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

THÈSE

Présentée pour obtenir le grade de Docteur en Sciences
de l'Université d'Avignon et des Pays de Vaucluse

SPÉCIALITÉ : INFORMATIQUE

Décodage conceptuel : co-articulation des processus de transcription et compréhension dans les systèmes de dialogue

par

CHRISTIAN RAYMOND

Soutenue publiquement le 8 décembre 2005 devant un jury composé de :

M.	Jean-Paul HATON	Professeur, LORIA, Nancy	Rapporteur
M.	Giuseppe RICCARDI	Professeur, Université, Trento	Rapporteur
Mme	Géraldine DAMNATI	Ingénieur, France Télécom R&D, Lannion	Examinateur
M.	Marc EL-BÈZE	Professeur, LIA, Avignon	Examinateur
M.	Renato DE MORI	Professeur, LIA, Avignon	Directeur de thèse
M.	Frédéric BÉCHET	Maître de Conférence, LIA, Avignon	Co-directeur de thèse

École Doctorale Science et Agronomie
Laboratoire Informatique d'Avignon

Résumé

Les systèmes de dialogue oral homme-machine fournissent des services (consultation d'horaires d'avions ou de trains, consultation de la météo, recherche de restaurants, ...) à des utilisateurs, tout en leur offrant la possibilité de s'exprimer en langage naturel. Ces systèmes sont couplés avec une base de données en relation avec le service fourni. La difficulté principale est de comprendre le sens des paroles de l'utilisateur. Pour effectuer cette tâche, la plupart des systèmes de ce type font appel à un module de reconnaissance de la parole permettant de transformer le signal vocal en version textuelle. Cette transcription est ensuite analysée de manière à extraire les informations sémantiques indispensables au système pour répondre aux attentes de l'utilisateur. Dans cette architecture séquentielle de traitement des tâches, la qualité de l'interprétation sémantique est très dépendante de la qualité du processus de reconnaissance automatique de la parole. Ce module utilise généralement des informations acoustiques pour convertir le signal en unités linguistiques de base (phonèmes, syllabes ou mots) et des informations linguistiques à portée réduite (N-grammes). Le module de compréhension s'appuie sur des unités sémantiques élémentaires, que l'on appelle concepts, qui sont ensuite composées pour obtenir une représentation sémantique.

Alternativement à l'approche séquentielle des deux processus de transcription et compréhension, nous proposons un modèle basé sur le formalisme des transducteurs à états fini qui met en relation les mots avec les concepts qu'ils représentent. Ce modèle permet d'enrichir un graphe de mots avec des informations conceptuelles. En considérant une interprétation comme étant une séquence de concepts avec leurs valeurs, le processus de décodage proposé permet de fournir une liste structurée des N-meilleures hypothèses d'interprétation de l'énoncé. Cette liste permet d'obtenir en quelques hypothèses, un résumé du graphe de mots, exhaustif et non-redondant du point de vue de la compréhension.

Afin de pallier les inévitables erreurs du processus de reconnaissance, nous présentons ensuite des mesures de confiance utiles pour diagnostiquer la qualité d'une interprétation. Ces mesures de confiance sont basées sur des connaissances acoustiques, linguistiques et sémantiques. Elles opèrent sur différents niveaux : mot, concept, phrase, *etc.*

Dans la dernière partie, nous proposons une stratégie d'aide à la décision pour le gestionnaire de dialogue. Cette stratégie s'appuie sur des unités de décision prenant en entrée la liste structurée des N-meilleures hypothèses d'interprétation ainsi que les mesures de confiance présentées. En sortie, chaque hypothèse est associée avec un état de fiabilité. Selon l'état et ses caractéristiques, des stratégies de correction d'erreurs adaptées sont proposées.

Abstract

In spoken dialog systems, the process of understanding consists in building a semantic representation from some elementary semantic units called concept.

We propose in this document a SLU (Spoken Language Understanding) module. First we introduces a conceptual language model for the detection and the extraction of semantic basic concepts from a speech signal. A decoding process is described with a simple example. This decoding process extracts, from a word lattice generated by an Automatic Speech Recognition (ASR) module, a structured n-best list of interpretations (set of concepts). This list contains *all* the interpretations that can be found in the word lattice, with their posterior probabilities, and the n-best values for each interpretation.

Then we introduces some confidence measures used to estimate the quality of the result of the previous decoding process.

Finally, we describes the integration of the proposed SLU module in a dialogue application, involving a decision strategy based on the confidence measures introduced before.

Remerciements

Je tiens à remercier Messieurs Giuseppe RICCARDI et Jean-Paul HATON qui m'ont fait l'honneur d'avoir été les rapporteurs de ce mémoire. Je remercie également Monsieur Marc EL-BÈZE qui en plus de sa participation au jury a contribué à la correction et l'amélioration de ce document.

J'ai été extrêmement heureux d'avoir Frédéric BÉCHET et Renato DE MORI comme directeurs durant cette thèse. Leur encadrement exemplaire et particulièrement complémentaire a été une chance pour moi. Je les remercie de s'être impliqués autant dans ce travail et m'avoir fait profiter de leur expérience.

Je remercie France Télécom Recherche et Développement qui nous a fourni le matériau nécessaire à cette étude et à toutes les personnes que j'ai eu la chance de rencontrer lors de mes brefs séjours à Lannion. Un merci particulier à Géraldine DAMNATI pour sa participation au jury et pour son amitié.

De façon générale, je remercie tous les membres du personnel du Laboratoire Informatique d'Avignon. Plus particulièrement, je pense à Yannick ESTÈVE avec qui j'ai passé deux années de thèse, ainsi que Mireille PALPANT et Jens GRIVOLLA. Merci également à Nathalie CAMELIN pour son amitié et son investissement dans l'organisation de la soirée de thèse.

Merci à mes parents et à ma sœur ...

Je pourrais également remercier le sort qui a permis à l'Olympique de Marseille et l'Inter de Milan de ne pas se rencontrer sur la scène européenne et ainsi préserver la relation amicale que j'ai avec Renato.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

Introduction

Depuis très longtemps, converser avec une machine est chose possible... dans les films de science-fiction. Mais ceux qui travaillent à l'élaboration des méthodes qui rendent ce genre d'applications possibles ou les utilisateurs de ces systèmes se rendent compte que le jour où l'on pourra tenir une conversation ouverte avec un ordinateur n'est pas encore arrivé. Les méthodes actuelles comme la reconnaissance automatique de la parole (*i.e.* conversion d'un signal de parole en texte) et l'analyse en compréhension (*i.e.* extraction du sens à partir du texte) rendent possible un dialogue homme-machine, dans des circonstances bien particulières : milieu non bruité, vocabulaire réduit et surtout une sémantique restreinte.

Objet de la thèse

La reconnaissance automatique de la parole utilise le plus souvent une approche statistique ; qu'il s'agisse d'applications de dictée vocale ou de systèmes de dialogue. Mais dans le cas de dictée vocale, le langage parlé est similaire au langage écrit, les modèles de langage statistiques N-grammes utilisés peuvent donc être construits sur la base de corpus de taille conséquente (*e.g.* corpus journalistiques). Le langage utilisé dans des applications de dialogue est soumis aux caractéristiques du langage parlé naturel, c'est à dire la présence d'hésitations, de reprises, de fautes grammaticales, ainsi qu'à un vocabulaire bien spécifique à l'application. Les modèles de langage pour ce genre d'applications doivent être établis sur la base de corpus spécialisés qui sont longs et coûteux à construire. Il en résulte des taux d'erreurs de reconnaissance assez élevés. Dans les applications de dialogue, l'objectif n'est pas de transcrire mais de comprendre le message porté par le signal. La transcription n'est qu'une étape intermédiaire nécessaire. La compréhension d'un message se fait par l'analyse de cette transcription, généralement par des grammaires sémantiques modélisant des relations entre les concepts élémentaires présents dans la phrase. Ces concepts élémentaires sont des mots ou des séquences de mots ayant un sens pour le système (*e.g.* lieu, date, prix, *etc.*). Or la transcription est effectuée à l'aide de modèles acoustiques et linguistiques à contraintes réduites (N-grammes), ceci entraîne que le processus de reconnaissance peut générer des phrases hors-domaine. Si la transcription est utilisée pour établir la compréhension

du message, la compréhension peut être utilisée pour guider le processus de transcription vers des phrases ayant un sens vis à vis du système. Les systèmes de dialogue auxquels nous nous intéressons sont ceux, tels les serveurs vocaux, fonctionnant sur une tâche finalisée dans un domaine particulier. Dans ces systèmes, le langage est limité au domaine de l'application et la sémantique est définie et restreinte. Afin de tenter d'améliorer la qualité de la transcription et de se concentrer sur les zones porteuses de sens, nous proposons dans le chapitre 5 un modèle de langage de niveau conceptuel assurant la correspondance mots/concept, permettant d'enrichir l'espace de recherche de la meilleure transcription par des informations utiles à la compréhension. Un processus de décodage y est présenté qui aboutit à une liste structurée des N-meilleures interprétations possibles (*i.e.* ensemble de concepts) associées à leur meilleure transcription qui ne sont pas redondantes pour le système du point de vue du sens exprimé.
Il est primordial dans les systèmes de dialogue, à cause des erreurs fréquentes de reconnaissance, de pouvoir diagnostiquer la qualité de cette reconnaissance afin de ne pas orienter le dialogue dans un mauvais sens et d'éviter le mécontentement de l'utilisateur. Nous proposons dans le chapitre 6 différentes mesures de confiance applicables sur la sortie de reconnaissance. Ces mesures faisant appel à différentes sources de connaissances, linguistiques, acoustiques ou sémantiques, permettent de diagnostiquer la sortie du module de RAP à différents niveaux : mot, concept et phrase.
Dans le chapitre 7 nous proposons une stratégie de validation de notre sortie de décodage (*i.e.* notre liste structurée) basée sur des consensus de classifieurs automatiques entraînés sur les différentes mesures de confiance présentées. Cette stratégie permet d'isoler des situations de confiance permettant de guider le gestionnaire de dialogue dans les choix à effectuer pour la gestion du dialogue.

Organisation du document

Ce document est divisé en deux grandes parties.
La première partie propose un survol des notions qui gravitent autour des systèmes de dialogue oraux. Après avoir présenté brièvement le fonctionnement de tels systèmes :

- le premier chapitre présente le fonctionnement général d'un système de dialogue pour produire la transcription du signal de parole ;
- le deuxième chapitre donne un aperçu du processus de compréhension de ce type de système basé sur l'analyse de la transcription.
- le chapitre trois, présente les outils qui seront utilisés dans cette thèse pour mener nos travaux : les méthodes formelles de description d'un langage, les machines d'analyse associées que sont les automates à états fini. Seront présentées également des méthodes de classifications textuelles utiles pour extraire des unités de sens à partir d'un texte.

La seconde partie du document concerne les travaux réalisés durant cette thèse :

- le chapitre 5 détaille l'implémentation d'un modèle de langage conceptuel sous la forme d'un transducteur à états fini qui permet d'inclure dans le processus de transcription des informations liées à la compréhension, ainsi que son utilisation lors d'un processus de décodage qui abouti à une liste structurée des N-meilleures hypothèses mots/interprétation qui existent dans le signal de parole ;
- le chapitre 6 propose un panel de mesures de confiance faisant intervenir plusieurs sources de connaissances, acoustique, linguistique et sémantique permettant d'estimer la qualité du processus de reconnaissance.
- nous proposons dans le chapitre 7 une stratégie basée sur les mesures présentées dans le chapitre précédent. Cette stratégie basée sur l'utilisation redondantes

de classifieurs permet de déterminer des situations de confiance en fonction desquelles le gestionnaire de dialogue peut determiner le choix à effectuer pour la suite du dialogue.

Les travaux présentés dans cette thèse, illustrations, expériences et résultats, sont en rapport avec des applications de dialogue oral homme-machine concrètes créées par France Télécom Recherche & Développement.

Première partie

Systèmes de dialogue : principes généraux

Introduction

Un système de dialogue oral, est un système informatique qui répond à un service. Le demandeur du service est un utilisateur humain, et le système doit interagir avec cet utilisateur comme aurait pu le faire un être humain ou du moins le plus naturellement possible. La communication est orale. Le système doit alors comprendre le sens des paroles de l'utilisateur, trouver une réponse et lui communiquer oralement. La majorité des systèmes de dialogue de ce type gère ce processus de manière séquentielle en enchaînant l'appel à des modules spécialisés dans le traitement des diverses tâches nécessaires pour la réalisation de ce processus. La figure 1 illustre le traitement opéré par un système de dialogue. Dans ce traitement, on peut comptabiliser 4 modules :

1. LE MODULE DE TRANSCRIPTION : le processus de compréhension de la parole de l'être humain est complexe. Les systèmes informatiques qui tentent de le reproduire font ce processus en 2 étapes. Ils tentent tout d'abord de générer une transcription et ensuite effectue une analyse textuelle de cette transcription pour en trouver un sens. Le module de transcription ou de reconnaissance de la parole (RAP), assure le passage du signal de parole à une version textuelle du message utilisateur. Le fonctionnement de ce module sera décrit plus en détail dans le chapitre 1 ;
2. LE MODULE DE COMPRÉHENSION : il se base sur la transcription générée par le module précédent pour trouver un sens aux paroles de l'utilisateur. Il doit alors mettre en œuvre des méthodes permettant de passer des mots au sens. Dans un premier temps, il fait une association mots/concepts et à partir des concepts, ainsi que de la connaissance du dialogue en cours construit une représentation sémantique qui sera exploitée par le gestionnaire de dialogue. Son fonctionnement sera détaillé dans le chapitre 2 ;
3. LE GESTIONNAIRE DE DIALOGUE : il est chargé d'assurer le bon déroulement de la conversation. Il se base sur les interprétations sémantiques fournies par le module de compréhension et l'historique du dialogue pour prendre les décisions sur l'action à entreprendre. L'action peut être d'interroger une base de données si l'utilisateur a émis une requête, ou bien de demander à l'utilisateur de répéter si le système ne comprend pas l'énoncé de l'utilisateur. Le travail présenté dans cette thèse ne s'interesse pas à ce niveau du traitement, pour plus de détails voir par

exemple [Sadek et De Mori, 1998] ;

4. LE MODULE DE SYNTHÈSE DE PAROLE : le synthétiseur de parole doit transformer la réponse textuelle générée par le gestionnaire de dialogue en signal de parole afin de converser oralement avec l'utilisateur de manière naturelle. Cette partie ne rentre pas dans les considérations de cette thèse, pour plus d'informations voir [Sorin et De Mori, 1998].

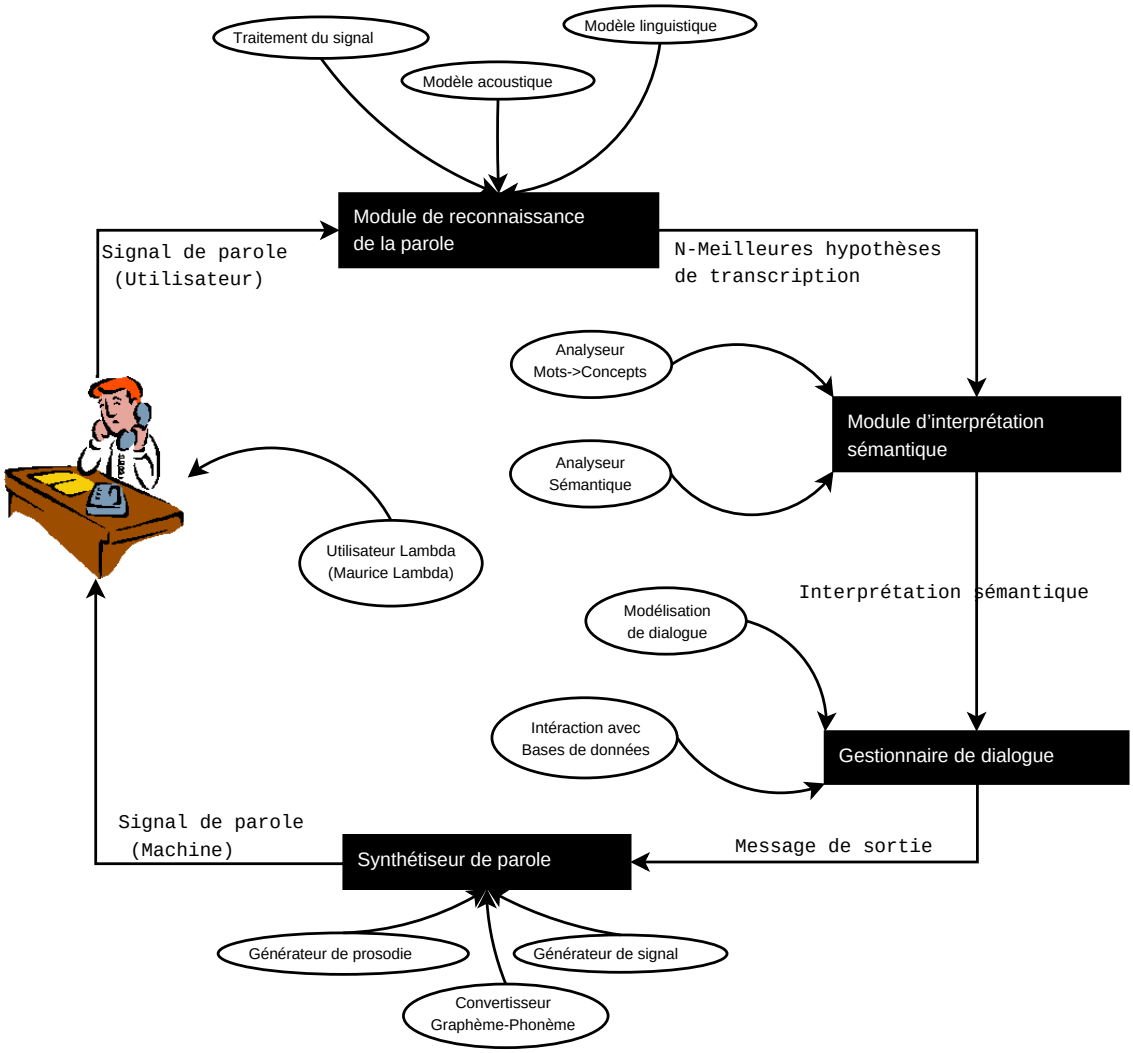

FIG. 1 – *Architecture générale d'un système de dialogue*

CHAPITRE

1

Transcription : modélisation statistique du langage

Sommaire

1.1 Principes généraux

La grande majorité des systèmes de reconnaissance automatique de la parole (RAP) fonctionne sur des principes probabilistes. Pour cela, à partir de la séquence d'observations acoustiques $A = a_1 a_2 \ldots a_m$ extraite du signal de parole, un système de reconnaissance de la parole recherche la séquence de mots $\hat{W} = w_1 w_2 \ldots w_k$ qui maximise la probabilité *a posteriori* $P(W|A)$. Soit la probabilité que la séquence d'observations acoustiques A génère la séquence de mots W. Or cette probabilité n'est pas calculable en l'état. En effet, une même personne prononçant deux fois une phrase, ne génère pas deux signaux

identiques. Il est alors difficile d'imaginer construire un corpus permettant de pouvoir estimer cette probabilité. Le théorème de Bayes permet de reformuler cette probabilité en :

$$P(W|A) = \frac{P(W)P(A|W)}{P(A)} \tag{1.1}$$

La séquence de mot $\hat{W}$ est celle qui maximise le produit de l'équation 1.1, comme noté dans l'équation 1.2.

$$\hat{W} = \underset{W}{\text{ArgMax}}\, P(W|A) = \underset{W}{\text{ArgMax}}\, \frac{P(W)P(A|W)}{P(A)} \tag{1.2}$$

Or, dans ce produit la probabilité *a priori* de la séquence d'observations acoustiques $P(A)$ n'est pas calculable pour les mêmes raisons que ne l'était pas $P(W|A)$. Mais ici la séquence d'observations acoustiques A est identique pour toutes les hypothèses W, $P(A)$ est une valeur constante qu'il est inutile de calculer pour trouver $\hat{W}$. On a donc :

$$\hat{W} = \underset{W}{\text{ArgMax}}\, P(A|W)P(W) \tag{1.3}$$

L'étape de reconnaissance revient a maximiser le produit des 2 termes, $P(A|W)$ et $P(W)$. $P(A|W)$ est la probabilité qu'une séquence de mots W génère la séquence d'observations acoustiques A. Elle est estimée par un modèle acoustique. $P(W)$ est la probabilité *a priori* de la séquence de mots, elle est estimée par un modèle de langage.

1.2 Modélisation acoustique

La fonction du modèle acoustique est d'estimer la première composante de l'équation 1.3. Étant donnée une séquence de vecteurs de paramètres extraits du signal de parole, le but des modèles acoustiques est de calculer la probabilité qu'un événement linguistique particulier (un mot, une phrase, *etc.*) ait généré cette séquence. La plupart des systèmes actuels recourent à l'utilisation des Modèles de Markov Cachés (HMMs). Les unités de base modélisées par ces systèmes sont souvent les *phonèmes*. L'utilisation de cette unité de base donne de bons résultats de reconnaissance, qui peuvent être améliorés en prenant en compte certaines caractéristiques des signaux de paroles. Il est connu que la réalisation d'un phonème est fortement influencée par les phonèmes qui l'entourent. Une description plus réaliste des sons de bases peut être effectuée en dédiant différents modèles au même phonème selon le contexte. Par exemple les modèles *allophones* ou *Tri-phones*. Un modèle de Markov peut être vu comme un automate stochastique. Les états sont associés à une densité de probabilité qui modélise les formes rencontrées $P(X_k|E_i)$ qui est la probabilité de produire l'événement X_k sur l'état E_i. Les transitions assurent les contraintes d'ordre temporel des formes pouvant être observées. $P(E_{i+1}|E_i)$ est la probabilité de transition d'un état à un autre. La figure 1.1 est un exemple de modèle pour les phonèmes. Les HMMs s'appuient sur un corpus d'apprentissage des différentes réalisations des phonèmes de la langue considérée pour les modéliser. Un modèle de mot peut alors être vu comme une concaténation successive de modèles de phonèmes, et la phrase complète comme une concaténation de modèles de mots. Les chemins dans cet automate représentent toutes les chaînes possibles de mots, le chemin ayant la plus forte probabilité est donc celui qui permet l'alignement optimal du signal acoustique sur le modèle de Markov. L'opération appelée *décodage* est celle qui permet de retrouver au mieux ce meilleur chemin. Différents algorithmes existent : Viterbi, le Beam Search, et le A^*.

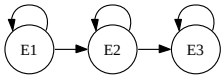

FIG. 1.1 – *Exemple de Modèle de Markov pour un phonème*

1.3 Modélisation statistique du langage

De façon générale, on appelle langage un ensemble de conventions permettant de conceptualiser la pensée et d'échanger avec d'autres interlocuteurs à l'aide d'un support comme la parole, l'écriture, *etc*. L'objectif de la modélisation du langage est de trouver un moyen de le décrire. Deux grandes approches existent : la modélisation basée sur l'utilisation de grammaires formelles mises au point par des experts en linguistique et la modélisation stochastique qui tente de décrire automatiquement un langage à partir de l'observation de corpus. Il est très difficile et long de représenter un langage naturel de manière formelle. Dans le cadre d'applications de dialogue oral autorisant l'utilisateur à s'exprimer librement, la modélisation stochastique du langage est privilégiée.

Les modèles de langage statistiques ont pour objectif d'estimer la seconde composante de l'équation 1.3. En effet, le modèle acoustique n'a pas d'*a priori* sur l'enchaînement entre les mots. Il est relaté dans [Mariani, 1990] qu'une suite de 9 phonèmes peut être transcrite en 32000 suites de mots différentes, dont quelques unes seulement sont syntaxiquement correctes. Les modèles de langage vont permettre de représenter des connaissances d'ordre linguistique afin de guider le décodage vers des hypothèses de phrase cohérentes du point de vue syntaxique ou grammatical. Les modèles de langage statistiques tentent alors de déterminer la probabilité *a priori* de la séquence de mots $W = w_1, w_2, \ldots, w_k$ selon l'équation 1.4.

$$\begin{aligned} P(W_k) &= \prod_{i=1}^{k} P(w_i|h_i) \\ \text{où } h_i &= \{w_1, \ldots, w_{i-1}\} \text{ pour } i > 2 \\ h_i &= \{w_1\} \text{ pour } i = 2 \\ h_i &= \{\varnothing\} \text{ pour } i = 1 \end{aligned} \tag{1.4}$$

Le principal problème dans l'utilisation de modèles de langage probabilistes tient en la longueur de l'historique considéré. En effet la taille des corpus d'apprentissage ne nous permet pas de calculer efficacement la probabilité $P(w_i|w_1, w_2, \ldots, w_{i-1})$. On approche alors la probabilité en fonction d'un historique de taille réduite et fixe. C'est ce que l'on nomme un modèle *N*-grammes. Le calcul considère alors, pour la prédiction d'un mot, que la suite des $N-1$ mots qui le précèdent est suffisante. Un N trop petit modélise mal les contraintes linguistiques tandis qu'un N trop grand va cruellement limiter la couverture du modèle. Les valeurs les plus utilisées sont $N = 2$ pour des modèles de langage appelé *Bi*-grammes et $N = 3$ pour des modèles de langage appelés *Tri*-gramme. Les termes de l'équation 1.4 pour un modèle *Tri*-gramme se résument alors à l'équation 1.5.

$$P(W_k) = P(w_1) \times P(w_2|w_1) \times \prod_{i=3}^{k} P(w_i|w_{i-2}w_{i-1}) \tag{1.5}$$

La probabilité d'apparition d'un mot est généralement estimée par le critère de maxi-

mum de vraisemblance. Pour un modèle *Tri*-gramme et pour un mot w précédé des mots $w_i w_j$ cela donne :

$$P(w|w_i w_j) = \frac{c(w_i w_j w)}{c(w_i w_j)}$$

où $c(w_i w_j w)$ correspond au nombre d'occurrences de la suite de mots $w_i w_j w$ dans le corpus d'apprentissage et $c(w_i w_j)$ au nombre d'occurrences de la suite de mots $w_i w_j$.

Le problème de cette modélisation est que les événements non-observés dans le corpus d'apprentissage du modèle ont une probabilité nulle. Afin de pallier le problème, plusieurs approches ont été développées pour pouvoir modéliser les événements qui n'ont pas été rencontrés lors de la phase d'apprentissage. Certaines utilisent des connaissances sur le langage pour générer des événements manquants, comme les modèles à base de classes (section 1.4), d'autres sont des techniques de lissage (section 1.5) dont les plus connues sont basées sur des méthodes de repli (back-off en anglais) sur des modèles N-grammes d'ordre inférieur.

1.4 Modèle *N*-grammes à base de classes

La quantité de données nécessaire à l'apprentissage d'un modèle de langage robuste et performant, malgré l'utilisation de l'approximation N-grammes, reste importante. En partant du constat que certains mots ont un comportement similaire, il est possible de les regrouper en classes :

- le nombre d'événements à modéliser est moindre, il nécessite donc moins de données d'apprentissage ;
- l'utilisation des classes permet d'établir une généralisation : certains événements non vus au niveau des mots dans le corpus d'apprentissage peuvent être modélisés au niveau des classes.

Dans ce cadre là, un mot w_i appartient à une classe c_i. Il est à noter qu'un mot peut très bien appartenir à plusieurs classes. Pour des raisons de simplicité, mettons qu'un mot w_i n'appartient qu'à une seule classe c_i, le modèle n-classes peut être construit à partir des $n-1$ classes précédentes :

$$P(w_i|c_{i-n+1}\dots c_{i-1}) = P(w_i|c_i)P(c_i|c_{i-n+1}\dots c_{i-1}) \tag{1.6}$$

où $P(w_i|c_i)$ est la probabilité du mot w_i dans 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 l'historique des $n-1$ classes précédentes. La probabilité d'une phrase W est donnée par :

$$P(W) = \sum_{c:w_i\in c}\prod_i P(w_i|c_{i-n+1}\dots c_{i-1}) = \sum_{c:w_i\in c}\prod_i P(w_i|c_i)P(c_i|c_{i-n+1}\dots c_{i-1}) \tag{1.7}$$

Si les classes ont une intersection vide, c'est à dire qu'à un mot ne correspond qu'une seule classe, et pour un modèle *Tri*-classes l'équation 1.7 peut être simplifiée en :

$$P(W) = \prod_i P(w_i|c_i)P(c_i|c_{i-2}\dots c_{i-1}) \tag{1.8}$$

Si $C(w_i)$ est la fréquence du mot w_i, $C(c_i)$ la fréquence de la classe c_i et $C(c_{i-1}c_i)$ la fréquence qu'un mot d'une classe soit suivi immédiatement d'un mot d'une autre classe, la probabilité *Bi*-grammes serait :

$$P(w_i|w_{i-1}) = P(w_i|c_{i-1}) = P(w_i|c_i)P(c_i|c_{i-1}) = \frac{C(w_i)C(c_{i-1}c_i)}{C(c_i)C(c_{i-1})} \tag{1.9}$$

1.5 Le lissage

Un des problèmes important en modélisation stochastique du langage est que les corpus d'apprentissage ne couvrent pas toutes les successions de mots possibles. Ceci est encore plus vrai dans les applications de dialogue où les corpus sont de taille restreinte. De nombreux événements, des successions de mots possibles, ne sont pas observés. La probabilité qui leur est associée est alors nulle. Une chaîne de mots où apparaît un de ces événements n'est pas considérée comme une transcription potentielle et ceci sans considération de son score acoustique. Le but du lissage est de prévenir ces potentielles erreurs de reconnaissance en rendant la distribution observée plus uniforme en attribuant une probabilité non nulle à ces événements et en ajustant à la baisse les probabilités trop fortes. Les principales techniques sont détaillées dans [Chen et Goodman, 1996] où est également présentée une discussion sur leurs performances respectives.

1.5.1 Principe

L'estimation des paramètres d'un modèle de langage de type N-grammes est le plus souvent obtenue par la combinaison de deux composants : un modèle de *discounting* (décompte) et un modèle de redistribution. Le principe général est de prélever une quantité à la masse des probabilités issue des événements observés, et de la redistribuer aux probabilités associées aux événements non vus.

La probabilité d'un mot jamais vu en présence d'un historique donné est, sans lissage, nulle. Au contraire, les méthodes de lissage présentées ici lui attribuent une valeur non nulle calculée à partir d'un historique réduit.

1.5.2 Décompte

La fréquence conditionnelle relative fr d'un mot w selon un historique h s'écrit :

$$\begin{cases} fr(w|h) = \frac{c(hw)}{c(h)} & \text{si } c(h) > 0 \\ fr(w|h) = 0 & \text{si } c(h) = 0 \end{cases} \tag{1.10}$$

Toutes les méthodes de *discounting* introduisent une fréquence conditionnelle décomptée $fr^*(w|h)$ telle que :

$$0 \leq fr^*(w|h) \leq fr(w|h) \quad \forall hw \in V^n \tag{1.11}$$

1.5.3 Redistribution

Pour un historique h donné, la redistribution de la masse de probabilités ôtée de fr s'effectue à l'aide d'une composante appelée la probabilité de fréquence nulle (*zero-frequency probability*), calculée à partir de fr^*.

La probabilité de fréquence nulle, notée λ, est définie comme suit :

$$\lambda(h) = 1 - \sum_{w \in V} fr^*(w|h) \tag{1.12}$$

Cette définition implique que pour un historique jamais observé ($c(h) = 0$), alors $\lambda(h) = 1$.

Pour un mot w jamais rencontré après l'historique h, la probabilité de fréquence nulle associée à h est utilisée pour pondérer la valeur de $P(w|h')$, où h' est un historique moins

restrictif que h et pour lequel on suppose que l'événement $h'w$ a plus de chance d'avoir été observé que hw.

1.5.4 Lissage par repli (*backing-off*)

Le lissage par repli [Katz, 1987] est un lissage de type hiérarchique. Le principe de cette technique consiste à utiliser un modèle de langage plus général lorsqu'un modèle spécifique ne détient pas suffisamment d'information pour un contexte donné.

Par exemple, lorsque pour un *n-gramme* hw, où h correspond aux $n-1$ mots précédant le mot w, aucune observation n'a été obtenue sur le corpus d'apprentissage, le modèle *n-gram* se tourne vers un modèle de niveau inférieur *(n-1)-gramme* : ce processus peut bien sûr être réitéré jusqu'au niveau le plus bas, le *zéro-gramme*, qui consiste en l'attribution d'une constante indépendante du mot w.

La probabilité d'un *n-gram* est donc estimée à partir du lissage de l'approximation la plus significative (du point de vue de la quantité d'observations) :

$$P(w|h) = \begin{cases} fr^*(w|h) & \text{si } fr^*(w|h) > 0 \\ \alpha_h \lambda(h) P(w|h') & \text{sinon} \end{cases} \tag{1.13}$$

avec

$$\alpha_h = \left(\sum_{w:fr^*(w|h)=0)} P(w|h') \right)^{-1}$$

qui permet à la distribution $P(w|h)$ de respecter la contrainte de sommation à 1.

1.6 Combinaison des modèles acoustiques et des modèles de langage

1.6.1 *Fudge factor*

Bien que la formule (1.3) suggère que la probabilité du modèle acoustique et la probabilité du modèle de langage peuvent être combinées à travers une simple multiplication, il est nécessaire en pratique d'effectuer une pondération. Sans cela, la participation d'un des modèles est négligeable à cause de la différence de l'ordre de grandeur de leurs probabilités. En effet, les probabilités du modèle acoustique sont beaucoup plus petites que celles du modèle de langage : $P(A|W) \ll P(W)$.

La solution la plus couramment utilisée pour atténuer ce problème consiste à ajouter un poids, noté lw (pour *linguistic weight*) et souvent appelé *fudge factor*, au modèle de langage. On a alors :

$$\hat{W} = \underset{W}{argmax}\, P(W)^{lw} P(A|W) \tag{1.14}$$

Le poids lw est déterminé empiriquement à partir d'expériences effectuées sur un corpus de développement : la valeur choisie est celle qui optimise les performances du système de reconnaissance. Généralement, $lw > 1$.

1.6.2 Pénalité linguistique

La contribution du modèle de langage peut aussi être interprétée comme une pénalité sur le nombre de mots. En fonction des valeurs des probabilités du modèle de langage, le système peut privilégier une séquence composée de peu de mots longs ou, au contraire, une séquence constituée de nombreux mots courts. Afin d'ajuster au mieux la tendance

du système à insérer ou supprimer des mots, une valeur appelée pénalité linguistique et notée lp est insérée dans la formule (1.14), qui devient :

$$\hat{W} = \underset{W}{argmax}\, P(W)^{lw} lp^{N(W)} P(A|W) \tag{1.15}$$

où $N(W)$ est le nombre de mots de la séquence W.

Tout comme le fudge factor lw, la pénalité linguistique lp est déterminée empiriquement : la valeur choisie doit optimiser les performances du système de reconnaissance pour des expériences effectuées sur un corpus de développement.

1.6.3 Utilisation des logarithmes

Les multiplications successives de probabilités, c'est-à-dire de valeurs comprises entre 0 et 1, mènent à manipuler des valeurs de plus en plus proches de 0. La limite de capacité de représentation de valeurs proches de 0 d'un ordinateur est rapidement atteinte, à moins de mettre en place des mécanismes coûteux en terme de temps de calcul. En pratique, les systèmes de reconnaissance de la parole ne manipulent pas directement les probabilités : ce sont les logarithmes de ces probabilités qui sont utilisés. Le passage aux logarithmes entraîne l'utilisation d'additions plutôt que de multiplications : ce type d'opérations conforte la propriété intéressante des logarithmes qui changent très lentement d'ordre de grandeur. Ainsi, la formule (1.15) se ré-écrit :

$$\hat{W} = \underset{W}{\text{ArgMax}}\, lw.\log P(W) + \log P(A|W) + N(W)\log(lp) \tag{1.16}$$

1.7 Espace de recherche et graphe de mots

À partir de l'observation d'événements acoustiques et de connaissances *a priori* (lexique, modèles acoustiques, ...), un système de reconnaissance génère un ensemble d'hypothèses de séquences de mots. Cet ensemble est appelé espace de recherche : le système doit en extraire la phrase qui satisfait l'équation (1.15). L'espace de recherche est généralement représenté sous la forme d'un graphe, appelé graphe de recherche, qui intègre les informations utilisées pour la génération des hypothèses : informations temporelles, unités acoustiques (phonèmes, syllabes, demi-syllabes, ...) associées à leurs scores acoustiques (probabilités données par le modèle acoustique), mots induits par les séquences d'unités acoustiques, ...

La recherche de la phrase de probabilité maximale au sein d'un graphe de recherche est analogue au problème de la recherche du chemin de poids minimal dans un graphe. De nombreux algorithmes existent pour résoudre ce problème [Cettolo *et al.*, 1998]. Cependant, pour la majorité des systèmes, la taille de l'espace de recherche est très importante et ralentit considérablement le traitement. Pour obtenir une solution dans un délai acceptable, une recherche en faisceau, appelée *beam search*, permet de restreindre l'espace de recherche en supprimant des hypothèses qui semblent localement peu probables [Ney *et al.*, 1992]. Cet élagage ne garantit pas l'obtention de la phrase la plus probable, mais le compromis entre la durée du traitement et la perte de précision est très souvent largement acceptable.

L'utilisation de modèles de langage sophistiqués, par exemple un modèle N-grammes avec un N assez grand, ralentit la recherche de la phrase de probabilité maximale. La solution la plus répandue consiste à utiliser ce type de modèle lors d'une deuxième passe : le graphe de recherche généré lors d'une première passe est élagué grâce à l'application d'un algorithme de *beam search*, et n'est plus composé que de mots. Chaque mot est

alors associé à un score acoustique calculé à partir des scores des unités acoustiques qui le composent. Le graphe obtenu pour la deuxième passe est un graphe de mots : il est l'objet de traitements linguistiques lourds qui auraient fortement ralenti le processus de reconnaissance s'ils avaient été appliqués sur l'intégralité de l'espace de recherche dès la première passe.

Généralement, le modèle de langage stochastique utilisé en première passe d'un processus de reconnaissance de la parole est un modèle *Bi*-grammes, voire *Tri*-gramme. Ces modèles ont la particularité d'être simples d'emploi et relativement peu coûteux en temps de calcul. Ces caractéristiques, combinées à l'influence largement bénéfique de ces modèles sur les résultats d'un processus de reconnaissance, sont à l'origine de leur très forte implantation dans les systèmes de reconnaissance de la parole. Les modèles de langage plus évolués, faisant appel à des historiques plus importants ou à des sources de connaissances supplémentaires, sont utilisés en seconde passe sur un espace de recherche réduit à un graphe de mots ou à une liste des N-meilleures phrases. Le graphe de mots ou la liste des N-meilleures hypothèses sont issus du décodage effectué en première passe. Cette seconde passe, qui consiste à utiliser un ou plusieurs modèles nécessitant plus de ressources qu'un modèle *Bi*-grammes afin d'améliorer encore la reconnaissance, est habituellement connue sous le nom de phase de rescoring.

Bien entendu, rien n'empêche d'utiliser ces modèles de langages gourmands en ressources dans un système de reconnaissance basé sur une seule passe. Malheureusement, les algorithmes utilisés et la technologie actuelle ne permettent pas d'obtenir des résultats satisfaisants dans des délais raisonnables. Dans une application conviviale de dialogue entre un homme et une machine, le système de reconnaissance de la parole doit avoir un temps de réponse proche du temps réel. L'utilisation de systèmes multi-passes permet d'utiliser des modèles de langage nécessitant de grosses ressources sans trop ralentir le processus global de reconnaissance.

CHAPITRE

2

Module de compréhension

Sommaire

2.1 Rôle du module de compréhension

Le but d'un module de compréhension est d'extraire le sens d'un signal de parole, afin d'interagir avec l'utilisateur. Actuellement, seuls les systèmes restreints à des domaines limités peuvent être conçus à avoir la faculté de comprendre le sens des paroles de l'utilisateur. Seule cette restriction autorise la création de modèles de langage contraints spécifiques et d'une description sémantique complète permettant une robustesse suffisante à la création d'un système utilisable. Pour réaliser cette opération, il existe des méthodes manuelles ([Minker et Bennacef, 1996]) ou des approches stochastiques ([Pieraccini et Levin, 1993, Schwartz *et al.*, 1997, Riccardi et Gorin, 1998]) qui

ont permis de réduire fortement le recours à l'expertise humaine lors du développement du module de compréhension.

Du point de vue pragmatique, le but du module de compréhension est de transformer le signal de parole en une structure sémantique qui sera utilisée par le gestionnaire de dialogue pour décider de l'action à entreprendre. Une définition précise d'une structure sémantique n'est pas ici quelque chose de critique. On peut la définir simplement comme une structure d'informations représentant une *intention* associée à des objets représentés sous la forme d'une paire attribut/valeur. Ces objets seront nommés concepts. Par exemple, le système PlanResto (présenté en section 4.2) de recherche de restaurant pourrait dire :

– « Bonjour, que recherchez vous ? »

et l'utilisateur peut répondre :

– « un restaurant japonais à Bastille »

Dans ce cas la structure sémantique Γ pourrait ressembler à :

[Recherche Restaurant :]
[spécialité]→(japonais)
[lieu]→(Bastille)
[prix]→()

Le principe d'interprétation consiste donc à détecter les segments correspondants aux paires attribut/valeur et à en déduire la représentation sémantique.

Certaines erreurs ont pu être commises, aux composants de cette structure peuvent être alors associés différents niveaux de confiance. En fonction de la structure sémantique décodée, de l'état actuel du dialogue, le gestionnaire de dialogue peut générer une structure sémantique encodant, par exemple, une requête pour confirmer un concept et obtenir le prix.

– « Quel est votre budget pour le restaurant japonais ? »

2.2 Analyse sémantique

L'analyseur sémantique d'un système de dialogue, qui aboutit à une compréhension, constitue la phase primordiale du traitement. Un tel module doit être capable de fournir une représentation du sens en dépit des difficultés inhérentes à la parole spontanée. En effet, le langage parlé de nature spontanée se manifeste par des répétitions, des hésitations ou des requêtes disloquées (ruptures de construction) qui ne respectent pas la grammaire de l'écrit. De plus les erreurs de reconnaissance du module RAP peuvent aggraver la situation. C'est pourquoi un analyseur robuste est nécessaire. Bien que l'interprétation sémantique puisse être obtenue à partir d'une analyse syntaxique [Roark, 2002, Chappelier *et al.*, 1999], l'extraction sémantique ne s'appuie généralement pas totalement sur l'analyse syntaxique telle qu'elle est menée par des grammaires de type hors-contexte ; elle se limite la plupart du temps aux éléments porteurs de sens de la requête tout en ignorant les parties redondantes ou non-essentielles pour l'application.

Des grammaires hors-contexte peuvent être utilisées pour analyser une phrase complète, par exemple la phrase « je veux aller à Marseille » pourrait être analysée par la grammaire suivante :

(a) S → NP VP
(b) NP → N
(c) VP → Vcluster PP
(d) Vcluster → veux V
(e) V → aller | voyager
(f) PP → prep NP
(g) N → Marseille | je
(h) prep → à

Le résultat serait :
[S [NP [N je]] [VP [VCluster veux [V aller]] [PP [prep à] [NP [N Marseille]]]]]
Cette structure peut fournir les bases d'une analyse sémantique. De plus, la grammaire est adéquate pour l'exemple donné et peut être facilement étendue pour couvrir plus de villes en augmentant la règle (g), en élargissant le lexique. Certains problèmes logiques peuvent se poser, par exemple la phrase « Marseille veux aller à je » est aussi analysée. Ceci peut être réglé facilement par exemple en utilisant des catégories plus fines. Mais ce genre de problèmes n'est pas crucial dans une application concrète car très rare. Le problème est surtout de savoir comment jongler avec des variations naturelles. Selon l'utilisateur d'un système, la phrase exemple aurait pu être :

à Marseille.
je vais à Marseille.
je suis en retard, j'ai un congrès dans Marseille.
je veux partir d'Avignon et joindre Marseille demain.
. . .

Ces phrases présentent différents types de variations pour lesquels une grammaire qui couvre toute la phrase peut avoir des problèmes d'analyse. C'est pourquoi les concepteurs de systèmes de dialogue ont gravité autour de l'idée d'*analyse robuste*. L'analyse robuste est l'idée d'extraire seulement les morceaux porteurs d'un sens basique (que nous nommerons *concepts*) d'une phrase, en ignorant le reste. Des petites grammaires peuvent être écrites pour analyser une phrase, voire un graphe de mots, afin de rechercher seulement les unités pour lesquelles elles sont spécialisées. Par exemple, une grammaire *Destination*, peut parcourir toutes les différentes séquences précédentes et trouver la destination dans chaque cas :

Destination → Préposition NomVille
Préposition → à | dans | joindre
NomVille → Marseille | . . .

