet, 68% des énoncés non-parole sont correctement détectés par la *1-best* alors que ce chiffre est de seulement 40% pour la *consensus hypothesis*. Ceci explique l'augmentation du nombre d'insertions et de substitutions sur cette catégorie pour la *consensus hypothesis*.

Distribution du même mot sur plusieurs classes adjacentes. Parcours topologique du graphe

Les systèmes de reconnaissance vocale rencontrent plus de difficultés à reconnaître les mots courts par rapport aux mots longs. Dans un graphe de mots, le système de reconnaissance vocale a tendance à générer un nombre élevé d'hypothèses pour le même mot mais avec des longueurs, instants de début et scores acoustiques différents. Il arrive fréquemment que, dans un CN, des hypothèses de mots différents ayant des longueurs différentes soient groupées dans la même classe. Des mots courts deviennent de surcroît des alternatives possibles pour des mots beaucoup plus longs. De ce fait, il arrive fréquemment que des hypothèses de mot représentant une meilleure alternative pour les mots longs de la classe ne puissent plus être insérées dans cette classe. Ceci se produit lorsque les mots courts, insérés lors des itérations antérieures, précèdent ces hypothèses de mots sur un chemin dans le graphe (on a une relation d'ordre entre l'hypothèse et le mot court de la classe).
La figure 6.1 montre un exemple réel, extrait d'un CN généré pour un enregistrement

2. Le *WER* calculé pour chaque catégorie est une contribution du *WER* total. On divise le nombre d'erreurs de chaque catégorie par le nombre total de mots de la référence sur l'ensemble du corpus.Pour les énoncés non-parole la référence est constituée d'une étiquette représantant un rejet.

du corpus de test. La référence de l'enregistrement contient le mot "**information**" qui est omis dans la *consensus hypothesis*. On observe que le mot "**information**" est présent dans deux classes adjacentes. Ceci est dû au fait que, dans le graphe, il existe des chemins qui contiennent la séquence de mots "**d' information**". Comme le mot "**d'**" est déjà inséré dans la première classe (du au parcours topologique du graphe), les transitions portant le mot "**information**" et qui sont précédées par une transition portant le mot "**d'**", insérée auparavant dans la classe, ne peuvent pas être insérées dans la même classe en raison d'une relation d'ordre entre les transitions.

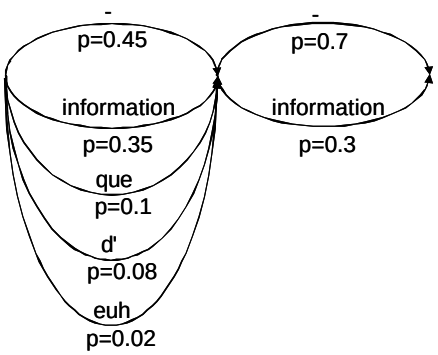

FIGURE *6.1* – Exemple de création forcée d'une classe ("-" représente l'omission)

La situation présentée dans la figure 6.1 est aussi due au fait que les transitions portant des mots courts dans le graphe ont des instants de début très différents. Une partie des transitions portant le mot "**information**", qui ont été insérées dans la première classe, sont aussi précédées des transitions portant le mot "**d'**". Mais en raison des instants de début très différents des transitions portant ce mot, certaines transitions sont regroupées dans la même classe avec le mot "**information**".
La conséquence de cette distribution d'une hypothèse de mot sur plusieurs classes adjacentes est l'omission du mot dans la *consensus hypothesis*. En effet, la masse de probabilité *a posteriori* est répartie sur deux classes au lieu d'être cumulée et dans chacune des deux classes la probabilité de "**information**" n'est pas assez élevée pour devancer celle de l'omission. Une analyse détaillée de ce phénomène nous a permis d'en déterminer les raisons. La figure 6.1 est très représentative dans ce sens, car elle montre clairement que si les transitions portant le mot "**information**" étaient traitées en premier elles auraient été insérées dans la "bonne" classe et aucune omission ne ce serait produite. Ce qui empêche cette insertion est l'ordre de traitement des transitions du graphe qui est donné par le parcours topologique du graphe de mots. On retrouve même des situations où des mots longs du *pivot* sont distribués sur plusieurs classes adjacentes à cause du traitement des transitions dans un ordre topologique.
Une conséquence de ce phénomène est le nombre inhabituellement élevé de classes par mot de la référence dans le réseau de confusion final. Le tableau 6.3 donne la largeur moyenne des CNs par rapport à la référence exprimée par le nombre moyen de classes par mot de la référence et la profondeur moyenne des CNs exprimée en nombre moyen de mots par classe. On observe que, pour un CN, l'algorithme génère en moyenne 34.8 classes par mot de la référence. On obtient ainsi des CNs avec une largeur moyenne de

	# Transitions / Graphe	# Transitions / Mot de la référence
Graphes de mots	16800	12000
	# Classes / Mot de la référence	# Mots / Classe
CNs	34.8	5.9

TABLE 6.3 – *Taille des CNs par rapport à la taille des graphes de mots*

73.1 classes (la longueur moyenne de la référence est de 2.1 mots). Au total cella représente une réduction de 98% du nombre de transitions par rapport aux graphes de mots initiaux. Cependant, la largeur reste importante et, en théorie, on pourrait considérer que pour chaque mot de la référence on a potentiellement 33.8 insertions de mots. Ce n'est pas le cas en pratique mais ce chiffre reste néanmoins élevé.
Une première modification de l'algorithme de génération des CNs que nous proposons est de ne plus parcourir le graphe dans un ordre topologique mais de privilégier d'abord certains ensembles de transitions suivant des critères que nous définissons par la suite. Ainsi, nous proposons de renforcer les mots du *pivot* afin d'éviter leur distribution sur plusieurs classes adjacentes. Les transitions portant des mots du *pivot* forment un groupe qui est traité en premier. A l'intérieur de ce groupe, les transitions sont traitées dans un ordre topologique. Dans la suite de la présentation nous appelons *pivot topologique*, l'algorithme initial du "**pivot**" tel qu'il est présenté au 5.3.2.

Prise en compte des connaissances sémantiques dans la construction des CNs

Un des objectifs principaux de ces travaux est de réaliser des CNs qui minimisent non seulement le *WER* mais, plus généralement, optimisent les performances du processus complet d'interprétation, et ceci afin d'améliorer les performances du système perçues par l'utilisateur. Pour ce faire nous devons tenir compte, dans le processus de génération des CNs, des spécificités du module de compréhension de la parole, tel que l'utilisation des concepts qui attribuent une plus grande importance aux mots non-vides (voir 3.3).

	IER	Correct	FR	Substitutions	FA
1-best ASR	22.7%	92.7%	1.6%	5.7%	15.4%
Consensus Hypothesis	21.8%	89.9%	4.1%	6.0%	11.7%

TABLE 6.4 – *Performances de l'algorithme du "**pivot**" en terme d'*IER.

Le tableau 6.4 présente les performances de la *1-best* ASR et de la *consensus hypothesis* en terme d'*IER*. Un détail des erreurs est aussi présenté (Substitutions, FR - Faux Rejets, FA - Fausses Alarmes). Contrairement aux performances en terme de *WER*, la *consensus hypothesis* améliore légèrement l'*IER*. Le point de fonctionnement en revanche est modifié.
Le tableau 6.5 présente cinq exemples d'énoncés, extraits du corpus de test, qui sont très représentatifs pour expliquer les différences entre les taux de FR et FA de la *1-best* et de la *consensus hypothesis*. Les trois premiers exemples correspondent à des FA et les deux suivants à des FR. Les deux premiers exemples correspondent à une FA présente dans

la *1-best* due au fait que l'hypothèse de reconnaissance est un mot non-vide qui produit une interprétation. La *consensus hypothesis* ne produit aucune interprétation. Le SRAP a plus tendance à produire une hypothèse de reconnaissance qui produit une interprétation pour ces types d'énoncés. Pour le troisième exemple, il s'agit d'une FA présente dans la *consensus hypothesis* sur un énoncé non-parole. En effet, une grande partie des erreurs de FA sur la *consensus hypothesis* provient de ce type d'énoncé. Pour la *1-best*, une partie des FA provient également des énoncés non-parole, mais une autre partie, assez importante, provient des énoncés comme celui présenté dans le deuxième exemple, ce qui n'est pas le cas pour la *consensus hypothesis*. L'accumulation des deux types d'énoncés font que le taux de FA est plus élevé pour la *1-best*. Le quatrième exemple est représentatif de la majorité des FR produits par la *consensus hypothesis*. En effet, sur cet exemple nous avons observé que le mot "**facture**" (qui est un mot non-vide) était distribué sur plusieurs classes adjacentes ce qui à conduit à son omission dans la *consensus hypothesis*. Le cinquième cas est moins fréquent et au total la *consensus hypothesis* génère plus de FR que la *1-best*.

	Mots		Interprétation	
	Référence	Hypothèse	Référence	Hypothèse
1-best	*bruit*	le l' aide	Rejet	Aide
consensus hypothesis		je		Rejet
1-best	*SPR*	annule	Rejet	Annuler
consensus hypothesis		euh ne s'		Rejet
1-best	*bruit*	<*REJET*>	Rejet	Rejet
consensus hypothesis		payer		PayerFacture
1-best	facture	facture	AmbiFacture	AmbiFacture
consensus hypothesis		*vide*		Rejet
1-best	paiement	je demande	PayerFacture	Rejet
consensus hypothesis		paiement		PayerFacture

TABLE 6.5 – *Exemples d'erreurs au niveau interprétation*

L'algorithme *pivot topologique* accorde une importance égale à tous les mots dans le graphe de mots. Ce n'est pas le cas du module d'interprétation pour lequel les mots non-vides ont une importance plus grande car ce sont eux qui allument les concepts qui, à leur tour, allument une interprétation. En plus de renforcer les mots du *pivot* en traitant en premier les transitions du graphe de mots contenant ces mots, nous proposons une deuxième modification importante de l'algorithme de génération des CN qui donne plus de poids aux mots non-vides par rapport aux mots vides lors du traitement des transitions restantes du graphe de mots. Pour ce faire, les transitions portant des mots non-vides seront traitées avant celle portant des mots vides. Les modifications proposées sont détaillées dans la section 6.3.

6.2 Nouvelle approche heuristique de la relation d'ordre entre transitions

La relation d'ordre joue un rôle très important dans la génération des réseaux de confusion. Elle doit être calculée à chaque fois qu'une classe d'équivalence, constituée d'une seule transition, est insérée dans le réseau (entre les états $\hat{S}_s$ et $\hat{S}_f$). Le calcul se fait pour savoir si la transition à insérer est en relation avec une des transitions de la classe d'équivalence (délimitée par ces états) dans laquelle elle doit être insérée. Le calcul ne se fait donc pas seulement pour une paire de transitions mais pour toutes les paires formées de la transition à insérer et d'une des transitions de la classe d'équivalence. En pratique on s'arrête dès qu'une paire de transitions se trouve en relation car cela signifie que toute la classe d'équivalence est en relation avec la classe d'équivalence correspondant à la transition à insérer. Compte tenu de la taille des graphes de mots du corpus de test **Test_II**, le nombre d'insertions de transitions est très grand ce qui entraîne un grand nombre de calculs de relations d'ordre.
Compte tenu de la définition de la relation d'ordre entre deux transitions e_1 et e_2 (donnée dans la section 5.2.3), les deux premières conditions, $e_1 = e_2$ et $Fstate(e_1) = Sstate(e_2)$ sont faciles et rapides à vérifier. La dernière condition doit déterminer s'il existe un chemin dans le graphe entre les deux transitions. Pour ce faire, elle doit soit vérifier l'existence d'un chemin dans le graphe à chaque fois, soit avoir une base de données qui répertorie l'existence d'un chemin entre toutes les paires de transitions du graphe de mots. Les trois conditions sont calculées successivement et on s'arrête dès qu'une condition est remplie. Si le nombre de fois où la troisième condition doit être testée est élevé, vérifier l'existence du chemin dans le graphe à chaque fois est coûteux en temps de calcul alors que la deuxième méthode peut s'avérer gourmande en termes de mémoire de stockage. Dans notre implémentation, c'est cette seconde solution qui a été retenue. Malgré une implémentation optimisée le parcours de la structure est coûteux en temps de calcul.
Nous avons décidé de compter le nombre de fois que chaque condition est utilisée lorsqu'une relation d'ordre entre deux transitions est déterminée. On observe que dans 84% des cas la relation d'ordre entre deux transitions est établie par une des deux premières conditions. Dans seulement 16% des cas une relation d'ordre est établie par la troisième condition. Ces pourcentages sont calculés par rapport au nombre de transitions pour lesquelles on a établi une relation d'ordre lors de leur insertion dans le CN. Pour les autres transitions du graphe, l'algorithme vérifie à chaque fois les trois conditions afin de conclure qu'aucune relation d'ordre avec les transitions des classes d'équivalences n'existe. Le nombre de transitions pour lesquelles une relation d'ordre est établie lors de leur insertion représente moins de 0.1% du nombre total de transitions dans un graphe. Le volume de calculs engendré par la vérification de la troisième condition est très important par rapport au nombre de fois ou celle-ci établi une relation d'ordre. Nous proposons une approche heuristique pour la mise en œuvre de la relation d'ordre entre deux transitions. Pour $e_1, e_2 \in E$, $e_1 \leq e_2$ si :

- $e_1 = e_2$ ou
- $Fstate(e_1) = Sstate(e_2)$ (où $Sstate$ et $Fstate$ sont les états de début et de fin d'une transition)

Nous n'examinons plus la troisième condition qui ne concernait que 16% des cas. Ceci engendre un gain de temps de calcul car on ne recherche plus un chemin entre deux transitions à chaque fois qu'une transition est insérée dans le CN. Du point de vue théorique, cette approche ne définit pas une relation d'ordre car elle ne vérifie plus la propriété de transitivité. Mais pour plus de simplicité nous continuons à l'appeler relation d'ordre dans la suite de ce document.
Cette nouvelle approche de la relation d'ordre soulève quelques questions. Elle permet désormais qu'une transition soit insérée dans une classe alors qu'il existe un chemin entre cette transition et une transition de la classe, les deux transitions n'étant pas adjacentes sur ce chemin. Le problème dans ce cas concerne la somme des probabilités *a posteriori* des mots de la classe qui doit être plus petite ou égale à 1. En effet, regrouper dans une classe deux transitions qui se trouvent sur un même chemin dans le graphe peut faire que la somme des probabilités *a posteriori* des mots de la classe dépasse 1.
Nous avons réalisé des tests sur le corpus de **Test_I** afin de mesurer la fréquence et l'importance de cet événement. Les CNs ont été générés avec l'algorithme du *pivot topologique* qui, pour le calcul de la relation d'ordre, utilise d'abord la définition décrite au 5.2.3, et ensuite l'approche que nous proposons. Nous avons observé, qu'en moyenne, la somme des probabilités *a posteriori* des transitions est plus grande que 1 pour moins de 0.01% des classes par enregistrement (cette somme est toujours plus petite que 1.1).

	# Classes / Mot de la référence	# Mots / Classe
définition de base	24.4	8.7
nouvelle approche heuristique	17.9	9.8

TABLE 6.6 – *Taille des CNs générés sur corpus **Test_I** avec la définition de base de la relation d'ordre et avec l'approche heuristique.*

Les tests effectués sur le corpus **Test_I** ont montré que les performances des CNs obtenus avec les deux définitions sont identiques en termes de *WER* et d'*IER*. Le *WER* oracle reste aussi constant. En ce qui concerne la taille des CNs, on observe dans le tableau 6.6 une diminution relative de 26% de la largeur des CNs due à l'utilisation de la nouvelle approche. L'utilisation de l'approche proposée permet également l'obtention d'un temps d'exécution d'environ 3 fois inférieur à celui obtenu avec la définition de base.
L'approche proposée pour le calcul de la relation d'ordre est utilisée dans les variantes de l'algorithme du "**pivot**" que nous proposons. Dans la suite du document, seul l'algorithme du *pivot topologique* utilise la définition de la relation d'ordre tel qu'elle est décrite au 5.2.3.

6.3 Regroupement des transitions guidé par le contexte applicatif

Comme nous l'avons décrit dans la section 6.1, suite à l'analyse approfondie du comportement de l'algorithme du *pivot topologique* sur un corpus de données réelles, nous proposons d'apporter des modifications (Minescu et Damnati, 2007) à l'algorithme

du *pivot topologique* dans le but de construire des CNs qui optimisent plus généralement les performances du processus complet d'interprétation.

6.3.1 Algorithme de génération multi-niveaux

Le principe de ce nouvel algorithme est d'utiliser des groupes de mots et d'insérer dans le CN les transitions correspondant à ces groupes en plusieurs étapes successives selon l'importance donnée aux groupes. Dans chaque étape les transitions correspondants à un groupe sont insérées dans un ordre topologique, mais sur l'ensemble du processus de génération ce n'est plus le cas. Il s'agit donc d'un algorithme multi-niveaux qui tient compte des mots portés par les transitions pour établir l'ordre d'insertion dans le CN lors de sa génération. L'objectif de ce nouvel algorithme est de rendre plus fiable la construction dans le CN des hypothèses portant sur des mots significatifs pour l'application (les mots non-vides).
L'insertion des transitions dans le nouvel algorithme se fait en quatre étapes. Comme nous l'avons expliqué, un traitement particulier est réservé d'abord aux mots du *pivot* qui sont traités en premier. Ensuite, on traite les mots non-vides ; leur insertion est guidée dans un premier temps par l'analyse en concepts de la séquence du *pivot*. Le calcul du *pivot* reste identique à l'algorithme *pivot topologique* et constitue le premier pas de l'algorithme suivi de la phase d'insertion des transitions. Nous décrivons ci-dessous les quatre étapes de la phase d'insertion des transitions dans le CN.

1. Pour toutes les transitions portant un mot du *pivot* :
   (1) *[Recherche de l'emplacement optimal]*
   (2) *[Insertion transition]*
2. Pour toutes les transitions portant un mot non-vide qui allume le même concept qu'un mot du *pivot* :
   (1) *[Recherche de l'emplacement optimal]*
   (2) *[Insertion transition]*
3. Pour toutes les autres transitions portant un mot non-vide :
   (1) *[Recherche de l'emplacement optimal]*
   (2) *[Insertion transition]* ou *[Création nouvel état et insertion transition]*
4. Pour toutes les transitions restantes qui portent un mot vide :
   (1) *[Recherche de l'emplacement optimal]*
   (2) *[Insertion transition]* ou *[Création nouvel état et insertion transition]*

Les trois processus *[Recherche de l'emplacement optimal], [Insertion transition]* et *[Création nouvel état et insertion transition]* ont été définis dans la description de l'algorithme du *pivot topologique* (voir la section 5.3.2). Les deux derniers sont des processus qui définissent l'insertion d'une transition dans le CN et le choix entre les deux se fait en fonction du calcul de la relation d'ordre entre la transition à insérer et les transitions de la classe d'équivalence correspondant à l'*emplacement optimal*. Nous utilisons l'approche proposée dans la section 6.2 pour le calcul de la relation d'ordre.

Une différence importante entre les deux premières étapes et les deux dernières réside dans le processus d'insertion d'une transition. Ainsi, dans les deux premières étapes on utilise seulement le processus *[Insertion transition]* qui implique que la transition à insérer ne doit pas être en relation d'ordre avec les transitions de la classe d'équivalence correspondant à l'*emplacement optimal*. Si toutefois cette relation d'ordre existe, la transition à insérer est traitée par une des deux dernières étapes (en fonction du mot porté). En procédant ainsi l'algorithme ne créé pas de nouvel état [3] dans le CN pendant les deux premières étapes. Le but est de générer des CNs plus compacts tout en évitant la dispersion des mots sur plusieurs classes adjacentes. Des précisions sur le fonctionnement de chaque étape sont données ci-dessous :

- **Etape 1.** Cette étape permet d'éviter la dispersion des hypothèses de mot du *pivot* sur plusieurs classes adjacentes, et donc de rendre plus représentative leur probabilité *a posteriori*.
- **Etape 2.** Un concept peut être allumé par plusieurs mots différents. Dans la majorités des cas, les mots ont une similarité phonétique [4] importante ce qui signifie que la probabilité de se retrouver en concurrence est assez forte. L'étape 2 permet ainsi de favoriser le regroupement des mots portant le même concept et de fiabiliser ainsi les hypothèses conceptuelles. Le cas des règles faisant correspondre une séquence de plusieurs mots à un concept n'est pas traité et les mots se trouvant dans ce cas sont traités à l'étape suivante.
- **Etape 3.** Cette étape permet l'insertion dans le CN des transitions portant des mots non-vides qui n'ont pas été traitées par les étapes précédentes. A la différence des deux premières étapes, la création d'un nouvel état lors de l'insertion des transitions est autorisée. Cette étape favorise le regroupement des mots non-vides et permet d'éviter une possible dispersion due aux mots vides.

La séparation des étapes 3 et 4 permet d'interrompre la construction des CNs dès l'étape 3. En effet, l'ajout des mots vides supplémentaires (seul ceux étant dans le *pivot* sont conservés) augmente la taille des réseaux alors que ces mots sont sans effet sur les traitements applicatifs en aval (l'allumage des concepts et l'obtention d'une interprétation). En effet, on observe une sur-génération des mots vides dans les graphes de mots (55% des occurrences) par rapport à seulement 35% dans le corpus de test.
Le fait d'arrêter l'algorithme à la fin de l'étape 3 donne la possibilité d'avoir deux variantes du même algorithme dont le choix réside dans les besoins de l'application choisie. Dans la suite de la présentation, nos appelons algorithme *multi-niveaux*, la variante qui comporte toutes les étapes et algorithme *multi-niveaux sans vides* la variante qui s'arrête après l'étape 3.

6.3.2 Performances du nouvel algorithme

Le tableau 6.7 montre les performances en terme de *WER* de la *consensus hypothesis* obtenue avec les algorithmes *multi-niveaux* et *multi-niveaux sans vides* et donne aussi le

3. La création d'un état est détaillé dans la section 5.3.2. Le processus *[Création nouvel état et insertion transition]* est le seul à pouvoir créer un nouvel état.

4. En général, lorsque plusieurs mots allument le même concept, ces mots sont les formes de singulier, pluriel, masculin, féminin du même mot

	WER	C	I	S	O
1-best ASR	42.0%	73.9%	15.9%	20.6%	5.5%
pivot topologique	49.0%	69.4%	18.3%	24.4%	6.3%
multi-niveaux	46.9%	70.3%	17.2%	23.1%	6.6%
multi-niveaux sans vides	43.1%	70.4%	13.5%	21.3%	8.3%

TABLE 6.7 – *Performances des variantes de l'algorithme* multi-niveaux *en termes de* WER.

détail des erreurs. Si le *WER* de la *1-best* reste légèrement meilleur que les performances des deux algorithmes, l'amélioration du celui-ci par rapport au *pivot topologique* est importante. La diminution de 2% du *WER* pour l'algorithme *multi-niveaux* s'explique par une meilleure gestion de l'insertion des transitions dans le CN qui se traduit par une baisse du nombre d'insertions et de substitutions pour un nombre d'omissions qui reste quasiment constant. En effet, comme montré dans le tableau 6.8, sur les mots vides, on obtient une augmentation de 3.7% de la précision par rapport à l'algorithme du *pivot topologique* ce qui se traduit par une réduction du nombre d'erreurs d'insertion et de substitution sur les mots vides. Dans le calcul des valeurs de précision et rappel dans le tableau 6.8, les énoncés non-parole ainsi que les énoncés vides sont assimilés à des mots vides. Pour les mots non-vides, l'augmentation de 1% du rappel pour l'algorithme *multi-niveaux* montre une amélioration de la reconnaissance de ces mots avec une augmentation du nombre de mots non-vides bien reconnus.