Une fois les unités sémantiques élémentaires détectées, des grammaires sémantiques sont écrites pour en déduire une représentation sémantique. C'est à dire remplir les attributs de la structure sémantique avec des mots terminaux ou des classes sémantiques récursives non-terminales. Ces grammaires servent à spécifier les relations entres les concepts plutôt que les expressions qui servent à les détecter. Ces expressions syntaxiques sont spécifiées par des grammaires spécifiques qui sont associées à chaque concept. Un exemple d'arbre d'analyse d'une grammaire sémantique est visible dans la figure 2.1.

Pour en savoir davantage, on peut se référer à [Huang et Hon, 2001].

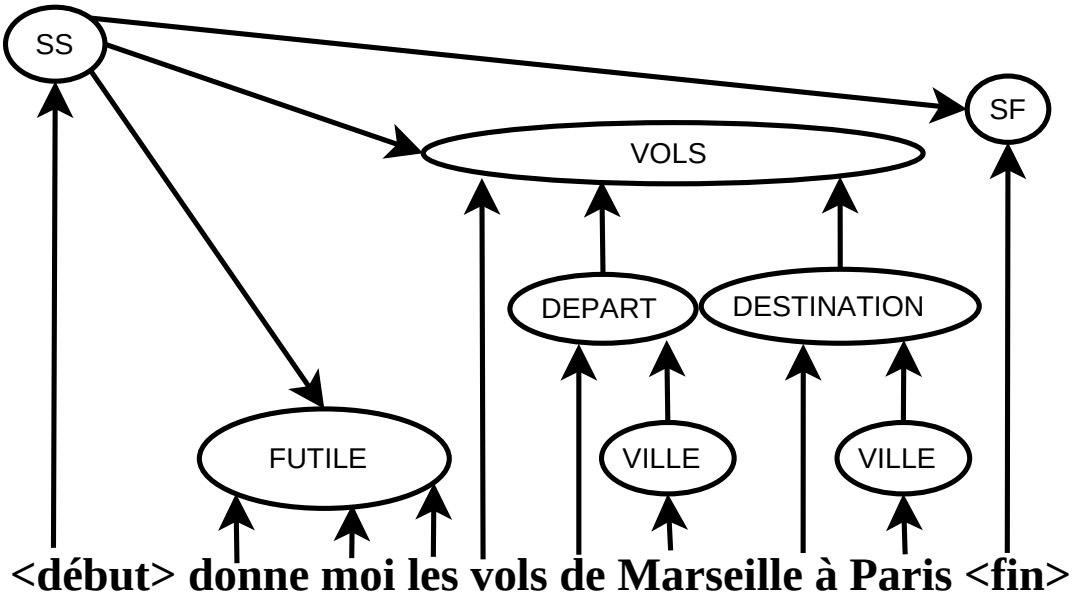

FIG. 2.1 – *Exemple d'un arbre d'analyse sémantique*

2.3 Analyse en compréhension

Un premier effort considérable dans la compréhension du langage parlé (SLU) a été effectué avec le projet DARPA qui a débuté en 1971. Le projet est passé en revue dans [Klatt, 1977]. Un nouveau projet DARPA débuté dans les années 90 dans le domaine des systèmes d'informations de voyage aérien (ATIS : *Air Travel Information System*) a réussi brillamment à accélérer le développement de systèmes SLU en se focalisant sur des tâches d'accès à des bases de données qui n'étaient pas trop éloignées des scénarios concrets. Les systèmes développés dans ce projet ainsi que les approches précédentes sur la compréhension du langage naturel sont passés en revue dans [Kuhn et De Mori, 1998]. Les sections suivantes présentent quelques approches classiquement utilisées.

2.3.1 Analyse fondée sur la grammaire de cas

[Minker et Bennacef, 1996] utilisent le formalisme de la grammaire des cas [Fillmore, 1968] pour construire une représentation sémantique sous forme d'un schéma. Le concept du schéma est identifié par un ou plusieurs mots de référence dans la phrase. Les attributs du schéma sont instanciés à partir de certaines parties de la phrase en utilisant des contraintes syntaxiques locales sous forme de marqueurs. Un schéma incomplet peut toutefois être complété par d'autres schémas à l'aide du gestionnaire de dialogue. Le processus de l'analyse considère seulement certaines parties de la phrase comme étant sémantiquement significatives. Les hésitations et les répétitions, par exemple, sont ignorées. Dans un premier temps, est alors appliqué un post-traitement sur la transcription, qui associe aux mots leur version de base (lemme par exemple) et les mots sémantiquement reliés sont regroupés (entités nommées). Les mots non-porteurs d'informations ou hors domaine sont assignés à une catégorie spéciale notée *filler*.

L'analyse sémantique procède alors de la manière suivante (considérant la représen-

tation schématique dans le tableau 2.1 sur une application qui fournirait un service de consultation d'horaires d'avion) : au niveau conceptuel, les MOTS DE RÉFÉRENCE permettent de sélectionner le SCHÉMA correspondant ; ensuite, les parties variables du schéma, ici introduites par le symbole @, sont instanciées en utilisant des marqueurs, sachant que des structures de niveau plus élevé font référence à des SOUS-SCHÉMAs de niveau inférieur. Le niveau intermédiaire abrite les relations entre les marqueurs et les attributs. Le SOUS-SCHÉMA itinéraire, par exemple, contient les attributs @Départ et @Destination. Les mots appartenant à ces attributs doivent être précédés de « de, départ » et « à, destination » respectivement. Le niveau de base regroupe la liste des attributs autorisés, par exemple les SOUS-SCHÉMAs ville et heure. Ils correspondent principalement aux valeurs dans la base de données. Pour chaque mot de la phrase, l'analyseur sémantique est appliqué de manière successive sur les SCHÉMAs et les SOUS-SCHÉMAs jusqu'à ce qu'il n'y ait plus de mots qui puissent remplir les champs du schéma. Une fois complété, le schéma sémantique représente le sens de la phrase.

TAB. 2.1 – Exemple de représentation sous forme de schéma qui contient l'ensemble des concepts et des attributs liés à l'application et les représente sous forme hiérarchique, le niveau conceptuel, intermédiaire et le niveau de base.

Niveau conceptuel	SCHÉMA vol {MOTS DE RÉFÉRENCE : tarif, prix, . . . itinéraire : @itinéraire }
Niveau intermédiaire	SOUS-SCHÉMA itinéraire {Départ : (de, départ) @ville. Destination : (à, destination) @ville. }
	SOUS-SCHÉMA horaire {}
Niveau de Base	SOUS-SCHÉMA ville {ville :Marseille, Paris, Lyon,. . . } SOUS-SCHÉMA heure {} SOUS-SCHÉMA minute {}

2.3.2 Template Matcher

L'entrée du Template Matcher (TM) est la meilleure hypothèse de mots générée par le module de reconnaissance de la parole, qui utilise un modèle de langage *Bi*-grammes [Appelt et Jackson, 1992]. Le TM tente simplement de remplir les champs d'une structure sémantique. Les différentes structures sont en compétition avec les autres sur chaque intervention ; à toutes, est associé un score ; et la structure avec le meilleur score génère la requête à la base de données. Les champs sont remplis en cherchant dans l'énoncé certaines séquences de mots. Le score d'une structure est simplement le pourcentage de mots de l'énoncé qui contribue à la remplir. Cependant, certains mots-clefs qui sont fortement corrélés avec une structure particulière augmentent fortement son score s'ils apparaissent. Si le meilleur score n'excède pas un seuil défini, le système préfère ne pas répondre plutôt que de faire confiance à cette structure.

2.3.3 DELPHI

DELPHI est un analyseur linguistique qui génère les *N*-meilleures hypothèses en utilisant un algorithme simple et rapide [Schwartz *et al.*, 1992], qui re-score répétitivement ces hypothèses au moyen d'un algorithme plus complexe et plus lent. De cette manière, plusieurs sources de connaissances peuvent contribuer au résultat final sans compliquer la structure de contrôle ou ralentir significativement la production du résultat final.

La première version de DELPHI consiste en un analyseur basé sur des schémas. Un dispositif intéressant de cet analyseur, a été l'incorporation de probabilités pour les différents sens d'un mot et pour l'application de règles grammaticales. Ces probabilités sont estimées à partir de données et utilisées pour réduire l'espace de recherche pour l'analyse. Un module de repli a été incorporé dans les versions suivantes. Un analyseur syntaxique utilise des règles étendues pour générer une analyse complète de l'énoncé. S'il échoue, un analyseur tente de remplir les champs de la structure à la manière du Template Matcher.

2.3.4 Phoenix

En beaucoup de points, le système Phoenix de l'université de Carnegie Mellon (CMU) est similaire au Template Matcher du SRI [Issar et Ward, 1994]. Le principe d'interprétation consiste à détecter les segments correspondant aux concepts (paires attribut/valeur) et à en déduire le schéma. Il n'y a pas de grammaire globale, mais plusieurs sous-grammaires dont chacune est associée à un concept. Le score pour un schéma est simplement le nombre de mots dans l'énoncé pris en compte.

2.3.5 TINA

L'analyseur linguistique TINA développé à l'institut technologique du Massachusetts (MIT), a connu la même évolution que d'autres analyseurs linguistiques : il consistait au départ en un analyseur syntaxique global. Cette analyse utilise une grammaire hors-contexte transformée de façon automatique en un automate portant des probabilités sur les arcs, ce qui permet d'avantager les constructions les plus courantes. Les nœuds de cet automate font référence à des catégories particulières, qui peuvent être sémantiques (comme les lieux) ou bien syntaxiques (par exemple les verbes ou les adjectifs) [Seneff, 1989]. Un analyseur robuste a été ajouté qui intervient lorsque il échoue [Seneff, 1992]. Il est obtenu en modifiant la grammaire autorisant des analyses partielles. Dans ce mode, l'analyseur effectue un traitement gauche-droite comme d'habitude, mais un ensemble exhaustif des analyses possibles est généré commençant à chaque mot de l'énoncé. L'aspect inhabituel de cet analyseur robuste est qu'il exploite l'historique du dialogue en autorisant les champs du schéma à être hérités des énoncés précédents.

2.3.6 CHRONUS

Le système CHRONUS (*Conceptual Hidden Representation of Natural Unconstrained Speech*) est un système de compréhension de la parole développé par le laboratoire AT&T [Pieraccini et Levin, 1995] qui propose un décodage conceptuel stochastique. Ce système considère l'énoncé utilisateur comme une séquence de concepts élémentaires. Ces concepts sont des séquences de mots correspondant à des unités de sens. Le rôle du décodage conceptuel consiste à découper l'énoncé en concepts. Le processus suit un modèle Markovien dont les états sont les concepts. La séquence de mots relative à un

concept est aussi modélisée par un processus Markovien représenté par un modèle de langage N-grammes conditionné sur les concepts.

2.3.7 CHANEL

Le système CHANEL [Kuhn et De Mori, 1995] est le résultat d'une recherche conduite à l'université Mc Gill de Montréal. CHANEL apprend des règles de détection de concepts au moyen de plusieurs arbres de décisions spécialisés appelés arbre de classification sémantique (SCTs). Il y a un SCT pour chaque concept. L'aspect le plus intéressant de CHANEL est que les SCTs tentent de découvrir des règles faisant intervenir aussi peu de mots ou d'unités syntaxiques que possible. Ils sont donc tolérants à un haut degré d'erreurs de reconnaissance sur les mots sémantiquement non-importants. Une autre différence importante entre CHANEL et les systèmes comme CHRONUS, est que CHRONUS fait une correspondance entre morceau de phrase et concept, tandis que chaque SCTs dans CHANEL utilise la phrase entière pour déterminer les concepts. Cela permet qu'un mot ou plusieurs mots contribuent à plus d'un concept.

2.3.8 Le HUM

Le *hidden understanding model* (HUM) [Miller *et al.*, 1994] propose la technique suivante. Soit W la chaîne de mots et S la représentation sémantique associée. En accord avec la règle de Bayes nous avons :

$$P(S|W) = \frac{P(W|S)P(S)}{P(W)} \tag{2.1}$$

La tâche du processus d'interprétation revient alors à trouver la représentation sémantique $\hat{S}$, telle que :

$$\hat{S} = \underset{S}{\text{ArgMax}}\, P(W|S)P(S) \tag{2.2}$$

$P(S)$ est le modèle de langage sémantique qui détermine la distribution statistique *a priori* de la représentation sémantique. Il est basé sur une représentation sémantique en forme d'arbre où les concepts sont les nœuds et les sous-concepts sont les nœuds fils (le tableau 2.2 illustre cette représentation pour la phrase « donne moi les vols de Marseille à Paris »). Le concept VOL possède les sous-concepts Vol_Indice,Origine,Destination.

TAB. 2.2 – Représentation sémantique sous forme d'arbre pour « donne moi les vols de Marseille à Paris » telle qu'elle pourrait l'être dans le système HUM

VOL [
VOL_INDICE[vols]
ORIGINE[ORIGINE_INDICE[de] VILLE[Marseille]]
DESTINATION[DESTINATION_INDICE[à] VILLE[Paris]]]

Les sous-concepts Destination et Origine contiennent respectivement les nœuds terminaux Destination_Indice, Ville ; Origine_Indice Ville. Chaque nœud terminal peut être composé d'un où plusieurs mots.

Le modèle de langage sémantique $P(S)$ est modélisé comme $P(S_i|S_{i-1}, concept)$ où *concept* est le concept parent pour S_i et S_{i-1}. Basé sur cette définition, la probabilité $P(Destination|Origine, Vol)$ est plus grande que la probabilité $P(Origine|Destination, Vol)$, les utilisateurs d'un système de réservation d'avions omettant souvent de préciser l'origine

du vol. $P(W|S)$ est lui appelé modèle de *réalisation lexicale*, c'est basiquement un modèle *Bi*-grammes de mots enrichi avec le contexte du concept parent :

$$p(W|S) = \prod P(w_i|w_{i-1}, concept) \tag{2.3}$$

Le modèle de langage sémantique ainsi que le modèle de réalisation lexicale sont appris sur un corpus étiqueté. L'algorithme de Viterbi est appliqué pour trouver le meilleur chemin correspondant à l'interprétation sémantique $\hat{S}$ en accord avec l'équation 2.2.

2.3.9 Le HVS

[He et Young, 2003] proposent pour le processus d'analyse sémantique un modèle vectoriel à états cachés (*hidden vector state model* HVS). Le principe est le suivant : si l'on considère l'arbre d'analyse de la figure 2.1, l'information sémantique reliée à chaque mot seul peut être codée en tant qu'un vecteur d'étiquettes sémantiques en partant de l'étiquette pré-terminale et finissant à l'étiquette racine. Par exemple le mot *Marseille* est décrit par le vecteur sémantique $[VILLE, DEPART, VOLS, SS]$ et l'arbre complet d'analyse peut être remplacé par une séquence de vecteurs telle que l'illustre le tableau 2.3. En prenant chaque vecteur d'état comme une variable cachée, l'arbre d'analyse peut être traité comme un processus markovien, ceci est le modèle HVS.

TAB. 2.3 – Vecteur d'état équivalent à l'arbre d'analyse

				SS		SS	
			SS	VOLS	SS	VOLS	
	SS	SS	VOLS	DEPART	VOLS	DESTINATION	SS
SS	FUTILE	VOLS	DEPART	VILLE	DESTINATION	VILLE	SF
<début>	donne moi les	vols	de	Marseille	à	Paris	<fin>

2.4 Coopération transcription/compréhension

Le processus de compréhension se fait généralement en deux temps : la génération d'une transcription du signal de parole en unités linguistiques (mots ou phrase) suivie de l'analyse de cette transcription. Certains travaux font intervenir les connaissances nécessaires à la compréhension dans le processus de transcription. Deux approches existent : la première, propose une architecture en deux passes où les informations de compréhension sont utilisées sur le graphe de mots issu du module RAP (2.3) ; la seconde, utilise une architecture de reconnaissance en une passe, où les informations de compréhension sont utilisées pour guider le processus de transcription (figure 2.2).

2.4.1 Architecture deux passes

La première architecture peut être illustrée par le processus de compréhension du système Philips [Aust *et al.*, 1995] qui travaille directement sur le graphe de mots proposé par le module RAP.

Les considérations suivantes sont utilisées : le sens de l'intervention utilisateur est exprimé par des séquences de mots significatives appelées concepts. Ces concepts s'agencent d'une manière arbitraire et des mots inutiles pour le processus de compréhension peuvent s'intercaler entre deux concepts. Ces mots inutiles sont désignés par *« filler »* .

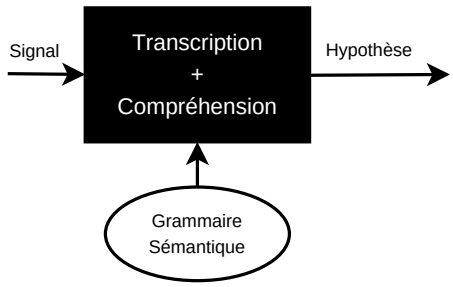

FIG. 2.2 – *Architecture reconnaissance/compréhension en une passe*

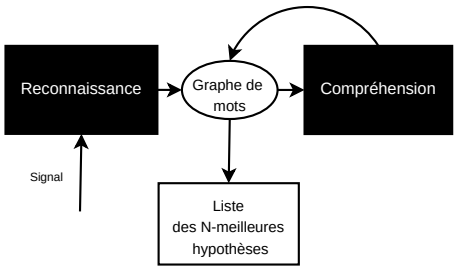

FIG. 2.3 – *Architecture reconnaissance/compréhension en deux passes*

Le traitement est réalisé en 4 étapes (voir figure 2.4) :

1. la première est un pré-traitement du graphe de mots consistant à construire un chemin dans le graphe permettant de sauter les silences et les hésitations ;
2. la seconde étape est l'étape essentielle de la compréhension. L'objectif est de construire le graphe de concepts à partir du graphe de mots. Une grammaire hors-contexte stochastique est utilisée pour modéliser les concepts. Toutes les dérivations possibles sont calculées et insérées dans le graphe conceptuel. Les scores de chaque concept sont donnés par la somme des scores des mots qui supportent le concept. C'est dire le score donné par la vraisemblance acoustique et le score linguistique délivré par la grammaire.
3. le graphe de concepts est ensuite complété avec des transitions *filler*, permettant de sauter les mots vides de sens afin de relier les concepts entre eux et assurer la présence d'un chemin du premier au dernier noeud du graphe. Des transitions filler sont également insérées entre le nœud de départ et de fin de chaque concept. Ces transitions filler ont le score optimal du chemin existant entre leur nœud de départ et de fin ;
4. le meilleur chemin de ce graphe étant celui donné par les transitions filler seules, leur score est pénalisé en fonction de leur durée. Un modèle *Bi*-grammes sur les concepts est ajouté afin de déterminer le meilleur chemin du graphe qui correspond à la séquence de concepts la plus probable.

2.4.2 Architecture une passe

Si le processus de transcription suit le principe statistique de l'équation 1.3, dans un système de dialogue, c'est le sens (S) de l'intervention utilisateur qui doit être trouvé. Étant donné un signal de parole A le problème de compréhension revient à chercher pour :

$$\hat{S} = \underset{S}{\text{ArgMax}}\, P(S|A) = \underset{S}{\text{ArgMax}}\, P(A|S)P(S) \tag{2.4}$$

Il est en général plus pratique de transcrire d'abord le signal A en unités linguistiques (mot ou phrase) pour que l'extraction sémantique soit réalisable :

$$\hat{S} = \underset{S}{\text{ArgMax}} \sum_{W} P(A|w,S)P(w|S)P(S) \tag{2.5}$$

Bien que le modèle acoustique $P(A|w,S)$ dans une application de dialogue peut être dynamiquement ajusté en fonction du contexte du dialogue, la plupart des systèmes

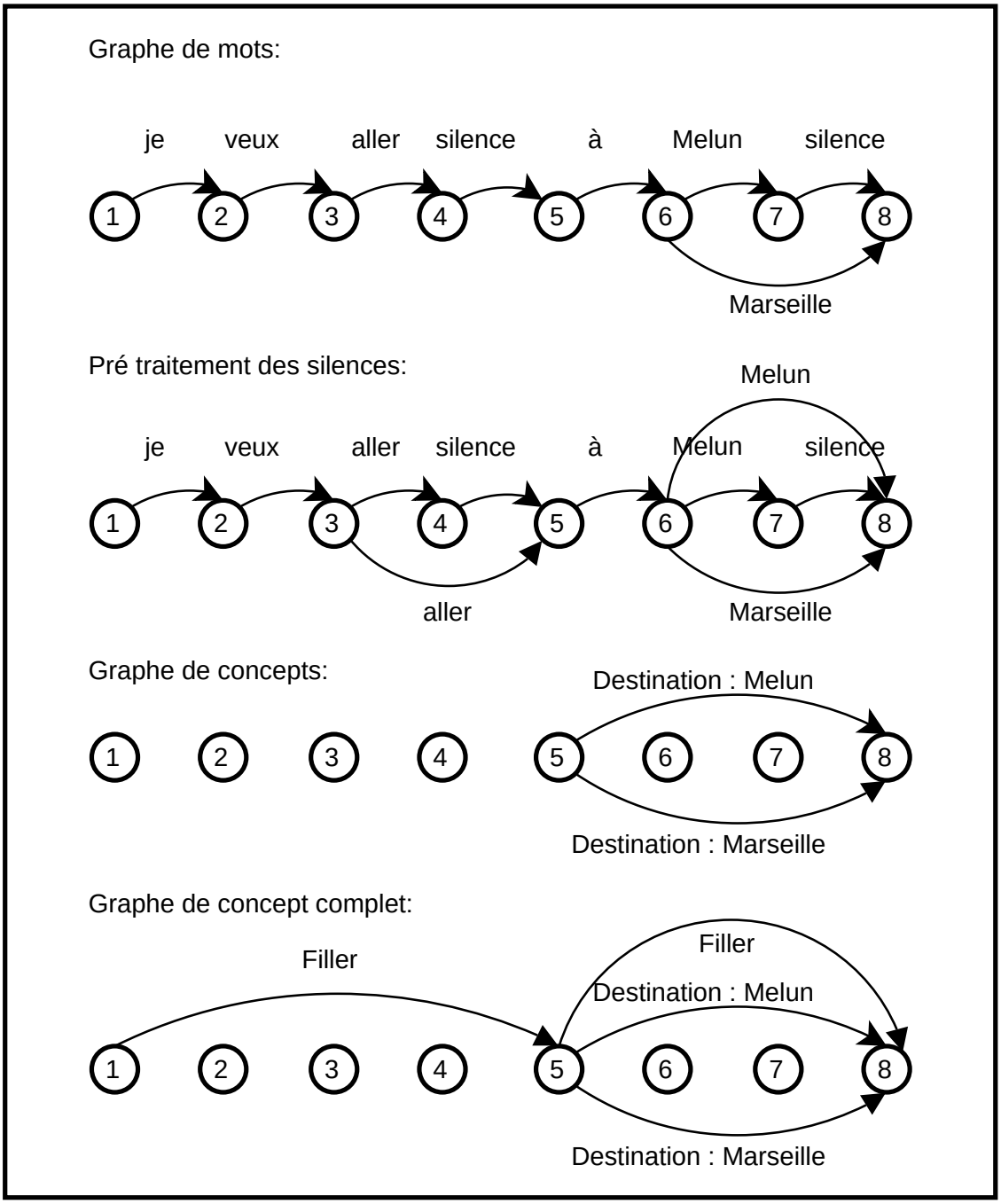

FIG. 2.4 – *Processus de compréhension du système Philips*

considèrent que les statistiques acoustiques sont satisfaisantes étant donné l'événement linguistique : $P(A|w,S) \simeq P(A|w)$. [Wang, 2003] considère que le modèle de langage sémantique $P(w|S)$ et le processus de décodage réalisant le ArgMax dans l'équation précédente est la clef pour articuler les processus de transcription et de compréhension. La différence entre un modèle de langage $P(w)$ et un modèle de langage sémantique $P(w|S)$ est que le second prend en considération l'hypothèse sémantique S pour prédire la séquence de mots, guidant le processus de transcription vers une chaîne de mots plus pertinente pour le processus de compréhension.

Pour incorporer les structures sémantiques dans le modèle de langage sémantique $P(w|S)$, [Wang, 2003] emploie une technique de *modèle de langage unifié* (ULM) qui combine des grammaires hors-contexte probabilistes (PCFG) avec des N-grammes. Les PCFG et les N-grammes ont des qualités et des défauts complémentaires, de nombreuses tentatives ont été faites pour combiner ces approches de modélisation en une seule cohérente appelée modèle de langage unifié (ULM). Une approche très utilisée en reconnaissance de la parole (par exemple [Riccardi *et al.*, 1996, Nasr *et al.*, 1999]) permet d'étendre la notion de classe dans les N-grammes à base de classes en associant à un mot une séquence de mots modélisés par une PCFG. Par exemple pour l'exemple : « *Un prix de cent francs soit trente deux pourcent de moins* », un *Tri*-gramme aurait à calculer les probabilités comme $P(\text{francs}|\text{de cent})$ et $P(\text{trente}|\text{francs soit})$ alors qu'un ULM a à traiter la phrase « *Un prix de* ***Prix*** *soit* ***Pourcentage*** *de moins* » et considère les probabilités $P(\textbf{Prix}|\text{prix de})$ et $P(\textbf{Pourcentage}|\textbf{Prix}\ \text{soit})$. Ce type d'ULM permet de réduire la perplexité pour le modèle de langage de reconnaissance, mais n'est pas forcément adapté pour découvrir la structure sémantique d'une phrase.

[Wang, 2003] utilise une seconde méthode permettant d'encapsuler des N-grammes dans des PCFG. Les symboles non-terminaux de la grammaire peuvent être modélisés par des N-grammes et notamment les symboles pré-terminaux. Avec cette approche, il est possible de garder la trace des règles invoquées dans les PCFG pour découvrir la structure de la phrase. Si les règles sont écrites pour refléter la construction d'une structure sémantique de l'application, le résultat est alors un arbre qui représente le sens de la phrase. Ce type de combinaison est désigné par « *ULM pour la compréhension* ». Tandis que les PCFG seules sont utilisées pour l'extraction sémantique, cet ULM est utile pour élargir la couverture lexicale grâce à la flexibilité des N-grammes.

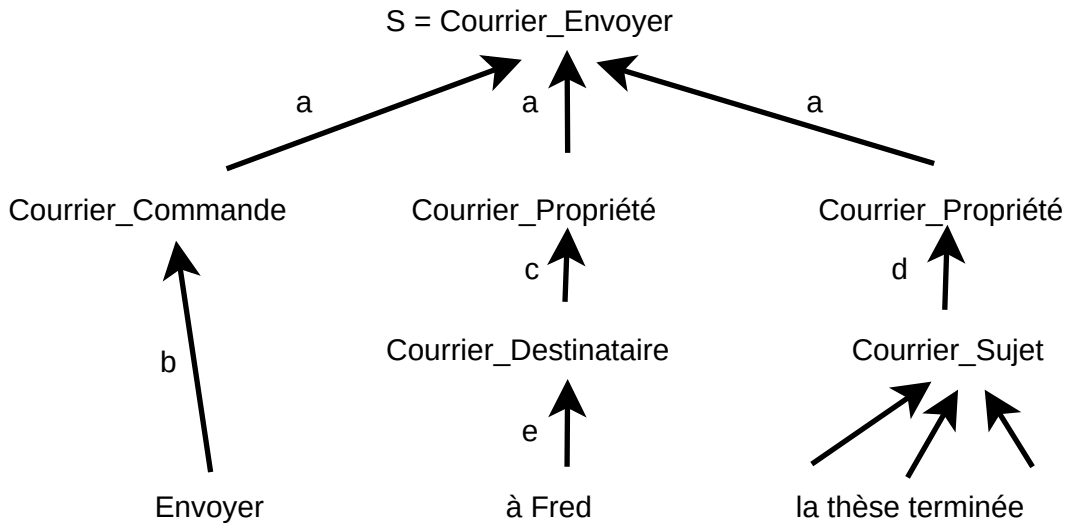

FIG. 2.5 – *Application du ULM pour la compréhension d'une phrase avec la structure sémantique modélisée par les règles PCFG et quelques éléments lexicaux, tels que « la thèse terminée » par des* N*-grammes*

La figure 2.5 illustre l'analyse sémantique de la phrase $w =$« Envoyer à Fred la thèse terminée ». Les règles de la grammaire amenant à l'interprétation S (Envoyer Courrier) sont les suivantes :

(a) S → Courrier_Commande Courrier_Propriété*
(b) Courrier_Commande → *Envoyer*

(c) Courrier_Propriété → Courrier_Destinataire
(d) Courrier_Propriété → Courrier_Sujet
(e) Courrier_Destinataire → *à Fred*

Dans cet exemple, seul le pré-terminal Courrier_Sujet est couvert par des *N*-grammes. La probabilité de la chaîne w sachant l'interprétation S, $P(w|S)$ est le produit des *N*-grammes pour P(la thèse terminée|Courrier_Sujet) et les probabilités de toutes les règles listées précédemment. Les *N*-grammes peuvent aussi être utilisés pour modéliser toutes les expressions à droite des règles de la PCFG et pas seulement les pré-terminaux comme dans l'exemple précédent. L'architecture de décodage proposée en utilisant ce modèle permet de pouvoir séquentiellement au fur et à mesure du décodage de repérer des hypothèses sémantiques et de prédire les objets sémantiques suivants afin de construire une structure sémantique complète.

CHAPITRE

3

Outils

Sommaire

3.1 Introduction

Deux points importants sont à retenir des chapitres précédents :

1. tout d'abord que le processus de compréhension se base sur la transcription du module RAP pour extraire le sens de l'énoncé utilisateur ;
2. deuxièmement, la compréhension se base dans la plupart des cas dans les systèmes de dialogue sur des unités conceptuelles élémentaires correspondant à des portions de phrases ayant un sens unitaire pour le système.

Nous proposons dans les travaux de cette thèse une architecture de décodage conceptuel, *i.e.* une méthode permettant de tenir compte de ces concepts dans la génération de la transcription. La détection des concepts dans une intervention utilisateur se fait généralement par une analyse à base de grammaire. Nous utiliserons ce formalisme et en particulier le formalisme des grammaires régulières encodées sous la forme d'automate à états fini pour effectuer cette opération. Le chapitre 3.2 introduit cette notion de modélisation formelle du langage. Le chapitre 3.4 présente les outils qui seront utilisés pour implémenter cette modélisation, à savoir les automates à états fini.

Plusieurs articles ont montré, [Schapire et Singer, 2000, Haffner *et al.*, 2003, Karahan *et al.*, 2003, Tur *et al.*, 2004], que les outils de classification de texte (les classifieurs à large-marge comme les machines à support vectoriels (SVM) ou les implémentations de l'algorithme de boosting peuvent être un moyen efficace pour détecter des concepts notamment dans les systèmes de routage d'appel, par exemple [Gorin *et al.*, 1997]. Les arbres de décisions sémantiques (SCT-Semantic Classification Tree) décrits dans [Kuhn et De Mori, 1995] ont également été utilisés dans des systèmes de dialogue similaires à ceux présentés dans ces travaux (application ATIS). Cette approche de détection de concepts a 2 principaux avantages :

- premièrement, l'intervention humaine est limitée dans le sens où aucun mot-clef ni aucune grammaire ne doivent être défini afin de caractériser un concept. Néanmoins, un corpus d'entraînement doit nécessairement être annoté ;
- deuxièmement, les classifieurs sont plus robustes au bruit généré par les erreurs du système de transcription automatique de la parole et aux effets dus à la parole spontanée parce qu'ils reposent sur des conditions suffisantes. Ils peuvent être directement entraînés sur les sorties du module RAP et donc modéliser ce bruit, alors que les grammaires génèrent habituellement des phrases correctes du language.

Nous proposons d'effectuer la détection de concepts au travers de ces méthodes. Le chapitre 3.5 introduit alors les notions de classification que nous utiliserons.

3.2 Les langages formels

3.2.1 Introduction

Bien que les applications de reconnaissance de la parole fonctionnent généralement au moyen de modèles de langage statistiques, plusieurs travaux présentent de nouvelles voies intermédiaires dans le but de combiner l'approche formelle et l'approche statistique afin de tirer parti du meilleur de ces 2 approches. [Estève, 2002] propose par exemple des modèles de langages hybrides s'appuyant sur l'intégration de connaissances linguistiques *a priori* (grammaires locales) dans un modèle statistique. Cette section présente l'approche formelle de modélisation du langage.

3.2.2 Définition

Soit un alphabet V, un ensemble fini de symboles ou de caractères, une chaîne $w = (v_1, v_2, \ldots, v_n)$ un n-uplet d'éléments appartenant à V, l'opération de base sur les chaînes, la concaténation :

$$
\begin{aligned}
V^2 &= V.V = \{w | w = v||v,\ v \text{ et } v' \in V\} \\
V^k &= \{(v_1 \dots v_i \dots v_k) v_i \in V, \forall i \in [1,k]\} \\
&= V.V^{k-1} = VV^{k-1} = V^{k-1}V \\
V^+ &= \bigcup_{k>0} V^k = V \bigcup V^2 \dots \bigcup V^k \dots \\
V^* &= \{\varepsilon\} \bigcup V^+ = V^0 \bigcup V^+ = \bigcup_{k \geq 0} V^k \text{ avec par convention } V^0 = \{\varepsilon\}
\end{aligned}
$$

Sur V^* la concaténation notée $||$ est une opération

$$
\left.\begin{array}{l} \text{- Interne} \\ \text{- Associative} \end{array}\right\} \Longrightarrow \langle V^*, || \rangle \text{ est un demi groupe} \left.\begin{array}{l} \\ \\ \text{- dotée d'un élément neutre } \varepsilon \end{array}\right\} \Longrightarrow \langle V^*, ||, \varepsilon \rangle \text{ est un monoïde libre}
$$

Un langage formel L construit à l'aide d'un vocabulaire V est un sous-ensemble du monoïde libre V^*. Un langage L sur V^* : $L \subseteq V^*$.

3.2.3 Les langages réguliers

L'ensemble des langages réguliers sur un vocabulaire V peut-être défini récursivement par les règles suivantes :

- le langage vide ϕ est un langage régulier ;
- le langage $\{\varepsilon\}$ est un langage régulier ;
- pour tout x de V, $\{x\}$ est un langage régulier ;
- si R_1 et R_2 sont des langages réguliers alors : $R_1 \cup R_2$, $R_1.R_2$ et R_1^* sont des langages réguliers.

Notons que certains langages ne sont pas réguliers, comme par exemple le langage $\{a^n.b^n | n > 0\}$ défini sur le vocabulaire $\{a,b\}$.

3.2.4 Expressions régulières

Les *expressions régulières* fournissent une autre méthode de description des langages réguliers. Les expressions régulières (E.R.) sur un alphabet V et les langages qu'elles décrivent sont définis récursivement de la manière suivante :

- ε est une E.R. qui décrit le langage $\{\varepsilon\}$;
- si $a \in V$, alors a est une E.R. qui décrit $\{a\}$;
- si r et s sont des E.R. qui décrivent respectivement les langages R et S :
  - alors $(r)|(s)$ est une E.R. décrivant $R \bigcup S$;
  - alors $(r)(s)$ est une E.R. dénotant RS ;
  - alors $(r)^*$ est une E.R. décrivant R^* ;
  - alors $(r)^+$ est une E.R. décrivant R^+ ;
- il n'y a pas d'autres expressions régulières.

Remarques : pour économiser des parenthèses, par convention, on établit les priorités décroissantes suivantes : $*$, concaténation, $|$.

Exemple : considérons le vocabulaire suivant $V_{fin} = \{f,i,n\}$, le langage $L_{fin} = \{(f|i|n)^* fin\} = \{fin, ffin, niffin, iiifin, \dots\}$ sur V_{fin}^* est un langage régulier où toutes les chaînes appartenant au langage sont composées de n'importe quelle chaîne composée des symboles du vocabulaire $\{f,i,n\}$, mais se terminant par la chaîne « fin ».

3.3 Grammaires formelles

3.3.1 Définition

Si un langage est fini, on peut le définir en listant l'ensemble des chaînes le composant : *définition en extension*. Si le langage est infini, la définition extensive est impossible, il est nécessaire d'avoir recours à une *définition en compréhension*. Le but est de trouver un moyen de spécifier un langage. Une grammaire formelle permet avec un nombre fini de règles de générer un langage donné.

[Chomsky, 1957, Chomsky, 1965] définit une grammaire G comme un quadruplet $G = (V_T, V_N, R, A)$, avec :

- V_T est un ensemble fini de symboles terminaux : l'alphabet sur lequel le langage généré est défini.
- V_N est l'ensemble des symboles non-terminaux
- $R \subseteq (V^+ \times V^*)$ où $V = V_T \bigcup V_N$ est un ensemble fini de règles de productions (ou règles de réécriture). Les règles sont donc de la forme $\alpha \rightarrow \beta$ où $\alpha \in V^+$ et $\beta \in V^*$.
- $A \in V_N$ appelé l'Axiome, est le symbole de départ

3.3.2 Classification de Chomsky

Chomsky a effectué une classification des grammaires formelles (appelée « hiérarchie de Chomsky ») en différents types selon leur capacité descriptive, c'est à dire selon la complexité des langages qu'elles engendrent. Les grammaires sont habituellement divisées en 4 types selon les contraintes posées sur les règles de productions :

Type 0 : Aucune restriction sur les règles.

Type 1 : *Grammaires sensibles au contexte*. Les règles doivent satisfaire la condition $|\alpha| \leq |\beta|$. Cela signifie intuitivement que le membre de droite doit contenir au moins autant de symboles que le membre de gauche.

Type 2 : *Grammaires hors-contexte*. Toutes les règles doivent avoir la forme $A \rightarrow \beta$ où $A \in V_N$. Intuitivement, cela signifie donc qu'une grammaire est hors-contexte si le membre de gauche de chaque règle est constitué d'un seul symbole non-terminal.

Type 3 : *Grammaires régulières*. Toutes les règles doivent avoir une des deux formes suivantes : $A \rightarrow \mu B$ ou $A \rightarrow \mu$ dans le cas des grammaires « régulières à gauche », $A \rightarrow B\mu$ ou $A \rightarrow \mu$ dans le cas des grammaires « régulières à droite » avec $A, B \in V_N$ et $\mu \in V_T$.

La relation entre ces 4 types de grammaire est bien sûr la suivante :

$$Type3 \subset Type2 \subset Type1 \subset Type0$$

Les grammaires hors contexte sont les plus utilisées pour le traitement du langage naturel, bien qu'il ait été prouvé que celui-ci n'est pas engendré par une grammaire de ce type [Pullum et Gazdar, 1982]. Cette utilisation répandue vient du bon compromis existant entre la capacité de description des grammaires hors-contexte et les restrictions qu'elles induisent au niveau de l'analyse grammaticale : ces restrictions permettent une analyse efficace, et la puissance de description des grammaires hors-contexte permet de décrire une grande partie de la structure d'un langage. Pour des applications restreintes concernant le traitement du langage naturel, les grammaires régulières sont préférées aux grammaires hors-contexte : puisque la partie visée du langage est déterminée, la capacité de description des grammaires régulières s'avère suffisante. Notamment dans le contexte de dialogue où nous avons à faire à des phrases courtes, il est aisément possible d'obtenir une grammaire régulière à partir d'une hors-contexte. De plus, leur

analyse grammaticale est plus efficace, en terme de rapidité, que celle des grammaires hors-contexte.

Dans la pratique, les langages réguliers ne sont donc pas utilisables pour décrire des phénomènes trop complexes, mais ils se prêtent facilement à la modélisation de phénomènes de portée réduite et localisée. Tout langage régulier peut être engendré par une grammaire régulière. Un exemple de grammaire régulière engendrant le langage de l'exemple 3.2.4 est la suivante :

$$G_{fin} = \left\{ V_{fin}, \{A,F,I\}, A, \left\{ \begin{array}{lcl} A & \rightarrow & f\ A \mid i\ A \mid n\ A \mid f\ F \\ F & \rightarrow & i\ I \\ I & \rightarrow & n \end{array} \right\} \right\}$$

À chacune des grammaires présentées, peut être associé un type d'automate qui accepte le langage engendré par cette grammaire. Ces automates sont des machines virtuelles qui permettent de signifier l'appartenance ou non d'une phrase à un langage. Le tableau 3.1 établit la correspondance entre les quatre grands types de grammaires présentées précédemment et quatre sortes d'automates.

TAB. 3.1 – Types d'automates acceptant les langages engendrés par les différentes grammaires

Grammaires	Automates
Grammaires non restreintes	Machine de Turing
Grammaires contextuelle	Automates linéaires bornés
Grammaires hors-contexte	Automates de type « push-down »
Grammaires régulières	Automates à états fini

Les grammaires utilisées dans le traitement effectué par les systèmes de dialogue sont généralement des grammaires régulières et hors-contexte. Dans un contexte de dialogue oral où les phrases sont finies et de taille limitée, il est possible d'obtenir une approximation d'une grammaire hors-contexte par une grammaire régulière [Mohri et Nederhof, 2001]. Les automates à états fini sont alors appropriés pour représenter les connaissances linguistiques dans les systèmes de dialogue et sont présentés dans la section suivante.

3.4 Automates À États Fini

3.4.1 Définition

Les automates sont des machines qui prennent en entrée une chaîne de symboles et qui effectuent un algorithme de reconnaissance de la chaîne. La chaîne est reconnue par l'automate *si et seulement si* celui-ci part d'un état initial, lit tous les symboles de la chaîne, puis s'arrête dans un état final. Le langage accepté par un automate à états fini est l'ensemble des chaînes dont les symboles font passer de l'état initial de l'automate jusqu'à un de ses états terminaux par une succession de transition utilisant tous ces symboles dans l'ordre.

Plus formellement, un automate à états fini est un quintuplet $A = (Q, V, \delta, s, F)$ où :
- Q est un ensemble fini d'états ;
- V est un ensemble fini de symboles (l'alphabet (d'entrée) ;
- $\delta : Q \times (V \cup \{\varepsilon\}) \rightarrow Q$ est la fonction (totale) de transition ;
- $s \in Q$ est l'état initial (ou de départ) ;

– $F \subseteq Q$ est un sous-ensemble de Q, les états accepteurs (ou terminaux).

δ est la fonction de transition de l'automate. $\delta(e,a) = e'$ indique que si l'automate est dans l'état e et qu'il rencontre le symbole a, il passe dans l'état e'. De plus, pour tout e de Q, $\delta(e,\varepsilon) = e$.

Un automate à états fini possède les propriétés suivantes :

- le langage accepté par un automate à état fini est un langage régulier ;
- à partir de toute expression régulière r décrivant un langage (régulier) R il est possible de construire un automate à état fini acceptant ce même langage ;
- à partir de tout automate à état fini acceptant un langage (régulier) R il est possible de construire un automate à état fini déterministe acceptant le même langage.

Un automate associé à l'expression régulière 3.2.4 est visible dans la figure 3.1. Cet automate à état fini est *non-déterministe*. C'est à dire qu'il existe au moins un couple, formé d'un état et d'un symbole, qui admette plus d'une image par la fonction de transition ($\delta(1,f) = 1$ et $\delta(1,f) = 2$ dans l'exemple de la figure 3.1). Dans ce cas l'automate doit faire des choix pour progresser. Un automate non-déterministe reconnaît une phrase si parmi tous les choix possibles pour l'analyser, il en existe au moins un qui le laisse dans un état final. L'inconvénient d'un automate non-déterministe est d'analyser les chaînes plus lentement que les automates déterministes. En effet, si un mauvais choix est pris dans une transition, il faut revenir en arrière jusqu'à ce choix pour parvenir à la reconnaissance de la chaîne. Par exemple la chaîne « ffin », à la lecture du premier symbole deux possibilités sont offertes, passage à l'état 1, ou rester en l'état 0. Si l'on passe à l'état 1, nous sommes dans un état ou il est alors impossible d'effectuer de transition, or la chaîne n'est pas encore entièrement lue, dans ce cas il faut donc revenir à l'état 0 et choisir de rester. La complexité de la reconnaissance d'une chaîne par un automate non-déterministe est dite exponentielle. Si, à chaque état et pour chaque symbole, existent n transitions possibles, la reconnaissance d'une chaîne de longueur p demandera jusqu'a n^p transitions. Un algorithme permet de déterminiser tout automate à états fini, c'est à dire de faire en sorte que sa fonction de transition n'admette pour tout état et tout symbole, qu'*au plus* un état dans lequel l'automate peut transiter. La version *déterministe* de l'automate de la figure 3.1 est visible dans la figure 3.2.

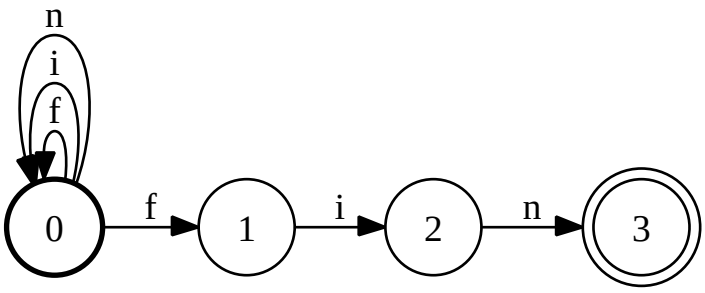

FIG. 3.1 – *Automate associé à l'expression régulière 3.2.4*

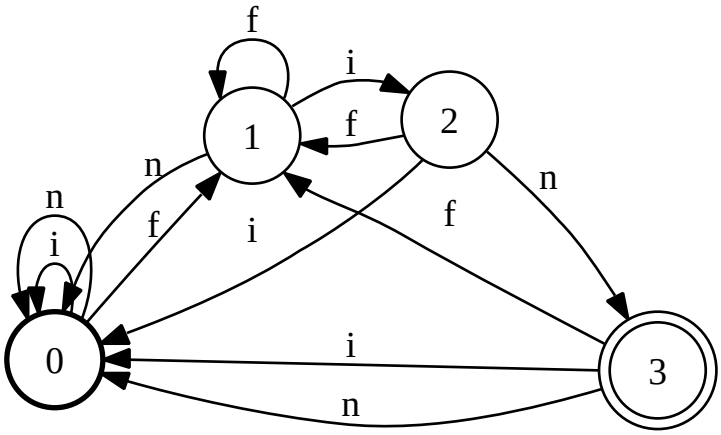

FIG. 3.2 – *Automate déterministe associé à l'expression régulière 3.2.4*

3.4.2 Définition d'un FSM et algorithmes associés

Toutes les opérations présentées sur les FSMs sont faites avec la suite d'outils d'AT&T ([Mohri *et al.*, 1997]). Suivant les définitions utilisées dans [Mohri *et al.*, 2002], les accepteurs et transducteurs utilisés dans cette étude sont définis en accord avec la notion générale algébrique du semi-anneau (*semiring*). Un semi-anneau K est un ensemble $\mathbb{K}$ avec une opération associative et commutative $\oplus$, une opération associative $\otimes$, ainsi que deux éléments neutres $\overline{0}$ et $\overline{1}$: $K = (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$. Les poids associés aux hypothèses générées par le module de reconnaissance automatique de la parole (RAP) représentent les probabilités codées en $-\log$. Le semi-anneau correspondant est appelé le *log semiring* : $(\mathbb{R}, +, ., 0, 1)$. Lorsque sont utilisées les probabilités en $-\log$ pour une approximation du meilleur chemin, un semi-anneau tropical (*tropical semiring*) est utilisé : $(\mathbb{R}_+ \cup \infty, min, +, \infty, 0)$.

TAB. 3.2 – Semi-anneaux utilisés

Semi Anneau	Ensemble	Addition (+)	Produit ($\otimes$)	Inférieur	El. Ne. (+)	El. Ne. $\otimes$
Log	$[-\infty, \infty]$	$-\log(e^{-x} + e^{-y})$	+	<	∞	0
Tropical	$[-\infty, \infty]$	*min*	+	<	∞	0

Les accepteurs et transducteurs sont définis comme suit :

Soient Σ un alphabet de symboles d'entrées ; Δ un alphabet de symboles de sorties ; ε un symbole vide ; Q un ensemble d'états (avec I=état initial et F=états terminaux) ; $\mathbb{K}$ un semi-anneau ; E un ensemble de transitions défini comme : $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathbb{K} \times Q$; w une fonction de pondération : $w : Q \to \mathbb{K}$.

Si $Path(R_1, x, R_2)$ est un ensemble de chemins de $R_1 \subseteq Q$ à $R_2 \subseteq Q$ avec un symbole d'entrée x et $Path(R_1, x, y, R_2)$ un ensemble de chemins dans $Path(R_1, x, R_2)$ avec un symbole de sortie y, alors :

– Accepteur $A = (\Sigma, Q, I, F, E)$ avec pour tout $x \in \Sigma$:

$$[A](x) = \bigoplus_{\pi \in Path(I,x,F)} w[\pi] \tag{3.1}$$

– Transducteur $T = (\Sigma, \Delta, Q, I, F, E)$ avec pour tout $x \in \Sigma^*, y \in \Delta^*$:

$$[T](x,y) = \bigoplus_{\pi \in Path(I,x,y,F)} w[\pi] \tag{3.2}$$

et $w[\pi] = w[t_1] \otimes w[t_2] \otimes \ldots \otimes w[t_n]$ pour un chemin π fait des transitions suivantes $t_1, t_2, \ldots, t_n$.

Différentes opérations fondamentales sur les FSMs utilisées dans ce document sont présentées :

L'union

Soit l'automate A_1 acceptant un langage $L_1 \subseteq V^*$ (exemple figure 3.3) et un automate A_2 acceptant un langage $L_2 \subseteq V^*$ (exemple figure 3.4), l'automate $A_{1\cup 2}$ (exemple figure 3.5) résultant de l'union des deux automates A_1 et A_2 au moyen de l'opération :

$$[A_1 \cup A_2](x) = [A_1](x) \cup [A_2](x)$$

accepte un langage $L_{1\cup 2} = L_1 \cup L_2$, le même en version déterministe est visible dans la figure 3.6.

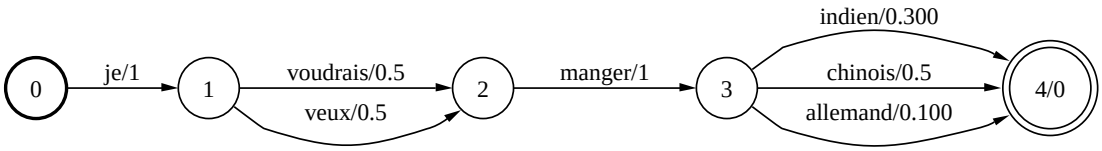

FIG. 3.3 – *Exemple d'automate A_1 acceptant un langage L_1*

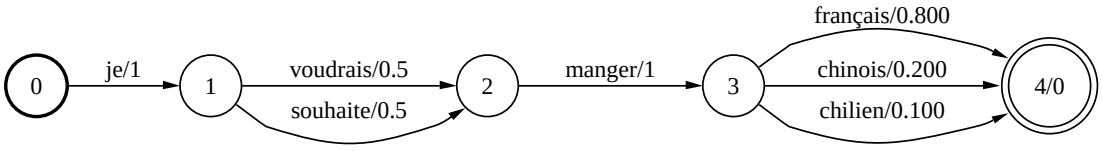

FIG. 3.4 – *Exemple d'automate A_2 acceptant un langage L_2*

L'intersection

Soit l'automate A_1 acceptant un langage $L_1 \subseteq V^*$ (exemple figure 3.3) et un automate A_2 acceptant un langage $L_2 \subseteq V^*$ (exemple figure 3.4), l'automate $A_{1\cap 2}$ (exemple figure

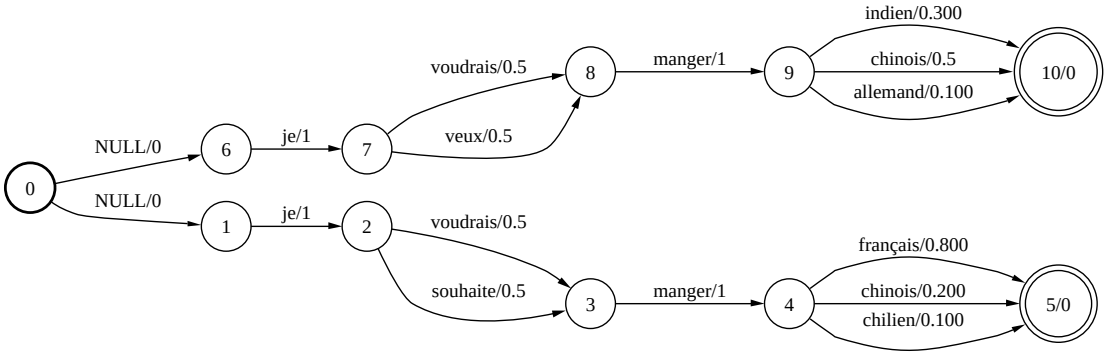

FIG. 3.5 – *Exemple d'automate $A_{1\cup 2}$ union des automates A_1 et A_2*

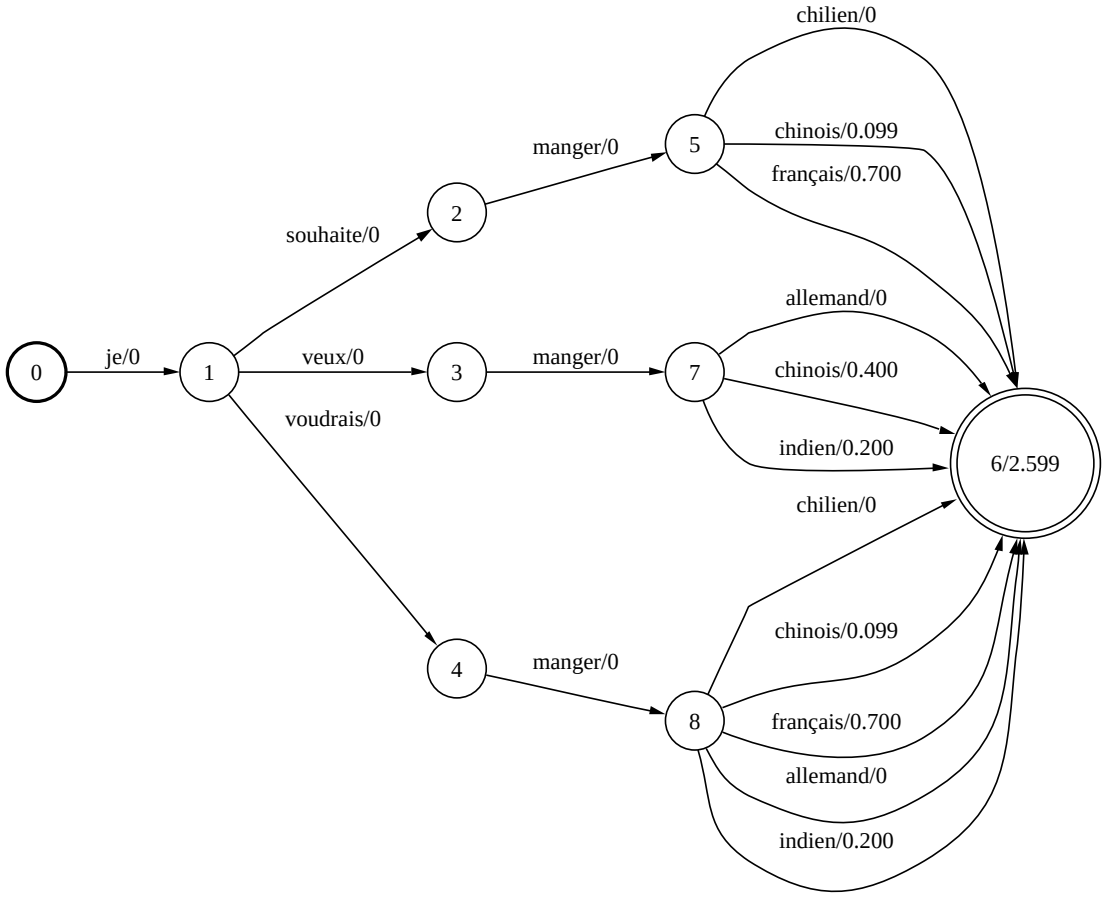

FIG. 3.6 – *Exemple d'automate $A_{1\cup 2}$ déterministe*

3.7) résultant de l'intersection des deux automates A_1 et A_2 au moyen de l'opération suivante :

$$[A_1 \cap A_2](x) = [A_1](x) \otimes [A_2](x)$$

accepte un langage $L_{1\cap 2} = L_1 \cap L_2$.

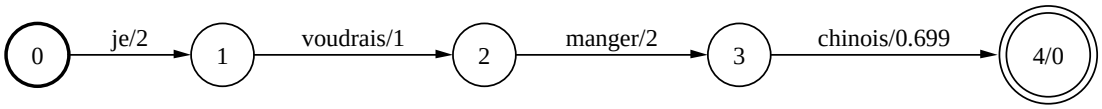

FIG. 3.7 – *Exemple d'automate $A_{1\cap 2}$ intersection des automates A_1 et A_2*

La différence

Soit l'automate A_1 acceptant un langage $L_1 \subseteq V^*$ (exemple figure 3.3) et un automate A_2 acceptant un langage $L_2 \subseteq V^*$ (exemple figure 3.4), l'automate $A_{1\cap\bar{2}}$ (exemple figure 3.8) résultant de la différence des deux automates A_1 et A_2 au moyen de l'opération suivante :

$$[A_1 - A_2](x) = [A_1 \cap \overline{A_2}](x)$$

accepte un langage $L_{1\cap\bar{2}} = L_1 \cap \overline{L_2}$.

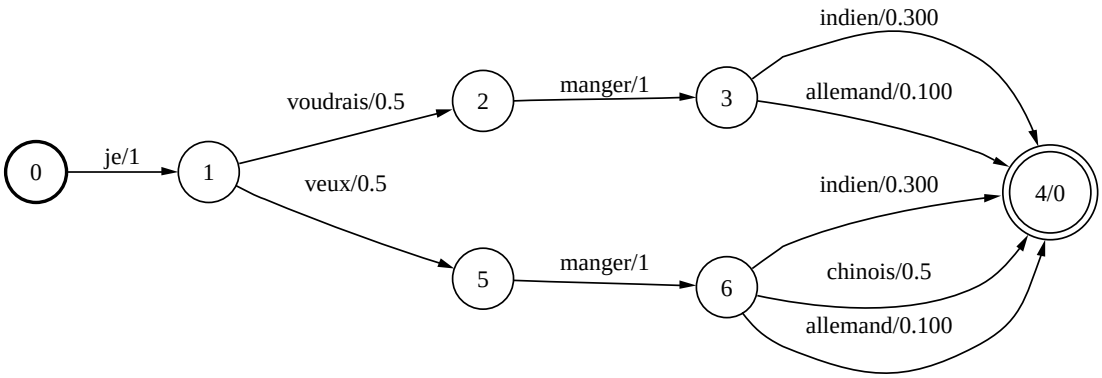

FIG. 3.8 – *Exemple d'automate $A_{1\cap\bar{2}}$ différence entre les automates A_1 et A_2*

La projection

L'opération de projection à partir d'un transducteur permet d'obtenir un accepteur de même topologie, avec les transitions étiquetées comme les entrées du transducteur si cette projection se fait sur les entrées, où comme les sorties si cette projection se fait sur les sorties :

$$[A](x) = \bigoplus_y [T](x,y) \text{ et } [A](y) = \bigoplus_x [T](x,y)$$

La composition

La composition est une opération entre deux transducteurs. Elle permet d'aligner les symboles de sortie du premier transducteur (exemple figure 3.9) avec les symboles d'entrée du second (exemple figure 3.10) et de produire un transducteur dont les entrées sont celles du premier transducteur et les sorties celles du second (exemple figure 3.11).

$[T_1 \circ T_2](x,y) = \bigoplus_z [T_1](x,z) \otimes [T_2](z,y)$

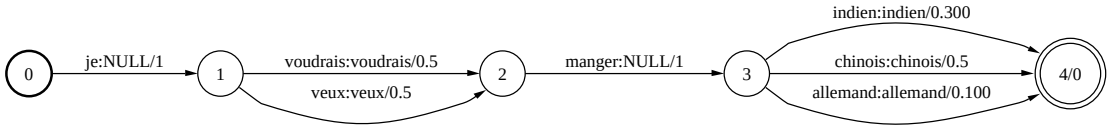

FIG. 3.9 – *Exemple de transducteur T_1*

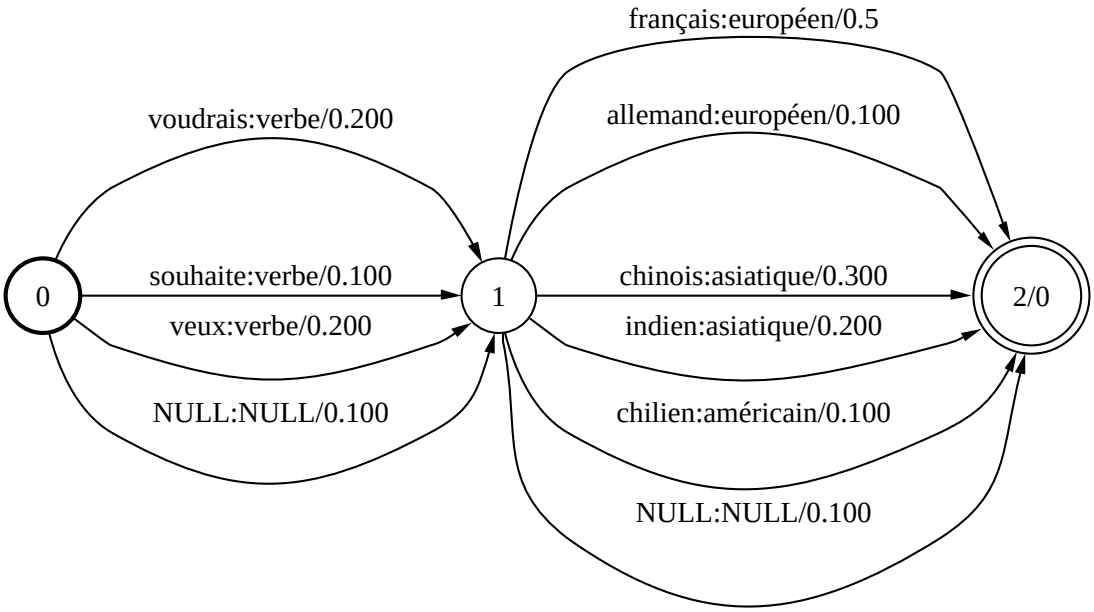

FIG. 3.10 – *Exemple de transducteur T_2*

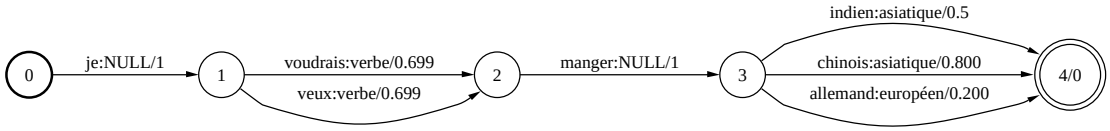

FIG. 3.11 – *Exemple de transducteur $T_{1\circ 2}$ résultant de la composition entre les transducteurs T_1 et T_2*

3.5 Méthodes de classification

3.5.1 Les arbres de décision sémantique

L'application des arbres de décisions sémantiques au langage naturel a été introduite dans [Kuhn et De Mori, 1995]. La nouveauté des algorithmes de SCT réside dans la construction des questions pour l'apprentissage de l'arbre de décision. Cette construction se fait à partir d'un ensemble d'expressions régulières Π_0 associé à la racine de l'arbre :

soient V, l'ensemble des mots du vocabulaire du corpus d'entraînement C_0, $w \in V$ n'importe quel mot du vocabulaire, le symbole « + » indiquant la présence de n'importe quelle séquence de mots non vide dans l'expression régulière, Π_0 contient les quatre éléments suivants :

$$\Pi_0 = \{w, +w, w+, +w+\}$$

L'ensemble des questions appliquées à la racine, N_0, est obtenu en considérant toutes les expressions régulières Π_0 possibles appliquées sur tous les w du vocabulaire V selon les quatre éléments de Π_0. Il y a donc $4 \times |V|$ possibilités où $|V|$ représente la taille du vocabulaire. Chaque expression régulière est alors testée à la racine de l'arbre. Le critère d'impureté de Gini est utilisé pour choisir la meilleure question à chaque nœud N_i. Soient les k classes $c_1, c_2, ..., c_k$ dont les probabilités de répartition sont $p_1, p_2, ..., p_k$ alors le critère de Gini du nœud N_i s'exprime par :

$$G(N_i) = 1 - \sum_{j=0}^{k} {p_j}^2.$$

La meilleure question pour N_i est celle qui apporte la plus grande variation d'impureté entre N_i et ses fils. Si les deux enfants de N_i sont notés N_{i+1}^{oui} et N_{i+1}^{non}, la variation d'impureté Δ_i est alors définie par :

$$\Delta_i = G(N_i) - \frac{|N_{i+1}^{oui}| \times G(N_{i+1}^{oui}) + |N_{i+1}^{non}| \times G(N_{i+1}^{non})}{|N_i|}.$$

Par exemple, si $+W$ est l'expression régulière qui maximise Δi, la phrase qui est acceptée par cette expression régulière fait partie d'un corpus C_1^{oui} associé au fils de la branche du corpus nommé *OUI*. Les autres font partie du corpus C_1^{non} associées au fils nommé *NON*. L'ensemble des questions associé à C_1^{oui} est obtenu par la substitution suivante : $+ \rightarrow \Pi_0$ nous menant à $\Pi_0 W$. En général, étant donnée une expression régulière : $+W_1+W_2+\ldots+W_i+\ldots$ où W_i est déjà déterminé par une série de mots, les questions sont générées en remplaçant chaque $+$ par Π_0, c'est à dire :

$$\begin{aligned} &\Pi_0 W_1 + W_2 + \ldots + W_i + \ldots \\ &+W_1 \Pi_0 W_2 + \ldots + W_i + \ldots \\ &+W_1 + W_2 \Pi_0 \ldots + W_i + \ldots \\ &\ldots \end{aligned}$$

Soit $|+|$ le nombre de symboles $+$ dans l'expression régulière originale, le nombre de cas est alors : $4 \times |+| \times |V|$.

Lorsque le SCT est construit, il prend des décisions sur la base de règles de classification statistique apprises sur ces expressions régulières. Les règles apprises par le SCT sont donc résistantes aux diverses formulations du locuteur car elles ne dépendent que d'un petit nombre de mots. En revanche, si un seul des mots composants de la règle est mal reconnu, elle ne s'applique pas.

La figure 3.12 est un exemple d'arbre.

3.5.2 Les algorithmes de Boosting

Le Boosting est une méthode générale pour améliorer les performances de n'importe quel algorithme d'apprentissage. Par définition, un algorithme de Boosting peut théoriquement transformer un algorithme d'apprentissage basique (appelé « weak learner »)

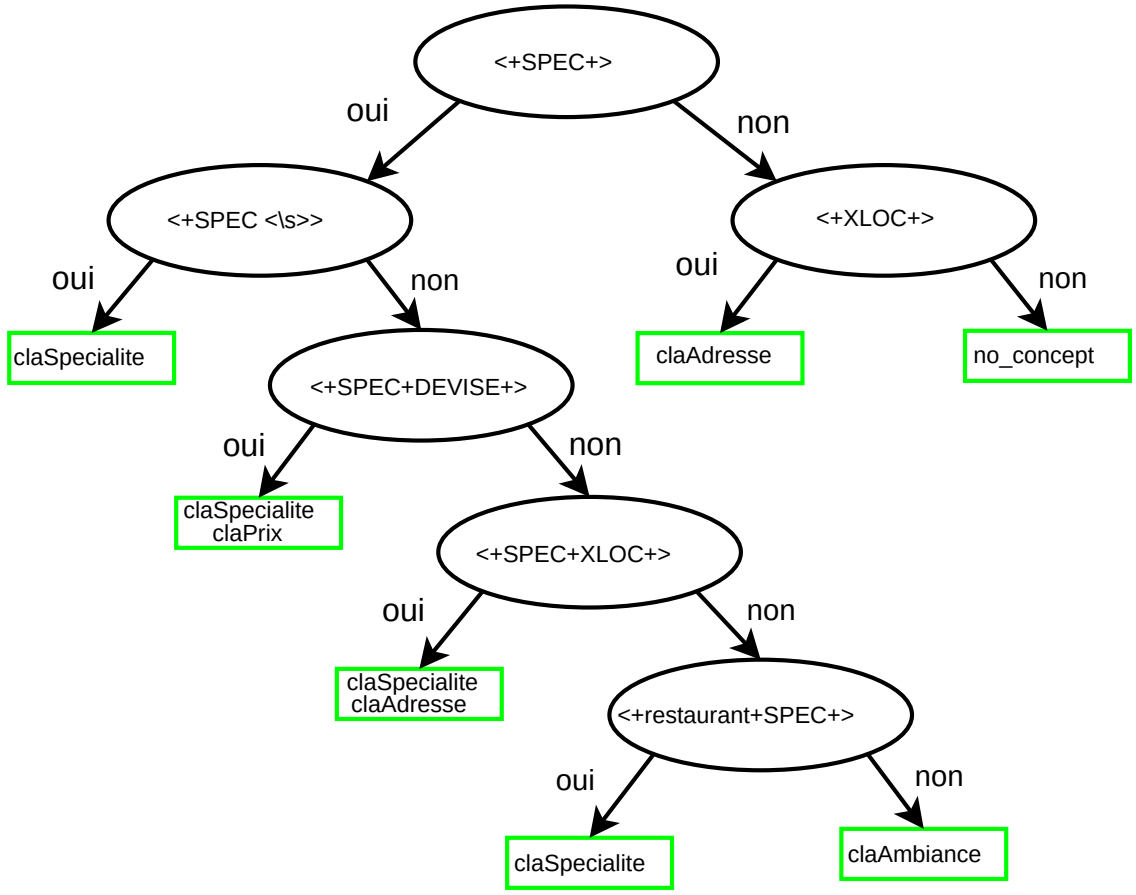

FIG. 3.12 – *Schéma simplifié d'un arbre de classification sémantique*

avec des performances juste un peu meilleures que le hasard en un algorithme très performant. Le principe de ces algorithmes de Boosting est re-pondérer à plusieurs reprises les exemples d'apprentissage et de refaire tourner l'algorithme d'apprentissage sur ces exemples re-pondérés. Les exemples mal classés sont re-pondérés à la hausse tandis que les biens classés sont re-pondérés à la baisse. Ainsi, le boosting force cet algorithme à concentrer ses efforts d'apprentissage sur les exemples les plus difficiles. L'hypothèse finale est un vote pondéré des différentes hypothèses obtenues à chaque instance de l'algorithme d'apprentissage. L'algorithme AdaBoost [Freund et Schapire, 1996] est présenté dans la figure 3.13.

3.5.3 Les Machines à Vecteur de Support, SVMs

Les SVMs, proposés par Vapnik ([Vapnik, 1982, Vapnik, 1995]), sont une classe d'algorithmes d'apprentissage qui ont été utilisés avec succès dans plusieurs tâches d'apprentissage. Ils permettent de construire un classifieur à valeurs réelles qui découpent le problème de classification en 2 sous-problèmes : transformation non-linéaire des entrées et choix d'une séparation linéaire *optimale*. Ils offrent en particulier une bonne approximation du principe de minimisation du risque structurel. L'idée de la minimisation du risque structurel est de trouver une hypothèse h pour laquelle l'erreur vraie minimale est garantie. L'erreur vraie de h est la probabilité que h fasse une erreur sur un exemple non-vu et extrait aléatoirement du corpus de test.
Le premier sous-problème à traiter est donc celui de travailler dans un espace où les

Entrées :

1. un ensemble d'apprentissage de m exemples $\langle (x_1,y_1),\ldots,(x_m,y_m)\rangle$ avec le label $y_i \in Y = \{1,\ldots,k\}$;
2. un nombre d'itérations T ;
3. un classifieur basique CB.

Initialiser la distribution $D_1(i) = \frac{1}{m}$ pour tout i
Faire pour $t = 1,2,\ldots,T$

1. Appel à CB avec la distribution D_t.
2. Récupérer une hypothèse $h_t : X \to Y$
3. Calculer l'erreur de $h_t : \varepsilon = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$.
4. Fixer $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
5. Mettre à jour la distribution D_t : $D_{t+1}(i) = \frac{D_t(i)}{Z_i} \times \left\{ \begin{array}{ll} \beta_t & si\ h_t(x_i) = y_i \\ 1 & sinon \end{array} \right\}$
   où Z_t est une constante de normalisation (choisie afin que D_{t+1} soit une distribution)

Sortir l'hypothèse finale : $h_{fin}(x) = \text{ArgMax}_{y\in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

FIG. 3.13 – *Algorithme AdaBoost*

données soient linéairement séparables. Les données sont alors projetées dans un espace de grande dimension par une transformation basée sur un noyau (voir figure 3.14). Le noyau est une fonction qui retourne la valeur du produit scalaire des images des 2 arguments $K(x_1,x_2) = \langle \Phi(x_1), \Phi(x_2)\rangle$. Il peut être linéaire, polynomial ou gaussien .

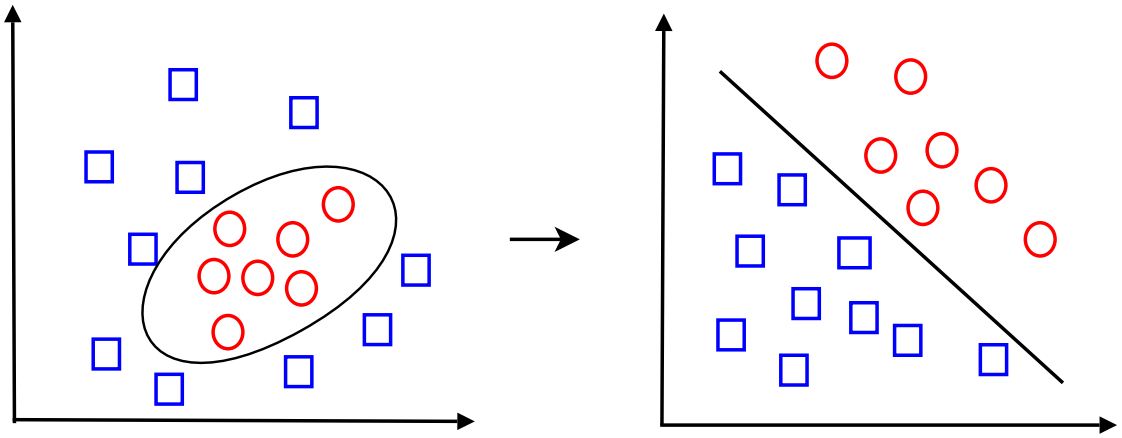

FIG. 3.14 – *Projection des données d'entrée dans un espace où elles sont linéairement séparables*

Le deuxième sous-problème est traité dans cet espace transformé. Les classes y sont séparées par des classifieurs linéaires qui déterminent un hyper-plan optimal. L'hyper-plan optimal est celui qui sépare correctement toutes les données et qui maximise la marge, la distance du point le plus proche à l'hyper-plan (représentée par d dans la figure 3.15). Les hyperplans peuvent être déterminés au moyen d'un nombre de points limité qui seront appelés les « vecteurs supports ».

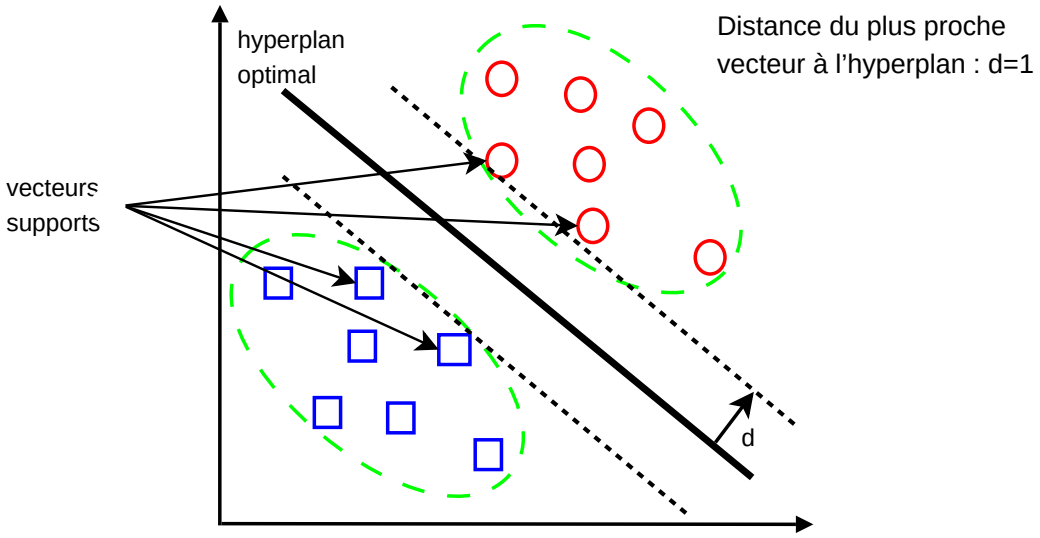

FIG. 3.15 – *Hyper-plan optimal et marge maximale*

Pour appliquer les SVMs sur du texte, la technique la plus simple employée est celle du sac de mots où tous les mots sont représentés par des chiffres. Le lexique complet représente alors un vecteur et chaque phrase sera codée par ce vecteur.
L'utilisation des SVMs dans ce cas est alors pleinement justifiée [Joachims, 1998].

- Ils ont le potentiel pour gérer ce grand nombre de données.
- Le vecteur représentant chaque phrase contient peu de données qui ne sont pas des 0. Les SVMs sont adaptés aux problèmes de ce type (vecteurs creux).
- La plupart des problèmes de catégorisation de textes sont linéairement séparables.

Deuxième partie

Contribution

Introduction

Dans les applications de dialogue oral, le module de compréhension analyse la transcription générée par le moteur de reconnaissance automatique de la parole (RAP) pour en produire une interprétation exploitable par le gestionnaire de dialogue. Or dans ce contexte de dialogue oral, la transcription automatique produite par le RAP n'est pas fiable. Ceci est dû à plusieurs facteurs :

– au contexte téléphonique : mauvaise qualité du signal, bruits parasites,. . . ;
– aux phénomènes de la parole spontanée, hésitations, agrammaticalité,. . . ;
– à la méthode de transcription elle même.

Le système de RAP utilise des connaissances acoustiques pour extraire les mots du signal de parole tandis que les connaissances linguistiques permettent d'assurer une cohérence syntaxique sur la phrase complète. Mais transcrire, c'est comprendre. En effet, si l'on considère les 3 hypothèses de transcription suivantes :

1 Patrick Bruel est dans l'île de France
2 Patrick Bruel est dans le lit de France
3 Patrick Bruel aidant le lit de France

on peut remarquer qu'elles sont phonétiquement quasi-identiques et qu'elles sont syntaxiquement correctes. Le modèle acoustique ne sera pas en mesure de les discriminer. Un modèle de langage *N*-grammes avec N>2 pourrait rejeter l'hypothèse 3, le syntagme « aidant le lit » étant vraisemblablement peu probable, tandis qu'un modèle *Bi*-grammes, avec N restreint à une historique de 1 mot ne pourrait détecter cette incohérence. En limitant la recherche de la bonne transcription aux seuls critères acoustiques et linguistiques, trouver la bonne solution devient hasardeux. Des informations de plus haut niveau, liées à la compréhension peuvent nous aider à discriminer des hypothèses qui ont un sens ou non : les hypothèses 1 et 2 ont un sens alors que l'hypothèse 3 n'en a pas. Afin de faire un choix entre l'hypothèse 1 et 2, qui sont, toutes deux, cohérentes et aussi probables, le contexte du dialogue est nécessaire pour trouver la transcription correcte.

La performance du module de compréhension s'appuyant sur la transcription pour extraire une interprétation de la parole de l'utilisateur est donc dépendante de la qualité de la transcription. Or, nous voyons que le processus de transcription pour être plus performant a besoin d'informations liées à la compréhension. L'utilisation séquentielle de ces deux modules est une des faiblesses de cette architecture de dialogue. Contrai-

rement à une application de dictée vocale où l'objectif est de transcrire fidèlement tous les mots prononcés par l'utilisateur, l'objectif d'une application de dialogue est de comprendre le sens des paroles de l'utilisateur. La transcription n'a pas vocation à être lue et n'est ici qu'une étape intermédiaire nécessaire. À la différence de la dictée vocale, la transcription n'a pas besoin d'être totalement fidèle, seuls les mots utiles à la compréhension doivent être correctement reconnus. Dans les deux cas des informations liées à la compréhension doivent aider la génération de la transcription. Dans le cas de dictées vocales où aucune restriction n'existe, il n'est pas possible de modéliser les informations sémantiques. Dans une application de dialogue, la sémantique est connue et restreinte, l'exploiter est alors chose possible.

Dans les applications de dialogue oral dans lesquelles nous situons notre travail, l'opération de compréhension s'appuie sur une interprétation conceptuelle de la phrase transcrite pour construire une représentation sémantique. Cette interprétation conceptuelle peut être vue comme l'ensemble des concepts qui ont été détectés dans la phrase. Indépendamment de la méthode d'analyse en compréhension utilisée, la détection de ces concepts est une étape particulièrement importante. Ces concepts sont obtenus en associant des mots ou groupes de mots à des étiquettes conceptuelles et contiennent les informations nécessaires au système, pour d'une part interroger la base de données afin de répondre à l'utilisateur, d'autre part permettre au gestionnaire de dialogue d'interpréter les attentes de l'utilisateur. Plus que la transcription en mots, c'est la génération de cette interprétation (ensemble de concepts) qui est importante pour le système.

Dans l'architecture séquentielle de la plupart de ces systèmes, l'extraction de ces concepts est faite postérieurement à la recherche de la meilleure transcription qui est assurée par des modèles basés sur l'acoustique et des contraintes linguistiques réduites (N-grammes). Nous proposons une architecture de système permettant de tenir compte de cette information conceptuelle lors du décodage. Elle donne la possibilité d'enrichir le graphe de mots, généré par le module RAP, de ces informations conceptuelles afin de pouvoir articuler les deux opérations de transcription et génération d'interprétation conceptuelle.

Dans le chapitre 5, nous proposons un modèle de langage conceptuel contenant les informations nécessaires afin de pouvoir passer des mots à ces concepts. Ce modèle associe à chaque concept une grammaire régulière permettant de le détecter. Notre modèle basé sur le formalisme des transducteurs à états fini permet de segmenter le graphe de mots selon les concepts qu'il contient. Un chemin dans ce graphe associe chaque phrase avec son ou ses interprétations (en cas d'ambiguïtés qui doivent être levées à un niveau supérieur). Il est alors aisé de passer du graphe de mots à un graphe de concepts. Nous proposons comme sortie de notre architecture une liste structurée des N-meilleures hypothèses. Cette liste créée à partir du graphe de mots enrichi par notre modèle, présente les N-meilleures interprétations existantes dans le graphe avec leur N-meilleures phrases supports qui ont un sens différent du point de vue de la compréhension (même séquence de concepts mais au moins un concept avec une valeur différente). Cette liste correspond à un résumé exhaustif de toutes les informations conceptuelles nécessaires à la compréhension dans le système de dialogue.

Dans les systèmes statistiques, les meilleures hypothèses de mots ou de compréhension sont choisies en fonction de la vraisemblance calculée selon des distributions statistiques apprises sur des corpus souvent de taille réduite. Elle n'est pas optimale. Il est important de mettre en place des critères permettant d'estimer la qualité du processus de RAP. Dans le chapitre 6 sont proposées différentes mesures de confiance opérant à des niveaux différents : acoustique, linguistique et conceptuel. Ces mesures

de confiance permettent de diagnostiquer la réponse de notre module RAP à différents niveaux : mot, phrase et concept.

En fonction de la sortie du module de RAP et des mesures de confiance associées le système de dialogue se doit d'avoir une stratégie cohérente : doit-il faire confiance à la sortie totalement ? partiellement ? et demander une confirmation ou au contraire rejeter l'intervention et demander une répétition. Cette décision appartient au gestionnaire de dialogue qui doit assurer la satisfaction maximale de l'utilisateur. Nous proposons dans le chapitre 7 une stratégie de décision qui, en fonction de la sortie de notre décodage (*i.e.* la liste structurée des N-meilleures hypothèses) et des mesures de confiance présentées, définit des états de fiabilité permettant de guider le gestionnaire de dialogue dans les choix qu'il aura à prendre.

Les travaux de cette thèse ont été financés par France Télécom Recherche et développement. Ces travaux seront illustrés par des exemples concrets d'applications développées par France Télécom et les expériences évaluées sur les corpus de données qui ont été mis à notre disposition. Le chapitre 4 décrit les deux applications de serveur vocal utilisées dans cette thèse ainsi que les données fournies.

CHAPITRE

4

Description des données expérimentales

Sommaire

Les expériences dans les travaux présentés dans ce document ont pu être effectuées grâce à France Télécom Recherche et Développement qui a fourni les données. Les données sont celles de deux applications de dialogue homme-machine par téléphone, AGS et PlanResto.

4.1 Application AGS

Le démonstrateur Audiotel Guide des Services (AGS) est une application de dialogue homme-machine par téléphone, elle est décrite dans [Sadek *et al.*, 1996]. Le démonstrateur AGS est utilisé afin de fournir à un utilisateur humain des numéros de téléphone de serveurs vocaux spécialisés dans les prévisions météorologiques ou la recherche d'emploi. Le dialogue qui s'établit par téléphone entre le démonstrateur et l'utilisateur hu-

main a pour but de guider l'utilisateur vers le serveur le plus pertinent vis-à-vis de sa demande de renseignements.

4.1.1 Données d'apprentissage

Les données d'apprentissage se présentent sous la forme d'un corpus de transcriptions de phrases prononcées par des utilisateurs du démonstrateur AGS. Il ne s'agit pas d'un grand corpus, puisqu'il est composé de 9842 phrases, pour 49591 mots, dont 821 différents. Ces phrases ont été récupérées à partir d'une collecte de données effectuées à l'aide de locuteurs naïfs et de locuteur experts. Les locuteurs naïfs sont des personnes externes ne travaillant pas pour France Télécom R&D et n'ayant pas de connaissances en reconnaissance de la parole. Les locuteurs experts travaillent pour France Télécom R&D. Les 821 mots du corpus d'apprentissage font partie des 880 mots du lexique du démonstrateur AGS. Plus de détails sur l'acquisition des corpora de test et d'apprentissage sont donnés dans [Damnati, 2000].

Les phrases du corpus d'apprentissage sont des questions, des requêtes, des réponses, ou des commandes ("annulation", par exemple). Elles concernent toutes l'application AGS. Une étude plus précise de ces phrases permet de noter qu'une grande partie d'entre elles (59%) sont des phrases courtes (1 à 4 mots). La figure 4.1 montre la répartition des phrases en fonction de leur nombre de mots.

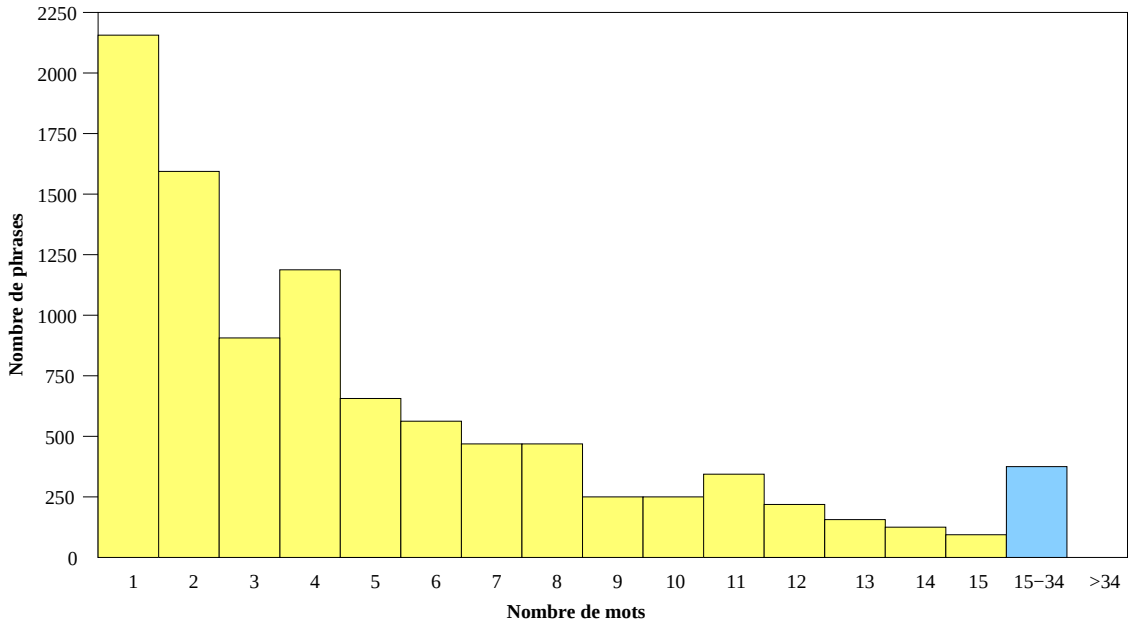

FIG. 4.1 – *Répartition des phrases du corpus d'apprentissage AGS en fonction du nombre de mots qui les composent*

4.1.2 Données de test

Les données de test sont des graphes de mots issus du processus de reconnaissance de la parole du démonstrateur AGS. Chacun de ces graphes de mots est associé à une phrase, appelée phrase de référence, qui correspond à la phrase effectivement prononcée par le locuteur. Les scores acoustiques associés aux mots dans un graphe sont calculés lors de la génération du graphe par le module de reconnaissance de la parole du démonstrateur AGS.