	Ensemble des mots		Mots non-vides		Mots vides	
	Précision	Rappel	Précision	Rappel	Précision	Rappel
1-best ASR	66.9%	73.9%	70.7%	86.7%	57.5%	50.7%
pivot topologique	61.9%	69.4%	68.4%	84.2%	46.2%	42.7%
multi-niveaux	63.6%	70.3%	68.9%	85.2%	49.9%	43.5%
multi-niveaux sans vides	67.0%	70.4%	69.6%	85.2%	59.3%	44.0%

TABLE 6.8 – *Performances des algorithmes en termes de précision et rappel*

En ne conservant que les mots non-vides présents dans le *pivot*, l'algorithme *multi-niveaux sans vides* permet d'obtenir une diminution de près de 4% du *WER* par rapport à l'algorithme *multi-niveaux* en évitant d'augmenter de façon importante le nombre d'insertions de mots vides. Pour les mots non-vides on observe une légère augmentation de la précision pour un rappel constant. Elle est due à une diminution du nombre d'insertions des mots non-vides. En effet, le fait de ne pas insérer dans le CN une grande partie des mots vides a pour résultat une augmentation de la probabilité *a posteriori* de l'omission dans certaines classes ce qui entraîne une diminution des insertions des mots non-vides provenant de ces classes. L'augmentation importante de la précision sur les mots vides est due à la diminution du nombre d'insertions sur ce type de mot. Cette diminution du nombre d'insertions a pour effet d'obtenir un certain nombre de *consensus hypothesis* vides. Comme pour ce calcul nous avons assimilé ces énoncés, ainsi que les énoncés non-parole, à des mots vides, l'obtention d'une *consensus hypothesis* vide pour des énoncés non-parole explique la légère augmentation du rappel sur les mots vides.
Le tableau 6.9 montre les performances en termes d'*IER* pour les deux algorithmes proposés ainsi que le détail des erreurs. Un des objectifs principaux de ces deux al-

	IER	C	FR	Sub	FA
1-best ASR	22.7%	92.7%	1.6%	5.7%	15.4%
pivot topologique	21.8%	89.9%	4.1%	6.0%	11.7%
multi-niveaux	20.2%	91.4%	3.4%	5.2%	11.6%
multi-niveaux sans vides	19.8%	91.5%	3.4%	5.1%	11.3%

TABLE *6.9* – *Performances des algorithmes en terme d'*IER

gorithmes était l'amélioration de la performance du système perçue par l'utilisateur qui passe par une amélioration du taux d'erreur interprétation. On observe ainsi une amélioration de 1.6% en absolu pour l'algorithme *multi-niveaux* par rapport au *pivot topologique*. Celle-ci est due principalement à une diminution du nombre de FR et de substitutions. Une légère amélioration de l'*IER* est obtenue avec l'algorithme *multi-niveaux sans vides* due principalement à une légère baisse du nombre de FA. Les performances équivalentes des deux algorithmes proposés s'expliquent, d'un côté, par le fait que le module de compréhension n'utilise pas les mots vides et de l'autre par le fait que le module de compréhension (plus précisément l'allumage des règles d'interprétation) n'est pas sensible aux insertions au niveau mots, qui contribuent principalement à la différence du *WER* entre les deux algorithmes. La diminution légère du nombre de FA est due à l'augmentation du nombre d'énoncés à rejeter pour lesquels la *consensus hypothesis* est vide ce qui produit également un rejet au niveau interprétation.

	# Classes / Mot de la référence	# Mots / Classe
pivot topologique	34.8	5.9
multi-niveaux	28.2	5.9
multi-niveaux sans vides	16.6	4.5

TABLE *6.10* – *Tailles des réseaux générés avec les différents algorithmes*

La réduction de la taille des CNs générés avec les algorithmes proposées est détaillée dans le tableau 6.10. On observe une réduction relative de 19% de la largeur des CNs générés avec l'algorithme *multi-niveaux* avec une profondeur qui reste constante. Ceci est dû principalement à l'ordre d'insertion des transitions dans le CN qui évite la création intempestive des classes. Le fait de ne pas insérer dans les CNs les transitions portant des mots vides (sauf ceux du *pivot*) permet d'obtenir des CNs encore plus compacts à l'aide de l'algorithme *multi-niveaux sans vides*. Ainsi, la réduction de la largeur des réseaux est de plus de 50% en valeur relative par rapport à l'algorithme *pivot topologique* et de 40% par rapport à l'algorithme *multi-niveaux*. La diminution de la profondeur des CNs de l'algorithme *multi-niveaux sans vides* est une conséquence logique de l'élimination des transitions portant un mot vide du processus de génération.
Le tableau 6.11 montre le détail des erreurs de l'oracle pour les algorithmes proposés. On observe, pour l'algorithme *multi-niveaux*, une légère dégradation des performances qui provient principalement d'une augmentation du nombre d'omissions. La légère augmentation du nombre de substitutions (de 0.8% à 1.4% pour l'algorithme *multi-niveaux* et 1.6% pour l'algorithme *multi-niveaux sans vides*) est due à la méthode d'évaluation du *WER* utilisée. En effet, si la *consensus hypothesis* ou la référence sont vides, la méthode **Methode 3** considère qu'il s'agit d'un rejet et les remplace par l'étiquette

	WER Oracle	C	I	S	O
Graphes de mots	18.9%	84.8%	3.6%	10.7%	4.6%
pivot topologique	6.2%	93.8%	0.1%	0.8%	5.3%
multi-niveaux	7.7%	92.5%	0.2%	1.4%	6.1%
multi-niveaux sans vides	14.4%	85.8%	0.2%	1.6%	12.6%

TABLE *6.11* – WER *oracle des algorithmes de génération des CNs*

<*REJET*>. Ainsi, l'augmentation des substitutions est due à une substitution d'un mot par cette étiquette ou inversement. En réalité, ces substitutions sont en fait des omissions ou des insertions. Dans le cas de l'algorithme *multi-niveaux sans vides* on observe une forte dégradation du *WER* oracle. Ceci est dû principalement à l'augmentation du nombre d'omissions qui est la conséquence directe de l'omission des transitions portant des mots vides (sauf ceux du *pivot*) dans la construction des réseaux.

Les mots vides sont filtrés	WER Oracle
Graphes de mots	14.4%
pivot topologique	2.8%
multi-niveaux	4.4%
multi-niveaux sans vides	4.0%

TABLE *6.12* – WER *oracle des algorithmes de génération des CNs en filtrant les mots vides*

Dans le tableau 6.12 nous avons réévalué le *WER* de l'oracle en ne gardant que les mots non-vides dans la référence et dans la solution oracle afin de faire une évaluation plus cohérente de l'algorithme *multi-niveaux sans vides*. Si la différence entre les *WER* oracle des algorithmes du *pivot topologique* et *multi-niveaux* est due principalement à une augmentation du nombre d'omissions, l'algorithme *multi-niveaux sans vides* améliore légèrement le *WER* oracle par rapport à l'algorithme *multi-niveaux*. Ceci montre bien que la forte dégradation du *WER* oracle observée dans le tableau 6.11 est due à l'absence de certain mots vides dans les CNs générés avec l'algorithme *multi-niveaux sans vides*.

6.4 Utilisation des mesures de confiance

Comme nous l'avons expliqué au 2.3, la probabilité *a posteriori* des mots dans les réseaux de confusion constitue une mesure de confiance fiable. Dans cette partie nous évaluons l'influence de cette mesure de confiance sur les performances en termes de *WER* et d'*IER* des CNs générés avec l'algorithme *pivot topologique* et avec les deux variantes de l'algorithme *multi-niveaux* proposées.
Nous utilisons ainsi la régression logistique (voir 2.2.4) pour évaluer la probabilité qu'un mot soit correct étant donné sa probabilité *a posteriori* ($P(COR|P(w|X))$). Un seuil est appliqué sur la probabilité ainsi évaluée des mots de la *consensus hypothesis*. Les mots ayant une probabilité d'être corrects étant donné leur probabilité *a posteriori* en dessous du seuil seront éliminés et nous calculons ensuite le *WER* et l'*IER* pour la nouvelle *consensus hypothesis*. Les paramètres de la régression logistique ont été évalués

sur le corpus de développement **Dev**. Nous utilisons la régression logistique afin de réaliser un calibrage des valeurs de la probabilité *a posteriori* des mots. Du fait d'une dynamique importante des valeurs de la probabilité *a posteriori* il est difficile d'établir une valeur absolue pour le seuil d'élagage. La régression logistique permet de réduire cette dynamique et une valeur pertinente du seuil d'élagage peut être établie.
La figure 6.2 montre les performances des algorithmes en termes de *WER*. Les courbes ont été obtenues en variant le seuil sur la mesure de confiance de 0 à 0.4. Chaque point sur la courbe correspond à une valeur du seuil en partant de la droite vers la gauche. Ainsi, le point le plus à droite sur chaque courbe correspond aux performances des algorithmes avec un seuil de 0, donc pas de filtrage de la *consensus hypothesis*. Le *WER* correspondant à chaque point est calculé en faisant la somme de l'abscisse, qui représente la somme des taux d'insertion et de substitution, et de l'ordonnée, qui représente le taux d'omission.

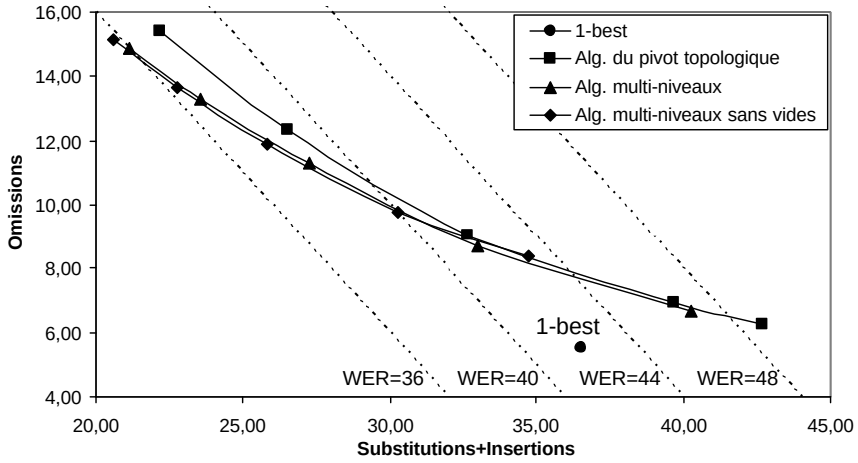

FIGURE 6.2 – *WER* de la *consensus hypothesis* en fonction du seuil sur la mesure de confiance

Comme on peut l'observer sur cette figure, plus on augmente la valeur du seuil plus le *WER* baisse. Ceci est vrai jusqu'à une valeur de 0.4 car pour des valeurs plus grandes le nombre d'omissions est trop important ce qui entraîne une augmentation du *WER*. Pour les trois algorithmes, l'utilisation de la probabilité *a posteriori* en tant que mesure de confiance permet d'améliorer le *WER* par rapport à la *1-best*. En ce qui concerne les performances des deux algorithmes proposés, on observe que leurs courbes sont quasiment superposées. L'utilisation de la mesure de confiance permet d'obtenir des performances équivalentes pour les deux algorithmes.
La figure 6.3 montre les performances en termes d'*IER* en fonction du seuil sur la mesure de confiance : la *consensus hypothesis* est filtrée en appliquant un seuil sur la probabilité $P(COR|P(w|X))$ et l'interprétation est calculée à partir de la séquence de mots filtrée.
L'utilisation des mesures de confiance permet une amélioration de l'*IER* avec un déplacement vers la gauche du point de fonctionnement du système. Contrairement aux performances en termes de *WER*, au delà d'une valeur du seuil de 0.2, l'*IER* augmente

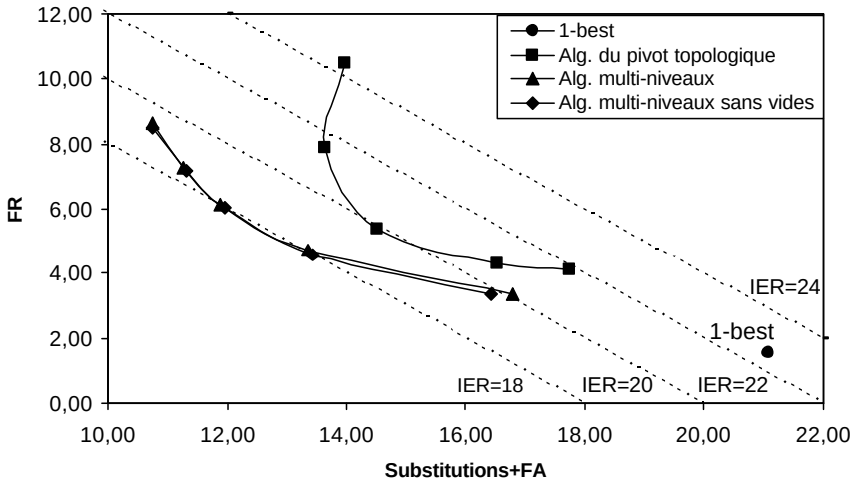

FIGURE *6.3* – Le point de fonctionnement en fonction du seuil sur la mesure de confiance

pour les trois algorithmes. Cette augmentation est très nette pour l'algorithme du *pivot topologique* par rapport aux algorithmes proposés. L'explication réside dans une augmentation plus forte du nombre d'omissions au niveau mot, en particulier pour les mots non-vides, pour l'algorithme du *pivot topologique*.
Afin d'intégrer l'utilisation des mesures de confiance dans le système de dialogue, le choix d'un point de fonctionnement est nécessaire. Ceci équivaut à choisir une valeur pour le seuil sur la mesure de confiance, est lié, d'un côté, à la valeur de l'*IER* et, de l'autre, à sa position sur la courbe. Dans tous les cas, une baisse du nombre de FA et des substitutions se fait au détriment du nombre de FR. Une substitution ou une FA peut entraîner une mauvaise décision du DM qui va orienter le dialogue dans une mauvaise direction. L'utilisateur se retrouve alors aiguillé vers un domaine qu'il n'a pas demandé et d'où il peut s'avérer difficile de revenir en arrière sans recommencer le dialogue de zéro. Un FR, en revanche, aura pour effet une réponse d'incompréhension du système et éventuellement une demande de répétition de la requête. Ceci est moins grave en apparence que le premier cas, mais trop de demandes de répétition peut rapidement faire baisser l'acceptation du système par l'utilisateur.

Diminution relative d'entropie

L'amélioration des performances des algorithmes de génération de CNs, obtenue grâce à l'utilisation de la probabilité *a posteriori* en tant que mesure de confiance, montre l'efficacité de celle-ci. Une autre méthode pour évaluer la fiabilité d'une mesure de confiance est le calcul de la diminution relative de l'entropie croisée ΔH (voir 2.1.3 pour plus de détail). Étant donné que celle-ci mesure l'information additionnelle apportée à l'hypothèse de mot par la mesure de confiance, plus ΔH est élevé plus la mesure de confiance est prédictive.
Le tableau 6.13 montre le détail des performances en termes de diminution relative

	ΔH (%)	H_{init}	H_{MC}
pivot topologique	29.3	0.6784	0.4795
multi-niveaux	30.1	0.6718	0.4694
multi-niveaux sans vides	26.3	0.6470	0.4767

TABLE 6.13 – *Performances de la mesure de confiance en termes de diminution relative d'entropie*

d'entropie. On observe ainsi que la mesure de confiance est plus performante sur l'algorithme *multi-niveaux* que sur les deux autres algorithmes. Effectivement, étant donné les performances équivalentes en termes de *WER* observées dans la figure 6.2 et une entropie croisée initiale plus élevée pour l'algorithme *multi-niveaux*, la réduction d'entropie est plus importante pour cet algorithme que pour l'algorithme *multi-niveaux sans vides*.
La réduction est donnée à titre indicatif, mais la comparaison est difficile dans la mesure où la précision initiale n'est pas la même.

6.5 Élagage *a posteriori* des classes du réseau de confusion

Nous avons montré qu'en utilisant l'algorithme *multi-niveaux sans vides* on peut obtenir une réduction de près de 99% du nombre de transitions dans un CN par rapport aux graphes de mots. Cette réduction est de 36% par rapport aux CNs générés avec l'algorithme *pivot topologique*. De plus ces algorithmes permettent une amélioration importante du *WER* oracle par rapport aux graphes de mots malgré un nombre d'hypothèses de mots très réduit par rapport aux graphes de mots. Toutefois, comme le montre le tableau 6.10, la largeur des CNs reste élevée alors que le nombre moyen de mots par classe est relativement petit. Nos efforts de réduction de la taille des CNs se concentrent donc sur une diminution de la largeur des CNs, tout en tenant compte de la valeur du *WER* oracle.
Une analyse détaillée des CNs a révélé que sur un nombre important de classes, l'omission a la plus grande probabilité *a posteriori* (souvent très proche de 1) de la classe et la différence avec le meilleur mot de la classe est très grande. Étant donné que ces mots ont des probabilités *a posteriori* très faibles, l'élimination de cette classe n'influe pas sur le *WER* oracle [5]. Nous proposons une étape d'élagage des CNs qui permet d'éliminer les classes qui satisfont les deux conditions suivantes :

1. L'omission à la plus grande probabilité *a posteriori* de la classe.
2. $\frac{P(omission)}{P(\text{la meilleure hypothèse de mot})} > Seuil$

Les figures 6.4 et 6.5 montrent les performances en termes de *WER* oracle ainsi que la taille des CNs en fonction du seuil d'élagage. Les figures ont été réalisées en variant le seuil à partir d'une valeur égale à 5 qui correspond au point le plus à gauche sur les courbes. Le point le plus à droite de chaque courbe correspond aux performances des CNs sans élagage.

5. Des expériences menées dans (Mangu, 2000) ont montré qu'en éliminant, dans une classe, les mots ayant une probabilité *a posteriori* inférieure à 0.1 le *WER* oracle reste inchangé.

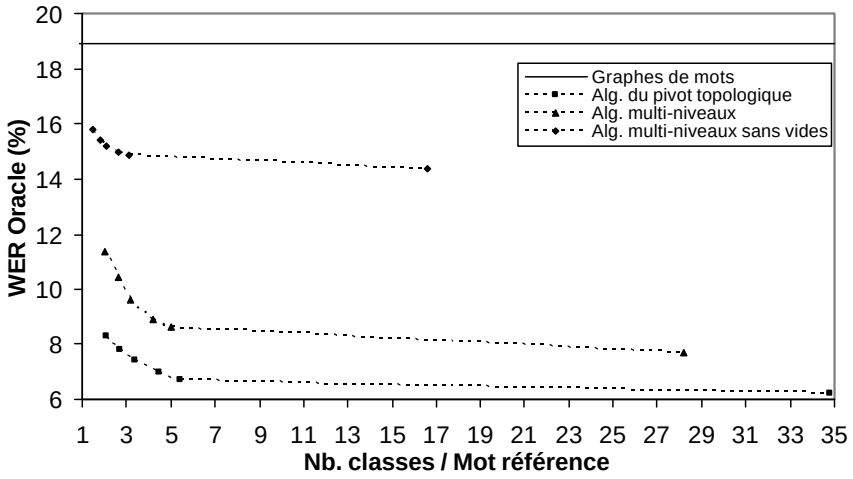

FIGURE *6.4* – *WER* oracle en fonction de la largeur des CNs pour différentes valeurs du seuil d'élagage

La figure 6.4 montre la variation du *WER* oracle et du nombre moyen de classes par mot de la référence en fonction du seuil d'élagage. La courbe correspondant à l'algorithme *multi-niveaux sans vides* montre un *WER* oracle plus élevé par rapport aux deux autres algorithmes pour les raisons que nous avons expliqué au 6.3.2 liés à la non-inclusion d'une grande partie des mots vides dans les CNs.
On observe sur la figure 6.4 que l'évolution de la taille des CNs en fonction du seuil d'élagage n'est pas la même pour les trois algorithmes. Ainsi, pour des valeurs de seuil petites (à gauche sur la courbe) le *WER* oracle pour l'algorithme *multi-niveaux* se dégrade plus rapidement comparé à l'algorithme *pivot topologique*. En ce qui concerne l'algorithme *multi-niveaux sans vides*, on observe une augmentation relative du *WER* oracle moins importante que pour les deux autres algorithmes ce qui montre que l'influence de l'élagage des classes est moins importante sur les performances de l'oracle. De plus, on observe que pour une valeur de seuil égale les CNs générés avec l'algorithme *multi-niveaux sans vides* ont une largeur moyenne 25% à 40% plus petites que pour l'algorithme du *pivot topologique*. La différence entre ce dernier et l'algorithme *multi-niveaux* est moins importante. La figure 6.5 montre l'évolution de la profondeur des CNs par rapport à leur largeur en fonction du seuil d'élagage des classes. On observe ainsi que plus l'élagage est fort (une valeur de seuil petite) plus le nombre moyen de mots par classe augmente fortement. Pour la même valeur du seuil on obtient une augmentation de seulement 60% pour l'algorithme *multi-niveaux* et de 70% pour l'algorithme *multi-niveaux sans vides*. Ceci montre bien l'existence des classes dont l'omission a la plus forte probabilité et qui contiennent un nombre d'hypothèses très faible.
L'effet de l'élagage des classes sur les performances des trois algorithmes est assez inégal d'un algorithme à l'autre. Toutefois, l'algorithme *multi-niveaux sans vides* permet de réduire fortement la taille des CNs avec une augmentation relative de *WER* oracle la moins importante. De plus, à seuil égal, les CNs générés ont la plus petite taille.
L'élagage des classes n'a aucun impact sur le *WER* et sur l'*IER*. Ceci est du au fait qu'on

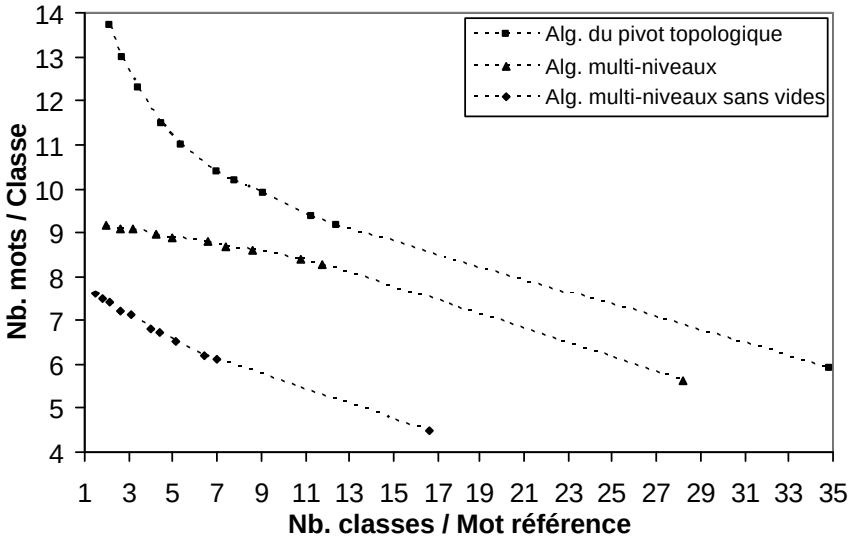

FIGURE *6.5* – Taille des réseaux en fonction des différentes valeurs du seuil d'élagage

élimine seulement des classes pour lesquelles l'omission a la plus grande probabilité *a posteriori*. Étant donné que lors de l'extraction de la *consensus hypothesis* on choisit les mots, ou l'omission, de chaque classe ayant la plus grande probabilité *a posteriori*, pour les classes éliminées le processus d'extraction aurait de toute manière choisi l'omission.