Les phrases de référence sont au nombre de 1422, composés de 7014 mots, dont 504 mots différents. La nature et la longueur de ces phrases sont semblables aux phrases du corpus d'apprentissage : la figure 4.2 illustre la répartition des phrases de référence en fonction de leur nombre de mots.

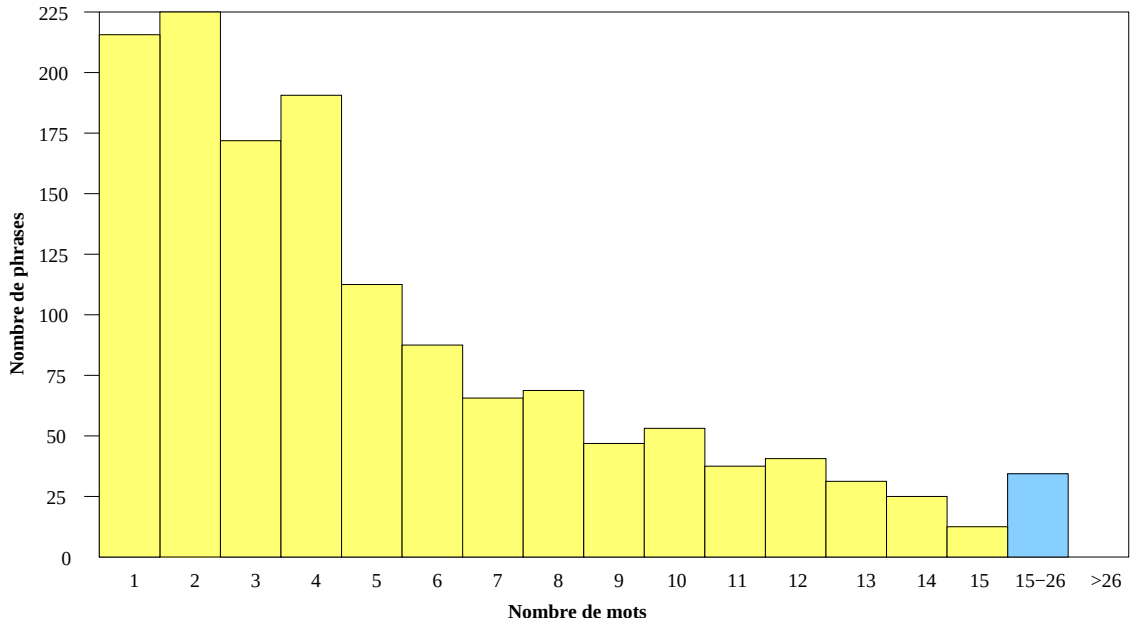

FIG. 4.2 – *Répartition des phrases de référence du corpus de test AGS en fonction du nombre de mots qui les composent*

Il est intéressant de noter que sur les 504 mots différents des phrases de référence du corpus de test, 109 mots n'apparaissent pas dans le corpus d'apprentissage. Certains de ces mots n'appartiennent pas au lexique : ce sont des mots dits hors-vocabulaire. Ces 109 mots affectent 187 phrases du corpus de test, soit $13,15\%$ des phrases de référence. Pour gérer les mots hors-vocabulaire, une entrée lexicale notée <UNK> représentant les mots inconnus est ajoutée au lexique. Au niveau de la modélisation du langage, les événements non vus sont gérés par les techniques de lissage[1].

En dehors du problème des mots hors-vocabulaire qui affectent les performances des modèles de langage et qui a donc une incidence sur les performances globales d'un système de reconnaissance, d'autres facteurs peuvent intervenir. Le décodage acoustique, qui génère les graphes de mots, peut connaître quelques difficultés. Dans le cas du démonstrateur AGS, les conditions d'acquisition de la parole sont difficiles : utilisation du téléphone, environnements sonores différents et bruités, locuteurs différents, ... Ces conditions, associées à un lexique fermé de 880 mots, et à un élagage plus ou moins fort de l'espace de recherche, compliquent la production de graphes de mots contenant des hypothèses acoustiquement fiables. Ainsi, pour environ $24,5\%$ des graphes, la phrase de référence n'est pas présente. Dans ce cas, il est impossible de retrouver la phrase prononcée par le locuteur à partir du graphe de mots : les hypothèses issues du processus de reconnaissance seront forcément erronées.

Les phrases du corpus de test peuvent être regroupées en fonction du locuteur qui les a prononcées. Il existe six locuteurs identifiés (l_1, l_2, l_3, l_4, l_5 et l_6), et un panel de locuteurs anonymes. Ce panel est nommé p_0. Le tableau 4.1 montre le nombre de phrases prononcées par chaque locuteur, ainsi que le nombre de sessions de dialogue corres-

[1] voir la section 1.5, consacrée au lissage

pondantes. Une session de dialogue correspond à un appel du locuteur et à l'intégralité du dialogue associé à cet appel.

TAB. 4.1 – Répartition des mots, des phrases et des sessions du corpus de test AGS en fonction du locuteur

locuteur	nombre de sessions	nombre de phrases
l_1	74	574
l_2	13	166
l_3	9	91
l_4	12	122
l_5	14	136
l_6	15	209
p_0	25	124

4.2 Application PlanResto

L'application PlanResto est une application de dialogue homme-machine par téléphone permettant à un utilisateur de rechercher un restaurant sur Paris. Il est censé fournir les mêmes services que l'application WEB PlanResto disponible à l'adresse http://paris.planresto.fr/.

4.2.1 Données d'apprentissage

Le corpus d'apprentissage est composé de 6608 transcriptions manuelles pour un total de 27838 mots dont 1130 uniques. La figure 4.3 illustre la répartition des phrases de référence en fonction de leur nombre de mots.

4.2.2 Données de développement

Le corpus de développement est composé de 3997 graphes de mots issus du moteur de reconnaissance de la parole (RAP) de France Telecom. À chaque graphe est associé sa référence, la phrase transcrite manuellement. Elles comportent 16239 mots dont 641 différents. La figure 4.4 illustre la répartition des phrases de référence en fonction de leur nombre de mots.

4.2.3 Données de test

Le corpus de test est lui composé de 1557 graphes de mots issu du moteur de reconnaissance de la parole (RAP) de France Telecom. Les phrases de référence comportent 6395 mots dont 439 différents. La figure 4.5 illustre la répartition des phrases de référence en fonction de leur nombre de mots.

4.2.4 Jeu d'étiquettes conceptuelles utilisé

Les étiquettes conceptuelles représentent les unités sémantiques élémentaires extraites à partir du texte pour permettre la construction de structures sémantiques. Dans l'application PlanResto, le nombre de concepts utilisés par France Télécom est de 59 et sont listés dans la tableau 4.2.

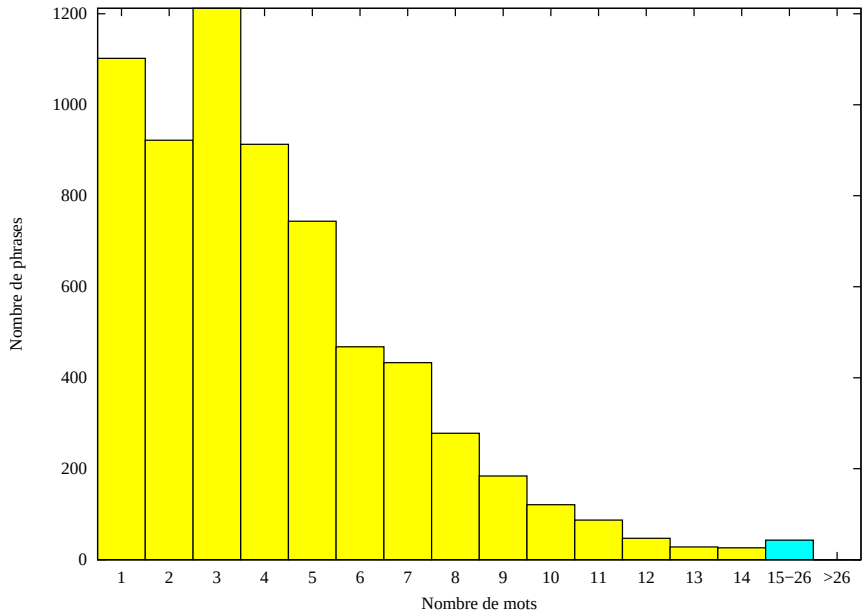

FIG. 4.3 – *Répartition des phrases de référence du corpus d'apprentissage PlanResto en fonction du nombre de mots qui les composent*

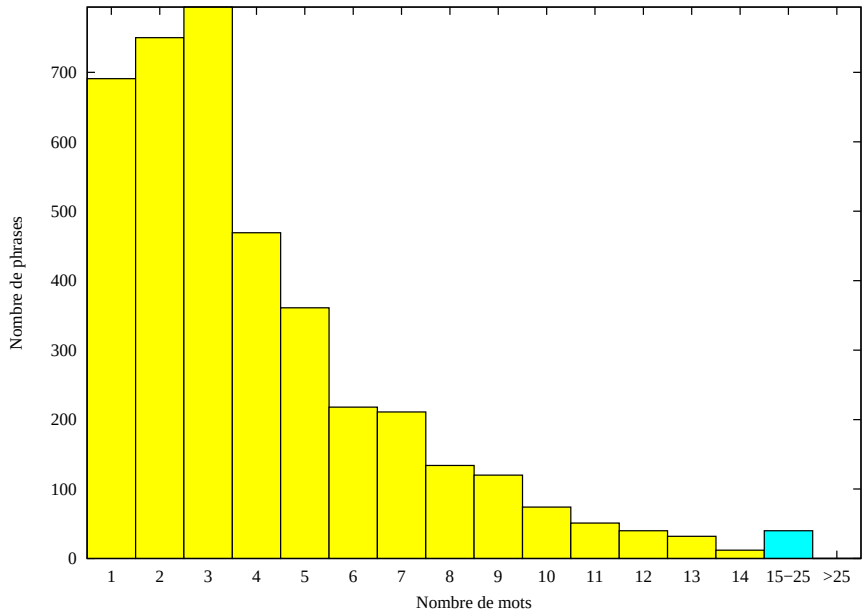

FIG. 4.4 – *Répartition des phrases de référence du corpus de développement Planresto en fonction du nombre de mots qui les composent*

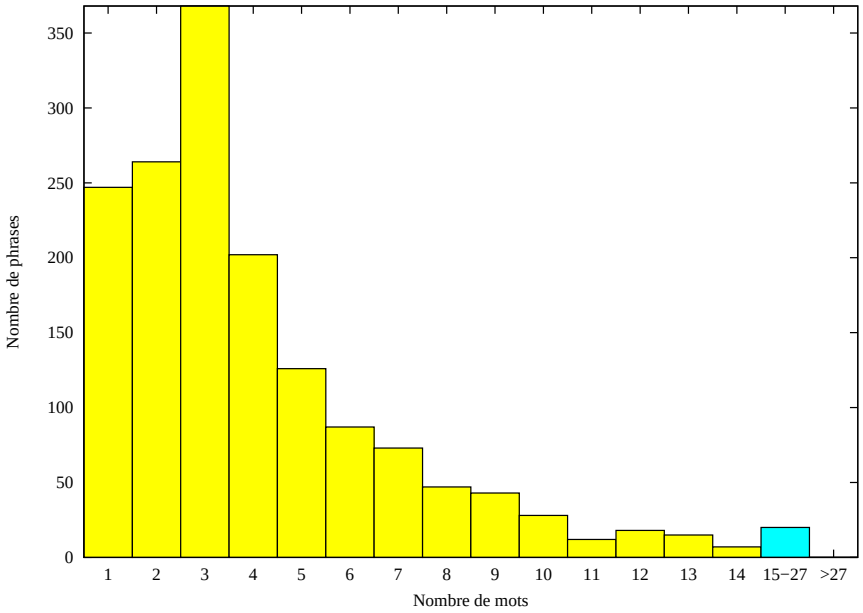

FIG. 4.5 – *Répartition des phrases de référence du corpus de test Planresto en fonction du nombre de mots qui les composent*

TAB. 4.2 – Liste des 59 concepts PlanResto

Concepts avec valeurs	
Lieux	un lieu
Prix	un prix
Specialite	une spécialité culinaire
valeur(ord)	un ordinal
valeur(card)	un cardinal

Classe Spécifiques	
claAdresse	
claAmbiance	
claArrondissement	
claCapaciteAccueil	
claConnexion	
claEspacesVerts	
claHautsLieuxReligieux	
claHoraire	
claInformation	
claLieu	
claMessage	
claMusees	
claNom	
claPlaces	
claPrix	
claPrixExterne	
claQuartiers	
claRestaurant	
claSpecialite	
claStations	
claTel	

verbe être	
vb(neg_rmoi)	
vb(pos_rmoi)	

Aucun concept (hors focus)	
BCK	Aucun concept

marqueurs d'actes communicatifs	
ma(aide)	demande d'aide
ma(end_of_session)	demande à quitter
ma(raz)	demande de remize à zéro
ma(reeng_Diag)	
ma(repeter)	demande de répétition
ma(modeGuide)	demande à être guidé
ma(petiteRelance)	

marqueurs linguistiques	
ml(contest)	contestation
ml(inver_v_suj)	inversion verbe/sujet
ml(neg_pre)	négation pécédent un verbe
ml(non)	réponse négative
ml(object)	pronom à la troisième personne
ml(ord(prec))	ordinal indiquant la précédence
ml(ord(svt))	ordinal indiquant le suivant
ml(ord(dernier))	ordinal indiquant le dernier
ml(tous)	toutes les réponses
ml(oui)	réponse positive

opérateurs modaux	
op(neg_krif_auditeur)	
op(neg_krif_locuteur)	
op(pos_kif_auditeur)	
op(pos_kif_locuteur)	
op(pos_krif_auditeur)	
op(pos_krif_locuteur)	

Divers	
consulter	
dans	
peu_importe	
retour	
mini	
maxi	
utilisateur_regulier	

Les figures 4.6 et 4.7 montrent respectivement la répartition des phrases en fonction du nombre de concepts présents pour le corpus de développement et de test.

4.3 Évaluation de la qualité de la reconnaissance

4.3.1 Le Taux d'Erreurs Mot

Le taux d'erreurs mot (ou Word Error Rate, WER) est une des mesures les plus utilisées pour estimer les performances d'un reconnaisseur sur la transcription produite. Un alignement est effectué entre une hypothèse de reconnaissance et la phrase de référence[2] et les erreurs sont comptabilisées et utilisées pour calculer le taux d'erreurs suivant la formule 4.1. Généralement un poids identique est accordé à chaque type d'erreur, toutefois il est possible de leur attribuer un poids différent.

$$\text{Taux d'erreurs} = \frac{(\#S + \#I + \#O) * 100}{\text{nombre de mots à reconnaître}} \tag{4.1}$$

Un système peut faire trois types d'erreur. Des substitutions, notées « S », correspondent aux mots substitués à d'autres. Des omissions notées « O », c'est-à-dire des mots qui n'ont pas été trouvés par le système. Enfin, des insertions, notées « I », lorsque des mots sont insérés par erreur. Le tableau 4.3 illustre un alignement entre une référence et une hypothèse qui aura comme taux d'erreurs :

$$WER = \frac{(1S + 1O + 1I) * 100}{5} = 60\%$$

Il est à noter qu'en raison des insertions, la mesure utilisée peut dépasser les 100%.

TAB. 4.3 – Alignement entre une phrase de référence et une hypothèse de reconnaissance

Référence :	je	veux	le	restaurant		indien
Hypothèse :	je	veux		restaurant	un	lien
Types d'erreur :			O		I	S

4.3.2 Le Taux d'Erreurs Concept, CER

Dans les systèmes de dialogue, l'objectif n'est pas de transcrire sans erreurs, mais de pouvoir comprendre les sens de ce qui est prononcé. Comprendre le sens, nécessite de pouvoir détecter tous les concepts élémentaires présents dans la phrase. Ceci reste possible même avec une transcription erronée, si les erreurs de reconnaissance n'affectent pas les mots porteurs de sens. Dans ce genre d'application le taux d'erreurs mot n'est alors pas le plus pertinent. Nous utilisons alors le taux d'erreurs sur les concepts (ou Concept Error Rate, *CER*). Il est associé aux étiquettes conceptuelles. Par exemple pour le contexte : *un restaurant à Bastille*, est associé la séquence de concepts `<claRestaurant> <Lieux>`. La séquence de concepts reconnue est alors alignée avec la référence et le taux d'erreurs concept est calculé de manière identique au WER, en tenant compte des séquences de concepts plutôt que des mots.

[2] la transcription manuelle de ce qui est énoncé dans la portion de signal à reconnaître

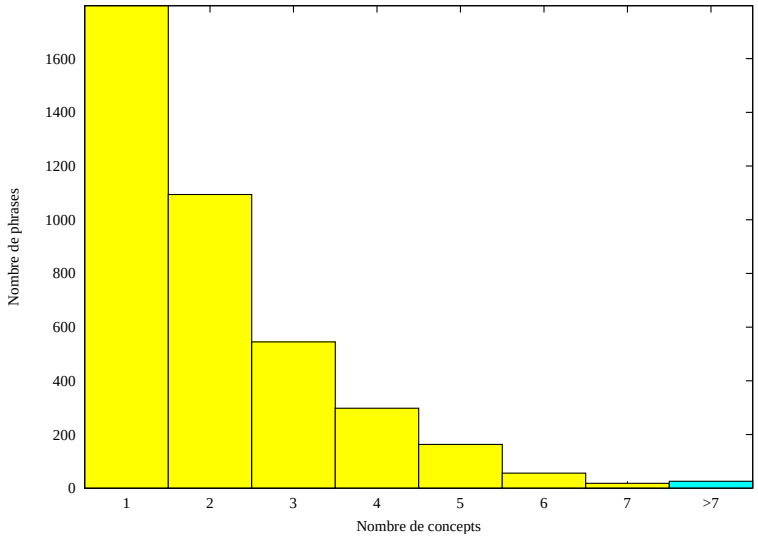

FIG. 4.6 – *Répartition des phrases de référence du corpus de développement Planresto en fonction du nombre de concepts qui les composent*

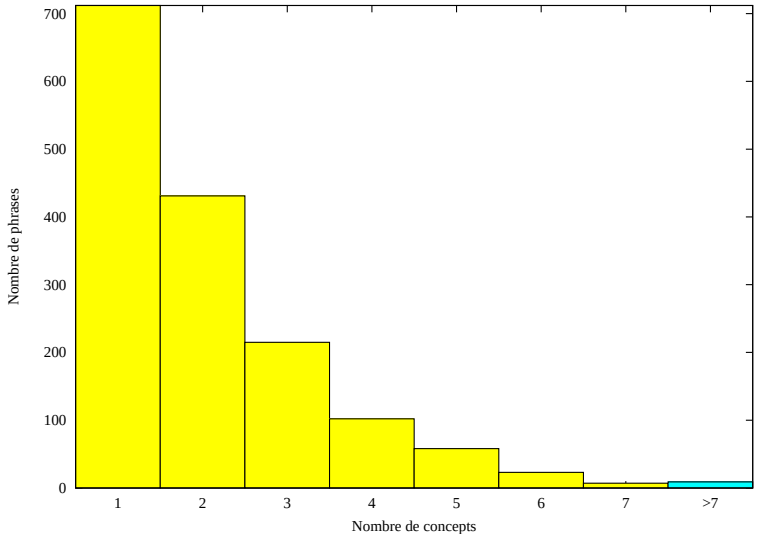

FIG. 4.7 – *Répartition des phrases de référence du corpus de test Planresto en fonction du nombre de concepts qui les composent*

4.3.3 Taux d'Erreurs en Compréhension

Le taux d'erreurs en compréhension (ou Understanding Error Rate, *UER*) est associé aux valeurs normalisées des concepts détectés. À la différence du CER, les valeurs des concepts sont prises en compte (pour les concepts en possédant). Ces valeurs sont obtenues par un ensemble de règles qui transforme la séquence de mots détectée comme concept en valeur significative. Par exemple pour le contexte : *un restaurant à Bastille pour cent francs* est associé `<claRestaurant> <Lieux:BASTILLE> <Prix:100 F>`. Le *UER* est défini comme suit :

$$UER = \frac{S_{c/v} + D_c + I_c}{T} \times 100 \tag{4.2}$$

où $S_{c/v}$ indique la substitution d'un attribut de Γ ou de sa valeur, D_c indique la suppression d'un attribut I_c indique une insertion. T est le nombre total de concepts dans la référence. Un UER à 0, donne donc une reconnaissance idéale même si son taux d'erreurs mot est supérieur à 0.

CHAPITRE

5

Stratégie de décodage conceptuel

Sommaire

5.1 Résumé

Dans ce chapitre, nous présentons une architecture de décodage intégrant des informations conceptuelles en plus des informations acoustico-linguistiques utilisées habituellement. Ces informations conceptuelles sont représentées par des paires attribut/valeur et sont les éléments à partir desquels le module de compréhension va pouvoir construire une représentation sémantique de ce que l'utilisateur a dit. Un concept

est représenté par un mot ou une séquence de mots ayant un sens pour l'application de dialogue. Nous présentons un modèle de language permettant de faire la correspondance entre ces mots et le ou les concepts qu'ils représentent. Ceci peut être vu comme une opération de traduction des mots vers les concepts.

Nous utilisons une grammaire régulière locale spécialisée pour détecter chaque concept de l'application et nous la codons sous forme de transducteur à états fini qui émet une étiquette correspondante au concept détecté. Ces transducteurs sont regroupés dans un super-transducteur qui correspond à notre modèle conceptuel capable d'analyser toute phrase et de produire la ou les interprétations conceptuelles (ensemble de concepts) associées.

L'espace de recherche de départ est la sortie d'un moteur de reconnaissance sous la forme d'un graphe de mots. Chaque chemin du graphe est une hypothèse faite par le système de RAP pour la transcription du signal de parole. Chaque transition dans ce graphe est un mot, et le score associé à chaque mot est le score combiné des scores donnés par le modèle acoustique et le modèle de langage.

En représentant le graphe de mots comme un automate à états fini et en le composant avec notre modèle, nous obtenons un espace de recherche enrichi sous la forme d'un transducteur avec comme entrée les mots et comme sortie des étiquettes conceptuelles. En ne considérant que les sorties de ce transducteur nous obtenons le graphe de concepts associé au graphe de mots. Nous présentons alors une recherche dans cet espace permettant de générer une liste structurée des N-meilleurs candidats en cherchant les meilleures interprétations disponibles. Cette liste fournit toutes les interprétations possibles qui existent dans le graphe de mots avec leurs meilleures phrases supports en terme de mots sans redondance du point de vue des informations conceptuelles.

5.2 Introduction

Dans les applications de dialogue considérées dans ces travaux, l'opération de compréhension consiste à construire une représentation sémantique de l'énoncé utilisateur. L'instanciation d'une structure sémantique s'appuie souvent sur un ensemble de concepts élémentaires de la phrase reconnue plutôt que sur sa forme syntaxique [Ward, 1991, Hacioglu et Ward, 2001, Pieraccini et Levin, 1995, Aust *et al.*, 1995, Jamoussi *et al.*, 2004]. À partir de la séquence d'observations acoustiques $A = a_1 a_2 \dots a_m$ extraite du signal, un système de compréhension de la parole cherche l'ensemble des concepts élémentaires $\hat{C} = c_1 c_2 \dots c_k$ qui maximise la probabilité *a posteriori* $P(C|A)$. Pour solutionner ce problème, il est commode de faire intervenir la séquence de mots W transportée par A :

$$\begin{aligned} \hat{C} &= \underset{C}{\text{ArgMax}} \sum_{W} P(C,W|A) \\ &\approx \underset{C,W}{\text{ArgMax}} P(C,W|A) \end{aligned} \tag{5.1}$$

d'après le théorème de Bayes la règle de décision (5.1) peut être re-formulée en :

$$\begin{aligned} \hat{C} &\approx \underset{C,W}{\text{ArgMax}} P(A|C,W)P(C,W) \\ \text{où} & \\ P(A|C,W) &\approx P(A|W) \\ P(C,W) &= P(C|W)P(W) \\ \text{donc} & \\ \hat{C} &\approx \underset{C,W}{\text{ArgMax}} P(A|W)P(W)P(C|W) \end{aligned} \tag{5.2}$$

Dans la pratique il est possible de scinder le problème de la détection de C en 2 étapes séquentielles qui consistent à effectuer une transcription et à utiliser le résultat de cette transcription pour effectuer le décodage conceptuel. La transcription est obtenue en utilisant la règle de décision suivante :

$$\hat{W} = \underset{W}{\text{ArgMax}} P(A|W)P(W)$$

La transcription est générée avec des modèles acoustiques permettant de calculer la probabilité $P(A|W)$ et des modèles linguistiques à portée réduites (N-grammes) calculant la probabilité $P(W)$. Nous partons du principe que la séquentialité du traitement amenant à l'interprétation est une faiblesse : les performances du module RAP pour assurer la transcription peuvent être améliorées en prenant en compte le fait que l'interprétation peut contraindre ce module afin d'obtenir une transcription plus pertinente vis à vis de l'application. La détection des concepts peut être suffisante dans le cadre de certaines applications comme le routage d'appels. Pour d'autres, il faut construire une représentation sémantique à partir de C. Mais dans les deux cas l'extraction de cet ensemble de concepts est une étape cruciale du processus de compréhension. Les travaux présentés ici proposent une architecture de décodage conceptuel alternative à un traitement purement séquentiel. Cette architecture permet d'exploiter les connaissances conceptuelles pour en faire bénéficier l'étape de transcription. L'architecture de décodage proposée est une architecture deux-passes exploitant un graphe de mots plutôt que la meilleure hypothèse comme espace de recherche de la meilleure interprétation.

5.3 Architecture proposée

La représentation sémantique et l'extraction de concepts sont deux opérations différentes. Construire une représentation sémantique consiste à utiliser des règles qui établissent des relations entre les concepts détectés. La détection de concepts correspond à la mise en œuvre de règles de séquences de mots associées à chaque entité conceptuelle. Ces règles utilisées pour détecter différents concepts dans la même phrase peuvent utiliser les dépendances contextuelles et donc partager des mots. Ces règles sont de taille finie car les phrases, spécialement en dialogue oral, contiennent un nombre fini et souvent petit de mots. Le formalisme des automates à états fini est donc approprié pour modéliser ces règles.
Les précédentes considérations suggèrent de concevoir l'étape de transcription comme un processus de décodage dans lequel les modèles de langage stochastiques (LMs) contiennent des séquences de mots acceptées par des Machines à états fini (FSM) dont les étiquettes de sortie représentent les constituants sémantiques, ou concepts : il y a un FSM pour chaque concept élémentaire. La définition d'un tel concept doit satisfaire

2 contraintes majeures :

- il doit être possible d'inférer un LM pour chaque concept ;
- il doit être possible de composer ou inférer toute structure sémantique à partir de ces concepts

Le LM pour chaque concept peut être vu comme un langage accepté par une approximation régulière d'une grammaire représentant un langage naturel. Une telle approximation est implémentée par un transducteur à états fini dont les sorties sont des instances des concepts. Nous avons voulu que ces différents automates, représentant différents concepts, soient capables de partager des séquences de mots, sans requérir, comme dans de nombreuses approches d'analyse sémantique restreinte que les constituants sémantiques ne puissent se chevaucher [Hacioglu, 2004, Pradhan *et al.*, 2004].

Pour satisfaire ces considérations nous proposons une architecture de décodage basée sur des connaissances à deux niveaux :

- le premier niveau est relié à la détection des concepts dans une phrase. Ces entités conceptuelles appartiennent à des ensembles de catégories ontologiques (tel que chose, événement, état, action, lieu, quantité,...) ;
- une interprétation correspond à l'ensemble des concepts contenus dans une phrase. Si les concepts sont tous modélisés par des séquences de mots contiguës et que ces séquences ne se chevauchent pas (elles n'exploitent pas de redondance au niveau du contexte), l'interprétation (ou les interprétations en cas d'ambiguïtés) d'une phrase correspond à la séquence de concepts. S'il est autorisé de détecter des concepts avec des règles qui peuvent s'appuyer sur l'intégralité d'une phrase ou si l'on décide que des règles associées avec différents concepts puissent se chevaucher pour tirer parti d'un contexte, l'interprétation n'est pas représentée par une séquence de concepts ordonnés en fonction des mots qui les génèrent, mais par une séquence de concepts composée à partir des concepts détectés. Cette composition est correcte, seulement s'il existe une séquence de mots qui supportent de façon cohérente tous les composants de la composition. Le second niveau consiste alors en des relations sémantiques entre ces entités conceptuelles afin de construire l'interprétation.

Notre approche se veut indépendante des unités conceptuelles. Leur nombre, la granularité de leur représentation est un choix qui appartient aux concepteurs d'un système. L'architecture de décodage proposée n'impose pas de restrictions. La section 5.4 présentera donc succinctement la notion de concept sans insister sur les différents choix possibles pour les représenter. Afin d'illustrer au mieux nos propos nous utiliserons des exemples en relation avec le système de dialogue oral PlanResto de France Telecom R&D (description dans le chapitre 4.2) qui est une application de serveur vocal proposant la recherche de restaurant dans Paris. La section 5.5 décrit la construction d'un modèle de langage conceptuel permettant d'enrichir un graphe de mots issu d'un module RAP avec des informations conceptuelles. La section 5.6 illustre le processus de décodage dans un nouvel espace de recherche conceptuel au travers d'un exemple simple. La sortie de ce processus permet de générer une liste structurée sémantiquement des N-meilleures hypothèses dont l'intérêt est discuté dans la section 5.7.

5.4 Les entités conceptuelles

Afin de pourvoir interagir avec l'utilisateur, le gestionnaire de dialogue doit construire une représentation sémantique correspondant aux attentes de l'utilisateur. La construction de cette représentation sémantique, se fait au fil du dialogue grâce aux interprétations faites par le module de compréhension sur les différentes interventions de l'utili-

sateur. Une interprétation est l'ensemble des concepts exprimés par l'utilisateur. Dans un contexte de dialogue oral, les concepts vont correspondre aux unités sémantiques élémentaires correspondant à l'application visée. Quels que soient le module de compréhension et le gestionnaire de dialogue utilisés, ces entités doivent être détectées et reconnues afin d'obtenir les paramètres essentiels à la résolution d'une tâche par le système. L'extraction de ces entités conceptuelles (concepts) représente le premier traitement dans le processus de compréhension. Leur définition est reliée à la stratégie de dialogue. Certains concepts sont reliés à la gestion du dialogue (confirmation, contestation, ...) et d'autres au domaine d'application (lieu, date, ...). Un concept peut-être vu comme une paire attribut/valeur.

On peut distinguer deux types d'entités conceptuelles (voir tableau 5.1) :

1. des concepts avec valeur qui correspond à un champ de la base de données utilisée par l'application. Ils sont souvent représentés localement dans une phrase par des séquences de mots contigus, telles les dates, les prix, les numéros de téléphone, *etc*. Nombreuses sont ces entités à pouvoir être partagées par plusieurs applications. Elles sont donc généralement représentées par des grammaires régulières écrites manuellement ;
2. des concepts sans valeurs particulières utiles pour pouvoir interpréter le sens du message utilisateur. Ces entités conceptuelles sont parfois détectables à travers des séquences de mots plus complexes où des séquences de mots parfois non contigus. Par exemple le concept de contestation pourrait être détecté par une expression régulière du type (non .* je ne VERBE pas).

TAB. 5.1 – Exemple de correspondance syntagme/concept

Type de concept	syntagme	exemple (attribut/valeur)
avec valeur	près de Bastille	(Lieu/Bastille)
	un restaurant indien	(Spécialité/indien)
	deux cents cinquante francs	(Prix/[250 F])
	le onze février	(DATE/[11/02])
sans valeur	non pas le	(Contestation/)
	je cherche un restaurant	(Requête/)

Toutes les entités conceptuelles seront codées au final sous la forme d'un transducteur.

5.4.1 Représentation des concepts

Les concepts avec valeurs sont détectables le plus souvent par des séquences de mots exprimées par l'utilisateur, et peuvent contenir 3 sortes de mots, des mots clefs (notés (1) dans les exemples qui suivront) qui sont indispensables car ils sont, soit caractéristiques du concept, soit ils en composent la valeur, des modifieurs (notés (2)), des mots qui informent sur la manière d'interpréter le ou les mots clefs, et des mots spécifiques (notés (3)) aux concepts dont la présence n'est pas indispensable, exemple :

- POUR UN LIEU
  - « près[2] de la tour[2] Montparnasse[1] »
  - « proche[2] du métro[2] Gaité[1] »
  - « dans[2] le troisième[1] arrondissement[3] »
- POUR UNE SPÉCIALITÉ CULINAIRE
  - « un restaurant[3] indien[1] »

  - « spécialité[3] du sud-ouest[1] »
  - « cuisine[3] végétarienne[1] »
- POUR UN PRIX
  - « inférieur[2] à cent[1] francs[2] »
  - « entre dix[1] et vingt[1] euros[2] »
  - « le moins[2] cher[1] possible »

Il est à noter que les modifieurs (mots de type 2) représentent une fonction sur le concept et peuvent être communs à plusieurs concepts. Ils peuvent alors être considérés comme des concepts à part entière (sans valeur) et être modélisés séparément. Le choix des unités conceptuelles ainsi que leur granularité appartient au concepteur du système. Dans les exemples qui suivent, il a été choisi de les modéliser au sein des entités dont ils sont la fonction.
L'apprentissage des règles a été effectué à partir du corpus d'apprentissage étiqueté manuellement. La séquence de mots représentant un concept a été choisie la plus longue possible. C'est à dire que sur les 3 types de mots présentés plus haut, la séquence composée des mots clefs est suffisante pour détecter le concept, mais nous avons voulu accepter les séquences les plus longues possibles pour deux raisons :

1. la première est que nous modélisons des structures linguistiques parfois plus longues que le modèle *N*-grammes utilisé pendant le décodage. Ce qui nous permet de localiser des phrases plus cohérentes sur leur totalité.
2. la seconde, c'est que la longueur du syntagme correspondant à un concept nous donne une information sur sa confiance de détection (*i.e.* présence d'un contexte désambiguïsateur). Si une grammaire reconnaît 5 ou 6 mots lors de la détection d'un concept, il est peu probable que ce soit dû à 5 ou 6 erreurs de reconnaissance. Toutefois, les séquences plus courtes composées des mots-clefs sont modélisées également afin de minimiser le risque de non-détection de concepts en cas d'absence de contexte ou de présence d'erreurs de reconnaissance.

Les règles apprises sur ce corpus ont ensuite été enrichies selon deux critères. Le travail de généralisation des règles de grammaires est intéressant, mais il n'est pas la priorité de ces travaux. Ce travail de généralisation a été fait de manière à ce que la couverture de nos modèles soit suffisante pour pouvoir les évaluer correctement.

Critère 1

Le premier critère d'enrichissement consiste à créer des classes d'équivalence entre mots qui peuvent être interchangés dans la grammaire sans altérer sa précision. C'est notamment le cas des mots clefs :

le troisième arrondissement	$\longrightarrow$	le ORDINAL arrondissement
type de cuisine indienne	$\longrightarrow$	type de cuisine SPEC
restaurant chinois	$\longrightarrow$	restaurant SPEC
pour moins de cent francs	$\longrightarrow$	pour moins de cent DEVISE
maximum trois cent euros	$\longrightarrow$	maximum UNITE cent DEVISE
près du métro Gaité	$\longrightarrow$	près du métro XLOC

Les règles obtenues après cet enrichissement sont suffisantes pour détecter quasiment tous les concepts et permettent de surcroît de diminuer la taille et la complexité de leur représentation sous forme de transducteur. L'utilisation de classes d'équivalence telle que nous l'avons effectué fait que l'on perd certaines contraintes, comme le genre et le nombre sur les mots clefs et les mots attenants. Mais nous avons décidé de ne pas

diviser ces classes en sous-classes tenant compte du genre et du nombre, car la cohérence en genre et en nombre est assurée en partie par le modèle de langage *N*-grammes et surtout une erreur en genre ou en nombre dans ce contexte n'a pas d'influence sur l'interprétation de la phrase.

Critère 2

Le deuxième critère exploite des connaissances syntaxiques du français pour déterminer des séquences de mots interchangeables au sein d'un même concept. Par exemple pour le concept de lieu les séquences suivantes sont équivalentes :

$$\begin{cases} \text{à côté de} \\ \text{dans le secteur de} \\ \text{proche de} \\ \ldots \end{cases}$$

Les règles contenant une de ces séquences interchangeables sont dupliquées avec toutes les autres séquences équivalentes. Cette généralisation permet donc de pouvoir accepter le plus souvent possible les séquences de mots les plus longues.

Les figures 5.1 et 5.2 illustre respectivement un exemple d'automate codant le concept de lieu et de prix.

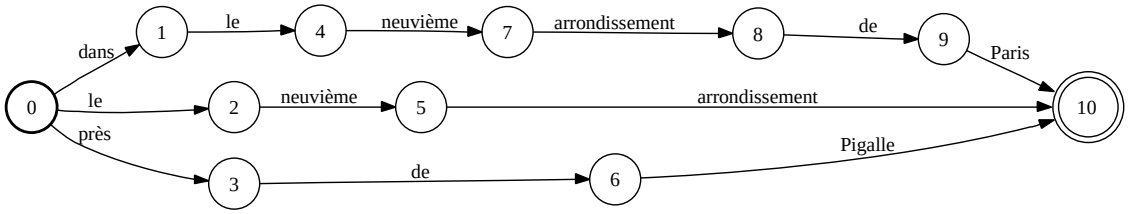

FIG. 5.1 – *Exemple d'automate représentant le concept de* LIEU

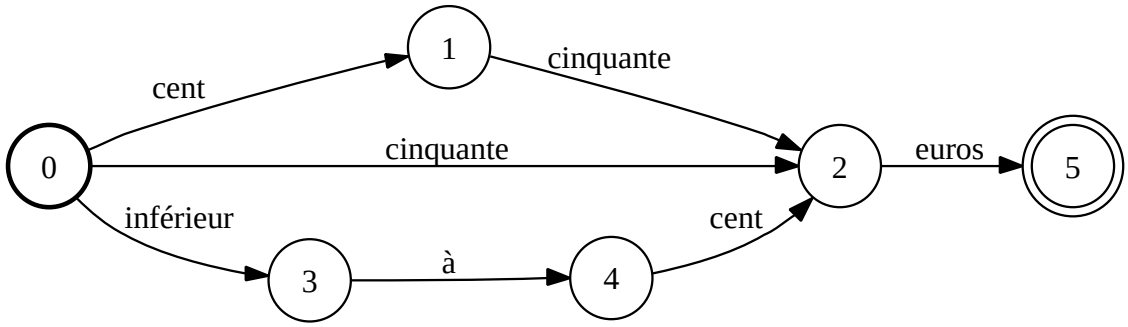

FIG. 5.2 – *Exemple d'automate représentant le concept de* PRIX

Ces transducteurs prennent les mots en entrée et donnent en sortie les étiquettes associées aux concepts transportés par la phrase reconnue.

5.5 Modèle conceptuel : un transducteur Mots/Concepts

Nous voulons construire un modèle de langage de plus haut niveau modélisant les informations conceptuelles relatives au module de compréhension. Notre modèle conceptuel a pour objectif de représenter les différentes associations mots/concepts qui existent dans une application particulière. Ceci peut être vu comme une opération d'association ou de traduction, c'est pourquoi nous allons représenter ce modèle sous la forme d'un transducteur à états fini, ou le langage d'entrée sera composé des mots du vocabulaire (V) de l'application et le langage de sortie sera composé des concepts, identifiés par des étiquettes ou balises (B). Ce transducteur devra être capable d'analyser une phrase quelconque et émettre la ou les séquences de concepts qui peuvent y être détectés.

L'objectif du modèle conceptuel est d'enrichir le graphe de mots généré par le système de RAP avec des informations conceptuelles. Le graphe de mots G sera codé sous la forme d'un automate à états fini. Notre modèle conceptuel MC sera représenté par un transducteur à états fini. L'objectif sera d'enrichir le graphe de mots par une opération de *composition* entre les deux.

$$GEC = G \circ MC$$

GEC est alors le graphe de mots enrichi conceptuellement par l'opération de *composition*.

5.5.1 Outil utilisé : AT&T FSM Library

Les automates à états fini (ou Finite State Machine - FSM) sont un formalisme puissant, à la fois pour représenter des structures linguistiques telles que des grammaires, mais aussi du point de vue de l'efficacité des algorithmes permettant d'effectuer des opérations fondamentales entre eux. Le toolkit développé à AT&T par Mehryar Mohri, Fernando Pereira et Michael Riley ([Mohri *et al.*, 1997]), permet de manipuler de telles structures, soit sous forme de FSM accepteur, soit sous forme de FSM transducteur, et implémente la plupart des opérations fondamentales (voir chapitre 3.4.2) nécessaires à leur application au traitement automatique de la langue (minimisation, déterminisation, composition, plus court chemin, *etc.*).

Voir http://www.research.att.com/sw/tools/fsm/ pour plus de détails.

5.5.2 Construction de MC

Chaque concept C_k est associé à un accepteur A_k (A_k pour le concept C_k) (exemple figure 5.1 et 5.2 pour les concepts « Lieux » et « Prix »). Dans le but de traiter les séquences de mots qui ne correspondent à aucun concept, un modèle *Filler* (« mange-mots ») appelé A_F est utilisé (voir figure 5.3). Chaque automate A_k est converti en transducteur T_k dont les symboles d'entrée sont les mots et les symboles de sortie sont des balises de début et fin de concept. Tous les accepteurs A_k deviennent alors des transducteurs T_k où la première transition émet le symbole <C_k> et la dernière le symbole </C_k>. De même, le modèle Filler devient le transducteur T_{bk} qui émet les symboles <BCK> et </BCK>. À l'exception de ces balises de début et fin de concept, aucun autre symbole n'est émis : tous les mots dans le Filler ou les transducteurs conceptuels émettent un symbole vide (transition ε notée « NULL » dans les différentes figures). Le modèle conceptuel, dans un premier temps, peut être vu comme un transducteur $T_{concept}$ étant l'union

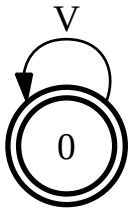

FIG. 5.3 – *Automate du modèle FILLER*

des transducteurs conceptuels et du transducteur Filler, où l'état final est re-bouclé sur l'état de départ afin de traiter une phrase complète, comme illustré dans la figure 5.4.