6.6 Parsing des réseaux de confusion

L'étape d'élagage des classes présentée au 6.5 permet de réduire de manière significative la taille des réseaux sans que les performances en terme de *WER* oracle soient dégradées de manière significative. Toutefois, cet élagage est une méthode générale de réduction de la taille des CNs qui ne tient pas compte des caractéristiques des modules applicatifs en aval. Nous proposons un algorithme de post-traitement des CNs qui vise à réduire la taille des CNs tout en privilégiant l'information porteuse de sens pour une application en aval.
Lors de l'analyse en concepts d'une séquence, le module de compréhension favorise l'allumage des concepts qui correspondent à des séquences contenant le plus de mots possibles. Dans un CN, les mots se trouvant dans des classes adjacentes ne forment pas nécessairement des séquences qui allument des concepts. Ceci est d'autant plus vrai pour les CNs générés avec l'algorithme *pivot topologique* ou *multi-niveaux* car les classes contiennent tous les mots vides présents dans les graphes de mots. Pour l'algorithme *multi-niveaux sans vides*, malgré le fait que seule une proportion réduite de mots vides sont présents dans les CNs, les mots non-vides présents dans des classes adjacentes ne forment pas nécessairement des séquences de mots qui correspondent à des concepts. En effet, certains mots non-vides n'allument un concept que s'ils font partie d'une séquence de plusieurs mots, alors que d'autres allument tout seuls un concept. Le pro-

cessus d'interprétation étant plus robuste si on privilégie l'allumage des concepts par des séquences de mots le plus longues possibles, l'algorithme de *parsing* que nous proposons vise à réduire la taille des CNs tout en favorisant la présence de ces séquences dans le CN.
Les graphes de mots, et par conséquence les CNs, contiennent des hésitations ou des erreurs dues aux disfluences. Si ces disfluences apparaissent dans la *consensus hypothesis* au milieu d'une séquence qui allume un concept, celui-ci ne pourra pas être allumé. Étant donné que les règles des concepts ont été construites manuellement elles ne contiennent pas ces disfluences et l'algorithme de *parsing* permet donc de les filtrer.

Fonctionnement de l'algorithme de parsing

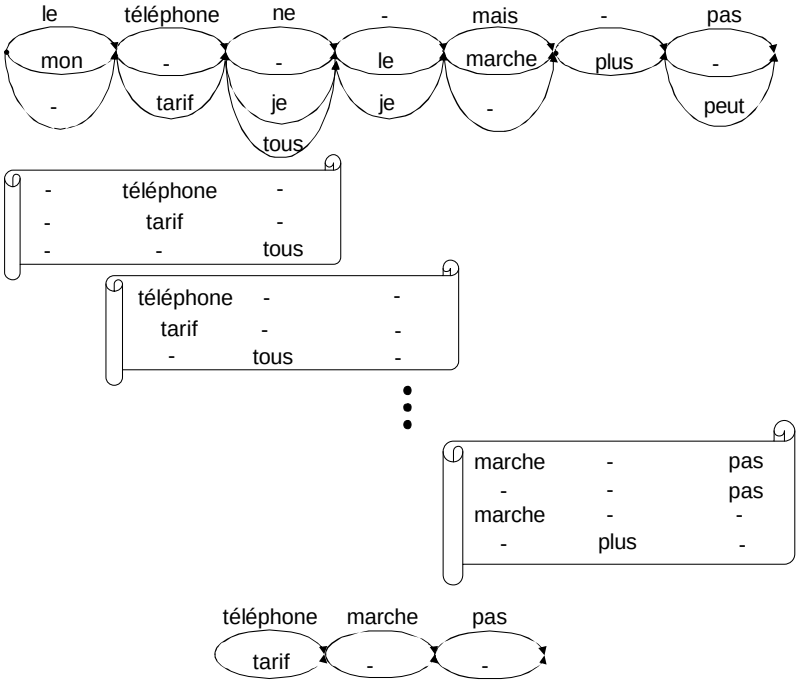

FIGURE 6.6 – Exemple illustrant le fonctionnement de l'algorithme de *parsing* d'un CN.

Nous illustrons le fonctionnement de l'algorithme de *parsing* à travers l'exemple présenté dans la figure 6.6. L'exemple correspond à l'énoncé "**mon téléphone ne marche pas**" et la *consensus hypothesis* extraite du CN est "**le téléphone ne mais pas**". Le tableau 6.14 présente un extrait des règles d'allumage de concepts qui ont dans leur composition un des mots non-vides du CN. Les mots présents dans le CN mais qui ne le sont pas dans les règles sont des mots vides.
Etant donné ces règles, l'analyse en concepts de l'énoncé et de la *consensus hypothesis* est donnée dans le tableau 6.15.
Le principe de l'algorithme de *parsing* est de chercher des séquences de mots qui al-

Séquence de mots	Concept
marche pas	[Probleme]
marche	[Fonction]
ne plus	[Plu]
plus	[Plus]
tarif	[Tarif]
téléphone	[Telephone]
tous	[Tout]

TABLE 6.14 – *Liste des règles d'allumage de concepts*

	Mots	Concept	Interprétation
énonce	mon téléphone ne marche pas	[Telephone] [Probleme]	Serv(DixQuatorze)
consensus hypothesis	le téléphone ne mais pas	[Telephone] [Pas]	Rejet

TABLE 6.15 – *Analyse en concepts de l'énoncé et de la* consensus hypothesis *avant le* parsing.

lument des concepts sur une fenêtre couvrant plusieurs classes adjacentes. L'algorithme de *parsing* est un algorithme itératif, à chaque itération la fenêtre se déplace d'une classe, le but étant de parcourir l'ensemble du CN. Dans notre exemple, la fenêtre couvre 3 classes adjacentes. Ainsi, comme le montre la figure 6.6, la première fenêtre extrait toutes les séquences des mots qui allument un concept. Par exemple, la séquence "**- téléphone -**" qui allume le concept "**[Telephone]**" est formée de deux omissions correspondant à la première et à la troisième classe et du mot "**téléphone**".
Pour chaque séquence de mot extraite ainsi on calcule une probabilité de la manière suivante :

$$P(\text{séquence}) = (\prod_{i=1}^{F} P(w_i))^{1/n} \tag{6.1}$$

où F est la longueur de la fenêtre et n le nombre de mots dans la séquence. Pour notre séquence, $n = 1$ car on a un seul mot dans la séquence, et la probabilité est $P(\text{- téléphone -}) = (P(-) \cdot P(\text{téléphone}) \cdot P(-))^{1/1}$.
Comme nous l'avons expliqué, nous voulons favoriser les séquences contenant plusieurs mots. Dans ce sens, de manière générale, plus il y a des mots dans une séquence plus sa probabilité est grande. En effet, $\prod_{i=1}^{F} P(w_i)) < 1$ du fait qu'on multiplie que des valeurs inférieures ou égales à 1, et la racine n^{ime} est plus grande pour une séquence contenant plus de mots (n plus grand). Dans notre exemple, cette situation est observée dans la dernière fenêtre.
A chaque itération de l'algorithme, la fenêtre se déplace d'une classe vers la droite, on extrait les séquences de mots et on calcule leurs probabilités. L'algorithme s'arrête lorsque toutes les classes du CN on été couvertes. Dans notre cas, la dernière fenêtre pour laquelle on extrait les séquences de mots est celle contenant les trois dernières classes du CN. Dans chaque fenêtre dans l'exemple, les séquences de mots sont ordonnées en fonction de leur probabilité. A la fin de l'algorithme, on construit le CN en

tenant compte des séquences de mots extraites dans chaque fenêtre et de leur probabilité. Ainsi, on peut choisir de prendre en compte que le N meilleures séquences dans chaque fenêtre. Dans l'exemple, on prend $N = 2$ et on reconstruit le CN avec seulement les mots contenus dans les deux premières séquences de chaque classe. Pour le reconstruire, on parcours le CN initial, et on regarde dans chaque classe si les mots (ou l'omission) qu'elle contient font partie des deux première séquences de chaque fenêtre dont la classe fait partie. Par exemple, pour la deuxième classe, les deux mots font partie des deux premières séquence, mais pas l'omission. Seuls les deux mots sont utilisés et forment une classe dans le CN reconstruit. Si aucun mot (ou omission) d'une classe n'est utilisé, le CN reconstruit ne contient pas de classe correspondante, comme c'est le cas dans l'exemple pour la première classe du CN initial.
La probabilité *a posteriori* des mots dans le CN reconstruit, est calculée à partir des probabilités *a posteriori* de mots dans le CN initial qu'on normalise pour que leur somme soit égale à 1 dans le CN reconstruit. La *consensus hypothesis* est calculée en se basant sur les nouvelles probabilités *a posteriori* et elle est ensuite traitée par le module de compréhension pour obtenir une interprétation. Comme le montre le tableau 6.16, suite au *parsing*, la nouvelle *consensus hypothesis* produit la même interprétation que celle de l'énoncé.

	Mots	Concept	Interprétation
énonce	mon téléphone ne marche pas	[Telephone] [Probleme]	Serv(DixQuatorze)
consensus hypothesis après *parsing*	téléphone marche pas	[Telephone] [Probleme]	Serv(DixQuatorze)

TABLE *6.16* – *Analyse en concepts de l'énoncé et de la* consensus hypothesis *après le* parsing.

L'algorithme de *parsing* se base sur deux paramètres qui peuvent varier en fonction des besoins applicatifs :

La largeur F de la fenêtre La largeur de la fenêtre est choisie en fonction de la longueur maximale des séquences de mots. Dans notre cas, les règles de concepts contiennent des séquences d'au maximum quatre mots. Un deuxième facteur qui entre en jeu est le nombre d'omissions entre deux mots d'une séquence. Dans l'exemple présenté, la séquence "**marche - pas**" contient une omissions entre les mots "**marche**" et "**pas**". Il convient d'établir un nombre maximal d'omissions entre les mots d'une séquence afin de ne pas former des séquences qui couvrent un nombre trop grand de classes (comme la séquence "**ne - - plus**" dans l'exemple) et qui ne reflète plus la réalité (un nombre élevé d'omissions allonge le support temporel de la séquence). Nous considérons qu'une seule omission peut exister entre le premier et le dernier mot d'une séquence. La largeur de la fenêtre est alors égale à la somme entre le nombre maximal de mots dans une séquence et le nombre d'omissions. Pour l'application de dialogue utilisée, $F = 5$.

Le nombre N de séquences par fenêtre Le nombre de séquences choisies est un compromis entre les performances des CNs en termes de *IER* et la taille des CNs. Nous avons choisi $N = 30$ pour nos évaluations.

Résultats expérimentaux

Les tests ont été effectués sur le corpus de test **Test_II** pour les algorithmes *multi-niveaux* et *multi-niveaux sans vides*. Nous nous sommes tout d'abord intéressé à l'influence de l'algorithme de *parsing* sur les CNs générés par l'algorithme *multi-niveaux* afin de voir si on pouvait améliorer d'avantage les performances en termes d'*IER*. Nous avons également comparé les performances des CNs sur lesquelles on applique une étape d'élagage de classes et ensuite le *parsing*, aux performances des CNs qui subissent seulement l'étape de *parsing*. Cette comparaison a pour but d'évaluer l'influence de la taille initiale des CNs sur l'algorithme de *parsing*.

	IER	C	FR	Sub	FA
1-best ASR	22.7%	92.7%	1.6%	5.7%	15.4%
multi-niveaux	20.2%	91.4%	3.4%	5.2%	11.6%
multi-niveaux + parsing	23.4%	90.6%	3.2%	6.2%	14.0%
multi-niveaux + élagage classes + parsing	22.7%	91.2%	3.2%	5.6%	13.9%

TABLE 6.17 – *Performances du parsing sur les CNs générés avec l'algorithme* multi-niveaux. *Evaluation au niveau interprétation*

Le tableau 6.17 montre les performances en terme d'*IER* de l'algorithme de *parsing* sur la *consensus hypothesis* des CNs générés avec l'algorithme *multi-niveaux*. Les deux premières lignes sont un rappel des performances de la *1-best* et de l'algorithme *multi-niveaux*. Dans la dernière ligne de tableau, les CNs subissent une étape d'élagage de classes avec un seuil de 10 avant le *parsing*. Ce seuil nous a paru être un bon compromis entre la taille et le *WER* oracle. On observe une dégradation des performances pour l'algorithme de *parsing* appliqué sur les CNs sans étape d'élagage qui est due principalement à l'augmentation du nombre de substitutions et de FA. Cette augmentation s'explique par une augmentation du taux de substitution et d'insertion au niveau concept, comme on observe dans le tableau 6.18. Les erreurs d'insertion au niveau concept proviennent principalement des concepts allumés par un seul mot. Le problème réside dans la taille du CN initial (28.2 classes par mot de référence, cf tableau 6.19) qui rend difficile l'extraction des séquences de plusieurs mots avec une fenêtre de largeur 5. Ceci résulte en un nombre important d'insertions de concepts allumés par un seul mot. Par ailleurs, on observe pour *multi-niveaux+élagage classes+parsing* une diminution du nombre d'insertions au niveau concept du fait d'un nombre de classes par mot de la référence beaucoup plus petit, ce qui engendre une baisse du nombre de substitutions au niveau interprétation. On obtient ainsi des performances équivalentes à celles de la *1-best*.
Le tableau 6.19 montre l'influence de l'algorithme de *parsing* sur la taille des CNs. On observe ainsi une réduction importante de la largeur de CNs sans l'étape d'élagage des classes. L'explication pour cette réduction du nombre de classes est donnée dans l'exemple montré dans la figure 6.6. L'algorithme de *parsing* permet aussi une légère réduction de la taille des CNs même si une étape d'élagage des classes est appliquée préalablement (on passe de 2.6 à 2 classes par mot de la référence).

	CER	C	FR	Sub	FA
multi-niveaux	21.0%	85.1%	6.2%	12.1%	2.7%
multi-niveaux + parsing	25.6%	83.5%	9.1%	13.9%	2.6%
multi-niveaux + élagage classes + parsing	23.3%	83.5%	6.8%	13.9%	2.6%

TABLE *6.18* – *Performances du parsing sur les CNs générés avec l'algorithme* multi-niveaux. *Evaluation au niveau concept*

	# Classes / Mot de la référence	# Mots / Classe
multi-niveaux	28.2	5.9
multi-niveaux + élagage classes	2.6	9.1
multi-niveaux + parsing	18.1	3.8
multi-niveaux + élagage classes + parsing	2.0	5.4

TABLE *6.19* – *Influence du parsing sur la taille des réseaux générés avec l'algorithme* multi-niveaux.

Nous avons aussi appliqué l'algorithme de *parsing* sur les réseaux générés avec l'algorithme *multi-niveaux sans vides* avec un élagage préalable des classes (le seuil d'élagage est égal à 10). Le tableau 6.20 montre les performances en termes d'*IER*. On observe ainsi une dégradation des performances par rapport à l'algorithme *multi-niveaux* due à l'augmentation des FA. Ceci s'explique par une tendance de l'algorithme de *parsing* à faire ressortir les séquences de mots formées d'un seul mot sur les CNs générés par l'algorithme *multi-niveaux sans vides* correspondant à des énoncés à rejeter.

	IER	C	FR	Sub	FA
multi-niveaux sans vides + élagage classes + parsing	24%	91.3%	3.2%	5.5%	15.3%

TABLE *6.20* – *Performances du parsing sur les CNs générés avec l'algorithme* multi-niveaux sans vides.

Discussions

Nous avons présenté dans cette section une étude préliminaire sur un algorithme de *parsing* des CNs. Nous avons observé que l'algorithme de *parsing* est plus efficace si les réseaux ne sont pas trop grands et donc une étape d'élagage des classes peut être nécessaire avant d'appliquer le *parsing*. Les performances moins bonnes en terme d'*IER* des CNs ayant subi un *parsing* ont pour cause principale un nombre trop important d'insertions des concepts allumés par un seul mot. Ceci est dû à un nombre réduit de règles d'allumage de concepts dont les séquences ont plus d'un mot (10% du total des règles). Nous pensons que dans un contexte avec des concepts plus couvrants le *parsing*

devrait être plus efficace.
Une autre piste pour l'amélioration de l'algorithme réside dans le calcul de la probabilité des mots dans le CN obtenu après le *parsing*. La nouvelle probabilité *a posteriori* pourrait être calculée à partir de la probabilité *a posteriori* du mot dans le CN initial mais avec un facteur de pondération donné par le nombre de séquences, extraites dans les fenêtres et qui contiennent le mot. Une autre possibilité serait d'estimer la nouvelle probabilité comme étant égale à la fréquence d'apparition d'un mot dans les séquences de mots extraites dans les fenêtres.

6.7 Généralisation de l'algorithme

Comme nous l'avons expliqué dans l'introduction de ce chapitre, le but des modifications présentées n'était pas de construire un algorithme dédié à un système de dialogue homme-machine en langage naturel comme celui utilisé par le service 3000. Les modifications présentées peuvent facilement être généralisées afin qu'elles puissent être utilisées dans d'autres domaines d'application.
L'objectif de ces modifications est de rendre plus fiable la construction des CNs en donnant une importance plus grande aux mots significatifs de l'application choisie. Le principe est de définir des groupes de mots auxquels on attribue des degrés différents d'importance qui définiront l'ordre d'insertion des transitions dans le CN et éventuellement les transitions qui ne sont pas insérées. L'algorithme généralisé comprend alors deux étapes. La première reste celle du calcul de la séquence *pivot*. La deuxième étape est une étape itérative qui traite les groupes de mots en fonction de leur degré d'importance. Ainsi, à chaque itération, l'algorithme insère dans le CN les transitions portant des mots d'un ou plusieurs groupes ayant le même degré d'importance. Un paramètre supplémentaire qui peut être changé à chaque itération est l'interdiction ou non de la création de nouvelles classes dans le CN lors de l'insertion d'une transition. Les groupes de mots qu'on ne souhaite pas introduire dans le CN reçoivent le degré d'importance le plus petit. De cette façon, on peut arrêter les itérations à partir d'un degré d'importance trop petit de manière à ce que les transitions portant ces mots ne soient pas traités par l'algorithme.
Nous avons appliqué cet algorithme pour la génération des CNs pour une application de traduction automatique de la parole continue. Les groupes de mots ont été construits avec les mots appartenant aux classes de mots utilisées par le modèle de traduction. Dans cette application de traduction nous avons également utilisé l'algorithme de *parsing* afin de réduire la taille des CNs tout en tenant compte des syntagmes de la table de traduction utilisée par l'application. Une explication plus détaillée ainsi que les résultats obtenus sont présentés dans les annexes B et C.

6.8 Conclusions

Ce chapitre a présenté dans un premier temps une analyse détaillée du fonctionnement de l'algorithme *pivot topologique*. Nous avons ainsi montré un phénomène de dédoublement de mots sur plusieurs classes du CN dû au parcours topologique des transitions et qui influe de manière négative sur le *WER* de la *consensus hypothesis* mais aussi sur la taille des réseaux. Un autre point abordé a été la prise en compte des caractéristiques du module de compréhension dans la génération des CNs. Les problématiques découvertes lors de cette analyse nous ont ensuite permis de proposer des modifications de l'algorithme du *pivot topologique*.
Tout d'abord, une approche heuristique pour la mise en œuvre de la relation d'ordre entre deux transitions a été proposée afin de rendre son calcul plus rapide et moins gourmand en ressources mémoire. Cette approche permet de générer des CNs plus compacts sans modifier leur performance en termes de *WER* et d'*IER*. Nous avons ensuite proposé deux versions d'un algorithme de génération multi-niveaux, permettant à la fois une amélioration significative des performances en termes de *WER* et d'*IER* et aussi de réduire de manière importante la taille des CNs. L'utilisation des mesures de confiance pour filtrer la *consensus hypothesis* permet d'améliorer davantage les performances. La différence entre les deux variantes de l'algorithme multi-niveaux réside dans le choix des objectifs de l'application choisie. Ainsi, si on a besoin d'un espace des hypothèses plus grand avec de bonnes performances au niveau interprétation, on choisi l'algorithme *multi-niveaux*. En revanche, si on se trouve dans un contexte avec de fortes contraintes de temps de calcul et de tailles des données, l'algorithme *multi-niveaux sans vides* peut constituer un meilleur choix.
Deux étapes de post-traitement des réseaux de confusion ont aussi été proposées. L'étape d'élagage des classes du CN permet de réduire de manière importante la taille des CNs avec une dégradation des performances de l'oracle moindre. L'algorithme de *parsing* permet de réaliser une réduction de la taille des CNs tout en tenant compte des caractéristiques des modules applicatifs en aval. Les résultats obtenus sont encourageants et montrent l'intérêt d'effectuer un élagage des réseaux en privilégiant l'information porteuse de sens pour une application en aval. Des perspectives d'amélioration de l'algorithme ont été discutées. Nous avons également utilisé l'algorithme de *parsing* dans un contexte de traduction automatique de la parole basée sur la traduction des CNs. Cet aspect est présenté dans les annexes B et C.

Chapitre 7

Stratégies de décision

Sommaire

Dans une application de dialogue, la compréhension d'un message est réalisée par l'analyse de la transcription issue du module de reconnaissance. En ce qui concerne le service 3000, l'interprétation de cette transcription passe par une succession d'étapes contrôlées par des sources de connaissance imparfaites (les règles d'allumage de concepts, les règles d'interprétation) qui traitent des données qui peuvent être incorrectes (les hypothèses de reconnaissance). Pour cette raison il est important dans ce type d'application de réduire la probabilité d'apparition des erreurs le plus tôt possible dans le déroulement du processus.
Pour des systèmes de dialogue en langage naturel, tel que le service 3000, déployé auprès du grand public, une analyse des données nous a montré qu'il existe différents types de messages, comme les énoncés non-parole, les segments de parole *Hors-Domaine* et la parole *Dans-le-Domaine*. Comme nous l'avons expliqué dans la section 4.1, les segments *Hors-Domaine* peuvent générer une interprétation et orienter le système dans une mauvaise direction pour la suite du dialogue. De plus la génération des CNs pour ces énoncés, mais aussi pour les énoncés non-parole, est très coûteuse en termes de temps de calcul car les graphes correspondant sont très bruités et donc très "volumineux". Le traitement de données réelles implique donc de devoir prendre en compte

et de pouvoir traiter certains types d'énoncés qui, du point de vue du système, sont à rejeter. Pour pallier cela nous proposons de rejeter ces énoncés dès la première passe de reconnaissance. Les CNs sont générés sur les énoncés considérés valides. Concernant le processus d'interprétation, il est, dans un premier temps, appliqué sur la meilleure solution du CN pour tous les CNs générés. Au delà de cette approche séquentielle, nous proposons également une stratégie qui permet l'utilisation de la recherche intégrée sur le graphe de mots et sur les CNs, décrite dans la section 3.3.3, dans le processus d'interprétation.
Le chapitre est organisé en trois parties. Dans la section 7.1, nous présentons les changements apportés au protocole expérimental du fait de la prise en compte des spécificités liées au traitement des données réelles. Une évaluation détaillée du comportement de l'algorithme du *pivot topologique* sur les différents types d'énoncés est aussi présentée. Dans la section 7.2, nous présentons une stratégie de décision à plusieurs niveaux (Minescu et Damnati, 2008) qui vise à rejeter les énoncés non-valides (bruits, commentaires) dès la première passe de reconnaissance et à construire les CNs seulement sur les énoncés valides. L'approche séquentielle (analyse en concepts suivie de l'analyse sémantique) du processus d'interprétation est appliquée sur la *consensus hypothesis* pour tous les CNs générés. Dans la section 7.3, nous présentons une stratégie de décision basée sur une approche intégrée [1] du processus d'interprétation. Cette approche introduit un niveau de décision supplémentaire de rejet des énoncés valides mais qui ne sont pas couverts par le modèle sémantique de l'application (aucune interprétation n'est trouvée).

7.1 Cadre expérimental

Dans les évaluations présentées dans les chapitres précédents, nous n'avions pas pris en compte la variabilité des données réelles utilisées. Par exemple, le sous-modèle de langage de détection des commentaire n'a pas été utilisé par la première passe de reconnaissance. Ainsi, les énoncés *Hors-Domaine* n'ont pas été traités différemment des autres énoncés et, par conséquence, les annotations des commentaires réalisées dans les transcriptions manuelles des énonces ont été ignorées. Les mots composant les commentaires ont été comptabilisés comme tous les autres mots de l'application.
Le tableau 7.1 présente deux énoncés *Hors-Domaine* avec la transcription manuelle annotée. Lorsque aucun traitement particulier n'est appliqué aux énoncés *Hors-Domaine* (on ne détecte pas les commentaires) la référence est constituée des mots présents dans la transcription manuelle (les annotations sont ignorées). En revanche, lorsqu'on souhaite appliquer des traitement spécifiques pour les énoncés *Hors-Domaine* (on détecte les commentaires), dans la référence de l'énoncé on remplace la séquence de mots, annotée comme étant un commentaire dans la transcription manuelle, par un symbole unique <*COMMENTAIRE*>. Afin de pouvoir réaliser une évaluation du *WER* qui soit pertinente dans le contexte du système de dialogue, nous utilisons la méthode de normalisation **Methode 4**, décrite dans la section 4.4.1. Les énoncés ne contenant que des

1. La recherche intégrée sur les graphes de mots et sur les CNs, décrite dans la section 3.3.3, est utilisée pour obtenir une interprétation.

<*COMMENTAIRE*>, des *OOV* ou des *SPR* sont considérés comme étant un rejet et la référence est remplacée par le symbole <*REJET*>. Nous utilisons cette normalisation afin de ne pas compter comme une erreur de reconnaissance une substitution du type <*COMMENTAIRE*> → *OOV*.

	Exemple 1	Exemple 2
Énonce	oh merde oh là oh	eh ben France Telecom
Transcription manuelle annotée	[com :] oh merde oh là oh [:com]	[com :] eh ben France Telecom [:com]
Référence sans prise en compte des commentaires	oh merde oh là oh	eh ben France Telecom
Référence avec prise en compte des commentaires	<*COMMENTAIRE*>	<*COMMENTAIRE*>

TABLE 7.1 – *L'influence de la prise en compte des commentaires sur la référence d'un énoncé* Hors-Domaine *au niveau mot.*

Interprétation	Exemple 1	Exemple 2
Énonce	Rejet	Serv(OffresFT)
Référence sans prise en compte des commentaires	Rejet	Serv(OffresFT)
Référence avec prise en compte des commentaires	Rejet	Rejet

TABLE 7.2 – *L'influence de la prise en compte des commentaires sur l'interprétation de la référence d'un énoncé* Hors-Domaine.

Le changement de la référence au niveau mot à aussi des répercussions au niveau interprétation. Ainsi, comme le montre le tableau 7.2, lorsqu'on souhaite appliquer des traitement spécifiques pour les énoncés *Hors-Domaine* (on détecte les commentaires), les énoncés étiquetés comme étant des commentaires deviennent des énoncés à rejeter. Dans le cas contraire, les mots de la référence peuvent produire une interprétation, comme pour l'exemple 2. Sur le corpus **Test_II** on observe ainsi une augmentation du nombre d'énoncés à rejeter de 2% (soit 114 énoncés) du nombre total d'énoncés dans le corpus. Ce pourcentage représente le nombre d'énoncés couverts par l'analyse sémantique (qui donnent lieu à une interprétation) si aucun traitement spécifique n'est appliqué aux énoncés *Hors-Domaine* (la première passe de reconnaissance ne détecte pas les commentaires). L'exemple 2 dans le tableau 7.2 illustre bien cette situation.

7.1.1 Description du corpus de test par catégorie d'énoncés

Dans la partie 4.1, l'analyse des données réelles issues du service 3000 nous a permis de distinguer trois catégories principales d'énoncés : les énonces non-parole (**C1-Non-Parole**), la parole *Hors-Domaine* (les commentaires, **C2-Hors-Domaine**) et la parole *Dans-le-Domaine* (**C3-Dans-le-Domaine**). Le tableau 7.3 présente une description du corpus de **Test_II** par catégorie, en détaillant le nombre d'énoncés par catégorie ainsi que la taille des graphes correspondants exprimée en nombre moyen de transitions par graphe.

Catégorie	# énoncés	# transitions par graphe
C1-Non-Parole	1333	24000
C2-Hors-Domaine	674	23000
C3-Dans-le-Domaine	4494	14000
Total	6501	17000

TABLE 7.3 – *Description du corpus de test **Test_II** par catégorie*

On observe que pour les deux premières catégories, **C1-Non-Parole** et **C2-Hors-Domaine**, le moteur de reconnaissance a tendance à être sur-génératif. Les graphes générés sont, en moyenne, 70% plus grands que pour la catégorie **C3-Dans-le-Domaine**. Non seulement la génération de ces graphes mais aussi la génération des CNs pour ces énoncés est très coûteuse en terme de temps de calcul car les graphes correspondant sont très bruités et redondants et donc très "volumineux". De ce fait, la génération des graphes de mots n'est pas envisageable en situation réelle pour les énoncés appartenant à ces deux catégories.

7.1.2 Analyse de l'algorithme du *pivot topologique* par catégorie d'énoncés

Le tableau 7.4 [2] montre le détail du *WER* sur chaque catégorie pour la *1-best* et la *consensus hypothesis* extraite à partir de l'algorithme du *pivot topologique*. Le *WER* sur chaque catégorie est calculé comme une contribution sur le *WER* de l'ensemble du corpus. La dernière colonne du tableau représente le nombre d'erreurs (substitutions, insertions et omissions) pour chaque catégorie.
Les performances en terme de *WER* sont globalement dégradées avec les CNs issus de l'algorithme du *pivot topologique*, la dégradation étant plus significative sur **C1-Non-Parole** et **C2-Hors-Domaine**. Malgré le fait que ces deux catégories ne représentent que 30% des énoncés, elle contribuent pour près de la moitié aux erreurs observées avec les CNs. De plus, comme le montre le tableau 7.5, la taille des CNs sur ces deux catégories est deux à trois fois supérieure à celle des CNs de la catégorie **C3-Dans-le-Domaine**. En effet, les graphes "volumineux" et hautement ambigus génèrent à leur tour des CNs bruités, pour lesquels la *consensus hypothesis* est source d'insertions et de substitutions. La génération de ces CNs est aussi trop lente est trop couteuse en terme de ressources mémoire. Ces résultats illustrent l'inadéquation des CNs pour rejeter des entrées non-valides.

2. Le *WER* de l'algorithme du *pivot topologique* présenté dans le tableau 6.2 a été évalué en comptant tous les mots des séquences annotées en commentaires, ce qui n'est plus le cas ici, car ces séquence ont été remplacées par le symbole <*COMMENTAIRE*>. Le nombre de mots dans la référence est donc différent ainsi que la valeur du *WER*, qui était de 49%.

3. Le *WER* calculé pour chaque catégorie est une contribution du *WER* total. On divise le nombre d'erreurs de chaque catégorie par le nombre total de mots de la référence sur l'ensemble du corpus.

		WER [3]	Nombre d'erreurs
C1-Non-Parole	1-best	6.8%	914
1333 én.	*pivot topologique*	11.6%	1549
C2-Hors-Domaine	1-best	10.9%	1452
674 én.	*pivot topologique*	12.8%	1704
C3-Dans-le-Domaine	1-best	24.1%	3236
4494 én.	*pivot topologique*	26.9%	3584
Total	1-best	41.8%	5602
6501 én.	*pivot topologique*	51.3%	6837

TABLE 7.4 – WER *de la* 1-best *et de la* consensus hypothesis *issue de l'algorithme du* pivot topologique.

	# classes / mot de la référence	# mots / classe
C1-Non-Parole	65.6	6.3
C2-Hors-Domaine	51.8	7.0
C3-Dans-le-Domaine	23.0	5.6
Total	34.8	5.9

TABLE 7.5 – *Taille des CNs issus de l'algorithme du* pivot topologique.

7.2 Stratégie de décision basée sur une approche séquentielle

Comme le montre l'analyse effectuée dans la section précédente, nous devons envisager un moyen efficace et rapide qui permette au système de rejeter les énoncés non-valides (les énoncés non-parole et les énoncés *Hors-Domaine*) sans avoir besoin de générer les graphes de mots ni par conséquent les CNs correspondants. La *1-best* présente une précision de 90.2% pour la détection des énoncés non-parole. Elle constitue ainsi un moyen fiable et rapide. De plus, le sous-modèle de détection des commentaires lui permet de détecter de manière efficace les énoncés *Hors-Domaine*. En situation réelle nous commencons la génération du graphe de mots en parallèle avec la *1-best*. En conséquence nous avons décidé d'utiliser la *1-best* afin de construire une stratégie qui vise à rejeter les énoncés non-valides (appartenant à **C1-Non-Parole** et **C2-Hors-Domaine**) et à ne générer les CNs que sur les énoncés contenant de la parole *Dans-le-Domaine* de l'application. Les CNs ne sont ainsi générés que si la première passe ne détecte ni un énoncé non-parole ni un commentaire. Dans le cas contraire, la génération du graphe de mots est arrêtée.

7.2.1 Étapes de la stratégie

La figure 7.1 montre les trois étapes de la stratégie. Pour les deux premières étapes, la décision est prise à partir de la *1-best* et c'est seulement lors de la troisième étape que les graphes de mots et les CNs correspondant sont construits. Le déroulement de la stratégie est le suivant :

Étape 1 : *Est-ce que la* 1-best *contient de la parole ?*

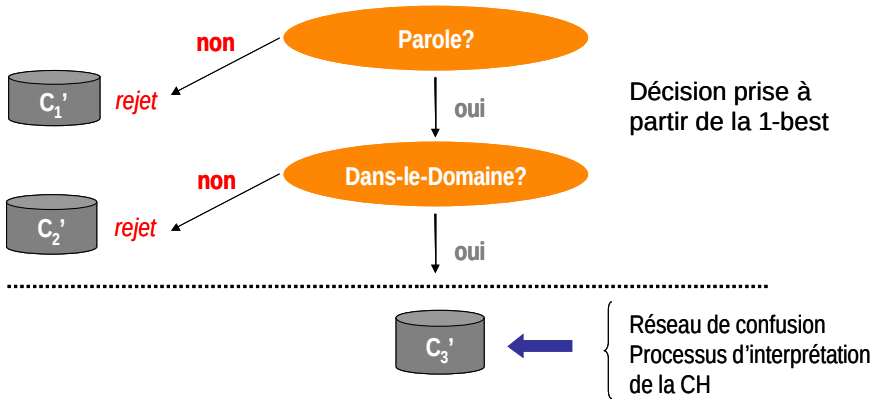

FIGURE 7.1 – Les différentes étapes de la stratégie de construction des CNs

- Si la réponse est non, il s'agit alors d'une détection d'un énoncé non-parole. Étant considéré comme un message non-valide, la stratégie le rejette et le classe dans la catégorie $C_1^{'}$.
- Si la réponse est oui, on passe à l'étape suivante.

Étape 2 : *Est-ce que la* 1-best *contient de la parole* Dans-le-Domaine *?*

- Si la réponse est non, il s'agit alors d'une détection de commentaires. Étant considéré comme un message non-valide, la stratégie le rejette et le classe dans la catégorie $C_2^{'}$.
- Si la réponse est oui, on passe à l'étape suivante.

Étape 3 : Tous les énoncés traités lors de cette étape sont considérés comme faisant partie de la catégorie $C_3^{'}$ ce qui signifie que la *1-best* est supposée contenir de la parole *Dans-le-Domaine*. Pour les énoncés dont la *1-best* ne contient que de la parole *Dans-le-Domaine*, la stratégie génère les réseaux de confusion et extrait la *consensus hypothesis*. Celle-ci est ensuite soumise au processus d'interprétation. [4].

Les catégories $C_1^{'}$, $C_2^{'}$ et $C_3^{'}$ ne sont pas identiques aux catégories **C1-Non-Parole**, **C2-Hors-Domaine** et **C3-Dans-le-Domaine**. Elles ont la même fonction, celle de regrouper les énoncés non-parole, les énoncés *Hors-Domaine* et respectivement les énoncés contenant de la parole *Dans-le-Domaine*, mais elles ne regroupent pas exactement les mêmes énoncés. En effet, les catégories **C1-Non-Parole**, **C2-Hors-Domaine** et **C3-Dans-le-Domaine** ont été calculées en fonction de la référence des énoncés, pour des raisons d'analyse et évaluation du corpus de test, alors que les catégories $C_1^{'}$, $C_2^{'}$ et $C_3^{'}$ sont calculées en fonction de la *1-best*.

4. Un traitement particulier est réservé aux énoncés dont la *1-best* contient des commentaires et de la parole *Dans-le-Domaine*. Comme nous l'avons expliqué dans la section 4.3, le sous-modèle de détection des commentaires n'est pas utilisé dans la génération des graphes de mots car il est difficile d'estimer la probabilité *a posteriori* sur les segments *Hors-Domaine* et donc de générer les CNs correspondant. La *1-best* est alors utilisée comme hypothèse de reconnaissance pour ce type d'énoncé et est ensuite soumise au processus d'interprétation.

7.2.2 Evaluation

Un premier jeu d'expériences présenté dans le tableau 7.6 vise à montrer les performances de la stratégie en terme de *WER*. On observe un gain absolu de 7% en terme de *WER* sur l'algorithme du *pivot topologique* grâce à l'utilisation de la stratégie. Toutefois, le *WER* de la *1-best* reste légèrement meilleur que celui obtenu avec la stratégie, la différence provenant essentiellement des énoncés appartenant à **C3-Dans-le-Domaine**.

WER	Total	**C1-Non-Parole**	**C2-Hors-Domaine**	**C3-Dans-le-Domaine**
1-best	41.8%	6.8%	10.9%	24.1%
pivot topologique	51.3%	11.6%	12.8%	26.9%
Stratégie + Alg. du *pivot topologique*	44.3%	6.6%	10.8%	26.9%

TABLE 7.6 – *Evaluation de la stratégie en terme de* WER.

Le tableau 7.7 montre le détail du *WER*, obtenu sur le trois catégories, de la stratégie utilisant l'algorithme du *pivot topologique*, l'algorithme *multi-niveaux* ou *multi-niveaux sans vides* pour générer les CNs sur les enregistrements du $C_3^{'}$. Le tableau 7.8 montre le détail d'*IER* sur les trois catégories. Nous rappelons que les taux de fausse alarme, de substitutions et de faux rejets sont calculés par rapport au nombre d'énoncés interprétables.

WER	Total	**C1-Non-Parole**	**C2-Hors-Domaine**	**C3-Dans-le-Domaine**
1-best	41.8%	6.8%	10.9%	24.1%
Stratégie + Alg. du *pivot topologique*	44.3%	6.6%	10.8%	26.9%
Stratégie + Alg. *multi-niveaux*	42.3%	6.3%	10.2%	25.8%
Stratégie + Alg. *multi-niveaux sans vides*	38.6%	5.3%	8.7%	24.6%

TABLE 7.7 – *Les performances en terme de* WER *de la stratégie en fonction de l'algorithme de génération des CNs utilisé à l'étape 3.*

L'utilisation de l'algorithme *multi-niveaux* à la place de l'algorithme du *pivot topologique* dans l'Etape 3 de la stratégie permet d'obtenir une diminution du *WER* de 2%, en valeur absolue. L'utilisation de l'algorithme *multi-niveaux sans vides* permet d'améliorer encore plus les performances de la stratégie avec une réduction absolue de 3.2% du *WER* par rapport à la *1-best*. En ne conservant que les mots vides présents initialement dans le *pivot*, on évite d'augmenter de façon trop importante le nombre d'insertions des mots vides. Ceci à d'autant plus d'effet sur les catégories **C1-Non-Parole** et **C2-Hors-Domaine** qui gèrent les graphes les plus bruités. L'amélioration des performances par rapport à la *1-best* sur les catégories **C1-Non-Parole** et **C2-Hors-Domaine** est due aux énoncés, appartenant à ces deux catégories, classés à tort par la *1-best* dans la catégorie $C_3^{'}$ et pour lesquels les CNs sont générés (30% pour **C1-Non-Parole** et 80% pour **C2-Hors-Domaine**, cf. tableau 7.9). En effet, sur ces énoncés, on observe une baisse relative

		C1-Non-Parole	C2-Hors-Domaine	C3-Dans-le-Domaine		
	IER	FA	FA	FA	Sub	FR
1-best	24.6%	4.6%	8.8%	4.1%	5.5%	1.6%
Stratégie + Alg. du *pivot topologique*	22.0%	2.4%	6.2%	3.2%	6.0%	4.2%
Stratégie + Alg *multi-niveaux*	20.1%	2.2%	6.3%	3.1%	5.1%	3.4%
Stratégie + Alg. *multi-niveaux sans vides*	19.9%	2.1%	6.2%	3.1%	5.1%	3.4%

TABLE 7.8 – *Les performances en terme d'*IER *de la stratégie en fonction de l'algorithme de génération des CNs utilisé à l'étape 3.*

de 20% des insertions et de 15% des substitutions dans la *consensus hypothesis* par rapport à la *1-best*. La légère dégradation du *WER* sur **C3-Dans-le-Domaine** est due à une augmentation relative de 25% du nombre d'omissions dans la *consensus hypothesis*. Sur cette catégorie, la baisse relative du nombre d'insertions et de substitutions et de 12% en moyenne. La diminution du nombre d'insertions et de substitutions sur les trois catégories explique la diminution du taux de fausse alarme. L'augmentation du taux de faux rejet est due à l'augmentation du nombre d'omissions sur **C3-Dans-le-Domaine**.

Total 6501 én.	C_1' 994 én.	C_2' 138 én.	C_3' 5369 én.	Rappel
C1-Non-Parole 1333 én.	897	47	389	67.3%
C2-Hors-Domaine 674 én.	76	54	544	8.0%
C3-Dans-le-Domaine 4494 én.	21	37	4436	98.7%
Précision	90.2%	39.1%	82.6%	

TABLE 7.9 – *Précision et Rappel pour la classification des énonces par catégorie*

Le tableau 7.9 montre le détail de la classification des énoncés dans chaque catégorie. Pour chacune des trois catégories nous calculons le rappel et la précision en utilisant les formules suivante :

$$\text{Rappel} = \frac{\text{\# énoncés bien détectés}}{\text{\# total d'énoncés dans une classe}} \tag{7.1}$$

$$\text{Précision} = \frac{\text{\# énoncés bien détectés}}{\text{\# total d'énoncés détectés comme appartenant à une classe}} \tag{7.2}$$

Suite à l'utilisation de la stratégie, 17% du nombre total d'énoncés dans le corpus de test ont été rejetés par la *1-best* dans les deux premières étapes de la stratégie, par rapport à 30% des énoncés à rejeter. La précision du rejet de la stratégie, due à l'utilisation de la *1-best*, sur les deux premières catégories est de 95%. Sur la catégorie **C3-Dans-le-Domaine** on obtient un rappel de 98.7%, ce qui signifie que la quasi totalité des énoncés

Dans-le-Domaine ont été correctement traités par la stratégie, en construisant les CNs correspondants.

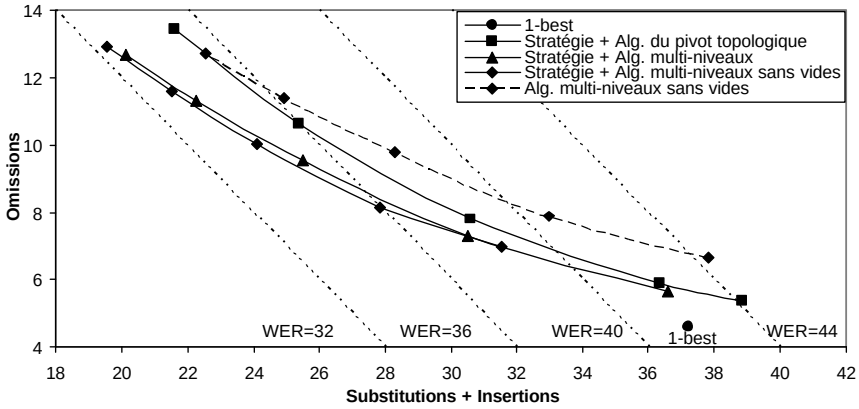

FIGURE 7.2 – Evaluation du *WER* de la stratégie en fonction de la variation du seuil sur la mesure de confiance pour la *consensus hypothesis*

Comme nous l'avons mentionné, dans la troisième étape de la stratégie, la *consensus hypothesis* peut être filtrée en fonction d'un seuil sur la mesure de confiance avant d'appliquer le processus d'interprétation. La figure 7.2 montre les performances de la stratégie en fonction du seuil sur la mesure de confiance qui permet de filtrer les mots de la *consensus hypothesis* extraite à partir des CNs générés à l'étape trois. Les courbes ont été construites en variant le seuil de 0 à 0.4, chaque point sur la courbe correspond à une valeur en partant de la droite vers la gauche. Dans cette figure, nous avons représenté également les performances obtenues en construisant les CNs à l'aide de l'algorithme *multi-niveaux sans vides* sur l'ensemble du corpus de test [5]. On observe des performances équivalentes en terme de *WER* pour la stratégie suite à l'utilisation de l'algorithme *multi-niveaux* ou de l'algorithme *multi-niveaux sans vides* à l'étape trois et l'utilisation d'un seuil sur la mesure de confiance afin de filtrer la *consensus hypothesis* correspondante. Par ailleurs, on observe que la stratégie permet une amélioration importante des performances par rapport à l'utilisation de l'algorithme *multi-niveaux sans vides* sur l'ensemble de corpus. Ainsi, pour un taux d'omission constant on observe une baisse de près de 4% du nombre d'insertions et de substitutions.
Un deuxième jeu d'expériences vise à montrer les performances de la stratégie en termes d'*IER* et le déplacement du point de fonctionnement du système en fonction du seuil sur la mesure de confiance. Les courbes de la figure 7.3 ont été construites de la même manière que pour la figure précédente, en variant le seuil sur la mesure de confiance. Le filtrage des mots de la *consensus hypothesis* en fonction du seuil sur la mesure de confiance est réalisé avant le processus d'interprétation. On observe une amélioration de l'*IER* en utilisant la stratégie avec l'algorithme du *pivot topologique*. Le point de fonctionnement est déplacé vers la gauche dû principalement à la réduction importante du nombre de FA. L'utilisation des algorithmes *multi-niveaux* et *multi-niveaux sans vides*

5. Cette courbe est différente de celle présentée dans la figure 6.2 car la référence est différente du fait de la prise en compte de la détection des commentaires.

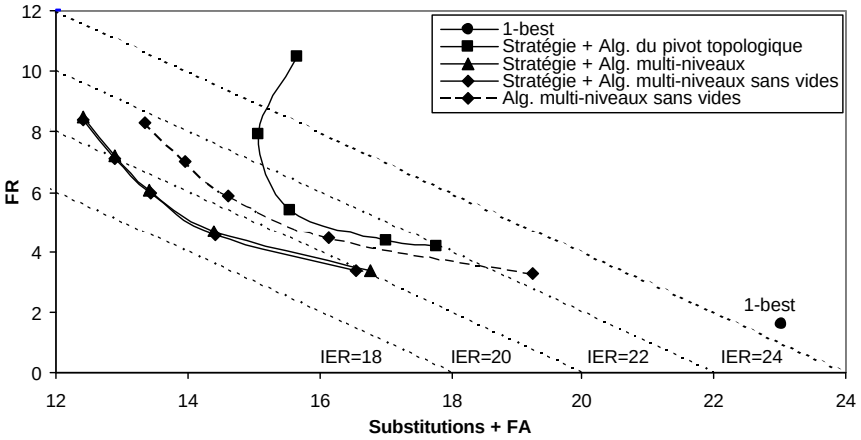

FIGURE 7.3 – La variation du point de fonctionnement de la stratégie en fonction du seuil sur la mesure du confiance

permet d'améliorer d'avantage l'*IER*. Le point de fonctionnement est déplacé encore plus vers la gauche en réduisant d'avantage le nombre de FA et de substitutions (cf. tableau 7.8). On observe également que la stratégie utilisant l'algorithme *multi-niveaux sans vides* dans la troisième étape permet une réduction importante de l'*IER* par rapport à l'utilisation de l'algorithme *multi-niveaux sans vides* sur l'ensemble du corpus. Ainsi, pour un taux de FR constant, on observe une réduction moyenne de 2% du nombre de substitutions et FA.