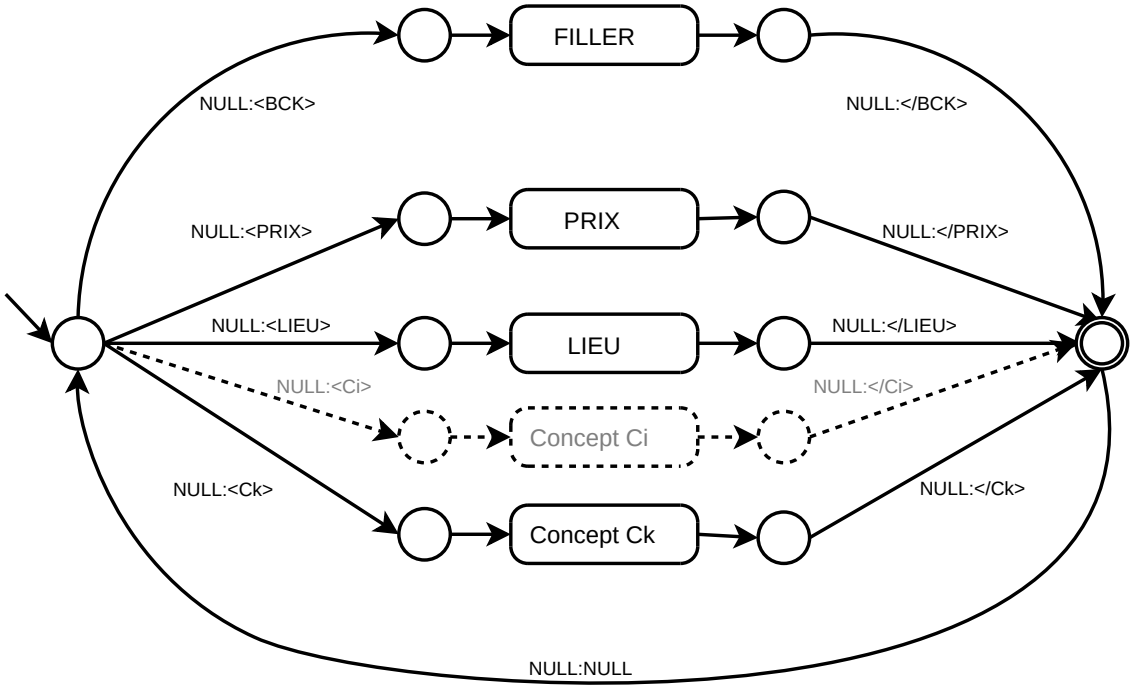

FIG. 5.4 – *Topologie de* $T_{concept}$

Dans cette configuration, le modèle conceptuel effectue des analyses que nous ne considérons pas optimales : pour une même chaîne de mots plusieurs analyses redondantes sont susceptibles d'être effectuées et certaines peuvent être erronées. Le but des différentes optimisations est de faire en sorte qu'en fonction d'une chaîne de mots notre modèle ne produise que des analyses non-redondantes et cohérentes.

5.5.3 Optimisation du modèle Filler N° 1

En premier lieu, l'accepteur A_F à partir duquel est construit le transducteur T_{bk} accepte la chaîne vide. Le transducteur T_{bk} peut alors émettre des balises de début et fin de *BCK* de manière anarchique dans la phrase analysée. Pour éviter cela nous modifions l'accepteur A_F afin de lui imposer l'analyse d'au moins un mot, voir figure 5.5.

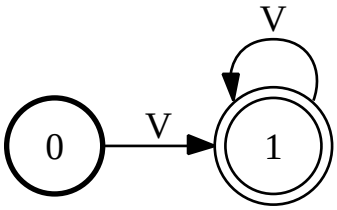

FIG. 5.5 – *Automate du modèle FILLER acceptant au minimum un mot*

5.5.4 Optimisation du modèle Filler N° 2

Le modèle Filler pouvant re-boucler sur lui même dans la topologie illustrée dans la figure 5.4, l'analyse d'une chaîne de mots non-conceptuelle (*e.g.* « euh oui alors euh ») peut amener à différentes segmentations, car plusieurs chemins dans le transducteur existent :
<BCK> euh oui alors euh </BCK>
<BCK> euh </BCK> <BCK> oui alors euh </BCK>
<BCK> euh oui </BCK> <BCK> alors </BCK> <BCK> euh </BCK>
...
Or, la segmentation désirée est celle qui analyse d'un bloc la séquence de mots non conceptuelle :
<BCK> euh oui alors euh </BCK>
Pour obtenir ce résultat la topologie du modèle conceptuel a été revue : le modèle Filler ne re-boucle plus sur lui même. Seule une séquence de mots conceptuelle peut suivre. On force donc l'analyse d'une séquence de mots « background » a être effectuée entièrement dans le Filler (figure 5.6).

5.5.5 Optimisation du modèle Filler N° 3

Un dernier inconvénient dû au Filler, est qu'il reconnaît toutes les séquences de mots (V^*) composées à partir du vocabulaire, y compris les séquences de mots conceptuelles. L'analyse d'une séquence de mots conceptuelle par notre modèle conceptuel va donner lieu à une double segmentation. Supposons que notre modèle conceptuel $T_{concept}$ ne modélise qu'un seul concept de LIEU et que l'automate A_{LIEU} représentant ce concept ne modélise qu'une règle : « près de Pigalle », comme illustré à la figure 5.7.

La chaîne de mots, « euh oui alors euh près de pigalle » va être analysée de 2 façons différentes par $T_{concept}$:
<BCK> euh oui alors euh </BCK> <LIEU> près de Pigalle </LIEU>
<BCK> euh oui alors euh près de Pigalle </BCK>
Pour supprimer cette double analyse tous les chemins dans les accepteurs A_k sont supprimés du modèle Filler A_F de la manière suivante :

$$A_F = V^+ - \bigcup_{k=1}^{K} A_k \tag{5.3}$$

Dans l'exemple représenté en figure 5.8 toutes les séquences de mots incluant une séquence conceptuelle sont retirées du Filler (figure 5.5). Nous obtenons un modèle Filler ne pouvant accepter les séquences de mots conceptuelles (figure 5.9).

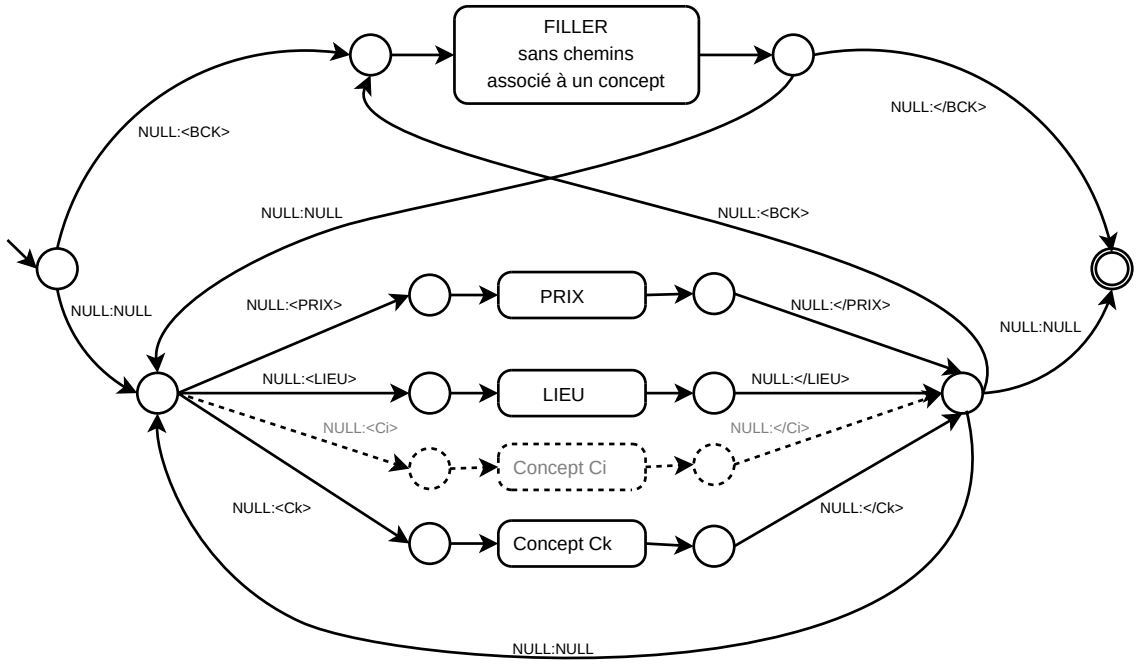

FIG. 5.6 – *Topologie finale de $T_{concept}$*

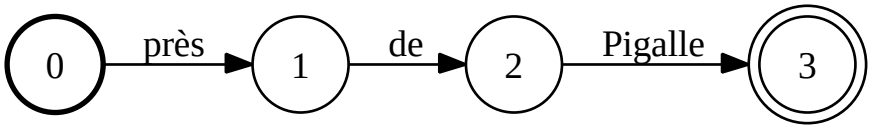

FIG. 5.7 – *Exemple d'automate représentant le concept de lieu*

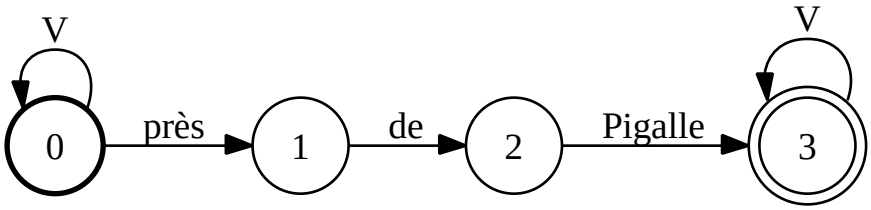

FIG. 5.8 – *Automate représentant les chemins à soustraire au FILLER*

5.5.6 Élimination des redondances et ambiguïtés intra-concept

Les différents automates conceptuels reconnaissent les séquences de mots relatives au concept qu'ils modélisent. Certaines séquences de mots extraites d'un corpus, peuvent partager les mots qui les composent, mais ont des tailles différentes en raison de la variabilité des expressions employées par les utilisateurs. Le tableau 5.2 montre un même concept, en l'occurrence de lieu, exprimé de plusieurs manières plus ou moins concises.

Dans cet exemple, certaines séquences de mots sont des sous-chaînes d'autres séquences. Certaines d'entre elles se chevauchent. Toutes ces séquences sont le produit de chemins parcourus dans l'accepteur modélisant le concept. Dans le cas

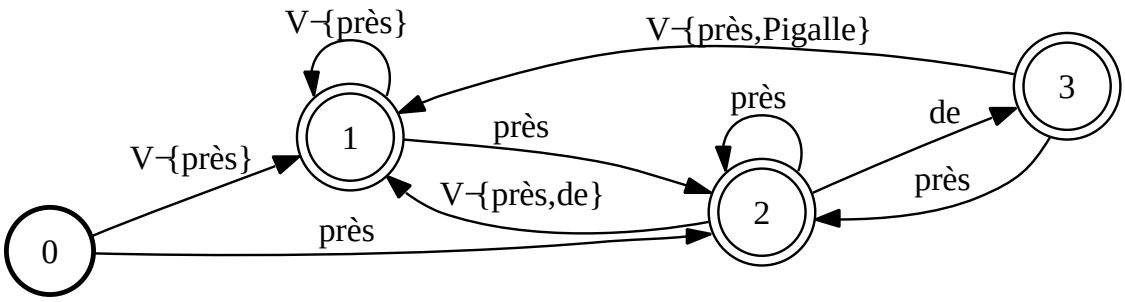

FIG. 5.9 – *Modèle FILLER après soustraction de l'automate des lieux*

TAB. 5.2 – Exemple de séquences de mots exprimant le même concept

Exemple 1 :	dans	le	neuvième	arrondissement	de	Paris
Exemple 2 :	dans	le	neuvième	arrondissement		
Exemple 3 :		le	neuvième	arrondissement		
Exemple 4 :	dans	le	neuvième			

où la chaîne à analyser contiendrait le contexte le plus long, plusieurs chemins dans $T_{concept}$ existent. En effet, même si ces séquences présentées dans le tableau 5.2, ont été supprimées des chemins acceptés par le Filler, les sous-séquences comme « dans » ou « arrondissement » sont acceptées et à juste titre, vu qu'elles ne représentent pas à elles seules un concept. Plusieurs segmentations sont alors données par $T_{concept}$ pour une même séquence de concepts :

<LIEU> dans le neuvième arrondissement de Paris </LIEU>
<LIEU> dans le neuvième arrondissement </LIEU> <BCK> de Paris </BCK>
<BCK> dans </BCK> <LIEU> le neuvième arrondissement </LIEU> <BCK> de Paris </BCK>
<LIEU> dans le neuvième </LIEU> <BCK> arrondissement de Paris </BCK>

Or, nous voulons laisser les segmentations multiples seulement dans le cas ou ces segmentations amènent à des interprétations différentes de la phrase à analyser. Ceci peut arriver dans le cas d'une ambiguïté sémantique qui ne peut être levée à ce niveau du processus. Par exemple « le neuvième » sans contexte peut faire référence à un arrondissement ou bien à une position dans une liste de restaurants. Sans un contexte dans la phrase permettant de désambiguïser l'analyse, les deux analyses possibles doivent être faites. C'est au gestionnaire de dialogue en fonction de l'historique du dialogue de choisir la bonne. Dans le cas présenté ici, nous désirons supprimer des analyses redondantes qui pour une même séquence de mots amènent à la même séquence de concepts (voir remarque plus bas), et nous désirons conserver celle qui donne la segmentation au profit de la plus longue séquence de mots représentant le concept (la n° 1 dans cet exemple). Au delà de ces considérations, ne conserver que le chemin le plus long dans l'automate permet surtout de supprimer des ambiguïtés

d'analyse en ce qui concerne les valeurs d'un concept. Considérons les séquences observées dans le corpus pour le concept PRIX dans le tableau 5.3.

TAB. 5.3 – Séquences de mots associées à un prix

Séquence 1 :	cent	cinquante	francs
Séquence 2 :		cinquante	francs

Le modèle $T_{concept}$ en l'état actuel, pour la chaîne « cent cinquante francs », va nous donner deux analyses redondantes en terme de concept si l'on ne considère que les attributs, mais une fausse en terme de valeur :

<BCK> cent </BCK> <PRIX> cinquante francs </PRIX>
<PRIX> cent cinquante francs </PRIX>

Or dans ce contexte seule la deuxième segmentation est correcte (voir remarque).

Remarque : il est possible que le contexte désambiguïsateur soit incorrect (*i.e.* dû à une erreur de reconnaissance) on pourrait donc penser dans ce cas qu'il faille conserver la segmentation donnant lieu à un autre concept ou une autre valeur pour un concept. Mais comme nous travaillons sur l'analyse du graphe de mots, nous faisons la supposition que si les mots formant le contexte sont proposés à tort par le système RAP, au moins un chemin (et dans la pratique plusieurs) du graphe ne les contient pas et fournira l'interprétation correcte.

Nous allons alors modifier notre modèle pour qu'il privilégie les séquences de mots conceptuelles les plus longues. Pour cela, nous allons supprimer les chemins amenant à des analyses non-convenables dans $T_{concept}$:

Pour chaque concept C_k, nous dressons la liste des séquences de mots qui sont incluses dans d'autres et nous générons un transducteur $T_{ForbiddenPath}(k)$ qui dresse la liste des segmentations indésirables. En considérant les accepteurs conceptuels illustrés en figure 5.1 et 5.2, les transducteurs 5.10 et 5.11 illustrent les transducteurs $T_{ForbiddenPath}(C_k)$ associés. Ces transducteurs contiennent tous les chemins possibles

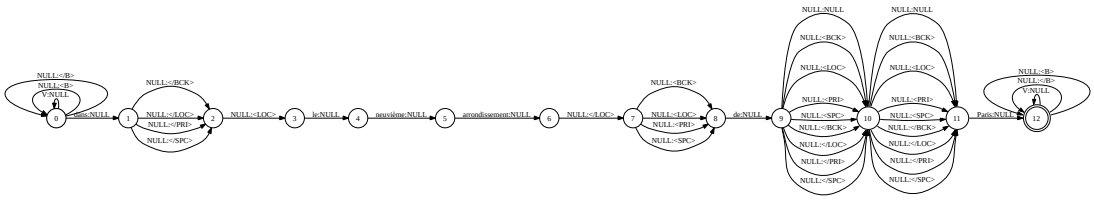

FIG. 5.10 – *Transducteur listant les chemins interdits pour le concept LIEU*

conduisant à ces segmentations incorrectes. Dans les transducteurs, les transitions « entrée :sortie » sont de la forme « ε :étiquette_conceptuelle » ou « mot :ε » (les transitions ε sont notées *NULL* dans les différentes figures), pour simplifier l'écriture nous ne considérons que la sortie dans le premier cas et l'entrée dans le second. Les chemins non-désirés sont alors représentés par l'expression régulière suivante :

$$(V|B)^*\ cent\ \ </B></PRIX>\ cinquante\ francs\ </PRIX>\ (V|B)^*$$

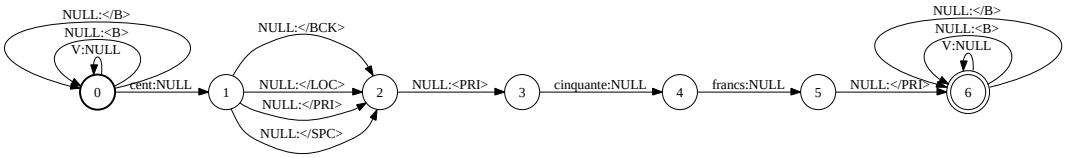

FIG. 5.11 – *Transducteur listant les chemins interdits pour le concept PRIX*

Le modèle $T_{concept}$ ainsi que les transducteurs $T_{ForbiddenPath}(C_k)$ sont convertis en accepteur par une concaténation des symboles entrée/sortie. Ces chemins sont alors enlevés du modèle général $T_{concept}$ au moyen d'une opération de soustraction. Le modèle est ensuite reconverti en transducteur.

Chevauchement de règles

Le même procédé est appliqué aux séquences de mots qui se chevauchent, seule la segmentation autour de la séquence la plus longue est conservée. En cas d'égalité, la première séquence apparaissant dans la phrase à analyser est gardée. Si le chevauchement est important, on peut supposer que la séquence de mots composée à partir de deux séquences est une séquence valide pour exprimer le concept. Si cette séquence ne fait pas partie des règles, alors des problèmes peuvent se poser lors des suppressions de chemins : une séquence de mots n'est plus analysée par notre modèle car plus aucun chemin n'existe dans $T_{concept}$. Cela peut être le cas dans l'exemple suivant : soient 3 règles associées à la détection d'un concept (tableau 5.4), en l'occurrence SPÉCIALITÉ, et la chaîne de mots reconnue « un restaurant diététique marocain ». Dans le modèle

TAB. 5.4 – Situation amenant à la suppression de toute analyse

Séquence 1 :	un	restaurant	diététique	
Séquence 2 :		restaurant	diététique	marocain
Séquence 3 :				marocain
Chaîne reconnue :	un	restaurant	diététique	marocain

$T_{concept}$ ont été supprimés plusieurs chemins : premièrement, aucune des séquences ne peut être analysée par le modèle Filler ; deuxièmement, dans le contexte de la séquence 2, la segmentation autour de « marocain » avec le concept de SPÉCIALITÉ n'est plus possible ; enfin dans un contexte où apparaissent les séquences 1 et 2, la segmentation, avec le concept de SPÉCIALITÉ, autour de la séquence 2 est supprimée (ceci étant dû au chevauchement de ces 2 séquences, la première est conservée). Dans un contexte où apparaîtrait la séquence « un restaurant diététique marocain[1] » : toutes les segmentations deviennent impossibles car cette séquence ne fait plus partie du langage reconnu par $T_{concept}$:

<BCK> un restaurant diététique </BCK> <SPEC> marocain </SPEC>
↪ Segmentation interdite : la séquence 1 a été soustraite du modèle Filler

[1] si, si, c'est possible !

<BCK> un </BCK> <SPEC> restaurant diététique marocain </SPEC>
↪ Segmentation interdite : on a privilégié la séquence 1 sur la séquence 2 :
Suppression de {.* un </B> <SPEC> restaurant diététique marocain </SPEC>}.

<SPEC> un restaurant diététique </SPEC> <SPEC> marocain </SPEC>
↪ Segmentation impossible : en raison des séquences 2 et 3, toutes les chaînes de type :
{.* restaurant <(B|/B)>? diététique </B> <SPEC> marocain </SPEC>} sont supprimées.

En fait, ces situations se produisent, lorsque nous avons deux règles qui se chevauchent (ici la séquence 1 et 2), et que celle qui n'est pas privilégiée (la 2) présente une inclusion dans une règle virtuelle qui serait obtenue par la concaténation de la règle privilégié (la 1) et d'une autre dans le modèle (ici la 3) :

$$\text{restaurant diététique marocain} \subset \text{un restaurant diététique } || \text{ marocain}$$

Or dans ces cas là, il est légitime de penser que cette règle virtuelle est valide pour représenter le concept ; comme solution, il a donc été choisi de l'ajouter. Une règle construit selon ce principe est, soit pertinente et dans ce cas lorsque la séquence de mots apparaît, il n'y a plus de segmentations impossibles, la segmentation se fait autour de cette nouvelle règle ; soit elle n'est pas pertinente, et alors la séquence de mots (relativement longue) a peu de chances d'apparaître et le fait que cette règle existe dans le modèle, même si elle n'est pas utile, ne perturbe pas l'analyse.

5.5.7 Élimination des ambiguïtés inter-concepts

Certaines règles peuvent être partagées par plusieurs concepts lorsqu'elles sont ambiguës. Par exemple la séquence de mots : « le deuxième » sans autre contexte peut faire référence à un lieu (le deuxième arrondissement) ou à une liste (le deuxième restaurant). Sans un contexte désambiguisateur, les deux concepts doivent être détectés et c'est au module de compréhension en fonction de ses connaissances sur l'historique du dialogue de prendre une décision sur la validité de l'un ou de l'autre. Au contraire, en présence de ce contexte, seul le bon concept doit être détecté. Le même procédé que dans la section 5.5.6 est appliqué entre les concepts pouvant être ambigus, afin d'éviter la génération du mauvais concept en présence d'un contexte clair. Exemple tableau 5.5. Le modèle ne génère plus le concept LIEU en présence du contexte droit « restaurant » et le concept LISTE en présence du contexte droit « arrondissement ».

TAB. 5.5 – Exemple de désambiguïsation inter-concept

Concept	LIEU	LISTE
Règles	le deuxième arrondissement le deuxième	le deuxième restaurant le deuxième
Chemins supprimés	.* <LIEU> le deuxième </LIEU> <B> restaurant .* .* <LISTE> le deuxième </LISTE> <B> arrondissement .*	

5.6 Processus de décodage

Le processus de décodage proposé aboutit à la création d'une liste structurée des N-meilleures hypothèses. En guise d'illustration, nous donnons ici un exemple simplifié.

5.6.1 Graphe de mots vers graphe de concepts

Lors d'un traitement d'une intervention utilisateur, le module de reconnaissance génère un graphe de mots codé en tant qu'accepteur G_W. Le semi-anneau utilisé est le *log semiring* (voir tableau 3.2) et la fonction $w(\pi)$ correspond au log des scores $P(A|W)P(W)$ où A est la séquence d'observations acoustiques, W la chaîne de mots représentant le chemin π, $P(A|W)$ la probabilité donnée par le modèle acoustique et $P(W)$ la probabilité donnée par le modèle de langage. Un exemple de G_W est observable à la figure 5.12. Les scores attachés à chaque transition sont donnés en $-logprob$.

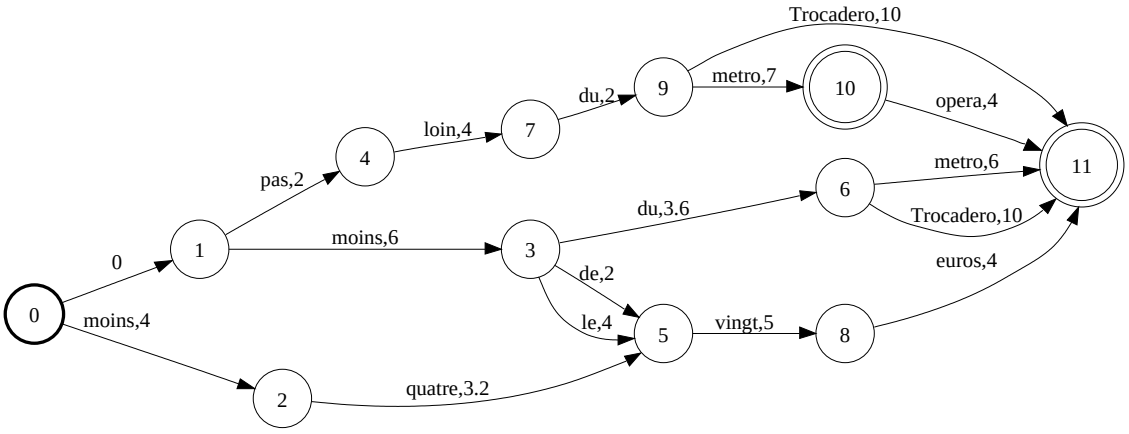

FIG. 5.12 – *Exemple de graphe de mots G_W généré par le module RAP*

G_W est composé avec le transducteur $T_{Concept}$ dans le but d'obtenir le transducteur mots-concepts T_{WC} : $T_{WC} = G_W \circ T_{Concept}$, illustré dans la figure 5.13.

Remarque : Dans l'espace de recherche T_{WC} il est maintenant possible de faire intervenir des connaissances sur l'agencement des concepts ou/et sur les prédictions des concepts à venir du système en fonction de l'état du dialogue.

Dans cet exemple, nous considérons 4 types de constituants conceptuels : les fonctions `NEAR` (pour un lieu) et `LESS` (pour un montant) ainsi que les étiquettes `LOC` pour les lieux et `AMOUNT` pour les valeurs monétaires. Ces constituants sont représentés par les patrons suivants dans cet exemple :

NEAR = pas loin du metro
NEAR = pas loin du $NAME
LOC = Trocadero
LOC = metro Opera
AMOUNT = $NUMBER euros
LESS = moins de $NUMBER

avec `$NAME` un nom propre et `$NUMBER` toute expression correspondant à un nombre.

Un chemin $Path(I,x,y,F)$ dans T_{WC} (avec I l'état initial et F un état final de T_{WC}) est une chaîne de mots si l'on considère seulement les symboles d'entrée x ou une séquence

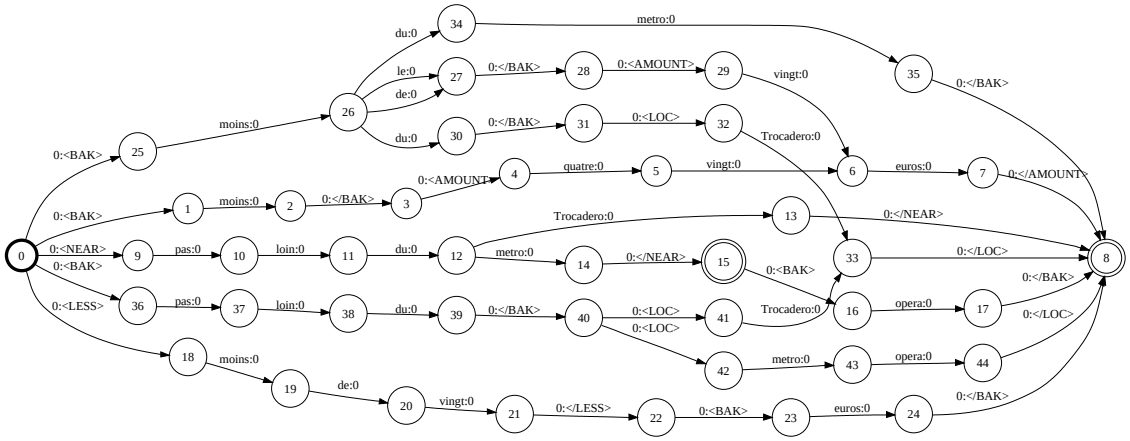

FIG. 5.13 – *Exemple d'un transducteur T_{WC} correspondant à la composition d'un accepteur représentant un graphe de mots généré par le module RAP et un transducteur mots-concepts $T_{Concept}$*

de concepts si l'on considère les symboles de sortie y. Dans le but d'obtenir toutes les interprétations possibles contenues dans G_W, nous projetons T_{WC} sur les symboles de sortie, puis nous déterminisons et minimisons le FSM résultant. L'accepteur obtenu est appelé G_C. Comme l'opération est effectuée sur le log semiring, le poids du chemin $Path(I,y,F)$ est la somme des poids de tous les chemins dans T_{WC} qui produisent y :

$$[G_C](y) = \bigoplus_x [T_{WC}](x,y) \tag{5.4}$$

Un exemple de G_C, obtenu à partir du transducteur T_{WC} de la figure 5.13 est visible dans la figure 5.14. Pour des raisons de clarté, nous avons omis les balises de fin de concept.

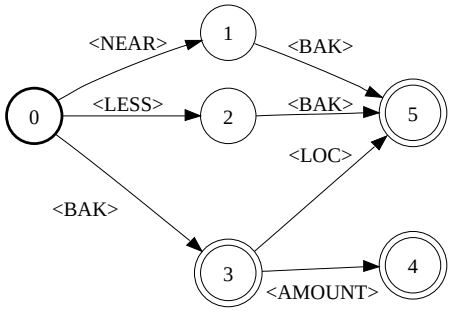

FIG. 5.14 – *Exemple d'accepteur G_C obtenu en projetant le transducteur T_{WC} sur les symboles de sortie*

La liste des N-meilleures interprétations conceptuelles $I_1, I_2, \ldots, I_n$ est obtenue en changeant le semi-anneau dans G_C, du « *log semiring* » au « *tropical semiring* » (voir tableau 3.2), puis nous énumérons les n meilleurs chemins S_1, S_2,... S_n dans G_C. Chaque interprétation I_i est une chaîne d'étiquettes γ représentée par un accepteur S_i avec le vocabulaire dans Γ. À chaque I_i est également attachée un accepteur G_{W_i} qui contient l'ensemble des chemins dans T_{WC} qui émet la chaîne I_i : $[G_{W_i}] = [T_{WC} \circ S_i]$. Ceci est représenté dans 5.15.

Cet ensemble de chaînes de concepts est la première étape du processus de génération de la *N-meilleure liste structurée* d'hypothèses L_{nbest} qui est la sortie finale de notre

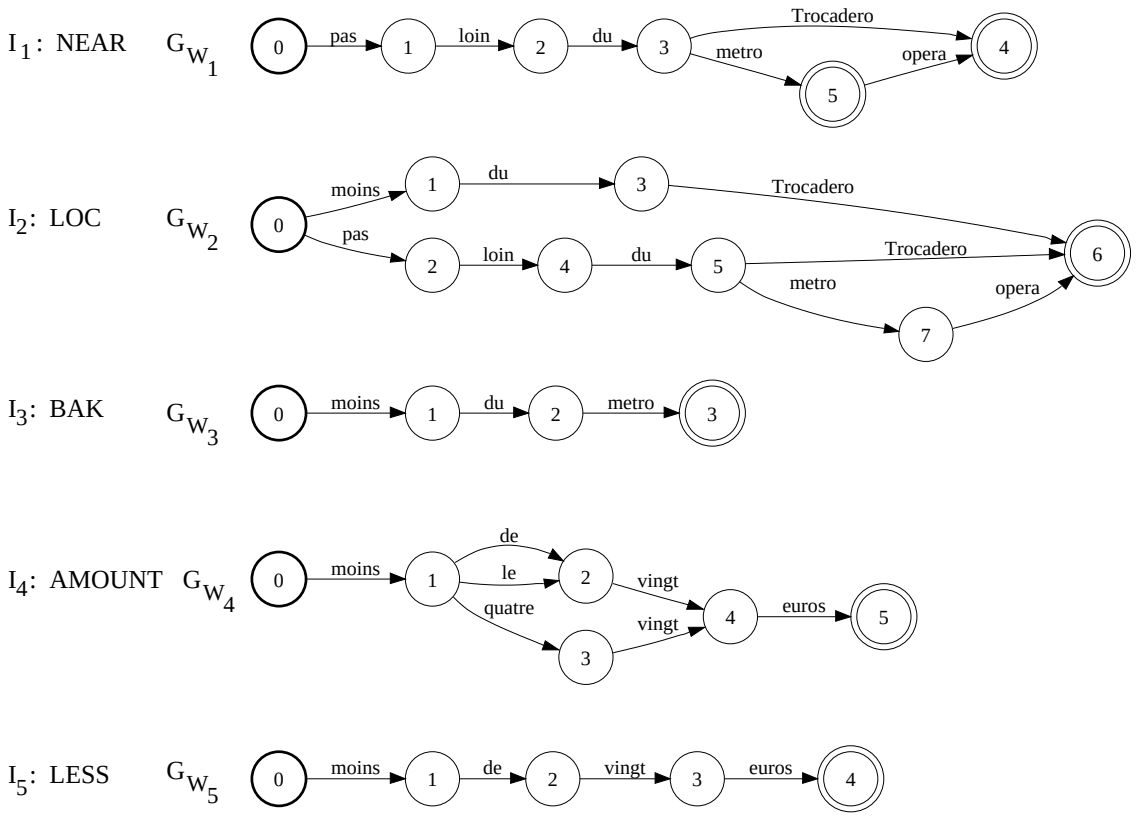

FIG. 5.15 – *Liste des* N*-meilleures interprétations conceptuelles* I_i *avec leur accepteur correspondant* G_{W_i}

architecture de décodage.

Il est à noter que les séquences de mots associées à différents concepts peuvent se chevaucher. En effet, les mots exprimant le sens d'une fonction peuvent être essentiels ou très utiles pour décider qu'un nom propre indique de manière non-ambiguë un lieu, par exemple, plutôt qu'un autre type d'entité, voir l'exemple plus bas. Un certain niveau de redondance peut aussi être très utile pour compenser les erreurs de reconnaissance. En raison de ce chevauchement, certains concepts (comme `NEAR` et `LOC` dans notre exemple) sont dans différentes interprétations (I_1 et I_2 dans la figure 5.15). Ceci est le second niveau dans notre processus de compréhension qui tente de fusionner, si c'est possible, ces concepts en appliquant des relations sémantiques, comme présenté dans la section suivante.

Il est à noter que ce processus a un impact négligeable sur la rapidité du décodage car les opérations sur les automates et transducteurs sont rapides.

Exemple : prenons la séquence de mots : « le deuxième restaurant italien », l'interprétation correcte associée est [< Accès_Liste_Restaurant :deuxième >< Spécialité :italien >]. Or, les valeurs seules de chaque concept ne sont pas caractéristiques des concepts qui doivent être détectés. Elles sont ambiguës sans la présence du mot restaurant. En effet « deuxième » sans contexte peut faire référence soit à un lieu (*e.g.* « deuxième arrondissement »), soit à une valeur d'accès à une liste et peut déclencher les deux concepts associés. « italien » seul peut faire référence à la spécialité culinaire où à un lieu (*e.g.* « quartier italien ») et déclencher les concepts de Spécialité et de Lieu. Le contexte désambigüisateur est pour les deux cas le mot « restaurant » qui doit être présent dans les deux règles permettant de détecter les concepts de l'interprétation : cela implique que les règles doivent se chevaucher

5.6.2 Application de relations sémantiques

Une fois que les constituents conceptuels ont été extraits, des relations sémantiques doivent être identifiées et utilisées pour instancier des structures sémantiques et faire de l'inférence. Il est à noter que les relations doivent avoir un support dans le graphe d'hypothèses de mots et le résultat de l'inférence également. Le support de la relation inférée est l'intersection des supports des interprétations basiques qui sont utilisées dans la relation d'inférence.

Une partie des connaissances sémantiques est faite d'implications exprimant des règles de constructions. Pour des raisons de clarté, un exemple d'inférence utilisant ces règles sera développé dans cette section. En utilisant la notation proposée dans [Jackendoff, 1990] la catégorie `path` est inférée par la règle suivante :

$$[PATH] \rightarrow \left[\begin{array}{l} \left\{ \begin{array}{c} TO \\ FROM \\ NEAR \\ TOWARD \\ \ldots \end{array} \right\} \quad \left(\left[\left\{ \begin{array}{c} THING \\ PLACE \end{array} \right\} \right] \right) \\ path \end{array} \right]$$

La règle établit que, par exemple, la composition de la fonction `NEAR` avec une instance de `PLACE` donne une instance de `PATH`. Toute composition obtenue est une structure sémantique correcte. Les règles de composition peuvent être utilisées dans des représentations sémantiques de structures plus complexes comme dans KLONE [Levesque et Brachman, 1985].

Par inférence, une instance de `<PATH>` : $\lfloor_{path} NEAR(\lfloor_{place} IN(\lfloor_{thing} LOC \rfloor)\rfloor)\rfloor$ peut être supputée par la présence d'hypothèses de la fonction `NEAR` et une instance de `<PLACE>`.

Si l'hypothèse `NEAR` est représentée avec l'accepteur $G_{W_{NEAR}}$ et `<PLACE>` avec l'accepteur $G_{W_{PLACE}}$ (obtenus avec la méthode présentée dans la section précédente), alors l'hypothèse $G_{W_{PATH}}$ pour une instance de `<PATH>` est générée si et seulement si :

$G_{W_{PATH}} = [G_{W_{NEAR}} \cap G_{W_{PLACE}}] \neq \emptyset$

Si la nouvelle structure sémantique générée rend possible l'application de nouvelles règles de composition, l'intersection des supports des constituants correspondants est appliquée et, si l'intersection n'est pas vide, la nouvelle structure sémantique inférée est ajoutée à l'ensemble des interprétations. Le processus est répété tant qu'il n'y a plus de compositions à effectuer.

À chaque structure sémantique I_i représentée par l'accepteur $G_{W_{I_i}}$ est attachée comme score la probabilité *a posteriori* suivante (avec X l'ensemble des chaînes acceptées par $G_{W_{I_i}}$ et Z l'ensemble des chaînes acceptées par W_G) :

$$P(I_i|Y) = \frac{\bigoplus_{x \in X}[G_{W_{I_i}}](x)}{\bigoplus_{z \in Z}[W_G](z)} \tag{5.5}$$

Par exemple, pour les 5 interprétations de la figure 5.15, 2 relations sémantiques peuvent être appliquées :

- une pour <PLACE>, basée sur les concepts NEAR (I_1) et LOC (I_2) ;
- une pour l'instance <MONEY> : $\lfloor_{money} LESS(\lfloor_{thing} AMOUNT \rfloor) \rfloor$ basée sur les concepts LESS (I_5) et AMOUNT (I_4).

Ceci amène à faire 6 opérations sur les accepteurs G_{W_i} :

$$\begin{aligned}
G_{W_{I_1 \cap I_2}} &= [G_{W_1} \cap G_{W_2}] && \rightarrow NEAR + LOC \\
G_{W_{I_1 - I_2}} &= [G_{W_1} - G_{W_2}] && \rightarrow NEAR \\
G_{W_{I_2 - I_1}} &= [G_{W_2} - G_{W_1}] && \rightarrow LOC \\
G_{W_{I_4 \cap I_5}} &= [G_{W_4} \cap G_{W_5}] && \rightarrow AMOUNT + LESS \\
G_{W_{I_4 - I_5}} &= [G_{W_4} - G_{W_5}] && \rightarrow AMOUNT \\
G_{W_{I_5 - I_4}} &= [G_{W_5} - G_{W_4}] && \rightarrow LESS
\end{aligned}$$

Comme $G_{W_{I_5 - I_4}} = \emptyset$, seulement 6 interprétations sont conservées : les 5 premières obtenues plus l'interprétation I_3 (BAK) qui n'a été impliquée dans aucune opération. Ces interprétations sont associées à un score en accord avec leur probabilité *a posteriori*. Sur le FSM de la figure 5.12, nous obtenons :

$$\begin{array}{llll}
P(I_1 \cap I_2) & = 0.58 & P(I_1 - I_2) & = 0.21 \\
P(I_3) & = 0.11 & P(I_4 - I_5) & = 0.07 \\
P(I_4 \cap I_5) & = 0.028 & P(I_2 - I_1) & = 0.002
\end{array}$$

La figure 5.16 montre les 6 interprétations conservées avec leur FSMs correspondant.

5.6.3 Liste des *N*-meilleures structurée des interprétations sémantiques

La dernière étape dans le processus de compréhension génère la liste des *N*-meilleures hypothèses en termes de valeur pour chaque hypothèse conceptuelle. Plusieurs valeurs peuvent être trouvées dans un FSM pour le même concept. C'est particulièrement vrai, quand les concepts représentent des entités numériques comme des numéros de téléphone ou des montants. Il est alors possible de générer pas seulement la meilleure chaîne de mots pour chaque interprétation I_i, mais plutôt la liste des *N*-meilleures chaînes de mots conduisant à des valeurs conceptuelles différentes.

L'extraction des *N*-meilleures valeurs peut être très utile dans un contexte de dialogue, ainsi que d'autres informations additionnelles (données clients, contraintes sur les valeurs, *etc.*) pour pouvoir sélectionner une valeur à l'intérieur d'une liste d'hypothèses. Par exemple [Rahim *et al.*, 2001] montrent que le fait d'utiliser un annuaire de téléphone pour filtrer automatiquement les numéros de téléphone de la liste des *N*-meilleurs candidats est très efficace : l'exactitude de la reconnaissance des chaînes de mots qui appartiennent à l'annuaire est de 94.5% (cela représente 61% des hypothèses) comparée à seulement 45% pour celles qui ne peuvent être trouvées dans un annuaire.

Ainsi, l'accepteur G_{W_i} attaché à chaque interprétation I_i est composé avec un transducteur qui émet seulement les différentes valeurs (principalement les noms propres et les valeurs numériques) contenues dans le FSM. Les *N*-meilleures valeurs ainsi que la chaîne de mots support correspondante sont associées à chaque I_i dans le but de construire une liste structurée des *N*-meilleures interprétations sémantiques (L_{nbest}).

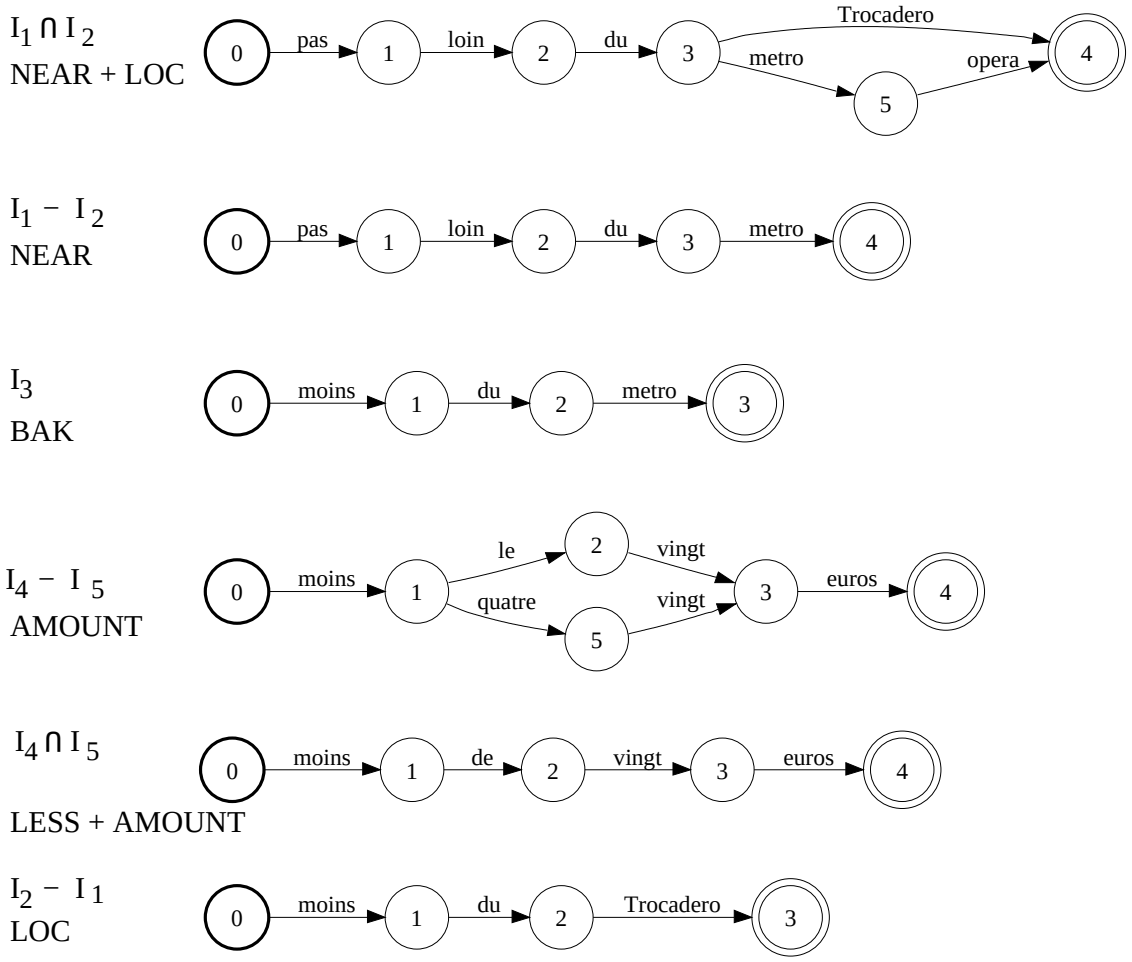

FIG. 5.16 – *Liste des* N*-meilleures interprétations (avec leur accepteurs correspondants) après application de relations sémantiques à la liste des* N*-meilleures de la figure 5.15*

La liste structurée des *N*-meilleures correspondant à notre exemple est présentée dans le tableau 5.6.

La liste structurée des *N*-meilleures hypothèses peut être vue comme le résumé de toutes les interprétations possibles d'une intervention utilisateur. Les 2 principaux avantages de cette liste structurée comparée à une liste standard sont les suivants :

1. la liste standard est produite en énumérant les *N*-meilleurs chemins produits par le module de RAP. Les scores utilisés sont une combinaison des scores du modèle acoustique et du modèle de langage ; aucun autre niveau linguistique n'est utilisé. Quand une liste standard est générée, les différences entre l'hypothèse *i* et l'hypothèse *i+1* sont souvent très petites, de l'ordre de un ou quelques mots. Ce phénomène est accentué quand le graphe de mots contient une zone peu fiable, due par exemple, à un mot hors-vocabulaire ou une entrée bruitée. Comme dans le contexte de dialogue oral, tous les mots ne sont pas importants pour le gestionnaire de dialogue, la différence au niveau mot entre deux hypothèses de la liste peut être sémantiquement non-pertinente. Un des avantages d'utiliser une liste structurée L_{nbest} est que toutes les hypothèses ont un sens différent du point de vue du gestionnaire de dialogue ;
2. en ayant les scores de confiance (les probabilités *a posteriori*) pour chaque inter-

TAB. 5.6 – Exemple de liste structurée des N-meilleures obtenue sur le graphe de mots de la figure 5.12 pondérée avec ses probabilités *a posteriori*

rang	**interprétation/valeur**	**score**
I_1	$\lfloor_{path}NEAR(\lfloor_{place}IN(\lfloor_{thing}LOC\rfloor)\rfloor)\rfloor$	0.58
$I_{1.1}$	$LOC(type:subway, valeur=opera)$	0.57
W	`pas loin du metro opera`	
$I_{1.2}$	$LOC(type:square, valeur=Trocadero)$	0.01
W	`pas loin du Trocadero`	
I_2	$NEAR$	0.21
W	`pas loin du metro`	
I_3	BAK	0.11
W	`moins du metro`	
I_4	$\lfloor_{thing}AMOUNT\rfloor$	0.07
$I_{4.1}$	$AMOUNT(type:euros, valeur=80)$	0.065
W	`moins quatre vingt euros`	
$I_{4.2}$	$AMOUNT(type:euros, valeur=20)$	0.005
W	`moins le vingt euros`	
I_5	$\lfloor_{money}LESS(\lfloor_{thing}AMOUNT\rfloor)\rfloor$	0.028
$I_{5.1}$	$AMOUNT(type:euros, valeur=20)$	0.028
W	`moins de vingt euros`	
I_6	$\lfloor_{thing}LOC\rfloor$	0.002
$I_{6.1}$	$LOC(type:square, valeur=Trocadero)$	0.002
W	`moins du Trocadero`	

prétation au niveau conceptuel et les interprétations avec les valeurs des concepts, L_{nbest} peut être utilisée par le gestionnaire de dialogue pour poser 2 types de questions :

- est-ce que les concepts attendus en accord avec l'état du dialogue sont dans la liste des possibles interprétations d'une intervention, et avec quel score de confiance ?
- pour un concept donné, quelles sont les valeurs possibles qui peuvent être trouvées et avec quel score de confiance ?

5.7 Intérêt de la liste structurée

Des expériences ont été menées sur l'enrichissement de l'espace de recherche au moyen de notre modèle de langage conceptuel construit à partir de la totalité des 59 concepts utilisés par France Télécom dans l'application PlanResto (section 4.2.4).

L'intérêt principal de la liste structurée est qu'elle permet de résumer un graphe de mots sans perdre d'informations. Elle propose en effet *toutes* les interprétations existantes dans le graphe, sans redondance, accompagnés des meilleures transcriptions les supportant. De ce fait, la liste structurée pour une taille équivalente permet de converger plus vite que la liste standard vers l'UER optimal qu'il est possible de trouver dans le graphe de mots du fait de l'absence de redondance au niveau conceptuel. Si l'on recherche dans une liste des N-meilleures hypothèses, l'hypothèse ayant le plus faible taux d'erreurs UER (taux Oracle) (voir figure 5.17) la liste structurée permet dès $N = 10$ d'obtenir l'UER le plus faible qui existe dans le graphe tandis qu'il faut $N > 40$ dans le cas d'une liste standard. L'écart est en rapport avec le nombre de concepts utilisés pour guider le décodage de la transcription. D'autres expériences ne faisant intervenir que les 3 principaux concepts avec valeur de l'application (*i.e.* Spécialité, Lieu, Prix) pour guider le processus de transcription donnent lieu à une convergence vers le plus faible taux

UER dès la troisième interprétation, *c.f.* tableau 5.7.

TAB. 5.7 – Comparaison de l'UER oracle lors d'un décodage faisant intervenir les 3 principaux concepts de l'application PlanResto

hypothèses	graphe de mots	meilleure hypothèse	liste standard des 3-meilleures	liste structurée des 3-meilleures
Taux UER Oracle (%)	9.3	26.0	17.1	9.8

Même si le WER et l'UER sont fortement corrélés, une réduction relativement forte de l'UER, n'induit pas une réduction du même niveau pour le WER. La (figure 5.18) montre le gain obtenu sur les taux d'erreurs WER et UER lorsque on sélectionne la meilleure hypothèse selon le critère de l'UER dans la liste. La réduction de l'UER va culminer à 60% tandis que le WER n'a été réduit que de 20%, ce qui tend à prouver le fait que tous les mots d'une phrase ne sont pas utiles pour la compréhension et que des informations linguistiques de niveau supérieur doivent être prises en compte. Réciproquement, en cherchant pour la meilleure hypothèse oracle selon le critère du WER (courbe 5.19), la réduction obtenu en WER passe rapidement à 30% contre 20% sur la courbe 5.18 mais la réduction sur l'UER culmine à 35% au lieu de 60%. L'optimisation d'un décodage en vue d'un meilleur taux d'erreurs mot n'implique pas obligatoirement un meilleur taux d'erreurs en compréhension. Dans tous les cas la liste structurée permet de converger plus vite vers la meilleure hypothèse que la liste standard grâce à l'absence des redondances. Un exemple de liste structurée utilisant les 59 concepts (voir section 4.2.4) de l'application PlanResto est présentée dans la figure 5.20. Le nombre en face de chaque transcription indique sa position dans une liste des *N*-meilleures standard.

5.8 Conclusion

Nous avons présenté l'implémentation d'un modèle conceptuel qui permet d'enrichir un graphe de mots des concepts utilisés par une application de dialogue. Ce modèle assure la détection de concepts autorisant les règles associées à chaque concept à partager des mots. Il effectue également, à son niveau, une première désambiguïsation conceptuelle. Une fois le graphe de mots enrichi conceptuellement, il est possible de faire des recherches guidées par les prédictions du gestionnaire de dialogue en plus des connaissances acoustiques et linguistiques. L'architecture de décodage proposée offre un moyen élégant et efficace de passer du graphe de mots au graphe de concepts. La sortie de reconnaissance fournit de manière automatique l'interprétation (séquence de concepts) d'une hypothèse. Ceci nous permet de générer une liste structurée sémantiquement des *N*-meilleurs candidats. Cette liste structurée met en compétition des hypothèses qui ne sont pas redondantes du point de vue du gestionnaire de dialogue car elles donnent lieu à des interprétations différentes. Nous avons montré à travers des mesures Oracle que la vraisemblance donnée par un système de RAP dans le choix d'une hypothèse n'est pas un critère optimal. La liste structurée propose très souvent, avec un nombre de candidats réduit, la solution optimale pour le module de compréhension. Le chapitre suivant propose un ensemble de mesures de confiance utiles pour diagnostiquer le résultat produit par notre décodage.

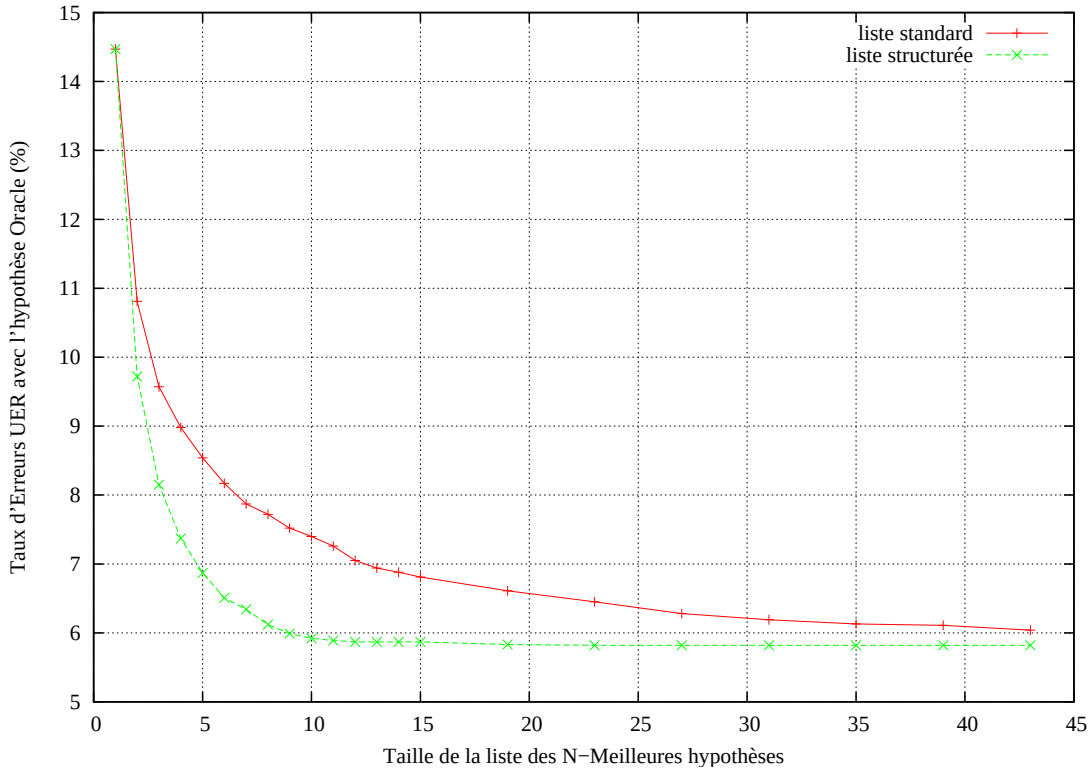

FIG. 5.17 – *Comparaison du plus faible UER (Oracle UER) dans une liste des* N*-meilleures hypothèses et une liste structurée*

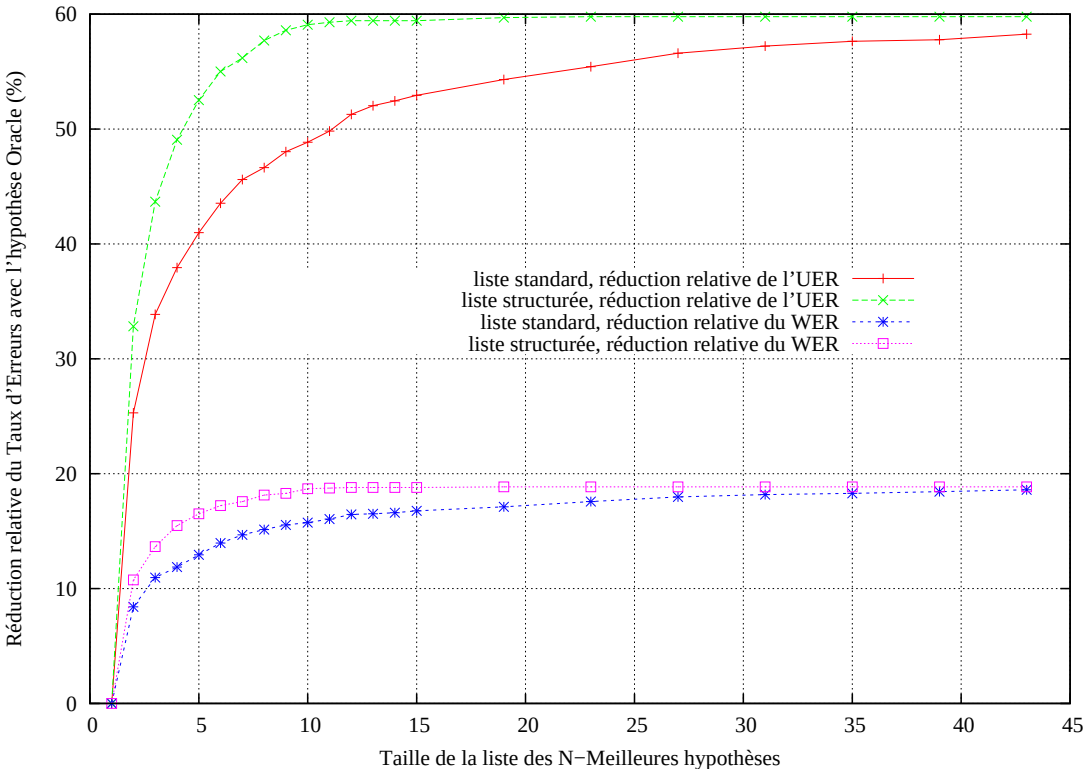

FIG. 5.18 – *Comparaison de la réduction d'erreur relative pour l'UER et le WER obtenue en choisissant manuellement l'hypothèse Oracle en fonction de l'UER dans une liste standard et structurée*

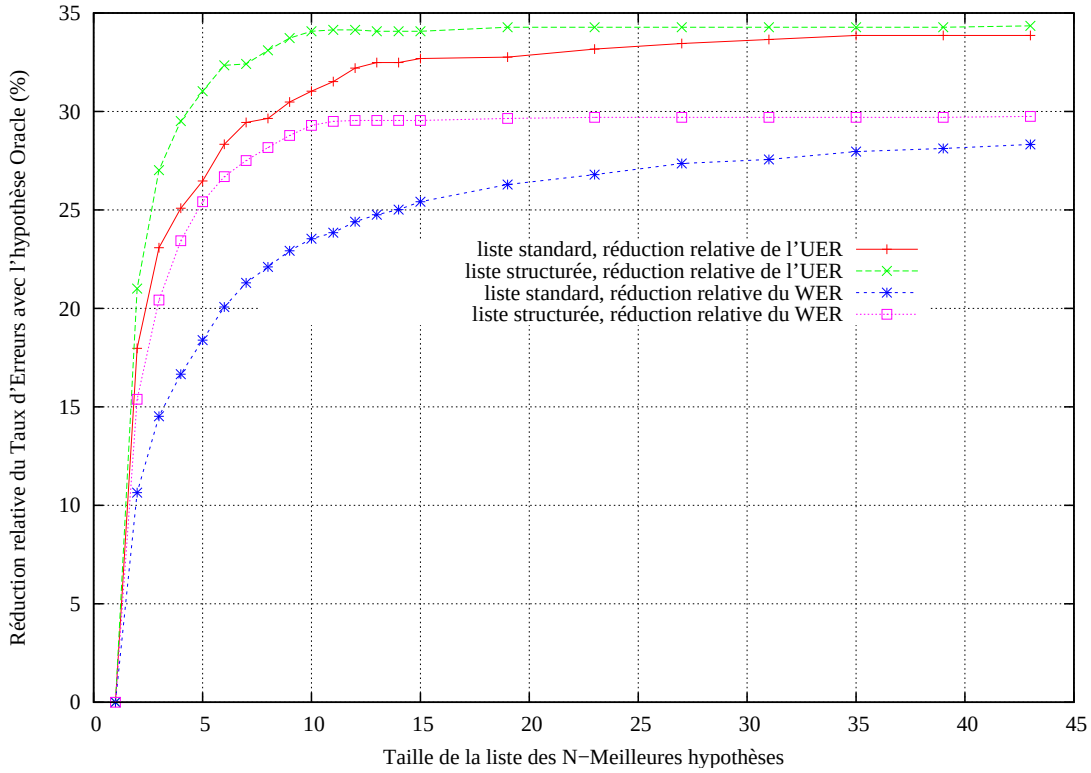

FIG. 5.19 – *Comparaison de la réduction d'erreur relative pour l'UER et le WER obtenue en choisissant manuellement l'hypothèse Oracle en fonction du WER dans une liste standard et structurée*

– **PHRASE PRONONCÉE**
  – dans le quartier des Halles le restaurant autour de dix euros par personne
– **PHRASE RECONNUE**
  – dans le quartier des vins euh le restaurant autour de dix euros par personne

✠ **INTERPRÉTATION 1** : < *dans* >< *claQuartier* >< *claRestaurant* >< *Prix* >
  – dans le quartier des vins euh le restaurant autour de *dix euros* par personne [1]
  – dans le quartier des vins euh le restaurant entre *deux mille euros* par personne [169]
  – dans le quartier des vins euh le restaurant de *trente-huit euros* par personne [1122]
✠ **INTERPRÉTATION 2** : < *Lieux* >< *claRestaurant* >< *Prix* >
  – dans le quartier des *Halles* restaurant autour de *dix euros* par personne [39]
✠ **INTERPRÉTATION 3** : < *dans* >< *claQuartier* >< *claRestaurant* >< *Specialite* >< *Prix* >
  – dans le quartier des vins euh le restaurant *breton* de *dix euros* par personne [809]
✠ **INTERPRÉTATION 4** : < *dans* >< *claQuartier* >< *claRestaurant* >< *valeur_card* >
  – dans le quartier des vins euh le restaurant entre deux euros par personne [1445]
  – dans le quartier des vins euh le restaurant entre deux des euros par personne [1834]

FIG. 5.20 – Exemple de liste structurée

CHAPITRE

6

Mesures de confiance

Sommaire

6.1 Introduction

Dans les applications de dialogue homme-machine, l'interprétation sémantique de la phrase exprimée par un utilisateur est effectuée sur la transcription générée par le moteur de reconnaissance de la parole. Cette reconnaissance est entachée d'erreurs, et la qualité de l'interprétation sémantique de la phrase est bien sûr très dépendante de la qualité de la reconnaissance. Une mauvaise interprétation peut amener le gestionnaire

de dialogue à faire de mauvais choix et donc mécontenter l'utilisateur. Le développement de mesures de confiance sur la transcription est alors crucial. Ces mesures de confiance sont utiles pour réaliser 2 objectifs :

1. aider le gestionnaire de dialogue sur les actions à entreprendre. Si les mesures de confiance estiment que la transcription et la détection des concepts sont sûres, le gestionnaire de dialogue pourra se fier à l'interprétation qui y en est faite. Dans le cas contraire, le gestionnaire de dialogue devra demander une confirmation ou une répétition ;
2. permettre de valider ou non la meilleure hypothèse d'une liste de *N*-meilleures hypothèses issue d'un RAP. Dans le cas, où la meilleure hypothèse n'est pas validée, ces mesures peuvent être utilisées pour tenter de la corriger. La correction peut revenir à trouver une alternative dans la liste structurée des *N*-meilleures hypothèses présentée dans le chapitre 5.

Typiquement, les mesures de confiance dépendent du type de tâche et de l'application. Les mesures populaires (voir par exemple [Falavigna *et al.*, 2002]) sont les scores donnés par le modèle acoustique et de langage, la densité du treillis, le comportement de repli du modèle de langage et les probabilités *a posteriori*. Dans la plupart, si ce n'est toutes les études, le moyen d'incorporer les informations sémantiques dans le processus de décision est plutôt ad-hoc. Une approche systématique est proposée dans [Sarikaya *et al.*, 2005].

Les mesures de confiance peuvent opérer à différents niveaux : mot, concept ou phrase. Elles peuvent exploiter différentes sources de connaissance : acoustique, linguistique ou sémantique. Les mesures linguistiques sont plus adaptées à estimer la qualité d'une phrase, tandis que les mesures acoustiques sont plus destinées a être utilisées au niveau du mot. Nous proposons dans ce chapitre un ensemble de mesures de confiance faisant intervenir différentes sources de connaissances permettant d'estimer la qualité de la reconnaissance à plusieurs niveaux : mot, concept et phrase. Ces mesures de confiance ne sont pas destinées à être comparées mais à nous donner un ensemble d'outils de diagnostics qui sera utilisé pour établir une stratégie d'aide à la décision présentée dans le chapitre 7.

Nous présenterons dans un premier temps, des mesures linguistiques intervenant au niveau de la phrase. L'observation qui a servi de point de départ à ces mesures est qu'une des raisons des taux d'erreurs mot élevés dans la transcription est que les modèles de langage *N*-grammes utilisés sont appris avec peu de données. Il en résulte que les probabilités des événements linguistiques sont peu fiables et que les méthodes de repli utilisées pour calculer la probabilité des événements non-observés sont parfois source d'erreurs de reconnaissance. Nous avions travaillé sur l'élaboration de techniques d'augmentation de données pour des modèles de langage *N*-grammes afin d'avoir recours aux méthodes de repli le moins souvent possible. Nous proposons une première mesure linguistique basée sur des situations de confiance obtenues par le consensus de différents décodages utilisant ces modèles, en parallèle avec un modèle classique (section 6.2). En suivant la même idée, (section 6.3), nous présenterons d'autres mesures de confiance mesurant la fréquence de l'utilisation des méthodes de repli.

Nous présentons ensuite une mesure acoustique classique permettant d'estimer la qualité de reconnaissance d'un mot, cette mesure est étendue au niveau conceptuel.

Nous présentons dans la dernière partie (section 6.5) une mesure de confiance sémantique adaptée pour estimer une confiance au niveau conceptuel. Cette méthode est basée sur l'utilisation redondante de classifieur.

6.2 Consensus de décodages en parallèle

6.2.1 Introduction

Si les données d'apprentissage d'un modèle de langage sont limitées, de nombreux *Bi*-grammes et surtout *Tri*-grammes qui apparaîtraient plus d'une fois dans un corpus idéalement dimensionné ont une probabilité estimée par un modèle de repli. Cette probabilité est souvent très différente de celle qui aurait été calculée avec un corpus d'apprentissage plus riche. De plus, dans beaucoup d'applications le corpus d'apprentissage disponible est biaisé par le fait qu'il a été collecté avec un nombre restreint de locuteurs et dans une période de temps limitée. L'effet produit est que la probabilité de certains *Bi*-grammes ou *Tri*-grammes est anormalement élevée. Ces considérations suggèrent que les comptes de certains événements plausibles devraient être générés même s'ils ne sont pas observés dans le corpus d'apprentissage. Nous présentons respectivement dans les sections 6.2.2 et 6.2.3, 2 méthodes permettant d'inférer de nouveaux événements et donc de limiter l'utilisation du back-off.

6.2.2 Augmentation de données par projection dans un espace réduit

La représentation des mots par des vecteurs dans un espace réduit peut être obtenue [Bellegarda, 1998], [Berry, 1992], [Searle, 1982] par décomposition en valeurs singulières (SVD)

Soit c_{ij} le nombre de fois où le mot w_i a été réellement observé dans le corpus d'apprentissage avec l'historique h_j. Soit a_{ij} le compte pour le même mot et historique obtenu par augmentation de données. Prenons d_{jk} la distance entre les vecteurs représentant les historiques h_j and h_k dans l'espace réduit. Notons $\Gamma_j(\Theta)$ l'ensemble des historiques dont les vecteurs sont proches du vecteur représentant h_j dans l'espace réduit. Le compte augmenté a_{ij} de la séquence $[(h_j = w_j)w_i]$ est obtenu, supposant que l'historique h_k contribue aux comptes de la séquence $[(h_j = w_j)w_i]$, en fonction du degré de similarité entre les deux historiques h_j et h_k :

$$a_{ij} = c_{ij} + \sum_{h_k \in \Gamma_j(\Theta)} c_{ik} f(d_{ik}) \tag{6.1}$$

Ce degré de similarité est représenté par une fonction $f(d_{ik})$ de la distance entre la représentation des deux historiques. La fonction $f(d_{ik})$ doit être égale à 1 quand $d_{ik} = 0$ et doit diminuer avec d_{ik}. Une fonction acceptable peut être celle de l'équation 6.2.

$$f(d_{ik}) = e^{-\frac{d_{ik}}{D}} \tag{6.2}$$

où D est un paramètre pour régler le système. Dans le modèle utilisé dans les évaluations, D a été fixé à 1. La distance ([Janiszek *et al.*, 2000]) Euclidienne entre chaque paire de vecteurs d'historique est calculée dans l'espace réduit. Il est intéressant de noter que le calcul de ces distances peut être obtenu sur un corpus différent de celui de l'application visée. Le modèle B_a utilisé dans la section 6.2.5 a été créé à partir des distances calculées sur un corpus généraliste composé d'articles du journal "Le Monde" de 40 millions de mots. Les distances peuvent être obtenues ou modifiées en utilisant d'autres critères comme la synonymie.

Cette augmentation de données accroît le nombre de *Bi*-grammes avec un compte supérieur à zéro. Néanmoins, un compte augmenté reste égal à zéro quand son compte avant augmentation était nul et qu'aucun historique proche du sien n'a été trouvé. Dans

ce cas, les *Bi*-grammes peuvent être augmentés par d'autres techniques. En pratique, l'ensemble $\Gamma_j(\Theta)$ est construit en considérant uniquement les distances inférieures à un seuil.

6.2.3 Augmentation de données par similarité

Prenons $t_g = xw_cw_d$, $t_c = w_gxw_d$, $t_d = w_gw_cx$, 3 *Tri*-grammes dans lequel le mot x apparaît. Considérons $c(t_q)$ étant le compte du *Tri*-gramme t_q(q=g,c,d). Nous souhaitons dériver par analogie le compte pour les *Tri*-grammes $t_g^{'} = yw_cw_d$, $t_c^{'} = w_gyw_d$, $t_d^{'} = w_gw_cy$ si x et y satisfont le prédicat $Analogue(x,y)$ défini dans la formule 6.3 :

$$Analogue(x,y) = [POS(x) = POS(y)] \wedge [SemComp(x,y)] \tag{6.3}$$

où $POS(w)$ indique la classe syntaxique (POS) du mot w. $SemComp(x,y)$ indique que x et y sont des mots sémantiquement compatibles. La compatibilité sémantique a été définie comme suit :

$$[SemComp(x,y) \text{ vraie } \Leftrightarrow d(x,y) < \delta]$$

où $d(x,y)$ est la distance entre les mots x et y définie dans [Janiszek *et al.*, 2000]. δ est un seuil fixé empiriquement.

La possibilité d'acquérir de nouvelles connaissances par analogie a été proposée dans [Evans, 1968, Polya, 1954, Brown, 1977, Yvon, 1996]. Le compte $c(t_q^{'})$ peut être obtenu de plusieurs manières.

Solution 1

Une solution simple consiste à positionner le compte à ϑ quand le compte original est inférieur à ce seuil ou que le *Tri*-gramme généré n'avait pas été observé, comme décrit dans la suite avec $t = hw$:

$$c(t_q^{'}) = \left\{ \begin{array}{ll} \rho_h\ c(hw) & si\ c(hw) \geq \vartheta \\ \vartheta & sinon \end{array} \right\} \tag{6.4}$$

Soit J_{qh} l'ensemble des *Tri*-grammes dont le compte a été augmenté à ϑ. Il inclut ceux dont le compte initial n'était pas nul et inférieur à ϑ_q. La constante ρ_h est dépendante de l'historique et peut être sélectionnée dans le but que la fréquence $R_a(hw)$ des comptes augmentés des *Tri*-grammes ayant un compte supérieur à ϑ avant augmentation soit proche de la fréquence $R(hw)$ que ces comptes avaient avant l'augmentation.

$$R_a(hw) = \frac{\rho_h c(hw)}{\sum\limits_{j \in J_h} \rho_h c(hj) + \vartheta N_h} = \frac{R(hw)}{1 + \frac{\vartheta N_h}{\sum\limits_{j \in J_h} \rho_h c(hj)}} \tag{6.5}$$

J_h est l'ensemble des *Tri*-grammes dont le compte avant augmentation était supérieur à ϑ, $R_a(hw)$ est la fréquence des comptes augmentés et N_h est le nombre de *Tri*-grammes dans J_{qh}. Dans le but d'être le plus proche de la condition désirée $R_a(hw) \approx R(hw)$ pour les *Tri*-grammes dans J_h, l'inégalité suivante doit être observée :

$$\frac{\vartheta N_h}{\sum\limits_{j \in J_h} \rho_h c(hj)} << 1$$

ce qui amène à :

$$\rho_h = \left\{ \begin{array}{ll} B\frac{\vartheta N_h}{\sum_{j\in J_h} c(hj)} & si\ N_h > 0 \\ 1 & sinon \end{array} \right\} \tag{6.6}$$

Solution 2

Le compte des nouveaux *Tri*-grammes ou des *Tri*-grammes dont le compte est inférieur à ϑ peut également être positionné ou réévalué en fonction du compte des *Tri*-grammes existants à partir desquels ils ont été générés (équation 6.7).

$$c(t'_g) = \left\{ \begin{array}{ll} c(yw_cw_d) & \text{si } c(yw_cw_d) > \vartheta \\ \alpha \left\{ \max_{z|Analogue(z,y)} c(zw_cw_d) \right\} e^{-d\left(y,\ \operatorname{argmax}_{z|Analogue(z,y)} c(zw_cw_d)\right)} & \text{sinon} \end{array} \right. \tag{6.7}$$

Où d est la mesure de distance proposée dans [Janiszek *et al.*, 2000] et α un paramètre choisi empiriquement pour régler le système.

On pourrait croire que l'augmentation des comptes des *Tri*-grammes selon ce principe revient à utiliser des classes de mots, comme pour les modèles *N-classes*. Mais ce n'est pas le cas, puisque les comptes modifiés des mots y analogues à x ne sont pas toujours les mêmes et dépendent du contexte. De plus, les probabilités obtenues sont différentes de celles obtenues avec des modèles à base de classes existants. Elles varient selon la distribution initiale des comptes des *Tri*-grammes et la distance entre les mots.

6.2.4 Évaluation

L'augmentation de données a été expérimentée pour construire des modèles de langage dans le but de générer automatiquement à partir du corpus d'apprentissage complet des événements manquants sur le corpus de test. Deux modèles de langage augmentés ont été créés à partir du corpus d'apprentissage de l'application AGS :

1. un modèle *Tri*-grammes (T_a), augmenté suivant la méthode décrite dans la section 6.2.3, en accord avec les formules 6.4, 6.5 et 6.6 avec $B = 10$. Pour certains langages, comme le français, de nombreuses erreurs sont dues à la reconnaissance erronée de prépositions, articles et d'autre mots courts apparaissant dans un *Tri*-gramme. Pour cette raison, l'attention a été portée sur les *Tri*-grammes faisant intervenir les plus fréquentes catégories de noms propres (*e.g.* villes, régions) ainsi que les noms géographiques et les expressions typiques ;
2. un modèle *Bi*-grammes (B_a), augmenté suivant la méthode décrite dans la section 6.2.2.

et ont été comparés à l'utilisation d'un modèle général *Tri*-grammes classique (T_g). Les expériences suivantes ont été menées sur les 1422 graphes du corpus AGS (voir section 4.1). Malgré différents tests, le modèle général reste le plus performant globalement avec un WER de 23% en moyenne sur les 1422 phrases de test, les deux autres modèles ne permettent pas d'améliorer globalement le WER sur ces 1422 phrases, avec un WER de 24% avec l'utilisation de T_a et 25% avec B_a. Toutefois, grâce à leur spécificité ils permettent ponctuellement de meilleures performances que le modèle général sur certaines instances, et notamment des phrases possédant des événements dont la probabilité est calculée par back-off dans l'utilisation de T_g. Si globalement les modèles de langages augmentés ne permettent pas d'améliorer le décodage, leur différence de comportement

est exploitée dans la section suivante pour isoler des situations de confiance en fonction des résultats donnés par un décodage en parallèle effectué avec ces 3 modèles.

Des expériences ont également été menées avec des modèles *Bi*-grammes construits suivant la méthode présentée dans la section 6.2.2 à partir de corpus de tailles différentes et comparées avec un modèle *Bi*-grammes classique construit à partir du même corpus. L'augmentation de données permet dans un premier temps un meilleur décodage que le modèle classique associé, mais il montre vite ses limites lors de l'utilisation d'un corpus de taille plus élevée.

6.2.5 Raisonnement sur des situations de consensus

Appelons $H(LM)$ la meilleure hypothèse de transcription obtenue en utilisant le modèle de langage LM. La comparaison des trois hypothèses obtenues lors de trois décodages en parallèle avec les modèles de langage T_g, T_a et B_a, nous permet d'isoler 3 situations de confiance différentes :

1. une situation de concordance des trois modèles $[H(T_g) = H(T_a) = H(B_a)]$, où les hypothèses concernées peuvent vraisemblablement être estampillées fiables ;
2. une situation regroupant les trois cas de concordance exclusive[1] des modèles deux à deux $[[H(T_g) = H(T_a)] \cup [H(T_g) = H(B_a)] \cup [H(T_a) = H(B_a)]]$, où les hypothèses sont peu fiables.
3. une situation de non-concordance où les modèles génèrent 3 hypothèses différentes $[H(T_g) \neq H(T_a) \neq H(B_a)]$, où les hypothèses ne sont pas fiables et vraisemblablement erronées.

TAB. 6.1 – Résultat Concordance sur le corpus de test de l'application AGS

	Situation	Nombre de Phrases	WER avec T_g
1	$H(T_g) = H(T_a) = H(B_a)$	980	12.86%
2	$H(T_g) = H(T_a)$ $\cup$ $H(T_g) = H(B_a)$ $\cup$ $H(T_a) = H(B_a)$	321	32.83%
3	$H(T_g) \neq H(T_a) \neq H(B_a)$	121	39.33%
WER Total avec $H(T_g)$: 22.92 % (1422 phrases)			

Le tableau 6.1 montre l'efficacité de la méthode à travers la situation de confiance (la situation n°1) qui regroupe 69% des hypothèses pour un WER presque deux fois plus faible que la moyenne sur la totalité des 1422 hypothèses. Il est intéressant de noter que dans la situation n°2, même si le modèle T_g donne des résultats globaux meilleurs que les modèles T_a et B_a, sur certaines hypothèses, ces derniers modèles assurent un meilleur décodage. En choisissant la meilleure hypothèse parmi les trois hypothèses produites en utilisant les trois modèles T_g, T_a et B_a, le WER dans la situation n°2 passe de 32.83% à 26.65%. 90 hypothèses sur 321 sont concernées (49% à 26%), et 31 sont totalement correctes. Cette stratégie ouvre donc la voie à l'application de stratégie de corrections d'erreurs.

On peut noter également, qu'en augmentant B de l'équation 6.6 pour la génération de T_a, le nombre de phrases dans la situation 1 diminue ainsi que le WER. La situation

[1] C'est à dire que dans la situation $[H(T_g) = H(T_a)] \Rightarrow [H(T_g) \neq H(B_a)]$

1 est donnée pour $B = 5$, par exemple, avec $B = 1000$, la situation 1 contient 898 phrases avec un WER de 10.73%.

6.3 CONS(LM) :une mesure linguistique

Peu de mesures de confiance purement linguistiques sont utilisées. [Eide *et al.*, 1995] et [Rayner *et al.*, 1994] en proposent certaines, mais pour être utilisées conjointement avec d'autres méthodes. Nous sommes partis dans une approche de ce type en raison de la constatation suivante : de nombreuses erreurs de reconnaissance sont dues à l'estimation incorrecte des probabilités des événements non observés dans le corpus d'apprentissage. Il est donc intéressant de vérifier si dans les hypothèses apparaissent de tels événements [Uhrik et Ward, 1997], et dans quelle proportion. Nous proposons dans la formule 6.8, une mesure appelée $CONS(LM)$, pour évaluer la consistance d'une hypothèse H_i par rapport au modèle de langage *Tri*-grammes *LM* utilisé, i représentant la i^{ieme} phrase du corpus de test.

$$CONS_i(LM) = \frac{n_{3g}(app_{LM} \cap H_i)}{n_{3g}(H_i)} \tag{6.8}$$

où $n_{3g}(app_{LM} \cap H_i)$ est le nombre de *Tri*-grammes de l'hypothèse H_i qui ont été observés au moins une fois dans le corpus d'apprentissage du modèle *LM*, et $n_{3g}(H_i)$ est le nombre de *Tri*-grammes que contient l'hypothèse H_i. Cette mesure de consistance est comprise entre 0 et 1 : 1 correspond à la mesure de consistance optimale. Cette mesure, simple à calculer, donne des résultats très intéressants pour la détection des hypothèses de reconnaissance erronées. Les résultats sur les hypothèses issues de l'utilisation du modèle général *Tri*-grammes T_g sont visibles dans le tableau 6.2. Ces résultats montrent que la mesure de consistance $CONS(LM)$ est très intéressante pour prédire le taux d'erreurs sur les mots des hypothèses de reconnaissance. Le second intérêt de cette mesure de consistance réside donc dans le type d'erreurs que cette mesure permet de détecter : il s'agit principalement d'erreurs dues au manque de données d'apprentissage caractérisées par la présence de *Tri*-grammes non vus dans l'hypothèse de reconnaissance[2], ou bien de graphes de mots ne proposant pas d'hypothèses de vraisemblance acoustique élevée contenant des *Tri*-grammes observés dans le corpus d'apprentissage.

TAB. 6.2 – Résultat de $CONS(T_g)$ sur le corpus de test de l'application AGS

$CONS(T_g)$	WER	Nb Phrases	cumul	
			WER	Nb Phrases
$= 1$	12.17 %	1011	12.17 %	1011
]1.00, 0.75]	29.65 %	222	17.08 %	1233
]0.75, 0.50]	45.57 %	160	21.67 %	1393
]0.50, 0.25]	69.89 %	27	22.89 %	1420
]0.25, 0.00]	75.00 %	2	22.92 %	1422
Taux Erreurs Mots Total : 22.92 % (1422 phrases)				

[2]Les valeurs du CONS(LM) sont plus élevées dans le cas du modèle T_a augmenté par analogie

6.3.1 Variante : critère de consistance sur des étiquettes morpho-syntaxiques

Le même critère peut être appliqué au niveau d'étiquettes morpho-syntaxiques (POS). Les hypothèses reconnues sont préalablement étiquetées de façon automatique, comme illustré dans le tableau 6.3 :

TAB. 6.3 – Exemple d'étiquetage morpho-syntaxique

je voudrais un serveur → PPER1S VREQ DETMS NMS		
avec :		
PPER1S	→	pronom personnel première personne du singulier
VREQ	→	verbe de requête
DETMS	→	déterminant masculin singulier
NMS	→	nom masculin singulier

Le corpus d'apprentissage est étiqueté de la même manière, et la mesure de consistance appelée $CONSPOS(LM)$ est définie comme $CONS(LM)$ explicité dans l'équation 6.8. Cette mesure permet de généraliser la détection de *Tri*-grammes incorrects au niveau grammatical. Un corpus, une fois transposé dans sa version grammaticale, même de taille limitée assure une couverture supérieure des événements, par rapport à sa version lexicale. Pour cette raison, le procédé permet d'être moins sévère, les hypothèses acceptées suivant cette méthode sont plus nombreuses, mais celles rejetées ont une grande chance d'être incorrectes. Les résultats sont présentés dans le tableau 6.4.

TAB. 6.4 – Résultat de $CONSPOS(T_g)$ sur le corpus de test de l'application AGS

CONS POS(LM)	WER	Nb Phrases	cumul WER	cumul Nb Phrases
= 1	15.86 %	1174	15.86 %	1174
]1.00, 0.75]	36.03 %	154	19.85 %	1328
]0.75, 0.50]	52.74 %	83	22.52 %	1411
]0.50, 0.25]	97.06 %	10	22.89 %	1421
]0.25, 0.00]	150.00 %	1	22.92 %	1422
Taux Erreurs Mots Total : 22.92 % (1422 phrases)				

6.3.2 La dépréciation des *Tri*-grammes peu plausibles

L'hypothèse de reconnaissance est parfois composée de *Tri*-grammes incohérents. Ceci est dû à un repli trop favorable : il arrive ainsi que des *Tri*-grammes non vus dans le corpus d'apprentissage soient favorisés au détriment de *Tri*-grammes observés dans une très faible mesure dans le corpus d'apprentissage. Ce phénomène est dû, comme énoncé par avant, à des techniques de repli peu fiables. Les critères de $CONS(LM)$ et notamment de $CONSPOS(LM)$ permettent d'identifier ces *Tri*-grammes peu plausibles. En effet, en faisant l'hypothèse que l'ensemble des *Tri*-grammes d'étiquette grammaticale observé sur un corpus d'apprentissage de très grande taille est très proche de l'ensemble des *Tri*-grammes d'étiquette grammaticale acceptable dans ce langage, il est

alors possible de considérer que tout *Tri*-gramme d'étiquette grammaticale n'apparaissant pas dans cet ensemble est peu plausible [Langlois *et al.*, 2003]. Cette hypothèse semble vraisemblable en raison de la relative stabilité des structures grammaticales d'un langage. La méthode proposée est donc la suivante : Lorsqu'une hypothèse de reconnaissance de première passe est associée à une mesure de consistance du modèle de langage $CONS(LM)$ inférieure à 1, nous vérifions la valeur de $CONSPOS(LM)$ et identifions les *Tri*-grammes qui impliquent que $CONSPOS(LM) < 1$, qui sont alors considérés comme peu plausibles. Dès lors, une phase de rescoring est effectuée avec un modèle adapté où ces *Tri*-grammes sont fortement pénalisés. La nouvelle hypothèse est soumise à la même procédure, et sera validée si $CONS(LM) = 1$, sinon il est possible d'itérer la procédure. Cette stratégie permet de baisser le WER de 38% à 32% sur un ensemble de 45 phrases concernées du corpus de test de l'application AGS. Il est intéressant de noter que certaines phrases sont totalement corrigées après l'opération (WER=0%).

Les mesures de confiance linguistiques précédentes ont de bonnes capacités pour diagnostiquer la qualité d'une phrase en terme de mots reconnus. L'utilisation conjuguée des différentes situations de confiance déduites de ces mesures peut amener à la construction d'une stratégie plus complexe de validation/rejet. Une stratégie élaborée faisant intervenir les notions précédentes a été développée au début de cette thèse [Estève *et al.*, 2003], elle n'est pas présentée dans ce document.

6.4 Mesure de confiance acoustique conceptuelle

Cette mesure de confiance fait la comparaison de la probabilité donnée par le modèle de reconnaissance de la parole pour une hypothèse donnée à celle qui aurait été obtenue par un modèle sans contrainte sur les boucles de phonèmes. Dans le but de rester consistant avec le modèle général, les unités acoustiques sont gardées identiques et la boucle est sur les phonèmes contexte dépendant, à savoir les allophones [Bartkova et Jouvet, 1991].

6.4.1 Probabilité au niveau des mots

Pour une hypothèse W identifiée par le modèle général (λ_G) de la trame t_0 à la trame t_n, la probabilité du signal de parole Y est comparée à la probabilité de la même portion de signal sur une boucle non-contrainte d'allophones. La probabilité est définie comme suit :

$$LR(Y \mid W) = \frac{P(Y \mid \lambda_G)}{P(Y \mid \lambda_{loop})} \tag{6.9}$$

Dans le but de pouvoir comparer les mesures pour des mots différents, nous comparons actuellement les log-prob et nous normalisons la différence par le nombre de trames sur lesquelles elles sont calculées.

$$\Delta_{loop}(Y \mid W) = \frac{\log LR(Y \mid W)}{N_{frame}(W)} = \frac{1}{N_{frame}(W)}[\log P(Y \mid \lambda_G) - \log P(Y \mid \lambda_{loop})] \tag{6.10}$$

En pratique, $\Delta loop(Y \mid W)$ est calculé à partir des scores acoustiques bruts produits par le processus de reconnaissance et :

$$\Delta_{loop}(Y \mid W) = \frac{1}{N_{frame}(W)}[Sc_G(Y) - Sc_{loop}(Y)] \tag{6.11}$$

Du fait du relâchement des contraintes, la probabilité du signal de parole sur λ_{loop} est supérieure à celle sur λ_G. Cela peut être vu comme une borne supérieure pour $P(Y \mid \lambda_G)$. Ainsi, $\Delta_{loop}(Y|W)$ est négatif et doit être interprété comme suit : au plus proche de zéro est la valeur, plus fiable est l'hypothèse W pour Y.