7.3 Stratégie de décision basée sur une approche intégrée

La stratégie présentée dans la section précédente permet non seulement de ne construire des CNs que sur les énoncés valides mais aussi d'améliorer les performances du système au niveau interprétation. Pour ce faire elle utilise la meilleure solution de la première passe de reconnaissance, capable de détecter et rejeter les énoncés non-parole et la parole *Hors-Domaine*, et ensuite la meilleure solution des CNs qui fournit une interprétation sur les énoncés valides. Toutefois, le module de compréhension travaille directement sur une séquence de mots et ne bénéficie pas de la richesse de l'espace de recherche correspondant (CN ou graphe de mots). De plus, on a montré que l'oracle calculé sur ces structures améliore de manière importante les performances en terme de *WER* (voir la section 6.3.2). On peut donc envisager qu'en retardant la prise de décision (ne pas calculer la meilleure séquence de mots avant le calcul de l'interprétation) en calculant une interprétation sur l'ensemble de l'espace des solutions on puisse améliorer les performances du système de dialogue en termes d'*IER*.
Le projet européen LUNA (voir 3.3.3 pour la présentation du projet) propose une architecture pour un système de dialogue qui se base sur la recherche intégrée pour calculer une hypothèse d'interprétation fournie ensuite au gestionnaire du dialogue, avec d'autres informations supplémentaires comme le contexte du dialogue, pour détermi-

ner la meilleure réponse du système à la requête formulée par l'utilisateur. Comme nous l'avons présentée dans la section 3.3.3, la recherche intégrée utilise des graphes de mots (ou autres structures similaires comme les CNs) afin d'obtenir directement une interprétation sans passer par le traitement d'une séquence de mots. La prise de décision de la meilleure hypothèse ne se fait plus au niveau mot mais est retardée au niveau interprétation. Pour ce faire, le graphe de mots est transformé en un graphe de concepts et ensuite une interprétation est obtenue. Les différentes transformations sont effectuées à travers des compositions avec des FSM représentant les règles d'allumage de concepts et les règles d'interprétation.
Dans cette partie nous utilisons la recherche intégrée sur les graphes de mots et sur les CNs construits pour les énoncés valides afin d'améliorer les performances du système. La stratégie présentée dans cette section (Minescu et al., 2007) rejette les énoncés non-valides de la même manière que la stratégie présentée dans la section 7.2 et n'utilise la recherche intégrée que sur les énoncés valides. Nous montrons également qu'une étape supplémentaire est nécessaire pour le rejet des énoncés valides qui ne sont pas couverts par l'analyse sémantique avant de réaliser une recherche intégrée sur les graphes de mots ou sur les CNs correspondants aux énoncés restants.

7.3.1 Analyse du processus d'interprétation

Une analyse plus approfondie de la catégorie **C3-Dans-le-Domaine** montre qu'une partie des énoncés en faisant partie sont des énoncés à rejeter car ils ne sont pas couverts par l'analyse sémantique. En effet, malgré le fait que l'énoncé ne contienne que de la parole *Dans-le-Domaine*, le système ne trouve pas une interprétation valide pour la séquence de mots prononcée. A la différence de la stratégie présentée au 7.2, où la *consensus hypothesis* extraite des CNs était utilisée pour obtenir une interprétation sur les énoncés valides, l'utilisation de la recherche intégrée sur des graphes de mots (ou des CNs) pour trouver un interprétation peut augmenter considérablement le nombre de FA sur les énoncés valides mais non couverts par l'analyse sémantique. Il convient donc de développer une stratégie capable de rejeter tous les énoncés non-valides, que ce soit des détections non-parole, de la parole *Hors-Domaine* mais aussi les énoncés valides non couverts par l'analyse sémantique.

Catégorie	# énoncés	# transitions par graphe
C1-Non-Parole	1333	24000
C2-Hors-Domaine	674	23000
C3-Dans-le-Domaine-sans-interprétation	355	17000
C3-Dans-le-Domaine-avec-interprétation	4139	13000
Total	6501	17000

TABLE *7.10* – *Description du corpus de test **Test_II** par catégorie*

Pour développer cette stratégie nous faisons la différence entre les énoncés de la catégorie **C3-Dans-le-Domaine** avec et sans interprétation afin de les traiter différemment. Le tableau 7.10 présente la nouvelle découpe par catégorie du corpus **Test_II**.

Les deux premières catégories restent inchangées, alors que la catégorie **C3-Dans-le-Domaine** est divisée en deux : la catégorie **C3-Dans-le-Domaine-sans-interprétation** qui contient les énoncés *Dans-le-Domaine* pour lesquels le processus d'interprétation ne produit aucune interprétation (ils ne sont pas couvert par l'analyse sémantique), qui sont donc à rejeter, et la catégorie **C3-Dans-le-Domaine-avec-interprétation** qui contient les énoncés *Dans-le-Domaine* ayant une interprétation valide. On observe que la taille des graphes de mots des énoncés appartenant à **C3-Dans-le-Domaine-sans-interprétation** est 25% plus grande que celle des énoncés valides couverts par l'analyse sémantique.
Les résultats en terme d'*IER* de la stratégie de construction des CNs, présentés dans le tableau 7.8, montrent que l'utilisation de l'algorithme *multi-niveaux sans vides* pour obtenir une interprétation sur la catégorie $C_3^{'}$ donne de meilleurs résultats que la *1-best* ou l'algorithme du *pivot topologique*, notamment grâce à la réduction du nombre de FA. Cette réduction du nombre de FA correspond en effet à un meilleur rejet par rapport à la *1-best* des énoncés sans interprétation valide appartenant à la catégorie **C3-Dans-le-Domaine** (les énoncés non couverts par l'analyse sémantique). De plus, l'utilisation des mesures de confiance pour filtrer la *consensus hypothesis* avant le processus d'interprétation permet de diminuer davantage le nombre de FA, comme montré dans la figure 7.3. Les performances en terme d'*IER* de l'algorithme *multi-niveaux* sont quasiment équivalentes mais les CNs générés sont plus grands et leur génération nécessite un temps de calcul plus élevé. La *consensus hypothesis* des CNs générés à l'aide de l'algorithme *multi-niveaux sans vides* constitue donc le meilleur choix pour rejeter les énoncés appartenant à la catégorie **C3-Dans-le-Domaine-sans-interprétation**.

7.3.2 Étapes de la stratégie

Nous avons identifié trois catégories d'énoncés à rejeter avant de pouvoir utiliser la recherche intégrée sur les énoncés restant. La stratégie utilise successivement différentes solutions pour rejeter les énoncés avant de calculer une interprétation sur les graphes de mots ou sur les CNs.
La figure 7.4 montre les quatre étapes de la stratégie et les solutions utilisées pour la prise de décision. On observe que le rejet dans les deux premières étapes est réalisé à l'aide de la *1-best* de la même manière que pour la stratégie décrite au 7.2. Le déroulement de la stratégie pour les deux dernières étapes est le suivant :

Étape 3 : Pour les énoncés dont la *1-best* ne contient que de la parole *Dans-le-Domaine*, la stratégie génère le graphe de mots et le CN correspondant utilisant l'algorithme *multi-niveaux sans vides* et extrait la *consensus hypothesis*. Celle-ci est ensuite soumise au processus d'interprétation. [6].

6. Un traitement particulier est réservé aux énoncés dont la *1-best* contient des commentaires et de la parole *Dans-le-Domaine*. Comme nous l'avons expliqué dans la section 4.3, le sous-modèle de détection des commentaires n'est pas utilisé dans la génération des graphes de mots car il est difficile d'estimer la probabilité *a posteriori* sur les segments *Hors-Domaine* et donc de générer les CNs correspondant. La *1-best* est alors utilisée comme hypothèse de reconnaissance pour ce type d'énoncé et est ensuite soumise au processus d'interprétation.

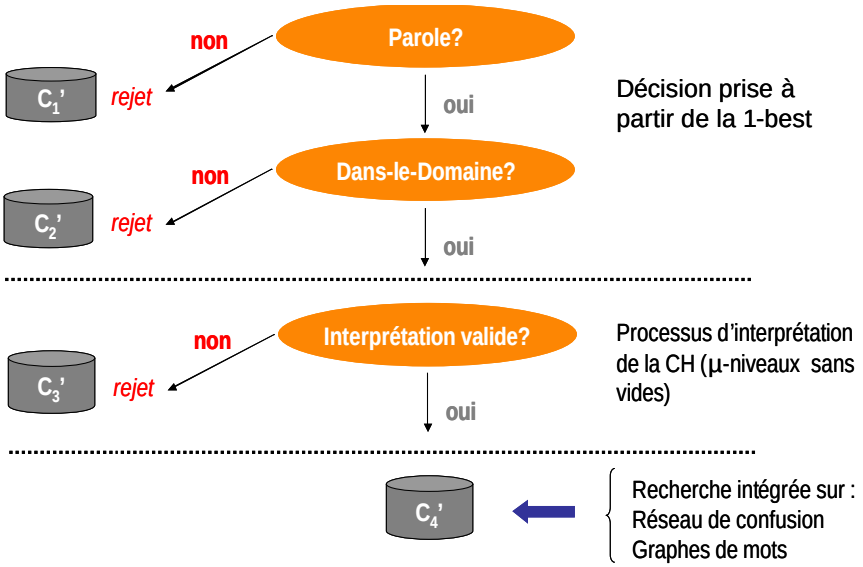

FIGURE 7.4 – Description des différentes étapes de la stratégie

Est-ce que le processus d'interprétation de la consensus hypothesis *a fourni une interprétation valide ?*

- Si non, l'énoncé est rejeté. Étant considéré comme un message *Dans-le-Domaine* sans interprétation valide, la stratégie le rejette et le classe dans la catégorie $C_3^{'}$.
- Si oui, la stratégie passe à l'étape suivante.

Étape 4 : Tous les énoncés traité lors de cette étape sont considérés comme faisant partie de la catégorie $C_4^{'}$. Il s'agit des énoncés *Dans-le-Domaine* ayant une interprétation valide. La stratégie calcule alors, à travers la recherche intégrée, la meilleure solution d'interprétation. Le calcul peut se faire sur le graphe de mots ou sur le CN.

Comme nous pouvons l'observer, pour les deux premières étapes, le rejet est effectué au niveau mot à l'aide de la *1-best*, alors que pour l'étape 3, le rejet est effectué au niveau interprétation à l'aide de la *consensus hypothesis* extraite des CNs générés avec l'algorithme *multi-niveaux sans vides*. Les graphes de mots et les CNs sont générés pour tous les énoncés traités à l'étape 3.

7.3.3 Evaluation

Avant d'évaluer les performances de la stratégie proposée, un premier jeu d'expériences vise à montrer les performances de chacun des algorithmes appliqué sur l'ensemble du corpus sans aucune stratégie. Le tableau 7.11 compare les solutions de six algorithmes en détaillant les erreurs au niveau interprétation sur chacune des quatre catégories. Étant donné que les trois premières catégories ne contiennent que des énoncés à rejeter, on ne peut avoir que des FA. En revanche, la dernière catégorie ne contient que des énoncés ayant une interprétation valide et, en conséquence, on ne peut avoir

que des Sub ou FR. La première colonne correspond à l'utilisation de la meilleure solution de la première passe de reconnaissance. Graphe *1-best* est l'interprétation obtenue sur le meilleur chemin du graphe de mots calculée au sens des probabilités *a posteriori* des transitions. Les deux colonnes suivantes correspondent à l'interprétation obtenue sur la *consensus hypothesis* des deux algorithmes. *Décodage Graphes* et *Décodage CN* correspondent aux interprétations obtenues en effectuant une recherche intégrée respectivement sur les graphes de mots et les CNs. Nous avons utilisé les CNs générés avec l'algorithme *multi-niveaux sans vides*.

	1-best	Graphe *1-best*	*pivot topologique*	*multi-niveaux sans vides*	Décodage CN	Décodage Graphes
FA sur **C1-Non-Parole**	4.6%	6.5%	4.7%	**4.5%**	20.3%	23.0%
FA sur **C2-Hors-Domaine**	8.8%	8.1%	6.6%	**6.5%**	14.5%	13.7%
FA sur **C3-Dans-le-Domaine sans-interp**	4.1%	3.0%	3.2%	**3.1%**	7.2%	6.5%
Substitutions sur **C3-Dans-le-Domaine avec-interp**	5.5%	6.1%	6.0%	5.0%	7.9%	5.9%
FR sur **C3-Dans-le-Domaine avec-interp**	1.6%	2.5%	4.0%	3.3%	0.3%	0.4%
Total *IER*	24.6%	26.3%	24.6%	22.5%	50.2%	49.5%

TABLE *7.11* – *Détail des erreurs sur l'ensemble de corpus pour chaque algorithme.*

L'utilisation d'un espace d'hypothèses plus grand, comme les graphes de mots, améliore les performances (en termes de FR et substitutions) sur **C3-Dans-le-Domaine-avec-interprétation** pour *Décodage Graphes* (de 7.1% à 6.3%) par rapport à la *1-best*, mais augmente en même temps le nombre de FA (jusqu'à 23% sur **C1-Non-Parole**). De plus, le temps de calcul pour *Décodage Graphes* est nettement plus élevé que pour la *1-best*. En revanche, la *consensus hypothesis* obtenue avec l'algorithme *multi-niveaux sans vides* donne les meilleurs résultats en ce qui concerne la réduction de nombre de FA sur les trois premières catégories. On observe également que les performances sur **C2-Hors-Domaine** et **C3-Dans-le-Domaine-sans-interprétation** de la Graphe *1-best* sont meilleures que celles de la première passe de reconnaissance mais le taux de substitution et FR sur **C3-Dans-le-Domaine-avec-interprétation** est plus élevé. De plus, l'obtention de Graphe *1-best* est beaucoup plus coûteuse en temps de calcul , surtout sur les graphes bruités.
Comme nous l'avons mentionné, à l'étape trois de la stratégie, le rejet des énoncés est réalisé au niveau interprétation, à l'aide de la *consensus hypothesis*, obtenue avec l'algorithme *multi-niveaux sans vides*, qui peut être ou non filtrée en fonction d'un seuil sur la mesure de confiance avant le processus d'interprétation. Nous avons choisi de filtrer la *consensus hypothesis* avant le processus d'interprétation avec une valeur du seuil égale à 0.1. En effet, les performances en termes d'*IER* de la stratégie basée sur une approche

séquentielle utilisant l'algorithme *multi-niveaux sans vides* à l'étape 3, présentées dans la figure 7.3, montrent une réduction importante du taux de FA (1.7%) pour une valeur du seuil de 0.1. Cette réduction est réalisée sur la catégorie **C3-Dans-le-Domaine** du fait du filtrage de la *consensus hypothesis* avant le processus d'interprétation. Une valeur plus grande du seuil permet certes de diminuer davantage le taux de FA, mais avec une augmentation importante du taux de FR.
Le tableau 7.12 compare les performances de la stratégie utilisant, à l'étape quatre, la recherche intégrée sur les CNs, que nous appelons *Strat1(CN)*, et celles de la stratégie utilisant la recherche intégrée sur les graphes de mots, que nous appelons *Strat2(Graphe)*, aux performances de la meilleure solution de la première passe de reconnaissance, la *1-best*. La *1-best* est utilisée sur l'ensemble du corpus tant pour le rejet des énoncés que pour la recherche de la meilleure interprétation.

total	*1-best*	Strat 1 (CN)	Strat 2 (Graphe)
FA	17.5%	9.6%	9.6%
Sub	5.5%	5.8%	4.2%
FR	1.6%	4.6%	4.6%
IER	24.6%	20%	18.4%

TABLE 7.12 – *Evaluation des deux stratégies proposées*

Une amélioration importante des performances est obtenue grâce à l'utilisation de *Strat1(CN)*. La stratégie permet de réduire le taux de FA de 7.9% en valeur absolue avec une augmentation de seulement 3% du taux de FR. L'utilisation d'un espace d'hypothèses plus grand dans *Strat2(Graphe)* pour les énoncés de la catégorie $C_4^{'}$ permet de diminuer davantage le taux de substitution. L'augmentation du taux de FR dans les deux cas est due aux énoncés de la catégorie **C3-Dans-le-Domaine-avec-interprétation** mal classifiés par la stratégie dans les catégories $C_1^{'}$, $C_2^{'}$ et $C_3^{'}$ (cf. au tableau 7.13).

Total 6501 én.	$C_1^{'}$ 994 én.	$C_2^{'}$ 138 én.	$C_3^{'}$ 1021 én.	$C_4^{'}$ 4348 én.	Rappel
C1-Non-Parole 1333 én.	897	47	325	64	67.3%
C2-Hors-Domaine 674 én.	76	54	327	217	8.0%
C3-Dans-le-Domaine-sans-interprétation 355 én.	14	13	208	120	58.6%
C3-Dans-le-Domaine-avec-interprétation 4139 én.	7	24	161	3947	95.4%
Précision	90.2%	39.1%	20.4%	90.8%	

TABLE 7.13 – *Précision et Rappel pour la classification des énonces par catégorie*

Le tableau 7.13 monte le détail de la classification des énoncés dans chaque catégorie. Suite à l'utilisation de la stratégie, 33% du nombre total d'énoncés ont été rejetés dans les trois premières étapes, par rapport à 36% des énoncés à rejeter. Sur les deux premières catégories, la précision du rejet due à l'utilisation de la *1-best* est de 95%.

Sur la catégorie $C_3^{'}$, l'utilisation de la *consensus hypothesis* permet d'obtenir une précision de rejet de 20.4%. On observe également que 32.5% des énoncés non-valides qui auraient dû être rejetés par la *1-best* sont rejetés par la *consensus hypothesis* car ils sont classifiés dans la catégorie $C_3^{'}$. Cette classification explique en partie la précision peu élevée sur cette catégorie malgré un rappel de 58,6% sur **C3-Dans-le-Domaine-sans-interprétation**. Sur l'ensemble de la stratégie, la précision du rejet est de 91%. Le rappel sur la catégorie **C3-Dans-le-Domaine-avec-interprétation** est de 95.4% ce qui signifie que la majorité des énoncés *Dans-le-Domaine* et couverts par l'analyse sémantique on été correctement traité par la stratégie en réalisant une recherche intégrée afin de trouver l'interprétation.

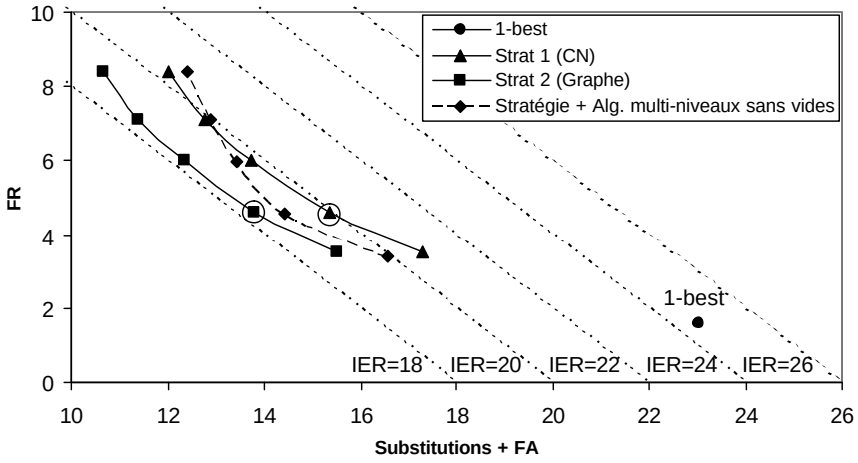

FIGURE 7.5 – Evaluation de la stratégie en fonction de rejet des énoncés à l'étape 3

Dans le tableau 7.12, les deux stratégies présentées utilisent, à l'étape trois, la *consensus hypothesis* obtenue avec l'algorithme *multi-niveaux sans vides* et un filtrage avec une valeur de seuil de 0.1 avant le processus d'interprétation . La figure 7.5 présente la variation du point de fonctionnement des deux stratégies en fonction du seuil sur la mesure de confiance qui est utilisé pour filtrer la *consensus hypothesis* à l'étape 3. Les courbes ont été construites en variant la valeur du seuil de 0 à 0.4, avec le point le plus à droite qui correspond à l'utilisation de la *consensus hypothesis* sans filtrage. Les deux points entourés sur les courbes correspondent respectivement au détail des erreurs dans le tableau 7.12. On observe que le choix d'une valeur du seuil égale à 0.1 est un bon compromis entre l'amélioration du nombre de FA par rapport à la dégradation du nombre de FR. La troisième courbe correspond à la stratégie présentée dans la section 7.2 utilisant l'algorithme *multi-niveaux sans vides* pour construire les CNs dans l'étape trois. On observe ainsi que pour *Strat2(Graphe)*, l'utilisation d'un espace de recherche plus riche (les graphes de mots) combiné à une stratégie de rejet adaptée, permet d'obtenir de meilleures performances par rapport à l'utilisation, sur l'ensemble des énoncés valides, de la *consensus hypothesis* extraite des CNs construits avec l'algorithme *multi-niveaux*. On observe dans la figure 7.5, que *Strat1(CN)* ne permet pas d'améliorer les performances de la stratégie proposée dans la section 7.2 ce qui montre que les CNs ne sont pas adaptés pour réaliser une recherche intégrée de la solution d'interprétation. En re-

vanche, les CNs s'avèrent un moyen très efficace de sélectionner les énoncés pertinents à travers l'utilisation de la *consensus hypothesis*.

7.4 Conclusions

Ce chapitre a introduit, dans un premier temps, un nouveau cadre expérimental défini par la détection des commentaires qui implique une prise en compte, dans la référence, des annotations indiquant les séquences de mots qui représentent des commentaires. Ces séquences sont remplacées par l'étiquette <*COMMENTAIRE*>. Nous avons ensuite réalisée une analyse détaillée des performances de la *1-best* et de l'algorithme du *pivot topologique* sur les trois types d'énoncés (détections non-parole, parole *Hors-Domaine* et parole *Dans-le-Domaine*) qui ont été identifiés dans le corpus de test. Cette analyse nous a permis de montrer que l'utilisation des CNs n'est pas adaptée pour la détection et le rejet des énoncés non-valides. De plus, la génération des CNs sur ces énoncés est très coûteuse en temps de calcul ce qui n'est pas compatible avec les contraintes temps-réel d'un système du dialogue. Ceci nous a amené à proposer une stratégie qui utilise la *1-best* de la première passe de reconnaissance pour rejeter les énoncés non-valides dès la première passe, et ceci avec une précision de 95%. Les CNs sont ensuite générés seulement pour les énoncés valides. Le processus d'interprétation est effectué sur la *consensus hypothesis* issue des CNs, qui peut, ou non, être filtrée en fonction d'un seuil sur la mesure de confiance. Cette stratégie permet d'améliorer le *WER* et l'*IER* par rapport à la *1-best*.
Nous avons montré par ailleurs que le recours à la recherche intégrée pour rechercher la meilleure interprétation sur le graphe de mots ou le CN ne s'avère pas pertinente dans tous les cas. En effet, il subsiste une partie des énoncés valides qui ne sont pas couverts par l'analyse sémantique et qui ne peuvent pas être interprétés par le système. L'utilisation de la recherche intégrée sur les graphes de mots ou sur les CNs correspondant peut donner lieu à des erreurs d'interprétation (notamment des fausses alarmes). Nous avons proposé une stratégie adaptée qui permet de rejeter également ce type d'énoncé en plus des énoncés non-valides grâce à un enchaînement d'étapes de rejet utilisant différentes solutions. Ainsi, pour le rejet des énoncés non-valides la stratégie utilise la meilleure solution de la première passe de reconnaissance. Ensuite, les CNs sont générés et les énoncés pour lesquels la *consensus hypothesis* ne peut pas être interprétée sont rejetés. La recherche intégrée est ensuite effectuée pour les énoncés restants. Les CNs s'avèrent un moyen très efficace de sélectionner les énoncés pertinents en rejetant correctement 58% des énoncés valides non couverts par l'analyse sémantique. Les résultats montrent une amélioration des performances au niveau interprétation par rapport à la stratégie décrite précédemment qui n'utilise que la *consensus hypothesis* pour obtenir une interprétation sur les énoncés valides.