6.4.2 Probabilité au niveau conceptuel

Dans le but, de donner un score aux différents concepts d'une hypothèse, la mesure précédente peut être facilement étendue au niveau conceptuel. En fait, le Δ_{loop} pour une séquence de mots est dérivé du Δ_{loop} de chaque mot la composant. Soit Γ une structure conceptuelle composée de n mots $W_1,...W_n$, $\Delta_{loop}(Y \mid \Gamma)$ est approximée par :

$$\Delta_{loop}(Y \mid \Gamma) = \frac{1}{\sum_{i=1}^{n} N_{frame}(W_i)} \times \sum_{i=1}^{n} N_{frame}(W_i)\Delta_{loop}(Y_i \mid W_i) \tag{6.12}$$

Cette mesure n'est pas évaluée ici, mais des résultats seront présentés dans le chapitre 7.

6.5 Mesure de confiance conceptuelle

6.5.1 Introduction

Lors de notre processus de décodage, un concept est détecté par notre modèle conceptuel lorsque une règle de grammaire reconnaît un certain patron. Nos grammaires tentent de récupérer la plus longue séquence de mots correspondant au concept détecté. Toutefois, en cas d'erreurs de reconnaissance, la grammaire peut ne pas détecter le concept si ces erreurs sont sur les mots-clefs, ou ne reconnaître qu'une séquence de mots limitée, si ces erreurs sont sur les mots autres que les mots-clefs. Dans le premier cas le concept n'est pas détecté par la grammaire (car dans ce cas il n'est pas possible d'en extraire une valeur), mais la présence d'autres types de mots peut infirmer ce résultat. Dans le second cas, si le concept est détecté avec une séquence de mots plus courte (et donc moins fiable car il peut s'agir d'un mot erroné isolé), la présence ou l'absence de certains mots contextuels peut également confirmer cette détection ou l'infirmer.

Comme nous le disions dans le chapitre 3.1, des méthodes à base de classifieurs peuvent être utilisées pour détecter des concepts dans une phrase. Nous proposons ici d'entraîner des classifieurs automatiques pour détecter la présence des concepts de l'application dans une intervention utilisateur. La détection conceptuelle faite par nos grammaires peut alors être confirmée ou infirmée en fonction de la décision du classifieur.

6.5.2 Méthodes de classification textuelle

Les outils de classification de texte peuvent se différencier par la méthode de classification utilisée et par les éléments choisis afin de représenter l'information textuelle (mot, étiquette de Part Of Speech, lemme, stemme, sac de mots, sac de n-grams, longueur de phrase, *etc.*). L'objectif est d'entraîner des classifieurs à détecter les occurrences de chaque concept dans les interventions. À chaque concept détecté par nos grammaires, le classifieur ainsi entraîné peut donner son avis sur la présence de ce concept dans l'intervention. L'avis du classifieur sur la présence du concept devient une mesure de confiance pour ce concept.

Le corpus utilisé pour entraîner les classifieurs, est composé d'interventions utilisateur sous leur forme transcrite. Chaque transcription est étiquetée par son interprétation conceptuelle représentée par une séquence de constituants conceptuels basiques (ou concepts).

Dans les expériences menées sont utilisées les transcriptions manuelles du corpus d'apprentissage et les transcriptions manuelles et automatique du corpus de développement de l'application PlanResto. Chaque intervention est représentée sur plusieurs niveaux. Trois niveaux ont été utilisés :

1. le premier niveau est le mot lui-même ;
2. le second niveau est la catégorie morpho-syntaxique (POS) du mot. Ainsi le mot « végétarien » porte l'étiquette « AMS » (Adjectif Masculin Singulier) et le mot « voudrais » l'étiquette « V1S » (Verbe première personne du singulier) ;
3. le troisième niveau est la catégorie d'équivalence sémantique (SEM), à un nom de lieu est associé l'étiquette « XLOC », à un ordinal l'étiquette « ORDINAL », à une devise l'étiquette « DEVISE », *etc.*

TAB. 6.5 – Exemple de données d'apprentissage pour les classifieurs

Niveau 1	Niveau 2	Niveau 3
des	DETMP	OTHER
restaurants	NMP	OTHER
italiens	NMP	SPECIALITE
pour	PREP	OTHER
des	DETMP	OTHER
menus	NMP	OTHER
inférieurs	AMP	OTHER
à	PREPADE	OTHER
deux	CHIF	UNITE
cent	CHIF	CENTAINE
cinquante	CHIF	DIZAINE
francs	NMP	DEVISE

Concepts présents
claRestaurant
Specialite
Prix

À chaque intervention est alors associé l'ensemble des concepts présents, un exemple est disponible dans le tableau 6.5. À chaque exemple du corpus, et pour chaque concept de l'application, est noté la présence ou l'absence de ce concept. Un classifieur par concept est alors entraîné pour apprendre automatiquement des règles afin de pouvoir décider de la présence ou non d'un concept dans une phrase. Les scores donnés par le classifieur sont utilisés comme score de confiance.

Trois classifieurs sont utilisés dans les expériences suivantes : un basé sur les arbres de décisions *LIA-SCT* et deux autres sont des classifieurs à large-marge : *BoosTexter* et *SVM-Torch*.

LIA-SCT

LIA-SCT [Béchet *et al.*, 2000] est un logiciel libre développé par le Laboratoire Informatique d'Avignon et disponible à l'adresse suivante : `http://www.lia.univ-avignon.fr/chercheurs/bechet/`. L'avantage principal de ce classifieur est de prendre en entrée une séquence de composants (qui peuvent être des descriptions de différents niveaux d'abstraction, comme des mots et des étiquettes morpho-syntaxiques, par exemple) afin de construire automatiquement, à chaque nœud de l'arbre, une expression régulière incluant ces composants qui peuvent s'appliquer à la globalité du tour de parole. À chaque feuille, les hypothèses conceptuelles sont associées à une probabilité.

BoosTexter

BoosTexter [Schapire et Singer, 2000] est un classifieur basé sur une méthode de boosting de classifieurs à faible performance. Le but de cette méthode de classification à large-marge est de trouver une fonction qui maximise la marge entre les différents exemples à classer. Les classifieurs à faible performance sont passés en entrée. Ils peuvent être l'absence ou la présence d'un mot ou d'un n-gram spécifique, une valeur numérique (comme la longueur de la phrase prononcée) ou une combinaison de cela. À la fin du processus d'entraînement, la liste des classifieurs sélectionnés est obtenue ainsi que les poids de chacun d'entre eux dans le calcul du score final pour chaque constituant de la liste des concepts. Les éléments choisis dans nos expériences sont des 1-gram, 2-gram et 3-gram sur les trois niveaux présentés dans la section 6.5.2.

SVM-Torch

SVM-Torch [Collobert *et al.*, 2002] est un classifieur basé sur les SVMs dont l'entrée est un vecteur d'éléments numériques. Dans nos expériences, la technique la plus simple du sac de mots est utilisée : un tour de parole est représenté comme un vecteur dont chaque composante correspond à un élément appartenant à un des trois niveaux présenté dans la section 6.5.2 et chaque composante a pour valeur le nombre d'occurrence de l'élément correspondant dans le tour de parole.

6.5.3 Quelques résultats

TAB. 6.6 – Exemple sur la mesure de confiance donnée par LIA-SCT sur la détection des concepts

Hypothèse	syntagme conceptuel	Proba. donnée par le classifieur que le concept soit présent
X	moins de <Lieux> Sens </Lieux> francs	0.17
Y	<Prix> moins de cent francs </Prix>	0.90
Référence	<Prix> moins de cent francs </Prix>	

Dans l'exemple du tableau 6.6, on peut voir que le système de reconnaissance propose comme candidat, une hypothèse X où le mot-clef est erroné. La grammaire détecte alors un LIEU, à la place d'un PRIX. Pourtant il paraît évident que dans ce contexte c'est un PRIX qui a été prononcé. En s'appuyant sur ce contexte l'arbre de classification nous donne une probabilité très faible que le concept LIEU soit présent dans l'hypothèse X (0.17) et très forte que le concept PRIX y soit (0.90), la mesure de confiance nous permet d'infirmer ici, la détection faite par la grammaire sur l'hypothèse X et confirme au contraire celle sur l'hypothèse Y.

6.6 Conclusion

Nous avons présenté dans ce chapitre des travaux sur un panel de mesures de confiance. Ces mesures de confiance permettent de diagnostiquer la sortie de reconnaissance à différents niveaux (mot, phrase, concept) et avec différents critères (linguistique, acoustique, sémantique). Au niveau de la phrase, deux mesures de confiance linguistiques ont été présentées. Elles sont fondées autour de l'idée que l'utilisation de méthodes de repli peut être source d'erreurs de reconnaissance. La première exploite des situations consensuelles obtenues par des décodages en parallèle effectués avec des modèles de langage « augmentés » créés pour pallier le manque de données d'apprentissage. La seconde mesure l'impact négatif d'utilisation de ces méthodes en identifiant la proportion d'événements dans la phrase reconnue ayant une probabilité calculée par une méthode de repli. Une mesure de confiance acoustique intervenant au niveau des mots a été présentée et étendue au niveau conceptuel (*i.e.* à la séquence de mots relative à un concept). Nous avons présenté une mesure de confiance fonctionnant au niveau conceptuel qui utilise des méthodes de classification automatiques pour détecter des concepts dans une transcription afin de valider les détections faites par les grammaires. Ce chapitre a mis en évidence un ensemble d'outils pour diagnostiquer des hypothèses de sortie du module RAP. Nous nous proposons dans le chapitre suivant de mettre en place une stratégie d'aide à la décision pour le gestionnaire de dialogue en utilisant ces mesures de confiance sur notre liste structurée.

CHAPITRE

7

Stratégie de validation

Sommaire

7.1 Stratégie par arbre de décision

Le gestionnaire de dialogue guide les échanges avec l'utilisateur en fonction de son état actuel, et de la représentation sémantique que lui a fourni le module de compréhension en rapport avec la dernière intervention de l'utilisateur. La représentation sémantique peut être entachée d'erreurs et notamment si le processus de transcription n'a pas été fiable. Ces erreurs peuvent amener le gestionnaire de dialogue à prendre de mauvaises décisions sur le choix de la continuité à donner au dialogue. Des mesures de confiance sont généralement associées aux éléments de la représentation sémantique. Elles aident le gestionnaire à prendre une décision sur la suite à donner au dialogue : faire totalement confiance à cette représentation et continuer le dialogue ou demander

une confirmation partielle, voire une répétition. Le gestionnaire de dialogue fait ce choix en vue d'assurer la satisfaction maximale de l'utilisateur.

Nous proposons dans ce chapitre une stratégie d'aide à la décision destinée au gestionnaire de dialogue. Cette stratégie va permettre de l'informer sur la qualité du processus de reconnaissance transcription/interprétation obtenu par notre stratégie de décodage présentée dans la section 5.6 afin de lui donner les moyens de prendre la décision optimale sur le choix à faire dans la gestion du dialogue.

Notre processus de reconnaissance génère une liste structurée des N-meilleures hypothèses L_{nbest}. Ces hypothèses correspondent aux N-meilleures interprétations trouvées dans le graphe de mots. Une interprétation est une séquence de concepts. Ce travail considère que le calcul de la probabilité qu'une interprétation soit correcte est basé sur un ensemble suffisant d'indicateurs de confiance qui peuvent utiliser de multiples sources de connaissances sur L_{nbest}. Dans un premier temps, il est nécessaire de vérifier la plausabilité de l'interprétation $\Gamma_{1,1}^{w}$.

Nous proposons une stratégie d'interprétation implémentée par un arbre de décision. Au nœud j de l'arbre est appliquée une Unité de Décision DU_j sur L_{nbest}. Les unités de décision font de la validation dans le but de mener à des états dans lesquels les erreurs de compréhension sont peu probables. Quand les résultats d'interprétation sont dans un de ces états, le gestionnaire de dialogue n'a pas à effectuer des demandes de clarification ou des demandes de répétitions. Un chemin dans l'arbre de décision stratégique défini alors un *état de fiabilité de l'interprétation*.

L'arbre de décision peut être automatiquement appris ou construit manuellement (comme c'est le cas dans nos expériences). Son objectif est de maximiser la couverture de cas pour lesquels l'erreur d'interprétation est en dessous d'un seuil donné. Ce seuil est choisi pour assurer la satisfaction de l'utilisateur en évitant un taux inacceptable de fausses directions dans le dialogue.

À chaque nœud de l'arbre de décision, le taux d'erreurs et la couverture sont calculées. Si une interprétation Γ est acceptée par la DU correspondant au nœud, Γ peut être traitée par une autre DU ou être transférée au gestionnaire de dialogue. Si l'interprétation est rejetée, elle peut aussi être traitée par une autre DU ou envoyée vers une unité de correction qui cherche dans L_{nbest} une correction Γ' à Γ. Γ' est alors transférée au gestionnaire de dialogue avec l'état de fiabilité attaché au dernier nœud traversé dans l'arbre, ou l'intervention de l'utilisateur est rejetée si aucune correction fiable de Γ n'est trouvée dans L_{nbest}. Cette stratégie est illustrée par la figure 7.1 avec une DU.

Nous présentons ici une stratégie qui s'appuie sur deux unités de décision. La première, DU_1, est une unité de décision qui s'efforcera de diagnostiquer si l'interprétation $\Gamma_{1,1}$ (séquence de concepts) supportée par l'hypothèse $W_{1,1}$ produite par notre décodage est correcte. La seconde, DU_2, va s'attacher à valider de manière indépendante les concepts présents dans une hypothèse. Ces deux unités de décision utilisent différents classifieurs appris sur différents paramètres pour prendre une décision.

7.2 DU_1 : validation d'interprétation conceptuelle

7.2.1 Objectif

Nous désirons mettre en place une mesure de confiance conceptuelle permettant d'estimer la qualité d'une interprétation. C'est à dire, vérifier si la séquence de concepts donnée par notre modèle est correcte. Nous avons présenté dans la section 6.5 une mesure de confiance permettant de confirmer ou d'infirmer la présence d'un concept dans une phrase. La première unité de décision DU_1 va étendre cette mesure au niveau d'une interprétation.

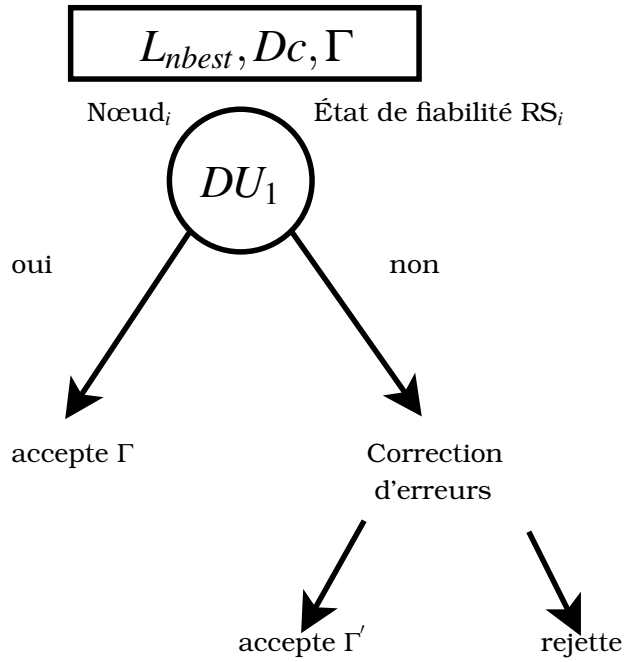

FIG. 7.1 – *Stratégie d'interprétation par arbre de décision avec une Unité de Décision (DU_i) validant l'interprétation* Γ *en accord avec* L_{nbest} *et le contexte du dialogue* Dc

7.2.2 Règle de décision consensuelle avec situations de confiance

La mesure de confiance présentée dans la section 6.5 permet de valider ou non un concept en demandant l'avis d'un classifieur entraîné à détecter sa présence selon un certain contexte. Le contexte dans ces expériences est la transcription complète. Nous interrogeons le classifieur sur tous les concepts de l'application avec la transcription complète. Il est alors possible pour chaque phrase de construire une interprétation conceptuelle associée au classifieur. Cette interprétation est la séquence de tous les concepts qui ont été détectés comme présents dans la phrase par un classifieur. Un concept est détecté présent par un classifieur, si la probabilité d'être présent donnée par le classifieur est supérieure à un seuil[1]. Ainsi nous avons 3 interprétations en plus de celle émise par notre processus de reconnaissance. Celle émise par LIA-SCT, Γ^{SCT}, celle émise par BoosTexter, Γ^{BoosT}, et celle émise par SVM-TORCH, Γ^{SVM}. Ces interprétations nous donnent une idée générale de l'interprétation réelle qui existe dans la phrase. Certaines informations que renferme l'interprétation produite par notre décodage n'existent pas ici : les classifieurs ne permettent pas l'association mots/concepts, alors ces interprétations ne donnent pas d'informations sur la valeur d'un concept et la relation temporelle entre les concepts est perdue. La présence multiple d'un concept dans une phrase n'est également pas détectée.

L'interrogation des classifieurs s'appuie pour chaque intervention sur $W_{1,1}$ la chaîne de mots la plus probable qui supporte $\Gamma_{1,1}$ dans le graphe de mots. Si $\Gamma_{1,1}$ est incorrecte parce que $W_{1,1}$ contient des erreurs de reconnaissance ou des expressions qui ne sont pas générées par notre décodage mais qui supportent des concepts, alors il est probable que les classifieurs ne soient pas en accord avec notre décodage. Dans ce cas, il est

[1] Ici 0.5. *i.e.* si la probabilité que le concept soit présent > la probabilité qu'il soit absent. Un seuil inférieur augmenterait le rappel sur la détection des concepts mais augmenterait la génération de faux concepts, tandis qu'un seuil supérieur améliorerait la fiabilité de détection des concepts au détriment du rappel. Nous avons choisi d'assurer cette fiabilité sur la concordance de plusieurs classifieurs plutôt que sur le seuil d'un seul d'entre eux

possible que l'interprétation correcte doit être trouvée dans une autre candidate de L_{nbest}. Au contraire, si tous les classifieurs sont en accord avec l'hypothèse $W_{1,1}$ générée par notre décodage, alors il est probable que l'interprétation suggérée soit correcte.

Si $\Gamma^{SCT}(w)$ est l'interprétation estimée par $LIA-SCT$ sur l'hypothèse de mots w, $\Gamma^{BOOST}(w)$ celle donnée par *BoosTexter* et $\Gamma^{TORCH}(w)$ celle fournie par $SVM-Torch$, la situation de confiance considérée est celle représentée par l'expression logique suivante :

$$HC(\Gamma_{1,1}, W_{1,1}) : \Gamma_{1,1} = \Gamma^{SCT}(W_{1,1}) = \Gamma^{BOOST}(W_{1,1}) = \Gamma^{TORCH}(W_{1,1})$$

Cette condition appelée *HC* pour Haute Confiance, correspond à l'unité de décision DU_1 choisie pour être à la racine de l'arbre de décision stratégique. La règle de consensus utilisée est motivée par la conjecture que différentes observations du même phénomène fournissent une interprétation plus sûre que chaque observation séparément. La figure 7.2 illustre ce procédé.

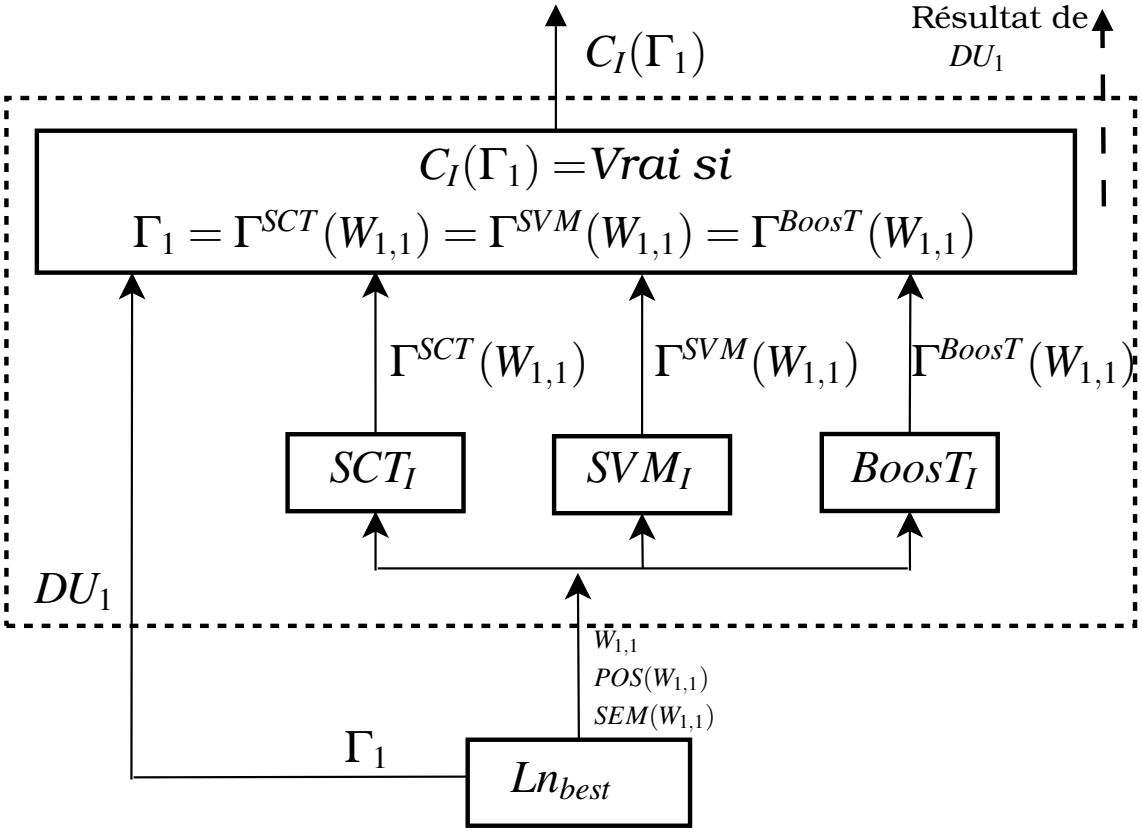

FIG. 7.2 – *Procédé de validation de l'unité de décision* DU_1

Cette unité de décision donne son avis sur la cohérence de l'interprétation fournie par notre processus de décodage en fonction de la meilleure chaîne de mots. En cas de non validation, cette unité permet également de fournir les informations suivantes :

- un classifieur peut avoir détecté la présence d'un concept dans $W_{1,1}$ qui n'a pas été détecté par notre décodage. Dans ce cas, et notamment si tous les classifieurs ont détecté sa présence, il est probable que l'interprétation $\Gamma_{1,1}$ ait subi une suppression :
- les classifieurs peuvent ne pas avoir détecté dans $W_{1,1}$ un concept présent dans $\Gamma_{1,1}$, dans ce cas il est fortement probable que l'interprétation $\Gamma_{1,1}$ ait généré une fausse

insertion.
Par contre, cette unité ne peut pas nous renseigner sur la cohérence d'une apparition multiple d'un concept.

7.3 DU_2 : validation conceptuelle

7.3.1 Objectif

L'unité de décision DU_1 valide les concepts détectés sur $W_{1,1}$. Cette unité de décision ne fournit pas d'informations sur le nombre d'occurrences d'un concept (en cas de d'apparition multiple), ni sur la valeur des concepts. Elle s'appuie sur des paramètres issus de $W_{1,1}$ qui peuvent être erronés et auxquels aucune information de confiance n'est passée. C'est pourquoi les entités conceptuelles dans $\Gamma_{1,1}$ sont validées à nouveau par une unité de décision DU_2 qui se charge de valider les concepts reconnus indépendamment en tenant compte de divers paramètres de fiabilité associés.

L'unité de décision DU_2 va s'efforcer de reconnaître un concept bien reconnu d'un mauvais. Cette distinction va se faire au moyen de différentes mesures de confiance. Les mesures de confiance qui peuvent être associées à un concept sont nombreuses. Certaines ont des performances globales bien meilleures que d'autres mais sont inefficaces dans certaines situations, *e.g.* la mesure acoustique *AC* qui globalement est une des plus performantes est inefficace pour estimer la confiance de concepts basés sur des mots homonymes (`<Lieu:Sens>` et `<Prix:cent>`), d'autres mesures sont alors plus pertinentes. L'idée est de combiner plusieurs mesures de confiance afin de tirer parti de leur potentiel spécifique. Cette combinaison sera faite en utilisant des classifieurs automatiques, les mêmes que ceux utilisés pour DU_1, autour des indices de confiance présentés dans la section 7.3.2.

7.3.2 Mesures de confiance

Les indicateurs de confiance suivants sont proposés :

LC, Mesure de confiance linguistique

La mesure utilisée est le *CONSLM* présenté dans la section 6.3.

AC, Mesure de confiance acoustique

La mesure conceptuelle présentée dans la section 6.4.2

R, Mesure de confiance sur le rang

Afin de prendre en compte le classement donné par le moteur de reconnaissance, le rang de l'hypothèse est considéré comme une mesure de confiance. Dans le cas d'une liste des N-meilleures standard, c'est le rang de l'hypothèse dans la liste. Dans le cas d'une liste structurée, le rang comporte deux nombres, le premier est le rang de l'interprétation, le deuxième est le rang de l'hypothèse pour cette interprétation.

SC, un descripteur de confiance sémantique

Cette mesure est dérivée des scores de classification donnés par les différents classifieurs (section 7.2) pour une phrase de contenir un concept spécifique. Il correspond

au nombre de classifieurs utilisés dans DU_1 qui ont fait l'hypothèse qu'un concept était présent.

DC, un descripteur du contexte de dialogue

Ce contexte est représenté par le prompt du système énoncé avant l'intervention utilisateur. Chaque prompt est étiqueté avec une étiquette correspondant au type de message donné à l'utilisateur (requête spécifique, confirmation, ...). Une distribution *a priori* de toutes les étiquettes conceptuelles pour chaque prompt est obtenue sur le corpus d'apprentissage. Pendant le décodage, la distribution attachée au prompt du système est comparée à celle détectée dans $\Gamma_{1,1}$.

Paramètres divers

Sont également ajoutés à l'apprentissage, le nombre de mots de la phrase, le nombre de concepts reconnus, la probabilité *a posteriori* de la phrase, ...

7.3.3 Application de méthodes de classification avec score de confiance

Le but est de valider chaque paire concept/valeur indépendamment des autres. L'unité de décision DU_2 calcule la probabilité que chaque constituant conceptuel γ_i ($\Gamma_{1,1} = \gamma_1, \gamma_2, \ldots, \gamma_i, \ldots$) est correct étant donnée une fonction de mesures de confiance pertinente, en accord avec le processus suivant :

- un corpus d'apprentissage est construit où chaque exemple correspond à un concept γ_i détecté par notre processus de décodage sur le corpus de développement ;
- à chaque exemple, est attaché l'ensemble des mesures de confiance introduit plus tôt ;
- si le concept γ_i est correct, l'exemple est étiqueté *correct*, sinon il est étiqueté *incorrect* ;
- les 3 outils de classification présentés dans la section 6.5.2 sont appliqués sur ce corpus d'apprentissage dans le but de discriminer les exemples *corrects* des *incorrects*.

Certains concepts ont des comportements différents : certains paramètres de confiance sont plus ou moins pertinents pour déterminer leur fiabilité. Il est alors plus judicieux, si les données d'apprentissage sont en nombre suffisant, d'entraîner un classifieur par concept. C'est ce qui a été réalisé dans nos expériences. Il est à noter que les données d'apprentissage font appel aux résultats de notre processus de reconnaissance sur le corpus de développement. Ce processus générant un liste structurée des N-meilleures hypothèses, les concepts ainsi que les mesures de confiance qui y sont attachés ne sont pas extraites que de $W_{1,1}$, mais de toutes ou partie de L_{nbest}. Ce qui permet d'augmenter le nombre d'exemples servant à l'apprentissage des classifieurs. Dans tous les cas, il est possible d'entraîner un classifieur pour tous les concepts ou par regroupement de concepts ayant le même comportement, si les données d'apprentissage sont en nombre insuffisant pour apprendre un classifieur par concept. Les résultats de cette classification ont été obtenus en considérant les concepts extraits d'une liste structurée élaguée à l'hypothèse $W_{3,5}$ et sont visibles dans le tableau 7.1. Dans ce tableau, la dernière colonne indique en choisissant le seuil optimal sur le corpus de développement le résultat de ce que l'on peut espérer de mieux de la discrimination des concepts *OK* ou *NOK* utilisant le score conceptuel acoustique *AC* présenté en section 6.4.2.

TAB. 7.1 – Résultats pour les corpus de développement et test de la classification *correct*/*incorrect* effectuée par les trois classifieurs

CORPUS	type	#exemples	LIA-SCT	BoosTexter	SVM-Torch	*AC*
Développement	NOK	13204	90.51	96.35	95.42	35.35
	OK	25895	95.65	98.38	97.86	93.06
Test	NOK	5236	85.19	86.45	85.37	46.21
	OK	9668	87.54	89.30	88.22	91.42

7.3.4 Validation de consensus

Comme dans l'unité de décision DU_1, le consensus des classifieurs donne un gain de garantie sur la validité d'un concept. Une validation de consensus sur les concepts γ_i est une variable binaire qui est vraie si γ_i est étiqueté *correct* par tous les classifieurs $V_{boost}(\gamma_i)$, $V_{tree}(\gamma_i)$ et $V_{SVM}(\gamma_i)$. Ce consensus est donc l'unité de décision DU_2. La fonction associée à DU_2 est la suivante :

$$F_2[L_{nbest}, Dc, \Gamma_{1,1}] : \forall \gamma_i \in \Gamma_{1,1}, V_{boost}(\gamma_i) = V_{tree}(\gamma_i) = V_{SVM}(\gamma_i) = correct$$

la figure 7.4 illustre ce procédé. L'unité de décision DU_2 a été choisie pour être exécutée à la suite de l'unité DU_1 dans notre arbre de décision stratégique.

En guise d'observation, un score S_{du_2} de validité de concept est calculé à partir des scores donnés par les 3 classifieurs. Ce score est comparé à la mesure de confiance *AC* au travers d'une courbe ROC (Receiver Operating Characteristic) dans la figure 7.3 sur le corpus de test. La courbe ROC compare, en fonction des valeurs de la mesure de confiance, les Fausses et les Correctes Acceptations d'un concept calculées de la manière suivante :

$$\text{Fausses Acceptations} = \frac{\text{\# de concepts accepté à tort}}{\text{Total \# de concepts erronés}}$$

$$\text{Correctes Acceptations} = \frac{\text{\# de concepts corrects accepté}}{\text{Total \# de concepts corrects}}$$

7.4 Résultats de la stratégie

La stratégie utilisée est maintenant un arbre de décision de profondeur 2 avec en racine l'unité de décision DU_1 suivie par l'unité de décision DU_2, illustrée dans la figure 7.5. Cette stratégie permet d'isoler 4 situations de confiance :

$RS_1 : DU_1 \wedge DU_2$
$RS_2 : DU_1 \wedge \overline{DU_2}$
$RS_3 : \overline{DU_1} \wedge DU_2$
$RS_4 : \overline{DU_1} \wedge \overline{DU_2}$

RS_1 correspond à la situation de validation totale, RS_2 et RS_3 des situations de validations intermédiaires, et RS_4 une situation d'invalidation totale. Ces situations de confiance engendrées par notre stratégie sur les corpus présentés dans la section 4.2 de l'application PlanResto, sont présentées dans les arbres des figures 7.6 pour le corpus de développement et 7.7 pour le corpus de test à travers 3 mesures :

– le *Taux d'Erreurs en Compréhension* (UER), similaire au word error rate mais au

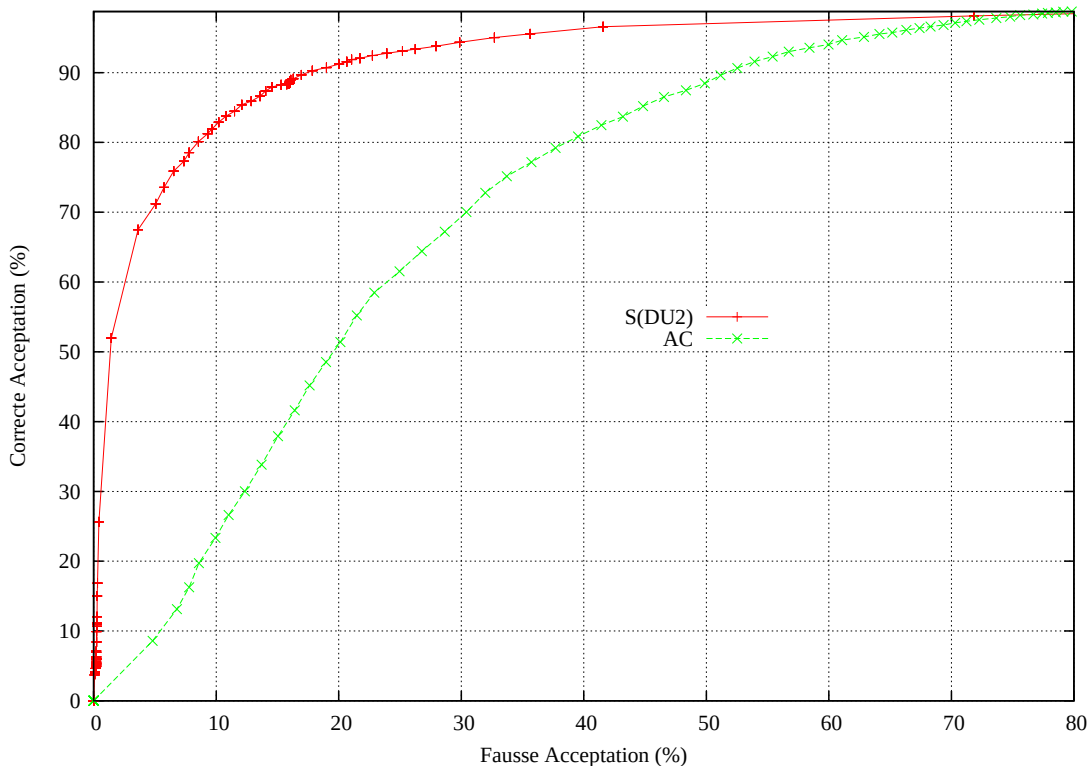

FIG. 7.3 – *Comparaison du score S_{du_2} avec la mesure AC sur leur capacité à diagnostiquer un concept*

niveau des concepts. Un concept est considéré correct seulement si l'étiquette et sa valeur sont correctes ;
- le *Taux d'Erreurs d'Interprétation Phrase* (SIER), qui indique le pourcentage d'interventions contenant au moins une erreur au niveau concept (étiquette ou valeur) ;
- la *Couverture* (Couv), qui indique le pourcentage d'interventions accepté par une unité de décision DU_i.

Nous pouvons effectivement remarquer que la situation de confiance optimale RS_1 est un ensemble contenant des hypothèses fiables. Comme nous pouvons le voir, en validant les deux unités de décision DU_1 et DU_2 sur le corpus de test l'UER chute vraiment significativement de 16.99% à moins de 6%, tandis que la couverture chute seulement à 58.44%. Cette situation optimale de confiance est comparée dans le tableau 7.3 avec la situation optimale que l'on pourrait obtenir avec la mesure de confiance acoustique AC (qui est la plus performante individuellement) pour une couverture identique. Le tableau 7.2 montre l'intérêt d'appliquer le consensus de différents classifieurs dans les unités de décision en présentant le gain en fiabilité apporté par chaque classifieur.

7.5 Correction d'erreurs

Les 4 situations de confiance RS_x présentées dans les figures 7.6 et 7.7 après applications des 2 unités de décision DU_1 et DU_2 peuvent logiquement être interprétées de la manière suivante :

1. la situation RS_1 ne contient *a priori* pas d'erreurs, les erreurs résiduelles qui peuvent persister peuvent être dues à des concepts non-détectés ni par la gram-

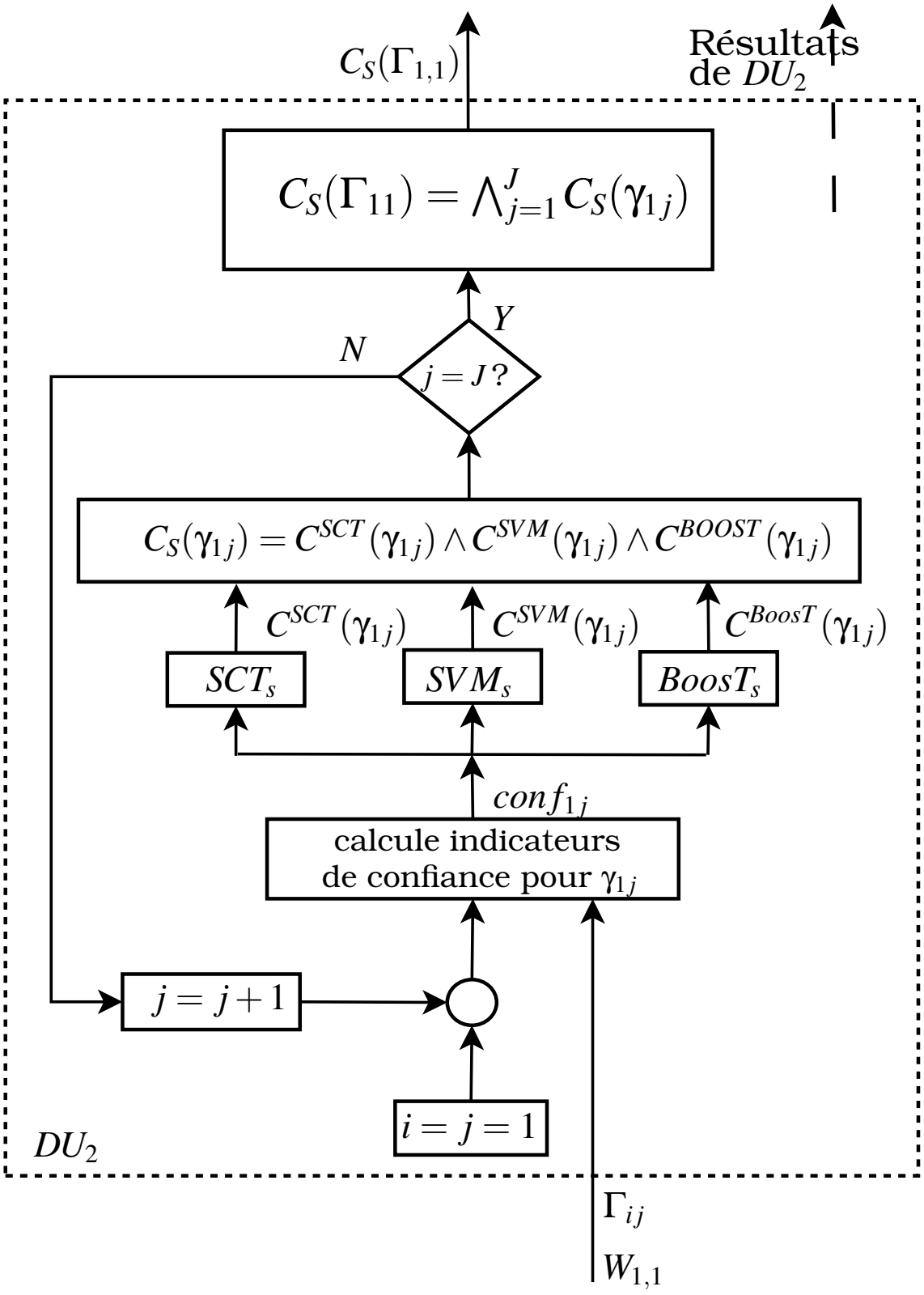

FIG. 7.4 – *Procédé de validation de l'unité de décision DU_2*

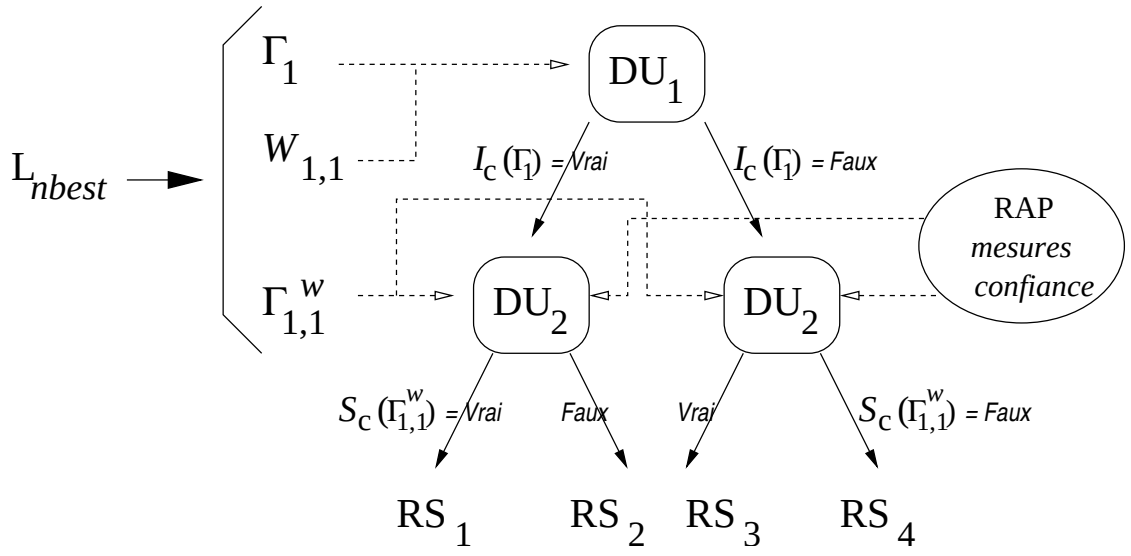

FIG. 7.5 – *Stratégie d'interprétation avec les unités de décision DU_1 et DU_2*

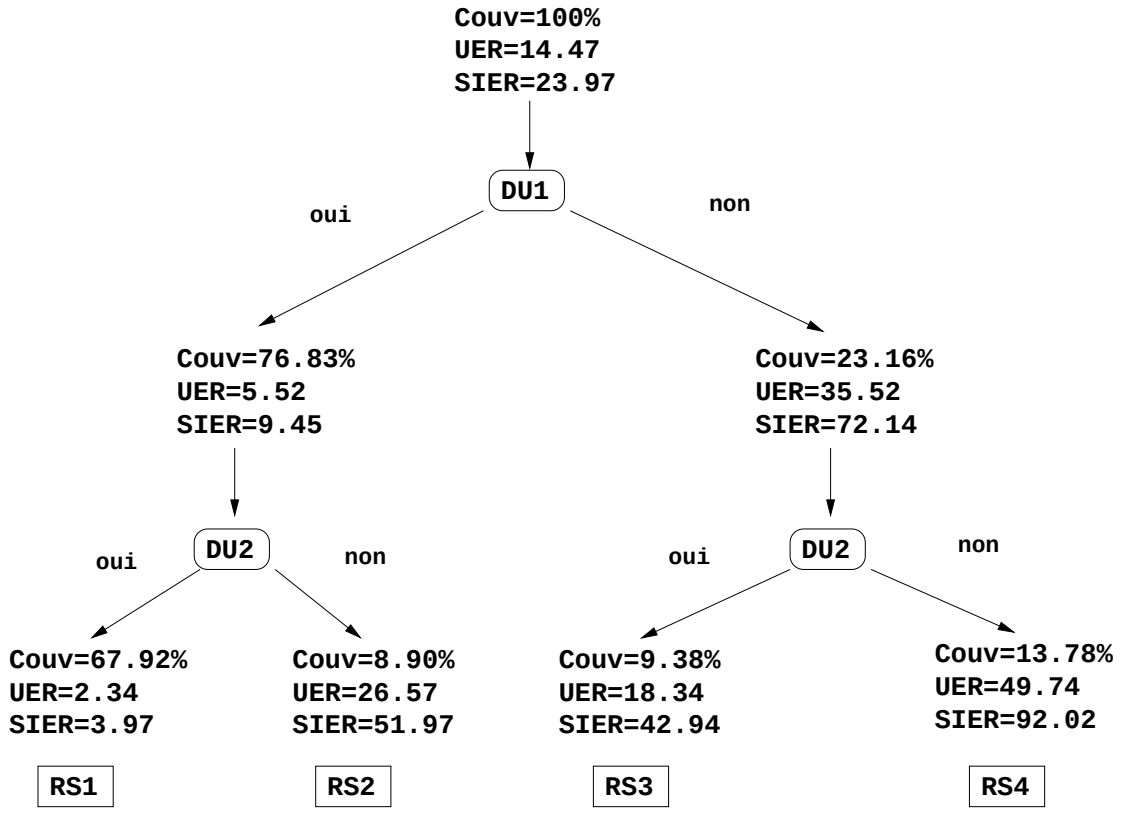

FIG. 7.6 – *Résultats de la stratégie d'interprétation pour les unités de décision DU_1 et DU_2 sur le corpus de développement*

maire, ni par les classifieurs, à cause d'erreurs de reconnaissance sur les mots, probablement des suppressions ;

2. la situation RS_2 contient les hypothèses validées par DU_1 mais invalidées par DU_2.