Conclusions et perspectives

Le travail de thèse présenté dans ce document s'inscrit dans le cadre de la compréhension de la parole continue.
L'application utilisée est un système de dialogue homme-machine en langage naturel déployé par France Telecom. Les expériences ont été menées sur des corpus de données réelles collectés à partir de l'application déployée.
Dans ce contexte, le travail présenté vise à améliorer les performances du système du point de vue utilisateur à travers l'utilisation des réseaux de confusion. Pour cela deux directions ont été suivies. La première avait pour objectif la réalisation d'une meilleure intégration des caractéristiques du système de dialogue en langage naturel dans la génération des CNs. La deuxième direction visait à trouver des stratégies permettant une meilleure prise en compte, dans le processus de reconnaissance et d'interprétation des énoncés, des spécificités liées à l'utilisation de données réelles.

Corpus de données réelles

Une analyse des données réelles utilisées nous a permis de déterminer l'existence de trois catégories d'énoncés utilisateur : les énoncés de type détection non-parole (bruits), les commentaires et les énoncés dans le domaine de l'application. Du point de vue fonctionnel du système, les énoncés des deux premières catégories sont considérés comme des énoncés à rejeter. Ils ne sont pas couverts par l'analyse sémantique du module de compréhension et leur interprétation peut diriger le dialogue dans de mauvaises directions. Nous proposons des techniques de rejet de ce type d'énoncés à travers la mise en place de stratégies de décision.
Un autre problème qui se pose est lié à la détection des énoncés à rejeter. Pour cela, le SRAP utilise, dans la première passe de reconnaissance, un sous-modèle de rejet des bruits (pour la première catégorie) et un sous-modèle de langage pour la détection des commentaires. Toutefois, le système peut aussi produire une hypothèse vide pour un énoncé non-parole ou bien un enchaînement des mots hors-vocabulaire pour un commentaire. D'un point du vue du SRAP, ce dernier exemple est considère comme une erreur de reconnaissance. Néanmoins, dans le contexte plus large du système de dialogue, un commentaire peut être considéré comme un énoncé à rejeter et une hypothèse de reconnaissance ne contenant que des *OOV* comme un rejet de l'énoncé. Nous avons

donc proposé des méthodes de normalisation des données qui permettent une évaluation pertinente du *WER* dans le contexte du système de dialogue.

Probabilité *a posteriori*

La génération des CNs nécessite une étape préalable d'estimation des probabilités *a posteriori* des transitions dans les graphes de mots à l'aide de l'algorithme *Forward-Backward*. L'implémentation de cet algorithme pose des problèmes numériques d'*underflow* qui sont généralement résolus par une méthode approximative de calcul des probabilités *a posteriori* qui force un élagage de certaines transitions pour des raisons numériques. Nous avons proposé une méthode de calcul exact de la probabilité *a posteriori* ne nécessitant aucun élagage des transitions. Cette méthode introduit des coefficients de normalisation par transition dans le calcul des variables de l'algorithme *Forward-Backward*.

Génération des CNs

La génération des CNs, telle qu'elle est décrite dans la littérature, est basée sur un critère de minimisation du *WER*. Or, dans le cadre d'une application de dialogue on cherche à minimiser l'*IER*. Dans ce contexte, nous avons proposé un algorithme de génération des CNs dont le but est de rendre plus robuste la reconnaissance des mots porteurs de sens. Le nouvel algorithme traite en priorité les mots porteurs de sens (les mots non-vides) grâce à un parcours modifié du graphe de mots. Nous avons également proposé une nouvelle heuristique pour remplacer le calcul de la relation d'ordre dans le processus de génération. Ces modifications permettent d'améliorer de façon significative l'*IER* de la *consensus hypothesis* (par rapport à l'algorithme *pivot topologique* et à la *1-best*), mais aussi de créer des réseaux de confusion plus compacts. Nous avons également observé une amélioration significative des performances de la *consensus hypothesis* en termes de *WER* (par rapport au *pivot topologique*) malgré la focalisation sur un sous-ensemble des mots dans la génération des CNs.

La probabilité *a posteriori* des mots dans un CN est une très bonne mesure de confiance. Nous avons montré que le filtrage de la *consensus hypothesis* en fonction d'un seuil sur cette mesure de confiance permet une amélioration encore plus importante des performances en termes de *WER* et d'*IER* (par rapport au *pivot topologique* et à la *1-best*).

Malgré une réduction importante de la taille des CNs créés avec l'algorithme proposé, leur largeur reste relativement importante. Après en avoir analysé les causes, nous avons présenté une méthode d'élagage des CNs qui permet d'éliminer certaines classes de mots en fonction d'un seuil sur le rapport des probabilités *a posteriori* de l'omission et du meilleur mot de la classe (à condition que l'omission ait la plus grande probabilité *a posteriori*). Cette méthode a permis de réduire très fortement la taille des CNs avec une

très légère diminution des performances oracle.
Nous avons également proposé un algorithme de *parsing* des réseaux de confusion qui a permis d'obtenir des CNs très compacts en éliminant l'information non-pertinente du point du vue des modules applicatifs (le module de compréhension dans notre cas). Les résultats préliminaires obtenus montrent qu'il est possible de réduire significativement la taille des réseaux avec une légère dégradation des performances.

Le nouvel algorithme de génération des CNs ainsi que l'algorithme de *parsing* ont été proposés dans un contexte de compréhension de la parole. Nous proposons de les utiliser dans un contexte de traduction automatique en réalisant quelques légères adaptations. Les résultats présentés en annexe montrent une amélioration de meilleures performances de traduction relativement à la *1-best*, proches de celle obtenues avec la transcription Verbatim du corpus de test.

Stratégies de décision

Nous avons montré que l'utilisation des CNs n'est pas adaptée pour la détection et le rejet des messages non-valides (énoncés non-parole, commentaires). De plus, la génération des CNs sur ces énoncés est très coûteuse en temps de calcul ce qui n'est pas compatible avec les contraintes temps-réel d'un système du dialogue. Nous avons proposé une stratégie de décision qui rejette ces énoncés dès la première passe de reconnaissance et qui permet de calculer l'interprétation seulement sur les énoncés valides. La stratégie utilise la meilleure solution de la première passe de reconnaissance pour rejeter les énoncés détectés comme non-valides et les CN sont générés ensuite sur les énoncés non-rejetés. Cette stratégie permet une amélioration des performances en terme d'*IER* par rapport à l'utilisation de la *1-best* pour tous les énoncés.

Nous avons montré par ailleurs que le recours à la recherche intégrée pour rechercher la meilleure interprétation non pas sur la meilleure séquence de mots issue du SRAP mais sur le graphe de mots complet ou le CN ne s'avère pas pertinente dans tous les cas. En effet, il subsiste une partie des énoncés valides qui ne sont pas couverts par les règles sémantiques et qui ne peuvent pas être interprétés par le système. L'utilisation de la recherche intégrée sur les graphes de mots ou sur les CNs correspondant peut donner lieu à des erreurs d'interprétation (notamment des fausses alarmes). Nous avons proposé une stratégie adaptée qui permet de rejeter également ce type d'énoncé en plus des énoncés non-valides grâce à un enchaînement d'étapes de rejet utilisant différentes solutions. Ainsi, pour le rejet des énoncés non-valides la stratégie utilise la meilleure solution de la première passe de reconnaissance. Ensuite, les CNs sont générés et les énoncés pour lesquels la *consensus hypothesis* ne peut pas être interprétée sont rejetés. La recherche intégrée est ensuite effectuée pour les énoncés restants. Les résultats montrent une amélioration des performances au niveau interprétation en utilisant à bon escient la recherche intégrée sur les graphes de mots. Les CNs s'avèrent un moyen très efficace de sélectionner les énoncés pertinents.

Perspectives

Cette thèse s'est attachée à réaliser une meilleure intégration de l'information du système de dialogue dans la génération des CN dans le but de rendre plus robuste la reconnaissance des mots porteurs de sens. L'analyse en concepts cherche à détecter en priorité les séquences de mots les plus longues allumant un concept. L'algorithme de génération des CNs proposé ne traite pas ces séquences, les mots les composant étant traités séparément comme étant des mots non-vides. Il serait alors intéressant de traiter ces mots comme une séquence dès leur insertion dans le CN. L'algorithme de génération devrait alors être modifié pour insérer dans le CN plusieurs transitions simultanément et de manière conjointe.
Une autre direction à suivre est la création des CNs de concepts. Les transitions porteraient alors non plus des mots mais des concepts. Deux possibilités existent pour cela. D'une part, générer des CNs de manière "classique", mais à partir de graphes de concepts. Un problème important qui se pose dans ce cas réside dans le calcul de la probabilité *a posteriori* pour des concepts allumés par des séquences de plusieurs mots. D'autre part, les CNs de concepts pourraient être générés directement à partir de CNs de mots. Ceci peut soulever un problème d'alignement temporel car des concepts allumés par des séquences de mots plus ou moins longues pourraient se trouver en parallèle. Il serait donc intéressant de créer des CNs dont les transitions peuvent couvrir plusieurs classes adjacentes. Une transition peut alors être en concurrence avec plusieurs transitions de classes adjacentes. On peut alors penser que la recherche d'une interprétation dans un CN de concepts produirait de meilleures performances.

Nous avons montré que la recherche intégrée n'est pas très adaptée aux CNs. D'un autre côté, les résultats préliminaires obtenus avec l'algorithme de *parsing* tant dans la compréhension de la parole que dans la traduction sont intéressants. Les CNs sont des structures qui mettent en parallèle des solutions localement. Nous pensons que des techniques plus complexes de recherche ou des techniques de rescoring des mots devraient s'appliquer de manière locale sur des fenêtres glissantes de plusieurs classes de manière à couvrir progressivement l'ensemble du CN.

Appendices

Annexe A

Calcul de la probabilité *a posteriori* dans un graphe de mots

Avant la description de l'algorithme, nous rappelons quelques notations utilisées. Le signal de parole est représenté par $X_T = x_1, \ldots, x_T$ une suite de T vecteurs d'observations acoustiques. On note t une transition du graphe de mots caractérisée par l'instant de début $\tau_d(t)$, l'instant de fin $\tau_f(t)$ et le mot porté $w(t)$.

A.1 Algorithme *Forward-Backward*

Procédure Forward

La procédure *forward* définit une variable $\alpha(t)$ qui représente la probabilité de la séquence partielle d'observations acoustiques, à partir de la trame 1 et jusqu'à $\tau_f(t)$, et de la transition t :

$$\alpha(t) = p(x_1, \ldots, x_{\tau_d(t)}, \ldots, x_{\tau_f(t)}, t) \tag{A.1}$$

On obtient par récurrence sur t :

$$\alpha(t) = \left[\sum_{t' | t' \in pred(t)} \alpha(t') P(w(t) | w(t')) \right] P(x_{\tau_d(t)}, \ldots, x_{\tau_f(t)} | t) \tag{A.2}$$

où $pred(t)$ désigne l'ensemble de prédécesseurs de t. Cet ensemble est défini comme contenant toutes les transitions t' dont l'état final coincide avec l'état de début de la transition t.
L'initialisation se fait de la manière suivante :

$$\alpha_0 = 1 \tag{A.3}$$

Le calcul de $\alpha(t)$ par récurrence représente en effet la somme des probabilités de tous les chemins dans le graphe qui arrivent dans la transition t.

Procédure Backward

La procédure *Backward* est très similaire. La variable $\beta(t)$ est définie comme étant la probabilité de la séquence partielle d'observations acoustiques, à partir de la trame $\tau_f(t)+1$ jusqu'à la dernière trame T, sachant que l'on part de la transition t :

$$\beta(t) = p(x_{\tau_f(t)+1}, \ldots, x_T | t) \tag{A.4}$$

On obtient par récurrence :

$$\beta(t) = \left[\sum_{t' | t' \in succ(t)} \beta(t') P(w(t') | w(t)) P(x_{\tau_d(t')}, \ldots, x_{\tau_f(t')} | t') \right] \tag{A.5}$$

où $succ(t)$ désigne l'ensemble de successeurs de t. Il est défini comme contenant toutes les transitions t' dont l'état de début coincide avec l'état de fin de la transition t. On a bien sûr $\tau_d(t') = \tau_f(t) + 1$.
L'étape d'initialisation pour les variables β se fait pour toutes les transitions t telles que $\tau_f(t) = T$ de la manière suivante :

$$\beta(t) = 1 \tag{A.6}$$

La probabilité *a posteriori* de la transition t se calcule comme étant la somme des probabilités de tous les chemins qui passent pas la transition t, normalisée par la probabilité de tous les chemins du graphes G :

$$p(t) = \frac{\alpha(t)\beta(t)}{\sum_{t' | \tau_f(t')=T} \alpha(t')\beta(t')} \tag{A.7}$$

A.2 Normalisation des probabilités *Forward* et *Backward*

Procédure Forward

On défini $\hat{\hat{\alpha}}(t)$ comme étant une version locale de $\alpha(t)$ avant normalisation. On obtient alors grâce à la formule de récurrence A.2 :

$$\hat{\hat{\alpha}}(t) = c_{\tau_d(t)} \ldots c_{\tau_f(t)-1} \left[\sum_{t' | t' \in pred(t)} \hat{\alpha}(t') P(w(t) | w(t')) \right] P(x_{\tau_d(t)}, \ldots, x_{\tau_f(t)} | t) \tag{A.8}$$

On définit le coefficient de normalisation pour l'instant τ par :

$$c_\tau = \frac{1}{\sum_{t | \tau_f(t)=\tau} \hat{\hat{\alpha}}(t)} \tag{A.9}$$

La version normalisée de α est calculée comme suit :

$$\hat{\alpha}(t) = c_\tau \hat{\hat{\alpha}}(t) \tag{A.10}$$

Compte tenu des nouvelles notations, la phase d'initialisation devient :

$$\hat{\alpha}_0 = c_0 \hat{\hat{\alpha}}_0 \tag{A.11}$$

où $c_0 = 1$ et $\hat{\hat{\alpha}}_0 = \alpha_0$.
On peut exprime la valeur normalisée $\hat{\alpha}(t)$ de la manière suivante :

$$\hat{\alpha}(t) = c_0 \dots c_{\tau_f(t)} \alpha(t) \tag{A.12}$$

Procédure Backward

La normalisation de la variable *backward* utilise les mêmes coefficients de normalisation que la variable *forward*. Du fait que l'ordre de grandeur des deux variables est similaire, l'utilisation des mêmes coefficients permet de maintenir les valeurs des β dans l'intervalle de précision de l'ordinateur.
La version normalisée de β devient alors :

$$\hat{\beta}(t) = c_{\tau_f(t)+1} \dots c_T \beta(t) \tag{A.13}$$

La probabilité *a posteriori* de la transition t s'écrit de la façon suivante :

$$p(t) = \frac{\hat{\alpha}(t)\hat{\beta}(t)}{\sum\limits_{t' | \tau_f(t')=T} \hat{\alpha}(t')\hat{\beta}(t')} \tag{A.14}$$

En replaçant les variables normalisés comme mentionné dans les équations A.12 et A.13 on obtient :

$$p(t) = \frac{c_0 \dots c_{\tau_f(t)} \alpha(t) c_{\tau_f(t)+1} \dots c_T \beta(t)}{\sum\limits_{t' | \tau_f(t')=T} c_0 \dots c_{\tau_f(t')} \alpha(t') c_{\tau_f(t')+1} \dots c_T \beta(t')} \tag{A.15}$$

Le produit des coefficients peut être écrit de la manière suivante :

$$c_0 \dots c_{\tau_f(t)} c_{\tau_f(t)+1} \dots c_T = \prod_{\tau=0}^{T} c_\tau = C \tag{A.16}$$

Le produit des coefficients du dénominateur est donc indépendant de la somme sur t'. Les coefficients s'annulent et on obtient la même équation que A.7.

Annexe B

Traduction statistique de la parole : Contexte applicatif

Le domaine de la traduction automatique de la parole est assez ancien, les premiers travaux ont été menés immédiatement après l'apparition des calculateurs électroniques qui allaient devenir les futurs ordinateurs. Au départ les systèmes de traduction automatique avaient pour but d'aider le traducteur humain dans son travail. De nos jours de plus en plus de travaux visent la construction de systèmes capables de produire de façon automatique une traduction qui soit utilisable telle quelle.
Les premiers systèmes de traduction étaient constitués de quelques règles de grammaire et d'un dictionnaire bilingue. Ainsi, en 1954, un système de traduction automatique présenté par l'Université de Georgetown et IBM, et constitué de 6 règles de grammaire et d'un dictionnaire de 250 mots, a été capable de traduire une soixantaine de phrases du russe à l'anglais. Au fil des années les systèmes de traduction voient leur dictionnaire grossir et le nombre de règles augmenter. Ainsi la nécessité d'automatiser l'acquisition des règles et leur généricité s'impose, et on observe un développement de la linguistique informatique avec, par exemple, l'apparition des grammaires formelles. Les systèmes sont de plus en plus spécialisés sur des domaines plus restreints. Un des systèmes commerciaux de traduction automatique le plus connu utilise l'approche de traduction à base de règles (Senellart et al., 2001).
Le triangle présenté dans la figure B.1, appelé aussi triangle de Vauquois, présente de manière synthétique une analyse du processus de traduction encore pertinente et employée de nos jours. Selon ce triangle, une traduction peut s'effectuer à plusieurs niveaux, avec des analyses de plus en plus approfondies de la phrase source. Au plus bas niveau on retrouve la traduction qui passe directement des mots de la langue source aux mots de la langue cible. En passant au niveau supérieur grace à une analyse syntaxique, le transfert vers la langue cible devrait être simplifié. Par exemple, les détails spécifiques à la constitution des groupes nominaux n'ont pas besoin d'être connus par les règles de transfert de la langue source à la langue cible. Au niveau supérieur, l'analyse sémantique de la phrase source rend le transfert vers la langue cible uniquement sémantique et donc plus simple. En revanche, l'analyse descendante au niveau de la

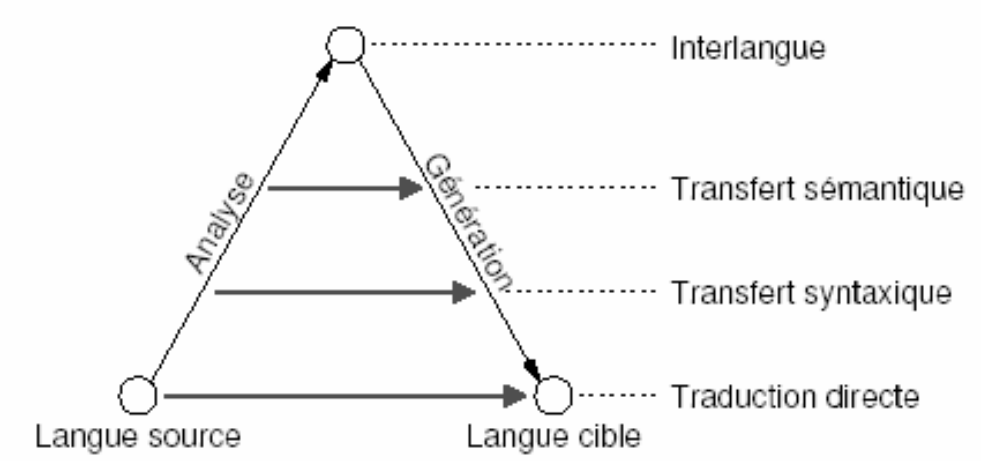

FIGURE *B.1* – Le triangle de "Vauquois" pour la traduction

langue cible (la génération des mots dans la langue cible) peut s'avérer plus complexe qu'au niveau inférieur. Le dernier niveau de l'analyse implique la création d'une "*interlangue*" artificielle qui permettrait une représentation complète du sens de la phrase source et une génération de la phrase cible.
Les vingt derniers années ont vu l'apparition de nouvelles méthodes de traduction basées notamment sur de grandes quantités de données ("*corpus-based approaches*"). Parmi ces méthodes on en retient particulièrement deux :

- La traduction automatique à base d'exemples ("*exemple-based machine translation*") est basée sur des exemples préalablement traduits ; un corpus parallèle de phrases dans la langue source ayant une traduction dans la langue cible. Pour réaliser la traduction d'une phrase le système parcourt sa base d'exemples et produit une traduction si la phrase est trouvée. Sinon, comme dans la majorité des cas, le système cherche des exemples qui contiennent des segments communs avec la phrase à traduire. Suit une phase d'alignement qui permet de retrouver la traduction des différents segments. Enfin, le système assemble les segments et produit une traduction dans la langue cible. Un tour d'horizon de cette approche est présenté dans (Somers, 1999).
- La traduction automatique par méthodes statistiques ("*Statistical machine translation - SMT*") (Brown et al., 1990) est, comme la traduction automatique à base d'exemples, basée sur un corpus parallèle. Un modèle statistique de traduction est défini comprenant une ou plusieurs lois de probabilités estimées sur un corpus parallèle d'apprentissage. A la différence des autres méthodes, la traduction par des méthodes statistiques permet de garantir que pour toute phrase dans la source une phrase dans la langue cible sera générée ; même si la syntaxe n'est pas forcément correcte elle pourra permettre de transmettre le sens de la phrase originale. On peut donc voir la traduction statistique comme une combinaison entre une modélisation linguistique et une prise de décision statistique.

B.1 Systèmes de traduction statistique

Les travaux expérimentaux de cette thèse se placent dans le contexte d'un système de traduction statistique. Ce type de système utilise, généralement, un des trois modèles de traduction suivants :

- Les modèles à base de mots : l'unité qui apparaît dans les lois de probabilité est le mot. (Brown et al., 1993) ont défini cinq modèles statistiques de complexité croissante est proposé un algorithme pour leur apprentissage. Celui-ci se fait à partir d'un corpus parallèle, aligné de phrases. Aucune information *a priori* n'est nécessaire. Il existe aussi des modèles dépendant du contexte (Berger et al., 1996) visant à limiter la complexité des modèles définis par (Brown et al., 1993). Les modèles statistiques à base de mots permettent une meilleure traduction que par une technique "mot à mot" car ils autorisent qu'un mot se traduise par plusieurs mots ou que les mots soient réordonnés. De plus, un modèle de langage est généralement employé par le système de traduction et donc le choix de traduction de chaque mot est pris en considérant son impact sur la phrase générée. Néanmoins, même entraînées sur des corpus adéquats, ces modèles sont susceptibles de mal traduire les expressions courantes du fait de certains paramètres qui entrent dans leur construction. De plus, un autre problème est l'asymétrie des alignements qu'il impose. Un mot de la langue cible ne peut être aligné à plus d'un mot de la langue source. Ceci rend difficile la modélisation des mots comme, par exemple, *n' est pas* en *isn't* pour une traduction français-anglais.
- Les modèles par groupes de mots ou syntagme (*phrase-based translation models*) permettent de corriger ces défauts en symétrisant l'alignement entre mots source et cible. Dans ces modèles, l'unité de traduction n'est plus le mot mais des groupes de mots, qui peuvent compter un ou plusieurs mots. Le principe du décodage est le suivant : chaque phrase source est segmentée en N groupes de mots. Chaque groupe de mots est ensuite traduit en un groupe de mots cible. Avant de constituer la phrase cible en recollant tout simplement les groupes cibles obtenus, une phase de réordonnancement des groupes de mots cibles peut être nécessaire (voir la figure B.2). Utiliser ce type de modèle permet donc d'aligner n mots source à m mots cible. Une propriété importante des modèles à base de groupes de mots est qu'ils sont capables de traduire directement des groupes nominaux (ex : *nom+adjectif*) observés dans le corpus d'apprentissage. Ces modèles utilisent comme loi de probabilité pour la traduction une "*table de traduction (phrase table)*" qui assigne une probabilité à chaque couple de groupes de mots en langue source et cible. Afin de pouvoir effectuer le réordonnancement des groupes de la langue cible, un modèle de distorsion est utilisé qui calcule un paramètre de distorsion exprimé par la distance entre la position initiale et finale du groupe de mots.
- Les modèles statistiques syntaxiques. Malgré le fait que les modèles statistiques à base de groupes de mots représentent aujourd'hui l'approche dominante dans la traduction statistique, des recherches sont menées pour le développement des modèles qui s'appuient sur la syntaxe de la phrase. Ces recherches sont motivées par plusieurs faiblesses des modèles "*phrase-based*". Ainsi, pour certaines langues la plupart des réordonnancements sont motivés par la syntaxe. De plus,

les modèles syntaxiques permettent une connaissance sur des mots nécessaires à la construction d'une phrase correcte (ex : le choix des déterminants ou des auxiliaires pour la conjugaison de verbes). Ils peuvent aussi conditionner la traduction d'un mot sur un autre avec qui il a un lien syntaxique. Les inconvénients des cette approche sont liés à la complexité des modèles et à leur mise en œuvre. De plus les analyses effectuées pendant l'apprentissage et le décodage sont dépendantes des deux langues voire du sens de la traduction. Toutefois, ces modèles obtiennent de plus en plus de résultats compétitifs dans des compagnes d'évaluation internationales (comme la campagne d'évaluation NIST 2006).