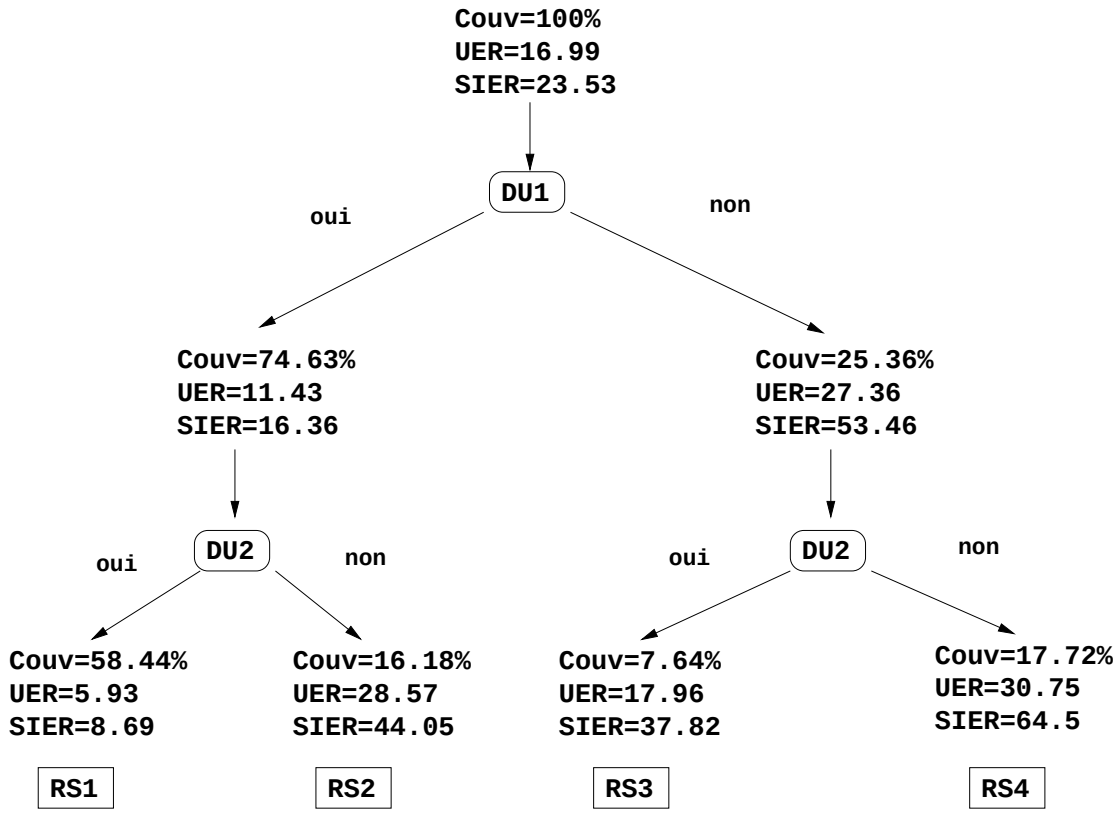

FIG. 7.7 – *Résultats de la stratégie d'interprétation pour les unités de décision DU_1 et DU_2 sur le corpus de test*

TAB. 7.2 – Performance des unités de décision DU_1 et DU_2 en fonction du nombre de classifieurs invoqué

DU	**classifieur**	**couverture**	**UER**
aucune	*aucun*	100%	17.0
DU_1	**+SCT**	86.5%	14.6
	+BOOST	76.1%	11.5
	+SVM	74.6%	11.4
+DU_2	**+SCT**	65.6%	9.1
	+BOOST	61.5%	6.9
	+SVM	58.4%	5.9

Dans ce cas il est probable que les erreurs produites sur les mots dans $W_{1,1}$ aient généré un concept validé dans l'interprétation par DU_1. Nous pouvons supposer que la majorité des erreurs sont des substitutions si ces mots non-erronés sont porteurs de sens et correspondent à un autre concept, ou a une fausse insertion s'ils ne sont pas porteurs de sens ;

3. dans la situation RS_3, les hypothèses sont invalidées par DU_1 mais validées par DU_2. Normalement, les concepts composant l'interprétation sont corrects, mais pas l'interprétation. Il est raisonnable de penser que les erreurs soient en majorité des suppressions ;

TAB. 7.3 – Comparaison des situations de confiance $RS1$ et celle obtenue avec la mesure de confiance AC

CORPUS	Couverture	Mesure	$RS1$	AC
Développement	67.92	UER	2.34	9
		SIER	3.97	14
Test	58.44	UER	5.93	10
		SIER	8.69	15.46

4. la situation RS_4 contient des hypothèses très peu fiables qui sont susceptibles d'être incorrectes vis à vis de leur interprétation ainsi que des concepts qui la composent.

Dans un cadre de fonctionnement plus ou moins idéal comme sur le corpus de développement, les réflexions précédentes semblent justifiées comme en atteste le tableau 7.4 qui présente les proportions de type d'erreurs commises en fonction des situations de confiance. Les expériences sur le test (tableau 7.5) montrent que ces réflexions dans une moindre mesure se confirment.

TAB. 7.4 – Type d'erreurs en fonction des situations de confiance RS_x sur le corpus de développement

Situation	Suppressions	Insertions	Substitution
RS_1	**84,03**	6,72	9,24
RS_2	10,34	**21,67**	**67,98**
RS_3	**98,06**	0,49	1,46
RS_4	13,04	32,00	54,96

TAB. 7.5 – Type d'erreurs en fonction des situations de confiance RS_x sur le corpus de test

Situation	Suppressions	Insertions	Substitution
RS_1	**54,84**	10,75	34,41
RS_2	31,94	18,06	**50,00**
RS_3	**62,26**	5,66	32,08
RS_4	22,31	22,31	55,38

En fonction de ces considérations des stratégies de corrections d'erreurs spécifiques peuvent être mises en place. Dans tous les cas, la solution a une forte probabilité d'être présente dans L_{nbest}. La correction revient alors à trouver une hypothèse $\Gamma_{x,y} \in L_{nbest} : \Gamma_{x,y}$ soit une correction pour $\Gamma_{1,1}$. Si une hypothèse $\Gamma_{1,1}$ est étiquetée avec un fort état de fiabilité tel RS_1, il peut être hasardeux de chercher une correction dans L_{nbest}, sinon il est intéressant de tenter de la corriger.

Pour évaluer le potentiel de cette méthode, il est intéressant d'estimer la limite inférieure de l'UER qui peut être trouvée dans la liste L_{nbest}. Cette limite est appelée *taux UER Oracle*. Il est obtenu à partir de la liste d'hypothèses en sélectionnant l'hypothèse avec le plus faible UER par rapport à la référence. Le taux UER Oracle est donné dans le tableau 7.6 pour chaque état de fiabilité $RS_{1,2,3,4}$ et pour deux types de liste d'hypothèses : une liste standard des N-meilleures hypothèses généré par un module de RAP et la liste structurée L_{nbest}. Nous pouvons observer que la L_{nbest} surpasse

significativement la liste standard : en gardant les 10 meilleures hypothèses, l'UER Oracle est atteint pour les 4 états. Il est intéressant de noter que le taux Oracle UER est bien corrélé avec les états de fiabilité : environ 1% pour l'état de haute fiabilité RS_1 à 20% pour l'état de faible fiabilité RS_4.

TAB. 7.6 – Taux Oracle moyen en UER dans la liste d'hypothèses attachée à chaque intervention avec le nombre minimum moyen d'hypothèses (n) qui doivent être conservées pour approcher cet UER dans les cas d'une liste standard et structurée des N-meilleures hypothèses

n-best	RS_1	RS_2	RS_3	RS_4
Oracle UER (%)	0.9	7.7	7.6	20.4
n liste standard	26	48	>50	>50
n dans L_{nbest}	6	9	9	10

La correction d'erreur I_q peut être vue comme un type spécial d'inférence dans laquelle une nouvelle interprétation $T_q(\Gamma^w_{i,j})$ est obtenue à partir de l'interpretation $\Gamma^w_{i,j}$ quand un certain pre-requis $F_q[L_{nbest},\Gamma_i]$ est vrai. La forme générale du $q^{\text{ième}}$ type de correction est :

$$I_q : \Gamma^w_{i,j} \wedge F_q[L_{nbest},\Gamma_i] \longrightarrow T_q(\Gamma^w_{i,j}) \tag{7.1}$$

où $F_q[L_{nbest},\Gamma_i]$ sont des expressions logiques conditionnant l'application d'une correction. Si les corrections ne s'appliquent que sur $\Gamma^w_{1,1}$, alors : $\Gamma^w_{i,j} = \Gamma^w_{1,1}$. Comme ces corrections sont cherchées dans L_{nbest}, nous avons : $T_q(\Gamma^w_{1,1}) = \Gamma^w_{c,r}$ avec $\Gamma^w_{c,r} \in L_{nbest}$.

Les types d'erreurs probables étant connus dans les situations RS_{2-3}, il est judicieux de mettre en place des règles de correction adaptées au type d'erreurs. Des propositions de méthodes de correction sont présentées dans la section 7.5.1. Dans la situation RS_4, se trouvent les hypothèses dont les erreurs sont les plus anarchiques, nous proposons une méthode de correction d'erreurs automatique dans la section 7.5.2.

7.5.1 Correction d'erreurs basée sur des règles

Dans les situations, RS_{2-3}, si une inconsistence sémantique est détectée dans $\Gamma_{1,1}$ il est intéressant de tenter de la corriger. 2 types de correction sont considérés pour les situations suivantes :

- l'absence d'un concept qui peut être retrouvé par inférence logique : ceci s'applique quand il y a une relation d'implication entre 2 concepts. Par exemple, si une instance d'un menu a été détectée , il est possible d'inférer en se basant sur le type de service et le contexte du dialogue que le menu est celui d'un restaurant ;
- l'incohérence d'une valeur : quand un concept γ_j apparaissant dans Γ_1 n'a pas de valeur ou une valeur incohérente dans $\Gamma_{1,1}$, il est intéressant de chercher une autre valeur pour γ_j dans $\Gamma_{1,x}$ avec $x > 1$, ou pour un autre concept corrigeant γ_j dans L_{nbest}.

Des exemples de correction de valeurs sont :

- si $\Gamma_{1,1}$ contient un concept γ_e pour lequel aucune valeur n'a été détectée, cela peut être parce que des mots-clefs sont présents, *e.g.* « quartier » dans $W_{1,1}$, mais pas de nom de lieu valide. Cependant une interprétation complète contenant un nom de lieu valide apparaît très probablement dans $\Gamma_{1,x}$.

– un autre exemple est la reconnaissance d'un nombre en réponse à une liste proposée par le système, qui est plus grand que la taille de la liste. Il est improbable que ce nombre réfère à un élément de la liste et il est judicieux alors de chercher un autre nombre dans la liste des N-meilleures valeurs ou un autre concept référant à un autre type de nombre.

Il est utile pour établir des règles de correction d'avoir des informations liées au gestionnaire de dialogue : état du dialogue, concepts attendus, *etc*. Nous ne disposons pas de ces informations mais à titre d'exemple une correction possible est la suivante : sur notre corpus de développement, des inconsistances linguistiques ont été observées en utilisant une approche de type *explanation based learning*. Chaque exemple a été généralisé manuellement afin de construire un patron pour détecter cette inconsistance et un patron représentant la correction. Si l'inconsistance est trouvée dans $\Gamma_{1,1}$ et la correction correspondante est trouvée dans L_{nbest} la correction est appliquée. Les patrons suivants ont été considérés pour corriger certaines suppressions :

– un verbe manquant au début de la phrase ;
– ajout d'un concept ou d'une valeur à la fin de la phrase.

Ces types de corrections sont appliqués au corpus de test dans l'état RS_3 où un nombre important de suppressions est observé dans le corpus de développement. Les résultats suivants sont observés :

– nombre de phrases= 119
– nombre d'erreurs avant correction= 45
– nombre d'erreurs après correction= 33

7.5.2 Corrections d'erreurs basées sur un arbre de décision

Si aucun de DU_1 ou DU_2 ne valide $\Gamma_{1,1}$, cela veut dire que $\Gamma_{1,1}$ est probablement au moins partiellement incorrecte. Dans cette situation les erreurs sont plus anarchiques et il est plus difficile de trouver des règles de correction manuelles. Nous proposons une méthode automatique pour trouver des règles de correction.
Toutes les corrections possibles sont considérées. Elles peuvent être exprimées comme :

– *insertion* : $I_{ins} : \Gamma^w_{1,1} \wedge F_{ins} \longrightarrow T_{ins}(\Gamma^w_{1,1}) = \Gamma^w_{c,r}$
  $\Gamma^w_{c,r} \in L_{nbest}$ contient au moins un concept de plus que $\Gamma^w_{1,1}$.
– *suppression* : $I_{del} : \Gamma^w_{1,1} \wedge F_{sup} \longrightarrow T_{del}(\Gamma^w_{1,1}) = \Gamma^w_{c,r}$
  $\Gamma^w_{c,r} \in L_{nbest}$ contient moins de concepts que $\Gamma^w_{1,1}$.
– *substitution* : $I_{sub} : \Gamma^w_{1,1} \wedge F_{sub} \longrightarrow T_{sub}(\Gamma^w_{1,1}) = \Gamma^w_{c,r}$
  $\Gamma^w_{c,r} \in L_{nbest}$ contient le même nombre de concepts que $\Gamma^w_{1,1}$.

Les fonctions des scores de confiance associées à F_q sont apprises au moyen d'arbres de décision avec la méthode suivante :

– toutes les interventions du corpus de développement sont traitées par notre module de décodage et notre liste structurée des N-meilleures hypothèses L_{nbest} est attachée à chacune d'entre elles ;
– un corpus d'exemples contenant chaque paire $(\Gamma_{1,1}, \Gamma_{c,r})$ de toute la liste L_{nbest} obtenue sur le corpus de développement est construit ;
– à chaque exemple est attaché l'ensemble des mesures de confiance présentées dans la section 7.3.2 associé aux deux hypothèses $\Gamma_{1,1}$ et $\Gamma_{c,r}$ ainsi que le type de modification (insertion, substitution, suppression) qui différencie les deux interprétations ;
– une étiquette, *correct* ou *incorrect*, est donnée à chaque paire ; *correct* signifie que les erreurs dans $\Gamma_{1,1}$ sont corrigés par $\Gamma_{c,r}$; *incorrect* signifie que $\Gamma_{c,r}$ n'est pas une correction pour $\Gamma_{1,1}$;

– finalement, un arbre de décision est entraîné sur ce corpus, les questions sont en relation avec les mesures de confiance sur $(\Gamma_{1,1}, \Gamma_{c,r})$ et le but du processus d'apprentissage est de réduire l'impureté entre les exemples *corrects* des *incorrects* attachés à chaque nœud.

Après la procédure d'apprentissage, les fonctions F_{ins}, F_{sup} et F_{sub} sont représentées par les différents chemins dans l'arbre. Chaque chemin est une expression logique composée de toutes les questions sur les mesures de confiance attachées aux nœuds traversés. Les probabilités de chaque correction $P\{q|F_j[L_{nbest}, \Gamma]\}$ (avec $q \in \{ins, sup, sub\}$) sont estimées en accord avec la distribution des exemples du corpus de développement parmi les différentes feuilles de l'arbre.

Durant le processus de décodage, cette stratégie de correction d'erreurs est utilisée seulement dans les états de faible fiabilité (RS_4) : toutes les paires $(\Gamma^w_{1,1}, \Gamma^w_{c,r})$ de L_{nbest} sont appliquées à l'arbre et celle avec la plus forte probabilité d'être correcte est passée à la stratégie de dialogue. Cette stratégie peut décider de rejeter la phrase si aucune solution alternative à $\Gamma^w_{1,1}$ n'est trouvée dans L_{nbest} avec une probabilité fiable. Le tableau 7.7 montre les résultats de cette correction d'erreur.

TAB. 7.7 – Résultats de la correction d'erreurs apprise automatiquement sur les corpus de développement et de test

Corpus	Mesure	Avant correction	Après correction
Développement	# Phrases corrigées	0	82
	UER(%)	49.74	43.74
Test	# Phrase corrigées	0	14
	UER(%)	30.75	28.96

7.6 Conclusion

Une stratégie d'interprétation séquentielle est proposée, basée sur un arbre de décision où les nœuds sont des unités de décisions effectuant des opérations de validation. La stratégie suit la conjecture que l'hypothèse faite à partir d'une intervention utilisateur doit être interprétée avec différents types de connaissances sémantiques. Nous avons proposé deux unités de décision, une pour vérifier une interprétation (séquence de concepts) extraite d'une intervention utilisateur, une autre permettant de valider un à un les différents concepts reconnus. Ces unités utilisent le consensus exprimé par différents classifieurs appris sur différentes sources de connaissances (*i.e.* mots, étiquettes morpho-syntaxiques, indices de confiance, *etc.*) pour donner une décision. La stratégie de validation proposée permet d'isoler des situations de confiance dans lesquelles nous pouvons prédire la qualité de la reconnaissance. Ceci est une information primordiale pour le gestionnaire de dialogue afin de guider ses choix sur la poursuite du dialogue. Dans l'état de haute fiabilité, les expériences ont montré que la probabilité d'exactitude est très haute, ce qui suggère que le gestionnaire de dialogue n'a pas à demander confirmation. Dans d'autres états, l'hypothèse est incertaine et le gestionnaire de dialogue devrait demander confirmation, voire une répétition. La prédiction de la probabilité des types d'erreurs en termes d'insertions, suppressions et substitutions est également utile pour le gestionnaire de dialogue. Nous introduisons, pour les états de moindre fiabilité, des méthodes de correction d'erreurs basées sur l'utilisation de notre liste structurée qui contient toutes les corrections possibles qui existent dans le graphe

de mots. Ces méthodes exploitent cette liste à travers des règles de correction manuelles ou automatiques.

CHAPITRE

8

Bilan

Le principal objet de cette étude a été la recherche de l'amélioration conjointe des étapes de reconnaissance de la parole et de compréhension dans les systèmes de dialogue oraux. Ces travaux sont partis du postulat que la séquentialité du traitement des tâches dans de tels systèmes, c'est à dire la génération de la transcription basée que sur des informations acoustiques et syntaxique réduite suivie de son analyse en compréhension en était une des ses faiblesses. Nous proposons une méthode permettant de faire coopérer les modules de transcription et compréhension.

Modèle de langage conceptuel

Nous avons présenté l'élaboration d'un modèle de langage basé sur le formalisme des transducteurs à états fini encodant des grammaires régulières permettant de faire l'association entre les mots et les concepts utilisés par le module de compréhension. Ce modèle de langage permet d'enrichir l'espace de recherche généré (graphe de mots) par le module de reconnaissance d'informations conceptuelles par une simple opération de composition.

Décodage conceptuel

Dans cet espace enrichi, représenté par un transducteur, les entrées correspondent aux mots tandis que les sorties correspondent aux concepts associés. Il est alors aisé de passer du graphe de mots au graphe de concepts en ne considérant que les sorties. Dans ce nouvel espace de recherche, il est possible de tenir compte des prédictions du gestionnaire de dialogue dans la recherche de la transcription. Nous proposons un processus de décodage qui cherche d'abord pour les meilleures interprétations (séquences de concepts) possibles et fournit une liste structurée des N-meilleures hypothèses conceptuelles accompagnées de leur meilleure chaîne de mots la supportant. Cette liste permet d'obtenir un résumé exhaustif et non-redondant du point de vue de la compréhension du graphe de mots.

Mesures de confiance

Nous avons proposé différentes mesures de confiance faisant intervenir des critères linguistiques, acoustiques ou sémantiques pour évaluer la qualité de la transcription à différents niveaux : mot, phrase et concept. Les mesures linguistiques chargées de diagnostiquer la qualité de la transcription au niveau de la phrase sont basées sur le fait que l'utilisation de méthodes de repli dans la construction d'un modèle de langage *N*-grammes est souvent générateur d'erreurs. Nous avons proposé de raisonner sur des situations de confiance obtenues par des décodages en parallèle effectués avec des modèles de langage augmentés permettant de faire appel aux méthodes de repli le moins souvent possible. Une mesure calculant la fréquence d'utilisation de ces méthodes de repli est également proposée et montre des résultats intéressants. Ont été également développées des mesures de confiance au niveau conceptuel qui utilisent des informations acoustiques ou plus originalement, des classifieurs textuels appris automatiquement.

Stratégie d'aide à la décision

La dernière partie de ces travaux s' est attachée à tirer parti des précédentes réalisations : la liste structurée et les mesures de confiance. Nous proposons une stratégie basée sur un arbre de décision ou chaque nœud est une unité de décision qui fait de la validation en fonction de la liste et des mesures de confiance. Nous aboutissons à des situations de fiabilité différentes avec dans certaines une probabilité très forte que l'hypothèse conceptuelle décodée soit correcte et d'autres non. Dans les situations où la probabilité que les hypothèses soient correctes est faible nous proposons différentes méthodes de correction d'erreurs. Cette stratégie permet de donner les moyens au gestionnaire de dialogue de prendre une décision plus optimale sur ses choix à effectuer dans la gestion du dialogue.

Perspectives

Cette thèse s'est attachée à faire coopérer le module de reconnaissance et le module de compréhension avec l'objectif d'améliorer conjointement ces deux modules. L'enrichissement conceptuel d'un graphe de mots que nous avons proposé permet un décodage conceptuel sur l'intégralité du graphe plutôt que sur la ou les meilleure(s) hypothèse(s) de mots. Avec suffisamment de données il serait intéressant de tester l'apport d'un modèle statistique de langage conceptuel lors de l'opération de décodage. Une des perspectives les plus intéressantes est de proposer une coopération plus poussée entre le module de transcription et d'autres modules du système de dialogue, telle que faire intervenir des informations liées au gestionnaire de dialogue : connaissances sur l'historique du dialogue, sur le prompt du système, sur les concepts décodés aux tours précédent, *etc*. On peut supposer qu'avec ces informations, il soit possible d'améliorer les processus suivants :

- le gestionnaire de dialogue en fonction du prompt du système et/ou des concepts déjà extraits peut estimer un ensemble de messages possibles que donnera l'utilisateur au prochain tour. Les concepts attendus peuvent être privilégiés grâce à l'ajout de scores dans le modèle conceptuel avant le processus de décodage ou ces informations peuvent être utiles pour filtrer efficacement la liste structurée des *N*-meilleures hypothèses conceptuelles ;
- la plupart des mesures de confiance sont mises au point à partir de paramètres liés au processus de reconnaissance de la parole. Les informations du gestionnaire de dialogue plus indépendantes de ce processus seraient certainement pertinentes, d'une part dans le diagnostic de la plausabilité du processus de recon-

naissance, mais surtout dans l'élaboration des méthodes de correction d'erreurs qui reviennent dans ces travaux à chercher une alternative crédible à la meilleure hypothèse dans la liste structurée des *N*-meilleures.

BIBLIOGRAPHIE

[Appelt et Jackson, 1992] Douglas APPELT et Eric JACKSON (1992). « Sri international february 1992 atis benchmark test results ». *In DARPA Workshop on Speech and Natural Language Processing*. 21

[Aust *et al.*, 1995] Harald AUST, Martin OERDER, Frank SEIDE et Volker STEINBISS (1995). « The philips automatic train timetable information system ». *Speech Communication*, 17:249–262. 24, 62

[Bartkova et Jouvet, 1991] Katarina BARTKOVA et Denis JOUVET (1991). « Modelization of allophones in a speech recognition system ». *In Proceedings of International Conference of Phonetic Science*, Aix-en-Provence, France. 95

[Béchet *et al.*, 2000] Frédéric BÉCHET, Alexis NASR et Franck GENET (2000). « Tagging unknown proper names using decision trees ». *In 38th Annual Meeting of the Association for Computational Linguistics, Hong-Kong, China*, pages 77–84. 98

[Bellegarda, 1998] Jérôme BELLEGARDA (1998). « Multi-span statistical language modeling for large vocabulary speech recognition ». *IEEE Transactions on Speech and Audio Processing*, 6(5):456–467. 89

[Berry, 1992] Michael W. BERRY (1992). « Large-scale sparse singular value computations ». *The International Journal of Supercomputer Applications*, 6(1):13–49. 89

[Brown, 1977] Richard BROWN (1977). « Use of analogy to achieve new expertise ». Rapport technique AI-TR-403, Artificial Intelligence Laboratory, Cambridge, MA. 90

[Cettolo *et al.*, 1998] Mauro CETTOLO, Roberto GRETTER et Renato DE MORI (1998). « Search and generation of word hypotheses ». *In* Renato DE MORI, éditeur : *Spoken Dialogues with Computers*, chapitre 9, pages 257–309. Academic Press. 15

[Chappelier *et al.*, 1999] Jean-Cédric CHAPPELIER, Martin RAJMAN, Ramón ARAGÜÉS PELEATO et Antoine ROZENKNOP (1999). « Lattice parsing for speech recognition ». *In Proceedings of the 6th conference on Traitement Automatique du Langage Naturel TALN'99*, Cargese, Corsica, France. 18

[Chen et Goodman, 1996] Stanley F. CHEN et Joshua GOODMAN (1996). « An empirical study of smoothing techniques for language modeling ». *In* A. JOSHI et M. PALMER, éditeurs : *Proceedings of the Thirty-Fourth Annual Meeting of the Association for*

Computational Linguistics, pages 310–318, San Francisco, USA. Morgan Kaufmann Publishers. 13

[Chomsky, 1957] Noam CHOMSKY (1957). *Syntactic structures*. Mouton, The Hague. 32

[Chomsky, 1965] Noam CHOMSKY (1965). *Aspects of the theory of syntax*. MIT Press, Cambridge, MA. 32

[Collobert *et al.*, 2002] Ronan COLLOBERT, Samy BENGIO et Johnny MARIÉTHOZ (2002). « Torch : a modular machine learning software library ». *In Technical Report IDIAP-RR02-46, IDIAP*. 98

[Damnati, 2000] Géraldine DAMNATI (2000). *Modèles de langage et classification automatique pour la reconnaissance de la parole continue dans un contexte de dialogue oral homme-machine*. Thèse de doctorat, Université d'Avignon et des Pays de Vaucluse, Avignon, France. 52

[Eide *et al.*, 1995] Ellen EIDE, Herbert GISH, Philippe JEANRENAUD et Angela MIELKE (1995). « Understanding and improving speech recognition performance through the use of linguistic tools ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, volume 1, pages 221–224, Detroit, MI. 93

[Estève, 2002] Yannick ESTÈVE (2002). *Intégration de sources de connaissances pour la modélisation stochastique du langage appliquée à la parole continue dans un contexte de dialogue oral homme-machine*. Thèse de doctorat, Université d'Avignon. 30

[Estève *et al.*, 2003] Yannick ESTÈVE, Christian RAYMOND, Renato DE MORI et David JANISZEK (2003). « On the use of linguistic consistency in systems for human-computer dialogs ». *IEEE Transactions on Speech and Audio Processing*, 11(6):746–756. 95

[Evans, 1968] Thomas G. EVANS (1968). « A program for the solution of a class of geometric analogy intelligence test questions ». *In* M. MINSLY, éditeur : *Semantic Information Processing*. MIT Press, Cambridge, MA. 90

[Falavigna *et al.*, 2002] Daniele FALAVIGNA, Roberto GRETTER et Giuseppe RICCARDI (2002). « Acoustic and word lattice based algorithms for confidence scores ». *In Proceedings of the International Conference on Spoken Language Processing*, pages 1621–1624, Denver, Colorado, USA. 88

[Fillmore, 1968] Charles J. FILLMORE (1968). « The case for case ». *In* Emmon BACH et Robert T. HARMS, éditeurs : *Universals of Linguistic Theory*, pages 1–88. Holt, Rinehart, and Winston, New York. 20

[Freund et Schapire, 1996] Yoav FREUND et Robert E. SCHAPIRE (1996). « Experiments with a new boosting algorithm. ». *In Thirteenth International Conference on Machine Learning*, pages 148–156. 41

[Gorin *et al.*, 1997] Allen L. GORIN, Giuseppe RICCARDI et Jeremy H. WRIGHT (1997). « How may i help you ? ». *Speech Communication*, 23(1-2):113–127. 30

[Hacioglu, 2004] Kadri HACIOGLU (2004). « A lightweight semantic chunker based on tagging ». *In Proceedings HLT-NAACL Conference*, pages 145–148, Boston, Massachussetts, USA. 64

[Hacioglu et Ward, 2001] Kadri HACIOGLU et Wayne WARD (2001). « A word graph interface for a flexible concept based speech understanding framework ». *In Proceedings of European Conference on Speech Communication and Technology*, pages 1775–1778, Aalborg, Denmark. 62

[Haffner *et al.*, 2003] Patrick HAFFNER, Gokhan TUR et Jerry WRIGHT (2003). « Optimizing SVMs for complex call classification ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, Hong-Kong. 30

[He et Young, 2003] Yulan HE et Steve YOUNG (2003). « Hidden vector state model for hierarchical semantic parsing ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, pages 268–271, Hong Kong. 24

[Huang et Hon, 2001] Xuedong HUANG et Hsiao-Wuen HON (2001). *Spoken Language Processing : A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA. Foreword By-Raj Reddy. 19

[Issar et Ward, 1994] Sunil ISSAR et Wayne WARD (1994). « Flexible parsing : Cmu's approach to spoken language understanding ». *In Proceedings of the ARPA Workshop on Spoken Language Technology*, pages 53–58. Morgan Kaufmann. 22

[Jackendoff, 1990] Ray JACKENDOFF (1990). *Semantic Structures*. MIT Press, Cambridge, MA. 79

[Jamoussi *et al.*, 2004] Salma JAMOUSSI, Kamel SMAÏLI, Dominique FOHR et Jean-Paul HATON. (2004). « A complete understanding speech system based on semantic concepts ». *In Proceedings of the* 4^{th} *International Conference on Language Resources and Evaluation*, Lisbonne, Portugal. 62

[Janiszek *et al.*, 2000] David JANISZEK, Frédéric BÉCHET et Renato DE MORI (2000). « Integrating MAP and linear transformation for language model adaptation ». *In Proceedings of the International Conference on Spoken Language Processing*. 89, 90, 91

[Joachims, 1998] Thorsten JOACHIMS (1998). « Text categorization with support vector machines : learning with many relevant features ». *In* Claire NÉDELLEC et Céline ROUVEIROL, éditeurs : *Proceedings of ECML-98, 10th European Conference on Machine Learning*, numéro 1398, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE. 43

[Karahan *et al.*, 2003] Mercan KARAHAN, Dilek HAKKANI-TÜR, Giuseppe RICCARDI et Gokhan TUR (2003). « Combining classifiers for spoken language understanding ». *In Proceedings IEEE Workshop on Automatic Speech Recognition and Understanding*, Virgin Island, USA. 30

[Katz, 1987] Slava M. KATZ (1987). « Estimation of probabilities from sparse data for the language model component of a speech recognizer ». *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401. 14

[Klatt, 1977] Dennis H. KLATT (1977). « Review of the arpa speech understanding project ». *In JASA*, volume 62, pages 1345–1366. 20

[Kuhn et De Mori, 1995] Roland KUHN et Renato DE MORI (1995). « The application of semantic classification trees to natural language understanding ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):449–460. 23, 30, 39

[Kuhn et De Mori, 1998] Roland KUHN et Renato DE MORI (1998). « Sentence interpretation ». *In* Renato DE MORI, éditeur : *Spoken Dialogues with Computers*, chapitre 14, pages 485–522. Academic Press. 20

[Langlois *et al.*, 2003] David LANGLOIS, Armelle BRUN, Kamel SMAÏLI et Jean-Paul HATON (2003). « Événements impossibles en modélisation stochastique du langage ». *Traitement Automatique des Langues*, 44(1):33–61. 95

[Levesque et Brachman, 1985] Hector J. LEVESQUE et Ronald J. BRACHMAN (1985). *A fundamental trade-off in knowledge representation and reasoning*. In Levesque H. J. and Brachman R.J. Eds. Readings Morgan Kaufmann publ, Los Altos, CA., USA. 79

[Mariani, 1990] Joseph MARIANI (1990). « Reconnaissance de la parole : progrès et tendances ». *Traitement du signal*, 7(4):239–266. 11

[Miller *et al.*, 1994] Scott MILLER, Richard SCHWARTZ, Robert BOBROW et Robert INGRIA (1994). « Statistical language processing using hidden understanding models ». *In Spoken Language Systems Technology Workshop*, pages 48–52, Plainsboro, New Jersey, USA. Morgan Kaufmann, Los Altos, CA. 23

[Minker et Bennacef, 1996] Wolfgang MINKER et Samir BENNACEF (1996). « Compréhension et évaluation dans le domaine atis ». *In Proceedings of 21èmes Journées d'Études sur la Parole*. 17, 20

[Mohri et Nederhof, 2001] Mehryar MOHRI et Mark-Jan NEDERHOF (2001). « Regular approximation of context-free grammars through transformation ». *In* Jean-Claude JUNQUA et Gertjan NOORD, van, éditeurs : *Robustness in Language and Speech Technology*, pages 153–163. Kluwer Academic Publishers, Dordrecht. 33

[Mohri *et al.*, 1997] Mehryar MOHRI, Fernando PEREIRA et Michael RILEY (1997). « AT&T FSM Library - Finite State Machine Library ». Rapport technique, AT&T. 35, 68

[Mohri *et al.*, 2002] Mehryar MOHRI, Fernando PEREIRA et Michael RILEY (2002). « Weighted finite-state transducers in speech recognition ». *Computer, Speech and Language*, 16(1):69–88. 35

[Nasr *et al.*, 1999] Alexis NASR, Yannick ESTÈVE, Frédéric BÉCHET, Thierry SPRIET et Renato DE MORI (1999). « A language model combining n-grams and stochastic finite state automata ». *In Proceedings of European Conference on Speech Communication and Technology*, volume 5, pages 2175–2178, Budapest, Hongrie. 27

[Ney *et al.*, 1992] Hermann NEY, Dieter MERGEL, Andreas NOLL et Annedore PAESELER (1992). « Data driven search organization for continuous speech recognition ». *IEEE Transactions on Signal Processing*, 40(2):272–281. 15

[Pieraccini et Levin, 1993] Roberto PIERACCINI et Esther LEVIN (1993). « A learning approach to natural language understanding ». *In NATO-ASI, New Advances & Trends in Speech Recognition and Coding*, volume 1, pages 261–279, Bubion (Granada), Spain. 17

[Pieraccini et Levin, 1995] Roberto PIERACCINI et Esther LEVIN (1995). « A spontaneous speech understanding system for database query applications ». *In ESCA Workshop on Spoken Dialogue Systems*, pages 85–88, Vigso, Danemark. 22, 62

[Polya, 1954] George POLYA (1954). « Induction and analogy in mathematics ». *In Mathematics and plausible reasoning*. Princetown University Press. 90

[Pradhan *et al.*, 2004] Sameer PRADHAN, Wayne WARD, Kadri HACIOGLU, James H. MARTIN et Dan JURAFSKY (2004). « Shallow semantic parsing using support vector machines ». *In Proceedings HLT-NAACL Conference*, pages 233–240, Boston, Massachusetts, USA. 64

[Pullum et Gazdar, 1982] Geoffrey K. PULLUM et Gerald GAZDAR (1982). « Natural languages and context-free grammars ». *Linguistics and Philosophy*, 4:471–504. 32

[Rahim *et al.*, 2001] Mazin RAHIM, Giuseppe RICCARDI, Lawrence SAUL, Jerry H. WRIGHT, Bruce BUNTSCHUH et Allen L. GORIN (2001). « Robust numeric recognition in spoken language dialogue ». *Speech Communication*, 34:195–212. 80

[Rayner *et al.*, 1994] Manny RAYNER, David CARTER, Vassilios DIGALAKIS et Patti PRICE (1994). « Combining knowledge sources to reorder n-best speech hypothesis lists ». *In Proceedings of the DARPA Human Language Technology Workshop*, pages 212–217, Princeton, New Jersey, USA. 93

[Riccardi et Gorin, 1998] Giuseppe RICCARDI et Allen L. GORIN (1998). « Stochastic language models for speech recognition and understanding ». *In Proceedings of the International Conference on Spoken Language Processing*, pages 2,087–2,090., Sydney, Australie. 17

[Riccardi *et al.*, 1996] Giuseppe RICCARDI, Roberto PIERACCINI et Enrico BOCCHIERI (1996). « Stochastic automata for language modeling ». *Computer Speech and Language*, 10(4):265–293. 27

[Roark, 2002] Brian ROARK (2002). « Markov parsing : lattice rescoring with a statistical parser ». *In Proceedings of the* 40^{th} *ACL meeting*, Philadelphia, USA. 18

[Sadek et De Mori, 1998] David SADEK et Renato DE MORI (1998). « Dialogue systems ». *In* Renato DE MORI, éditeur : *Spoken Dialogues with Computers*, chapitre 15, pages 523–561. Academic Press. 8

[Sadek *et al.*, 1996] David SADEK, Alexandre FERRIEUX, Alain COZANNET, Philippe BRETIER, Franck PANAGET et Jacques SIMONIN (1996). « Effective human-computer cooperative spoken dialogue : the AGS demonstrator ». *In Proceedings of the International Conference on Spoken Language Processing*, pages 546–549, Philadelphia, Pennsylvanie, USA. 51

[Sarikaya *et al.*, 2005] Ruhi SARIKAYA, Yuqing GAO, Michael PICHENY et Hakan ERDOGAN (2005). « Semantic confidence measurement for spoken dialog systems ». *IEEE Transactions on Speech and Audio Processing*, 13(04):534–545. 88

[Schapire et Singer, 2000] Robert E. SCHAPIRE et Yoram SINGER (2000). « BoosTexter : A boosting-based system for text categorization ». *Machine Learning*, 39:135–168. 30, 98

[Schwartz *et al.*, 1992] Richard SCHWARTZ, Steve AUSTIN, Francis KUBALA, John MAKHOUL, Long NGUYEN, Paul PLACEWAY et George ZAVALIAGKOS (1992). « New uses for the n-best sentence hypotheses within the byblos speech recognition system ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, volume 1, pages 1–4, San Francisco, California, USA. 22

[Schwartz *et al.*, 1997] Richard SCHWARTZ, Scott MILLER, David STALLARD et John MAKHOUL (1997). « Hidden understanding models for statistical sentence understanding ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, volume 2, page 1479, Washington, DC, USA. IEEE Computer Society. 17

[Searle, 1982] Shayle R. SEARLE (1982). *Matrix Algebra Useful for Statistics*. John Wiley & Sons, New York, NY, USA. 89

[Seneff, 1989] Stéphanie SENEFF (1989). « Tina : A probabilistic syntactic parser for speech understanding systems ». *In Proc. of the Speech and Natural Language Workshop*, pages 168–178, Philadelphia, PA, USA. 22

[Seneff, 1992] Stephanie SENEFF (1992). « TINA : A natural language system for spoken language applications ». *Computational Linguistics*, 18(1):61–86. 22

[Sorin et De Mori, 1998] Christel SORIN et Renato DE MORI (1998). *Spoken Dialogue with Computers*, pages 563–582. Academic Press. 8

[Tur *et al.*, 2004] Gokhan TUR, Dilek HAKKANI-TÜR et Giuseppe RICCARDI (2004). « Extending boosting for call classification using word confusion networks ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, volume 1, pages 437–440. 30

[Uhrik et Ward, 1997] Carl UHRIK et Wayne WARD (1997). « Confidence metrics based on *N*-gram language model backoff behaviors ». *In Proceedings of European Conference on Speech Communication and Technology*, pages 2771–2774, Rhodes, Greece. 93

[Vapnik, 1982] Vladimir N. VAPNIK (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag. 41

[Vapnik, 1995] Vladimir N. VAPNIK (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc. 41

[Wang, 2003] Kuansan WANG (2003). « Semantics synchronous understanding for robust spoken langage applications ». *In Proceedings IEEE Workshop on Automatic Speech Recognition and Understanding*. 26, 27

[Ward, 1991] Wayne WARD (1991). « Understanding spontaneous speech : the phoenix system ». *In Proceedings of the International Conference on Acoustic Speech and Signal Processing*, pages 365–367, Toronto, Canada. 62

[Yvon, 1996] François YVON (1996). *Prononcer par analogie : motivation, formalisation et évaluation*. Thèse de doctorat, École Nationale Supérieure des Télécommunications. 90

Références bibliographiques personnelles

Revues internationales

CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET, NATHALIE CAMELIN, RENATO DE MORI, GÉRALDINE DAMNATI (2006). « Sequential decision strategies for machine interpretation of speech ». *IEEE Transactions on Speech and Audio Processing*, à paraître.

CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET, RENATO DE MORI ET GÉRALDINE DAMNATI (2006). « On the use of finite state transducers for semantic interpretation ». *Speech Communication*, 48 :288-304.

YANNICK ESTÈVE, CHRISTIAN RAYMOND, RENATO DE MORI ET DAVID JANISZEK (2003). « On the use of linguistic consistency in systems for human-computer dialogs ». *IEEE Transactions on Speech and Audio Processing*, 11 :746-756.

Conférences internationales

CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET, RENATO DE MORI ET GÉRALDINE DAMNATI (2005). « On the use of finite state transducers for semantic interpretation ». *Proceedings of the International Conference on Acoustic Speech and Signal Processing, ICASSP'05*, Philadelphie, USA.

CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET, RENATO DE MORI ET GÉRALDINE DAMNATI (2004). « On the use of confidence for statistical decision in dialogue strategies ». *In 5th SIGdial Workshop on Discourse and Dialogue*, Boston, USA.

CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET, RENATO DE MORI, GÉRALDINE DAMNATI ET YANNICK ESTÈVE (2004). « Automatic learning of interpretation strategies for spoken dialogue systems ». *Proceedings of the International Conference on Acoustic*

Speech and Signal Processing, ICASSP'04, Montréal, Canada.

CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET, RENATO DE MORI ET GÉRALDINE DAMNATI (2004). « Stratégie de décodage conceptuel pour les applications de dialogue oral ». *XXV^{ième} Journée d'Études sur la parole, JEP'04*, Fès, Maroc.

CHRISTIAN RAYMOND, YANNICK ESTÈVE, FRÉDÉRIC BÉCHET, RENATO DE MORI ET GÉRALDINE DAMNATI (2003). « Belief confirmation in spoken dialogue systems using confidence measures ». *Proceedings IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU'03*, St. Thomas, US-Virgin Islands.

YANNICK ESTÈVE, CHRISTIAN RAYMOND, FRÉDÉRIC BÉCHET ET RENATO DE MORI (2003). « Conceptual decoding for spoken dialog systems ». *Proceedings of European Conference on Speech Communication and Technology, Eurospeech'03*, Genève, Suisse.

RENATO DE MORI, YANNICK ESTÈVE ET CHRISTIAN RAYMOND (2002). « On the use of structures in language models for dialogue ». *In Proceedings of the International Conference on Spoken Language Processing, ICSLP'02*, Denver, Colorado, USA.

YANNICK ESTÈVE, CHRISTIAN RAYMOND ET RENATO DE MORI (2002). « On the use of structure in language models for dialogue : specific solutions for specific problems ». *In ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, Kloster Irsee, Allemagne.

Conférences nationales

CHRISTIAN RAYMOND (2003). « Mesures de confiance pour la reconnaissance de la parole dans des applications de dialogue homme-machine ». *Majecstic*, Marseille, France.

CHRISTIAN RAYMOND, PATRICE BELLOT ET MARC EL-BÈZE (2002). « Enrichissement de requêtes pour la recherche documentaire selon une classification non-supervisée ». 13ème *Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA'2002)*, pages 625 à 632, Angers, France.