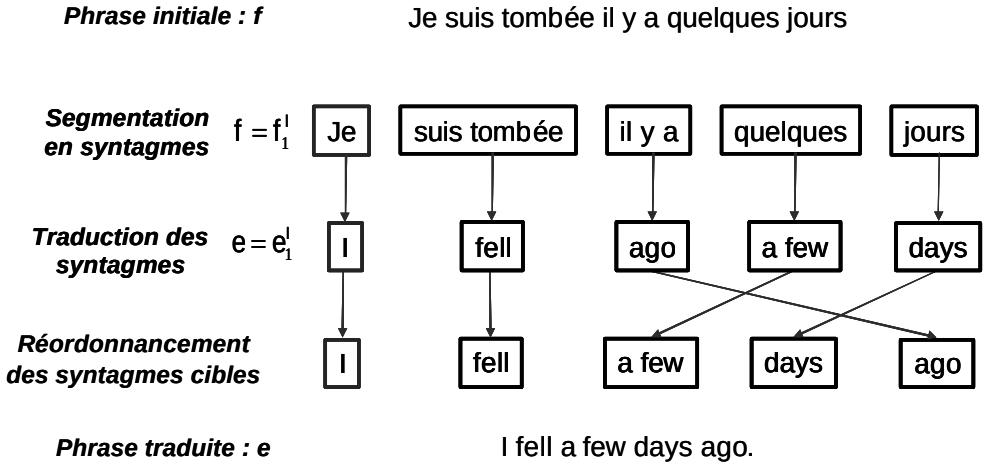

FIGURE *B.2* – Le processus de traduction d'un phrase source "f" (français) vers une phrase cible "e" (anglais) pour un système de traduction statistique à base de groupes de mots.

Comme nous l'avons évoqué, les travaux expérimentaux de cette thèse se placent dans le contexte d'une approche de traduction statistique utilisant des modèles de traduction à base de groupes de mots. Le système de traduction utilisé dans les travaux de cette thèse est un système de traduction statistique qui utilise un modèle de traduction à base de groupes de mots appelé **Moses**. Ce système a été développé pendant le JHU Workshop en 2006 (Koehn) et correspond à l'état de l'art dans la traduction statistique à base de groupes de mots. A l'aide d'un dictionnaire bilingue de groupes de mots, le décodeur développe, dans une recherche en faisceau ("*beam-search*"), toutes les segmentations possibles de la phrase source et leur traductions respectives comme dans l'exemple B.3. Pour chacune de ces traductions, Moses calcule un score associé obtenu par une combinaison log-linéaire de différents paramètres comme le modèle de langage n-grammes de la langue cible, la table de traduction qui fournit une probabilité de traduction aux couples de groupes de mots de la langue source et cible et différents modèles de pénalités (comme les pénalités de distorsion ou de longueur de l'énoncé cible).
Moses est capable de traduire aussi bien des séquences de mots que des réseaux de confusion. Toutefois, l'utilisation des CN nécessite l'optimisation d'un certain nombre de paramètres supplémentaires par rapport à la traduction d'une seule phrase, notamment car il inclut les probabilités *a posteriori* dans le processus global d'optimisation du score. Cette optimisation demande l'utilisation d'un corpus de développement conte-

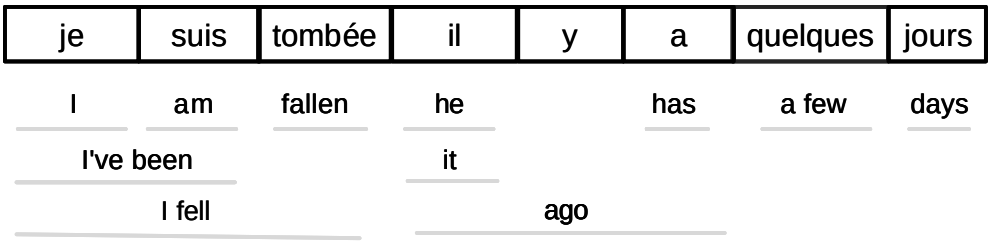

FIGURE *B.3* – Le développement des segmentations possibles de la phrase source et leur traduction

nant un nombre très grand de données. Comme nous le montrons dans la section B.2, nous ne disposons pas d'une telle quantité de données et pour cela nous introduisons deux techniques qui nous permettent d'utiliser Moses avec une quantité de données de développement réduite :

- L'utilisation des classes *a priori* dans les différents modèles de langage et de traduction pour améliorer le degré de généralisation des modèles. Cette technique est décrite dans la section B.3.
- L'utilisation de l'algorithme de parsing des CNs (décrit au 6.6) afin de bénéficier de l'espace de recherche important des CNs tout en produisant une structure très compacte et adaptée pour le moteur de traduction. Le fonctionnement de l'algorithme de parsing dans le contexte de la traduction est décrit dans la section C.2 de l'annexe C.

B.2 Description des données expérimentales

Dans cette partie nous décrivons le domaine d'application et les corpus associés ainsi que les différents modèles nécessaires au fonctionnement de Moses (la table de traduction, le modèle de langage de la langue cible).
Pour cette tâche de traduction de la parole continue nous avons travaillé dans le domaine de la pharmacie (Mami et Damnati, 2006). Les corpus ont été collectés dans des conditions de laboratoire. Imaginons une situation où une personne dans un pays étranger (dans notre cas, une personne francophone dans un pays anglophone) va dans une pharmacie pour demander des renseignements, des médicaments etc. Le module de traduction est alors vu comme un intermédiaire ayant pour but de faciliter un dialogue naturel dans un domaine spécifique entre deux personnes ne parlant pas la même langue. La langue source est donc le français et la langue cible de la traduction est l'anglais. On retrouve trois situations principales correspondants à différentes étapes du dialogue. La première situation correspond à la politesse, quand le client salue le pharmacien ou le remercie. La deuxième est la plus spécifique au domaine de la pharmacie ; le client explique ses symptômes et demande des conseils ou des médicaments, etc. La dernière situation est liée aux modalités de paiement. Plusieurs situations peuvent coexister dans un même énoncé.
Les corpus utilisés ont été créés en utilisant plusieurs méthodes. Tout d'abord, un cor-

pus bilingue a été construit manuellement par un interprète professionnel. Chaque phrase a ensuite été découpée en situations pouvant être combinées par la suite selon différents scénarios. Pour une situation donnée, des phrases supplémentaires ont été créées en utilisant des classes *a priori*. Les classes on été construites suivant des considérations sémantiques et syntaxiques. 7 catégories sémantiques ont été sélectionnées (médicaments, produits pharmaceutiques, parties de corps, etc.), chacune étant déclinée en genre et en nombre pour aboutir à un total de 32 classes. Étant donné cette base de données structurée manuellement et un ensemble de grammaires de génération des phrases il est possible d'extraire un grand nombre d'énoncés. Afin d'éviter les redondances et la surgénération, nous avons sélectionné un groupe de 3542 énoncés appelé corpus automatique. Le corpus d'apprentissage est constitué de ce corpus automatique. Plus de détails sur la construction du corpus sont fournis dans (Mami et Damnati, 2006).

Corpus	Type	# énoncés	# mots	longueur moyenne des énoncés
App	Verbatim	5131	48811	9.5
	Corpus propre	5131	44003	8.6
Dev	Verbatim	1023	14999	14.7
	Corpus propre	1023	14479	14.1
Test	Verbatim	286	2953	10.3
	Corpus propre	286	2646	9.3

TABLE *B.1* – *Description des corpus utilises*

Une autre partie du corpus est composée d'énoncés collectés à l'aide d'un *Magicien d'Oz*. Ces énoncés sont plus longs que ceux du corpus automatique et contiennent beaucoup de disfluences. Le corpus a été transcrit et les énoncés annotés de façon détaillée afin de pouvoir extraire les disfluences et d'obtenir un "*corpus propre*". Le corpus contenant les disfluences est appelé *Verbatim*. Les traductions de référence ont été réalisées à partir du *corpus propre*. Le tableau B.1 décrit les corpus d'apprentissage (*App*), de développement (*Dev*) et de test (*Test*) utilisés. Pour chaque corpus, le tableau présente le détail du nombre d'énoncés, nombre de mots dans la référence en français et la longueur moyenne de cette référence pour les transcriptions *Verbatim* et *Corpus propre* de chacun des trois corpus. Le lexique utilisé est composé de 2420 mots, dont 477 sont groupés dans les 32 classes utilisées. Le SRAP utilise un modèle de langage de type bigramme qui incorpore les 32 classes *a priori*. Les mots dans les classes sont équiprobables.
Les énoncés en français ont été traduits et alignés manuellement utilisant le positionnement relatif des groupes de mots. Un alignement automatique, obtenu avec le logiciel GIZA en partant de l'alignement manuel, est aussi utilisé sur le corpus d'apprentissage. Ensuite, la méthode de *refined alignements* (Koehn et Marcu, 2003) a été appliquée sur les énoncés alignés automatiquement afin d'obtenir les paires de groupes de mots dans le deux langues. Les groupes de mots bilingues ont été ensuite combinés pour obtenir une table de traduction. Les différentes probabilités de traduction ont été estimées en utilisant le maximum de vraisemblance en donnant un poids cinq fois plus grand pour les alignements manuels, car considérés plus fiables. Ce facteur de cinq a été déterminé

empiriquement.

B.3 Modèle de traduction avec classes *a priori*

Nous avons décidé d'intégrer les classes *a priori* dans les différents modèles stochastiques utilisés par le module de traduction (la table de traduction, le modèle de langage cible). Les principales motivations pour cette intégration sont d'une part une meilleure intégration entre le SRAP et le module de traduction statistique en utilisant les mêmes connaissances *a priori* et donc les mêmes classes *a priori* que celles utilisées dans la construction du modèle de langage du SRAP. D'autre part, l'introduction de ces classes *a priori* améliore le degré de généralisation des modèles stochastiques surtout dans le contexte d'un domaine restreint avec une quantité de données limitée qui est le notre.
Les 32 classes *a priori* sont définies pour les deux langues. D'un coté, les classes *a priori* de mots en français qui contiennent la liste des mots en français, et de l'autre coté les classes correspondantes en anglais qui contiennent toutes les traductions possibles des mots français. Un groupe de dictionnaires bilingues est ainsi défini. Les alignements de groupes de mots ne seront plus appris au niveau mot mais en mélangeant les mots et les classes *a priori* ; les mots étant remplacés par les classes correspondantes.
Toutefois, la création de ces classes n'est utile que si les mots qui les composent se traduisent par des mots qui font partie de la classe correspondante dans la langue cible. Mais chaque langue a ses particularités. Si on prend l'exemple "*J'ai mal à la tête*", une traduction possible est "*I have a headache*". Alors que l'énoncé en français contient le mot "tête" appartenant à la classe de parties du corps, l'énoncé en anglais contient "headache" et non "head" qui est la traduction de "tête". C'est une exception de l'anglais qui utilise d'habitude le mot "hurt" pour désigner un mal (ex : "*my arm hurts*"). Nous voulons donc préserver ces particularités malgré l'introduction des classes *a priori* ce qui impose une adaptation de la manière d'annoter le corpus d'apprentissage en classes.
Chaque paire d'énoncés français/anglais a été annotée indépendamment et vérifiée ensuite de façon automatique. Ainsi, pour chaque paire, les énoncés doivent contenir le même nombre de classes et ces classes doivent être identiques pour que la paire d'énoncés annotés soit gardée dans le corpus d'apprentissage (la paire d'énoncés sans annotation est conservée si cette condition n'est pas satisfaite). Sinon, seules les classes identiques présentes dans les deux énoncés sont gardées. Ainsi, pour la paire "*J'ai mal à la tête/I have a headache*" la classe partie de corps activée par le mot "tête" n'est pas gardée parce qu'elle n'est pas présente dans la traduction en anglais. Ce critère de vérification automatique permet d'éviter des fausses annotations, notamment pour des expressions idiomatiques (ex : "*jeter un coup d'oeil*" qui se traduit par "*have a look*").
Le modèle *n-grammes* de la langue cible est aussi appris en utilisant les classes *a priori* sur le corpus d'apprentissage annoté en anglais obtenu après vérification. Les alignements des groupes de mots et les probabilités de traduction sont aussi appris sur ce corpus.

B.4 Processus de traduction avec classes *a priori*

L'utilisation des classes *a priori* dans le processus de traduction basé sur Moses nécessite le rajout de deux étapes d'annotation en amont et en aval de Moses comme le montre la figure B.4.

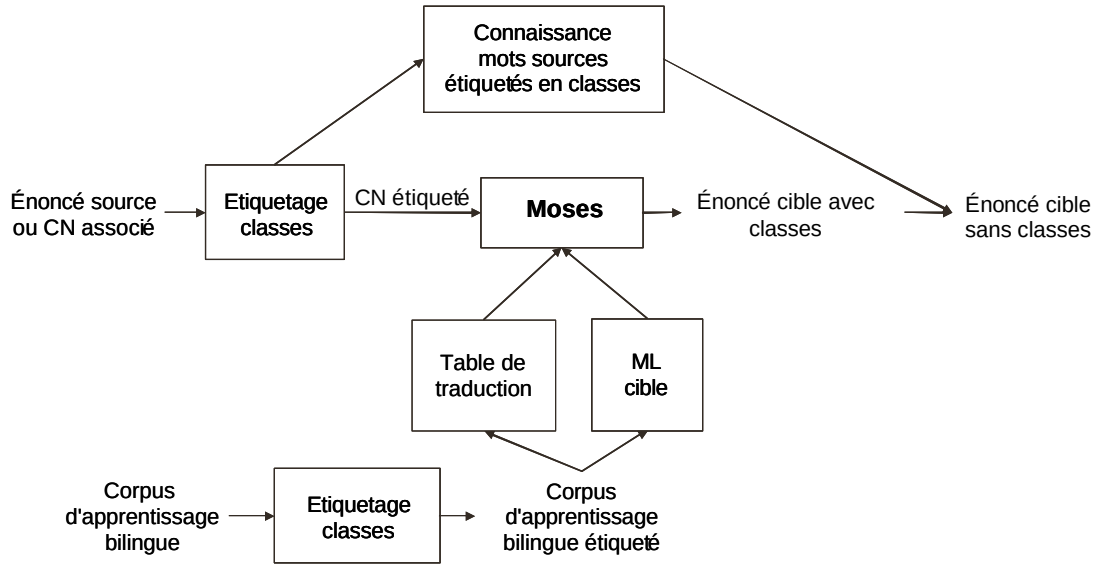

FIGURE *B.4* – Le processus de traduction avec des classes *a priori* basé sur Moses

Comme nous l'avons décrit, des ambiguïtés peuvent exister pendant le processus d'étiquetage en classes. Afin de garder toutes les possibilités, nous avons décidé d'exploiter la capacité de Moses de traduire des réseaux de confusion. Une séquence de mots à traduire est alors transformée en un CN à travers l'annotation des classes *a priori*. Le CN ainsi créé est ensuite fournie à Moses à la place de la séquence de mots. Le processus de traduction se déroule alors en trois étapes :

1. Étape 1 - **Étiquetage mots $\longrightarrow$ classes dans la langue source** : Création du CN à travers l'annotation de la séquence de mots en classes
2. Étape 2 - **Traduction basée sur le modèle utilisant les classes *a priori*** : La traduction est effectuée à l'aide de Moses
3. Étape 3 - **Étiquetage classes $\longrightarrow$ mots dans la langue cible** : Les éventuelles classes dans la solution fournie par Moses sont remplacées par la traduction du mot initial correspondant à ces classes

Pendant l'étape 1, (*étiquetage mots $\longrightarrow$ classes dans la langue source*), un CN est donc construit à partir de la séquence de mots en créant des transitions alternatives pour les mots qui appartiennent à différentes classes *a priori*. Afin d'éviter une surgénération des transitions (certains mots appartenant à des classes *a priori* n'apparaissent jamais tout seuls dans le corpus d'apprentissage), les transitions alternatives sont crées seulement pour les mots appartenant à une classe et qui apparaissent aussi seuls dans le corpus. La première transition porte le mot et sa probabilité *a posteriori* est égale à la probabilité

que le mot soit gardé pendant la phase d'étiquetage en classes (estimée sur le corpus d'apprentissage par le maximum de vraisemblance). La deuxième transition porte la classe *a priori* associée au mot avec la probabilité complémentaire.
Lors de la deuxième étape du processus de traduction, (*Traduction basée sur le modèle utilisant les classes* a priori), Moses est capable de traduire le réseau de confusion créé (la traduction des CNs par Moses est détaillée dans (Bertoldi et al., 2007)).
Si l'hypothèse obtenue suite à l'étape 2 contient une ou plusieurs classes *a priori*, l'utilisation du dictionnaire bilingue et l'information sur le découpage de l'énoncé source permettent, à l'étape 3, d'obtenir l'hypothèse de traduction finale au niveau mot.
Afin de traduire des CNs générés à partir des graphes de mots, à l'étape 1 du processus de traduction, les mots du CN sont annotés en classes et ils sont remplacés de la même manière que celle décrite ci-dessus pour une séquence de mots. Les deux autres étapes ne changent pas.

Annexe C

Traduction statistique de la parole : Génération des réseaux et résultats

La traduction des CNs nécessite une exploration de tous les chemins du réseau. De part sa structure linéaire et la propriété que les chemins passent par tous les nœuds, la traduction des CNs est très similaire à la traduction d'un texte car on doit rechercher toutes les traductions possibles sur chaque segment de la phrase source. Le problème principal qui se pose dans le cas des CNs est que le nombre de phrases source augmente de manière exponentielle avec la largeur des segments (le nombre de classes couvertes). Comme nous l'avons précisé dans la description du cadre expérimental, dans la section B, nous ne disposons pas d'un volume de données de développement suffisant pour optimiser correctement un certain nombre de paramètres supplémentaires, nécessaires pour le fonctionnement optimal de MOSES, par rapport à la traduction d'une seule phrase. Ces observations expliquent le fait que la traduction obtenue en utilisant les CNs sans aucun post-traitement particulier n'a pas produit les performances attendues. Pour cela nous avons introduit, dans un premier temps, l'utilisation des classes *a priori* (voir la section B.3 pour plus de détails) dans les différents modèles utilisés. Ensuite, au lieu d'utiliser les CNs tels quels, une phase de post-traitement est introduite avant leur traduction. Celle-ci consiste en un *parsing* à l'aide de l'algorithme décrit dans la section 6.6 qui ne gardera que les syntagmes importantes pour la traduction, tout en réduisant la taille des CNs. Par ailleurs, la construction des CNs tient compte des classes *a priori* utilisées à travers l'algorithme de génération *multi-niveaux* généralisé décrit dans la section 6.7.

C.1 Génération des réseaux de confusion

La génération des réseaux de confusion tient compte des classes *a priori* définies afin de favoriser les mots pertinents pour la traduction contenus dans ces classes. Pour cela, nous utilisons l'algorithme *multi-niveaux* généralisé qui consiste en une première phase de calcul du *pivot* et en une deuxième phase qui insère, en plusieurs étapes, les

transitions du graphe de mots dans le CN. Cette deuxième phase est composée, dans ce cas, de trois étapes successives, et insère les transitions correspondant aux critères de chaque étape :

Étape 1 : Les transitions portant un mot du *pivot* sont insérées dans le réseaux si leur insertion ne nécessite pas la création de nouvelles classes.

Étape 2 : Les transitions portant un mot appartenant à une des classes *a priori* sont insérées dans le réseau. Aucune restriction n'est faite en ce qui concerne la création de nouvelles classes. Toutes les classes *a priori* ont le même degré d'importance.

Étape 3 : Les transitions restantes sont insérées dans le réseau.

Pour chacune des étapes, les conditions de recouvrement temporel et de relation d'ordre s'appliquent exactement comme pour l'algorithme *multi-niveaux* décrit dans la section 6.3. Aucune transition du graphe de mots n'est écartée du processus de génération de CNs.

C.2 Parsing des réseaux de confusion

L'algorithme de *parsing* décrit dans la section 6.6 réalise un élagage des CNs en tenant compte des séquences de mots qui allument des concepts. De la même manière, on peut appliquer cet algorithme pour tenir compte de la table de traduction utilisée par Moses. En effet, la table de traduction contient des correspondances entre des séquences de mots (syntagmes) de la langue source et des séquences de mots de la langue cible. Pour adapter les CNs à la tâche de traduction, l'algorithme de *parsing* tient compte des syntagmes de la langue source de la même manière que pour les séquences de mots allumant un concept.
Les graphes de mots, et par conséquent les CNs, peuvent contenir des hésitations ou des erreurs dues aux disfluences présentes dans le langage naturel. Ces disfluences ne sont pas modélisées, car le modèle de traduction a été appris sur des données propres, et peuvent donc gêner la traduction. L'avantage de l'algorithme de *parsing*, à part le fait de réduire la taille de CNs en les adaptant à la tâche de traduction, réside dans le filtrage de ces disfluences car elles ne forment pas des syntagmes.
Le déroulement de l'algorithme de *parsing* est identique à celui décrit au 6.6 avec l'extraction des syntagmes de la langue source pour chaque fenêtre et le calcul d'une probabilité pour chaque syntagme. On utilise le même processus itératif pour couvrir l'ensemble de CN et extraire ainsi pour chaque fenêtre les N meilleures syntagmes. Le CN est ensuite reconstruit et la probabilité des mots dans le nouveau CN est calculée.
Trois paramètres doivent être spécifiés pour cet algorithme de *parsing* : la longueur des syntagmes utilisées, qui donne implicitement la largeur de la fenêtre ; la valeur de N qui donne le nombre de phrases prises en compte pour chaque fenêtre ; la méthode de calcul de la probabilité *a posteriori* des mots dans le CN reconstruit. Nous avons déterminé de manière empirique le jeu de paramètres optimal :

- Nous avons utilisé des syntagmes contenant un maximum de 3 mots. Par conséquence, la largeur de la fenêtre est de 4 classes adjacentes.

- Nous avons choisi de n'utiliser que la meilleure syntagme de chaque fenêtre afin d'obtenir des CNs avec une taille réduite.
- La probabilité des mots dans le nouveau CN a été estimée comme étant égale à la fréquence d'apparition du mot dans les syntagmes choisies à une position donnée.

Une étape d'élagage des classes avec une valeur du seuil de 10 (voir 6.5 pour la description de l'algorithme d'élagage) est appliqué aux CNs avant le *parsing*. La longueur maximale relativement réduite des syntagmes ont amené à l'introduction de cette étape afin de favoriser la formation des syntagmes avec des mots ayant une forte probabilité *a posteriori*. Le CN formé suite à l'étape de *parsing* est alors utilise par Moses pour fournir une hypothèse de traduction.

C.3 Résultats expérimentaux

Les tests, l'apprentissage des modèles et l'optimisation des paramètres ont été réalisés en utilisant les différents corpus décrits dans le tableau B.1. En ce qui concerne le modèle de traduction, la table de traduction et le modèle de langage anglais ont été appris sur le corpus d'apprentissage *App* dénommée *Corpus propre* dans le tableau B.1, qui contient seulement de la parole propre (sans disfluences). Pour le modèle sans classes, la table de traduction a été apprise sur le corpus d'apprentissage à travers la combinaison des phrases "manuelles" et de phrases "automatiques". Le modèle de langage anglais utilisé est un 4-gramme appris sur la traduction en anglais du corpus d'apprentissage. Le modèle avec classes a été appris sur les corpus d'apprentissage annoté en classes à l'aide du critère automatique décrit au B.3. Le modèle de langage anglais correspondant est un 6-gramme.
Les poids des paramètres utilisés par le modèle log-linéaire de Moses (qui donne un score de confiance à chaque hypothèse de traduction grâce à la combinaison des paramètres) ont été estimés à l'aide de la procédure de *Minimum Error Rate Training* (Och, 2003) qui essaie de maximiser le score BLEU sur un corpus de développement. Pour cela nous avons utilisé le corpus *Dev* (cf. tableau B.1). Étant donné que nous utilisons un paramètre supplémentaire pour les CNs (la probabilité *a posteriori* des mots du CN), l'ensemble des poids des paramètres est ajusté sur un ensemble de développement constitué des CNs générés pour les énoncés en français du corpus de développement et des traductions correspondantes annotés en classes.
Les tests on été réalisés sur le corpus de test *Test* (cf. tableau B.1). Le tableau C.1 montre les performances de la traduction sur la transcription de l'énoncé de parole, Verbatim et Corpus propre, sur la *1-best* de SRAP et sur les CNs issus du *parsing* (Nouveau CN). Les performances ont été mesurées en termes de *WER*, *PER* (*Position-independent Error Rate* - Taux d'erreur mot indépendant de la position dans la phrase) et score BLEU ; toutes les valeurs dans le tableau sont exprimées en pourcentage. Pour l'évaluation, nous disposons d'une seule traduction de référence en anglais, ce qui est une limitation pour une évaluation plus pertinente d'un système de traduction automatique.
Les résultats montrent tout d'abord une différence importante entre les performances obtenues avec la transcription et celles de la *1-best*. L'utilisation des transcriptions pro-

Corpus	Modèle	*WER*	*PER*	*BLEU*
Corpus propre	sans classes	41.4	33.4	45.3
	avec classes	35.6	28.7	52.5
Verbatim	sans classes	68.9	60.5	30.8
	avec classes	62.6	54.0	35.3
1-best	sans classes	81.6	72.9	17.2
	avec classes	75.6	67.0	20.0
Nouveaux CNs	sans classes	69.2	63.3	16.5
	avec classes	68.8	62.6	17.8

TABLE *C.1* – *Résultats de la traduction*

duit tout naturellement de meilleures résultats, mais même en utilisant la transcription Verbatim les performances ne sont pas optimales, comme le montre la valeur de *PER* de 60.5%. Ceci confirme la complexité de la traduction automatique dans un contexte de parole spontanée.
On observe un impact important de l'utilisation des modèles avec des classes dans les performances globales de la traduction sur le corpus de test (*Corpus propre* et*Verbatim*). L'introduction des classes permet d'améliorer les résultats. On observe également un impact positif de l'utilisation des classes sur les performances obtenues sur les hypothèses de reconnaissance (*1-best* et CN).
L'utilisation des CNs et de l'algorithme de *parsing* permet d'améliorer les performances en termes de *WER* et *PER* par rapport à la *1-best*. Les *WER* et *PER* des CNs sont plus proches des performances obtenues avec la transcription Verbatim. Néanmoins, le taux d'omission est plus élevé sur les CNs, ce qui conduit à des hypothèses de traduction plus courtes et pénalise le score BLEU.

Nous pensons que les résultats obtenus avec les réseaux de confusion peuvent être améliorés en utilisant un volume de données plus important. Un corpus d'apprentissage plus grand permettrait d'obtenir des modèles de traduction à base de classes *a priori* plus fiables ce qui pourrait améliorer les performances des CNs avec le modèle à base de classes. La calibration des paramètres utilisés par Moses pour réaliser la traduction des CNs est une étape très importante qui influe beaucoup sur les performances globales. Nous avons observé que le moindre changement d'un paramètre produit des changements importants dans le comportement de Moses et par conséquence dans les hypothèses de traduction fournies. Pour cela nous pensons qu'un corpus de développement important est nécessaire afin d'obtenir de performances optimales avec Moses. Malgré ces observations, nous avons montré que l'algorithme de génération *multi-niveaux* est facilement adaptable à différents domaines d'application de même que l'algorithme de *parsing*.

Liste des illustrations

Liste des tableaux

Bibliographie

(Bahl et al., 1983) L. Bahl, F. Jelinek, et L. Mercer, 1983. A maximum likelihood to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 179–190.

(Baum, 1972) L. Baum, 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of markov processes,. *Inequalities 3*, 1–8.

(Berger et al., 1996) A. Berger, S. Della Pietra, et V. Della Pietra, 1996. A maximum entropy approach to natural language processing. *Computational linguistics 22*(1), 39–71.

(Bertoldi et al., 2007) N. Bertoldi, R. Zens, et M. Federico, 2007. Speech translation by confusion network decoding. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Honolulu, Hawaii, USA.

(Breiman, 2001) L. Breiman, 2001. Random forests. *Machine Learning 45 (1)*, 5–32.

(Brown et al., 1990) P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, et P. Roossin, 1990. A statistical approach to machine translation. *Computational linguistics 16*, 79–85.

(Brown et al., 1993) P. Brown, S. Della Pietra, V. Della Pietra, et R. Mercer, 1993. The mathematics ofstatistical machine translation : Parameter estimation. *Computational linguistics 19*, 263–311.

(Burges, 1998) C. Burges, 1998. A tutorial on support vector machine for pattern recognition. *Data Mining and Knowledge Discovery 2*(2).

(Charlet et al., 2001) D. Charlet, G. Mercier, et D. Jouvet, 2001. On combining confidence measures for improved rejection of incorrect data. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Aalborg, Danemark.

(Chen et Goodman, 1996) S. Chen et J. Goodman, 1996. An empirical study of smoothing techniques for language modeling. Dans les actes de *A. Joshi M.P., editor, Thirty-fourth Annual Meeting of the Association for Computational Linguistics*, Morgan Kaufmann Publichers, San Francisco, USA, 310–318.

(Chomsky, 1957) N. Chomsky, 1957. *Syntactic Structures*. The Hague : Mouton.

(Chomsky, 1965) N. Chomsky, 1965. *Aspects of ht theory of syntax*. Cambridge : MIT Press.

(Cox et Dasmahapatra, 2002) S. Cox et S. Dasmahapatra, 2002. High-level approaches to confidence estimation in speech recognition. *IEEE Transactions on Speech and Audio Processing 10*(7).

(Cox et Rose, 1996) S. Cox et R. Rose, 1996. Confidence measures for the switchboard database. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, 511–514.

(Damnati, 2000) G. Damnati, 2000. *Modèles de langage et classification automatique pour la reconnaissance de la parole continue dans un contexte de dialogue homme-machine*. Thèse de Doctorat, Université d'Avignon et des Pays de Vaucluse.

(Damnati et al., 2007) G. Damnati, F. Béchet, et R. De Mori, 2007. Spoken language understanding strategies on the france telecom 3000 voice agency corpus. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 4, Honolulu, USA, 9–12.

(Davis et Mermelstein, 1980) S. Davis et P. Mermelstein, 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. Dans les actes de *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Volume 28, 357–366.

(Dempster et al., 1977) A. Dempster, N. Laird, et D. Rubin, 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*(39), 1–38.

(Duda et Hart, 1973) R. Duda et P. Hart, 1973. *Pattern classification and scene analysis*. New York.

(Estève et al., 2003) Y. Estève, C. Raymond, R. De Mori, et D. Janiszek, 2003. On the use of linguistic consistency in systems for human-computer dialogues. *IEEE Transactions on Speech and Audio Processing 11*(6), 746–756.

(Evermann et Woodland, 2000a) G. Evermann et P. Woodland, 2000a. Large vocabulary decoding and confidence estimation using wordposterior probabilities. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 3, Istambul, Turquie, 1655–1658.

(Evermann et Woodland, 2000b) G. Evermann et P. Woodland, 2000b. Posterior probability decoding, confidence estimation and system combination. Dans les actes de *Proc. of the NIST Speech Transcription Workshop*, Washington DC, USA.

(Fu et Du, 2005) Y. Fu et L. Du, 2005. Combination of multiple predictors to improve confidence measure based on local posterior probabilities. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, Philadelphi, USA, 93–96.

(Furui, 1986) S. Furui, 1986. Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing 34*(1), 52–59.

(Gillick et al., 1997) L. Gillick, Y. Ito, et J. Young, 1997. A probabilistic approach to confidence estimation and evaluation. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Munich, Allemagne, 879–883.

(Goel et al., 1998) V. Goel, W. Byrne, et S. Khundanpur, 1998. Lvcsr rescoring with modified loss functions : a decision theoretic perspective. Volume 1, Seatle, USA, 425–428.

(Goel et al., 2001) V. Goel, S. Kumar, et W. Byrne, 2001. Confidence based lattice segmentation and minimum bayes-risk decoding. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Aalborg, Denmark.

(Gorin et al., 1997) A. Gorin, G. Riccardi, et J. Wright, 1997. How may i help you ? *Speech Communication Journal*, 113–127.

(Guo et al., 2004) G. Guo, C. Huang, H. Jiang, et R.-H. Wang, 2004. A comparative study on various confidence measures in large vocabulary speech recognition. Dans les actes de *Proc. of the fourth International Symposium on Chinese Language Processing*,, Hong Kong, Chine.

(Hakkani-Tur et al., 2006) D. Hakkani-Tur, F. Béchet, G. Riccardi, et G. Tur, 2006. Beyond asr 1-best : Using word confusion networks for spoken language understanding. *CSL, Computer Speech and Language 20*(4), 495–514.

(Hakkani-Tur et Riccardi, 2003) D. Hakkani-Tur et G. Riccardi, 2003. A general algorithm for word matrix decomposition. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, 596–599.

(Hamimed et Damnati, 2002) H. Hamimed et G. Damnati, 2002. Integration of phonetic length properties in the acoustic models of false starts and out-of-vocabulary words. Dans les actes de *ICSLP, International Conference on Spoken Language Processing*, Denver, USA, 865–868.

(Hazen et al., 2002) T. Hazen, S. Seneff, et J. Polifroni, 2002. Recognition confidence scoring and its use in speech understanding system,. *CSL, Computer Speech and Language 16*(1), 49–67.

(Jelinek, 1997) F. Jelinek, 1997. *Statistical Methods for Speech Recognition*.

(Jiang, 2005) H. Jiang, 2005. Confidence measures for speech recognition : a survey. *Speech Communication Journal 45*, 455–470.

(Jouvet, 1988) D. Jouvet, 1988. *Reconnaissance de mots connectés indépendamment du locuteur par des méthodes statistiques*. Thèse de Doctorat, Ecole Nationale Supérieure des Télécommunications.

(Kampari et Hazen, 2000) S. Kampari et T. Hazen, 2000. Word and phone level acosutic confidence scoring. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, 1799–1802.

(Katz, 1987) S. Katz, 1987. Estimation of probabilities from sparse data for the language model component of speech recognizer. *IEEE Transactions on Acoustic, Speech and Signal Processing 35*.

(Kemp et Schaaf, 1997) T. Kemp et T. Schaaf, 1997. Estimating confidence using word lattice. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Rhodes,Grèce.

(Ketabdar et al., 2006) H. Ketabdar, J. Vepa, S. Bengio, et H. Bourlard, 2006. Using more informative posterior probabilities for speech recognition. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, Toulouse, France, 29–32.

(Kobus, 2006) C. Kobus, 2006. *Exploitation d'informations d'ordre sémantique pour la reconnaissance vocale dans le contexte d'une application de compréhension de la parole*. Thèse de Doctorat, Université d'Avignon et des Pays de Vaucluse.

(Koehn et Marcu, 2003) O. F. J. Koehn, P. et D. Marcu, 2003. Statistical phrase-based translation. Dans les actes de *HLT, Human Language Technology Conference*, Edmonton, Canada, 127–133.

(Koehn,) P. Koehn. Moses : Open source toolkit for statistical machine translation. Dans les actes de *Annual meeting of Association for Computational Linguistics*, 177–180.

(Lee, 2001) C.-H. Lee, 2001. Statistical confidence measures and their applications. Dans les actes de *Proc. of ICSP*, Daejon, Corée du Sud, 1021–1028.

(Levensthein, 1966) V. I. Levensthein, 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys.-Dokl. 10*, 707–710.

(Makhoul, 1975) J. Makhoul, 1975. Linear prediction : A tutorial review. Dans les actes de *IEEE*, Volume 63, 561–580.

(Mami et Damnati, 2006) Y. Mami et G. Damnati, 2006. On the use of multigram models for automatic translation. *TC-STAR Workshop*.

(Mangu, 2000) L. Mangu, 2000. *Finding consensus in speech recognition*. Thèse de Doctorat, John Hopkins University.

(Mangu et al., 1999) L. Mangu, E. Brill, et A. Stolcke, 1999. Finding consensus among words : lattice-based word error minimisation. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Volume 1, Budapest, Hongrie, 495–498.

(Mangu et al., 2000) L. Mangu, E. Brill, et A. Stolcke, 2000. Finding consensus in speech recognition : Word error minimization and other applications of confusion networks. *CSL, Computer Speech and Language 14*(4), 373–400.

(Martin et al., 1997) A. Martin, T. Doddington, G.and Kamm, M. Ordowski, et M. Przybocki, 1997. The det curve in assessment of detection task performance. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Rhodes, Greece, 1895–1898.

(Metze et al., 2000) F. Metze, T. Kemp, T. Schaaf, T. Schultz, et H. Soltau, 2000. Confidence measure based language identification. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Istambul, Turkie.

(Minescu et Damnati, 2007) B. Minescu et G. Damnati, 2007. Amélioration des performances des confusion networks dans le contexte d'une application du dialogue en langage naturel. Dans les actes de *RJCP, Rencontre des Jeunes Chercheurs en Parole*, Paris, France, 104–107.

(Minescu et Damnati, 2008) B. Minescu et G. Damnati, 2008. Construction et exploitation des réseaux de confusion dans le contexte d'une application de dialogue en langage naturel. Dans les actes de *JEP, Journées d'Étude sur la Parole*, Avignon, France.

(Minescu et al., 2007) B. Minescu, G. Damnati, F. Bechet, et R. De Mori, 2007. Conditional use of word lattices, confusion networks and 1-best string hypotheses in a sequential interpretation strategy. Dans les actes de *Interspeech, International Conference on Speech Communication and Technology*, Anvers, Belgique.

(Minker et Bennacef, 1996) W. Minker et S. Bennacef, 1996. Compréhension et évaluation dans le domaine atis. Dans les actes de *JEP, Journées d'Étude sur la Parole*.

(Mohri, 1996) M. Mohri, 1996. On some applications of finite-state automata theory to natural langauge processing. *Natural Language Engeneering 2*, 1–20.

(Mohri, 1997) M. Mohri, 1997. Finite state transducers in language and speech processing. *Computational Linguistics 23*.

(Mohri et al., 1998) M. Mohri, F. N. Pereira, et M. Riley, 1998. A rational design for a weighted finite-state transducer library. *Lecture Notes in Computer Science 1463*.

(Mohri et al., 2002) M. Mohri, F. N. Pereira, et M. Riley, 2002. Weighted finite-state transducers in speech recognition. *CSL, Computer Speech and Language 16*(1), 69–88.

(Moreno et al., 2001) P. Moreno, B. Logan, et B. Raj, 2001. A boosting approach for confidence scoring. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Aalborg, Danemark, 2109–2112.

(Ney et Essen, 1991) H. Ney et U. Essen, 1991. On smoothing techniques for bigram-based natural language modelling. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 12, Toronto, Canada, 825–828.

(Ney et Ortmanns, 2000) H. Ney et S. Ortmanns, 2000. Progress in dynamic programming search for lvcsr. *Proceedings of the IEEE 88*(8), 1224–1240.

(Ney et Ortmans, 1997) H. Ney et S. Ortmans, 1997. Extentions to the word graph method for large vocabulary continuous speech recognition. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, 1787–1790.

(Och, 2003) F. Och, 2003. Minimum error rate training for statistical machine translation. Dans les actes de *41st Annual Meeting of ACL*, Sapporo, Japon, 160–167.

(Ortmanns et Ney, 1997) S. Ortmanns et H. Ney, 1997. A word graph algorithm for large vocabulary continuous speech recognition. *CSL, Computer Speech and Language 11*(1), 43–72.

(Palmer et Ostendorf, 2001) D. D. Palmer et M. Ostendorf, 2001. Improved word confidence estimation using long range features. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*.

(Pao et al., 1998) C. Pao, P. Schmid, et J. Glass, 1998. Confidence scoring for speech understanding systems. Dans les actes de *ICSLP, International Conference on Spoken Language Processing*.

(Pereira et Riley, 1997) F. N. Pereira et M. Riley, 1997. Speech recognition by composition of weighted finite automata. *E. Roche and Y. Schabes, editors, Finite-State Language Processing. MIT Press*.

(Pieraccini et Levin, 1993) R. Pieraccini et E. Levin, 1993. A learning approach to natural language understanding. *NATO-ASI, New Advances& Trends in Speech Recognition and Coding 1*, 261–279.

(Preti et al., 2007) A. Preti, J.-F. Bonastre, F. Capman, C. Matrouf, et B. Ravera, 2007. Confidence measure based unsupervised target model adaptation for speaker verification. Dans les actes de *Interspeech, International Conference on Speech Communication and Technology*, Anvers, Belgique.

(Pullum et Gazdar, 1982) G. Pullum et G. Gazdar, 1982. Natural language and context free gramars. *Linguistics and Philosophy 4*, 471–504.

(Rabiner, 1989) L. Rabiner, 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE 7*, 257–286.

(Rabiner et Juang, 1993) L. Rabiner et B.-H. Juang, 1993. *Fundamentals of Speech Recognition*. Prentice Hall.

(Raymond, 2005) C. Raymond, 2005. *Décodage conceptuel : co-articulation des processus de transcription et compréhension dans les systèmes de dialogue*. Thèse de Doctorat, Université d'Avignon et des Pays de Vaucluse.

(Raymond et al., 2004) C. Raymond, F. Béchet, R. De Mori, G. Damnati, et Y. Estève, 2004. Automatic learning of interpretation strategies for spoken dialogue systems. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, Montréal, Canada, 425–428.

(Raymond et al., 2006) C. Raymond, F. Bechet, R. De Mori, et G. Damnati, 2006. On the use of finite state transducers for semantic interprétation. *Speech Communication 48*, 288–304.

(Riccardi et Gorin, 1998) G. Riccardi et A. Gorin, 1998. Stochastc language models for speech recognition and understanding. Dans les actes de *ICSLP, International Conference on Spoken Language Processing*, Sydney, Australie, 2087–2090.

(Roark, 2002) B. Roark, 2002. Markov parsing :lattice rescoring with a statistical parser. Dans les actes de *Proceedings of the 40th ACL meeting*, Philadelphia, USA.

(Rueber, 1997) B. Rueber, 1997. Obtaining confidence measure from sentence probabilities. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Rhodes, Grece, 739–742.

(San-Segundo et al., 2001) R. San-Segundo, B. Pellom, K. Hacioglu, W. Ward, et J. Pardo, 2001. Confidence measures for spoken dialogue systems. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, Salt Lake City, USA, 393–396.

(Schaaf et Kemp, 1997) T. Schaaf et T. Kemp, 1997. Confidence measures for spontaneous speech recognition. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Munich, Allemagne, 875–878.

(Senellart et al., 2001) J. Senellart, P. Dienes, et T. Varadi, 2001. New generation systran translations system. Dans les actes de *MT Summit*, Santiago de Compostela, Espagne.

(Siu et Gish, 1999) M. Siu et H. Gish, 1999. Evaluation of word confidence for speech recognition systems. *CSL, Computer Speech and Language 13*(4), 299–319.

(Siu et al., 1997) M. Siu, H. Gish, et F. Richardson, 1997. Improved estimation, evaluation and applications of confidence measures for speech recognition. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Volume 2, Rhodes, Grèce, 831–834.

(Somers, 1999) H. Somers, 1999. Review article : Exemple-based machine translation. *Machine Translation 14*, 113–157.

(Soong et al., 2004) F. Soong, W. Lo, et S. Nakamura, 2004. Generalized word posterior probability (gwpp) for measuring reliability of recognized words. Dans les actes de *SWIM*.

(Sorin et De Mori, 1998) C. Sorin et R. De Mori, 1998. Sentence generation. Spoken Dialogues with Computers, 563–582.

(Stolcke et al., 1997) A. Stolcke, Y. Konig, et M. Weintraub, 1997. Explicit word error minimization in nbest list rescoring. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, Rhodes, Grèce, 163–166.

(Uhrik et Ward, 1997) C. Uhrik et W. Ward, 1997. Confidence metrics based on n-gram language model backoff behaviors. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Rhodes, Grece.

(Vergyri, 2000) D. Vergyri, 2000. Use of word level side information to improve speech recognition. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 3, Istambul, Turquie, 1823–1826.

(Wessel et al., 1999) F. Wessel, K. Macherey, et H. Ney, 1999. A comparison of word graph and n-best list based confidence measures. Dans les actes de *Eurospeech, European Conference on Speech Communication and Technology*, Budapest, Hongrie, 315–318.

(Wessel et al., 1998) F. Wessel, K. Macherey, et R. Schlüter, 1998. Using word probabilities as confidence measures. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Seatle, USA, 225–228.

(Wessel et Ney, 2001) F. Wessel et H. Ney, 2001. Unsupervised training of acoustic models for large vocabulary continuous speech recognition. Dans les actes de *IEEE Automatic Speech Recognition and Understanding Workshop*, Trento, Italie.

(Wessel et Ney, 2005) F. Wessel et H. Ney, 2005. Unsupervised training of acoustic models for large vocabulary continous speech recognition. *IEEE Transactions on Speech and Audio Processing 13*, 23–31.

(Wessel et al., 2001) F. Wessel, R. Schlüter, K. Macherey, et H. Ney, 2001. Confidence measures for large vocabulary continuous speech recognition. *9*(3), 288–298.

(Wessel et al., 2000) F. Wessel, R. Schlüter, et H. Ney, 2000. Using posterior probabilities for improved speech recognition,. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Istambul, Turquie, 1587–1590.

(Xue et Zhao, 2006) J. Xue et Y. Zhao, 2006. Random forests-based confidence annotation using novel features from confusion network. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, Toulouse, France, 1149–1152.

(Young, 1994) S. Young, 1994. Detecting misrecognitions and out-of-vocabulary words. Dans les actes de *ICASSP, International Conference on Acoustics, Speech, and Signal Processing*, II–21–II–24.

(Young et al., 1997) S. Young, M. Adda-Decker, X. Aubert, C. Dugast, J. Gauvain, D. Kershaw, L. Lamel, D. Leeuwen, D. Pye, A. Robinson, H. Steeneken, et P. Woodland, 1997. Multilingual large vocabulary speech recognition : the european sqale project. *11*, 73–89.

(Zhang et Rudnicky, 2001) R. Zhang et A. I. Rudnicky, 2001. Word level confidence annotation using combinations of features. *Eurospeech, European Conference on Speech Communication and Technology*.