Description de l'interaction vocale

Sommaire

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

3.1 Le service 3000

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

3.2 Systèmes de dialogue oral

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

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

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

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

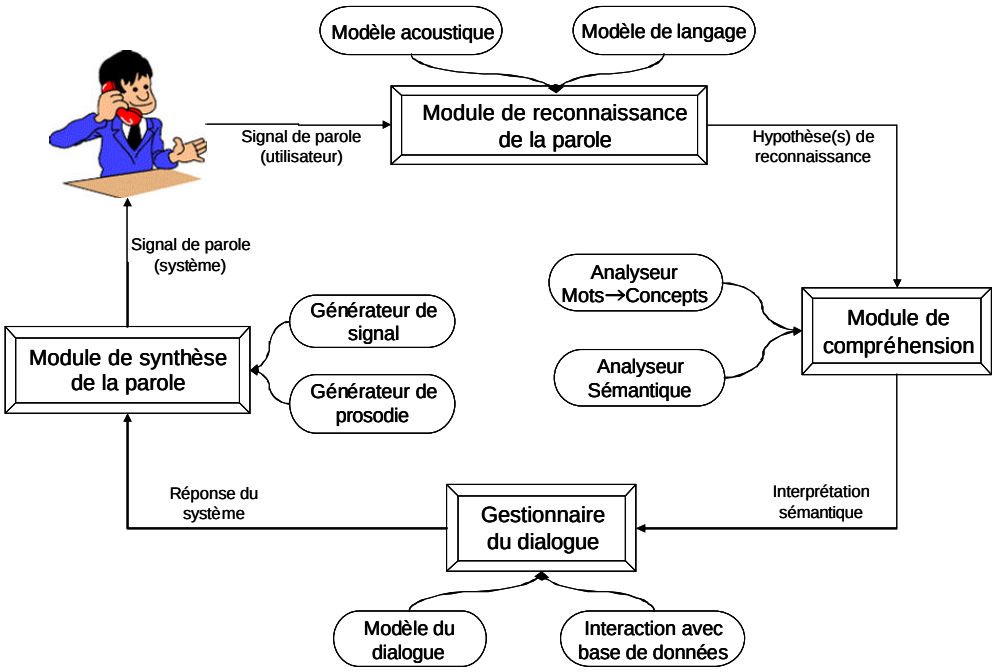

FIGURE *3.1* – Diagramme des modules d'un système de dialogue

3.3 Compréhension de la parole

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

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

3.3.1 Analyseur Mots $\longrightarrow$ Concepts

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

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

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

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

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

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

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

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

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

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

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

3.3.2 Analyseur sémantique

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

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

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

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

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

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

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

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

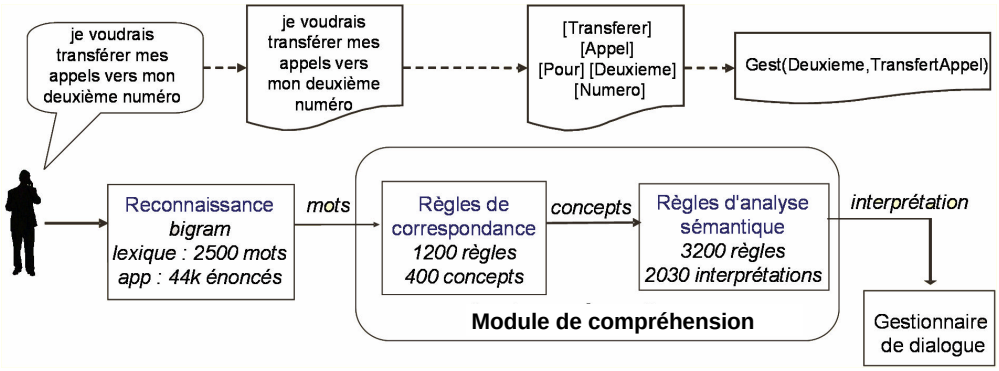

FIGURE 3.2 – Le service 3000

3.3.3 Projet LUNA

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

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

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

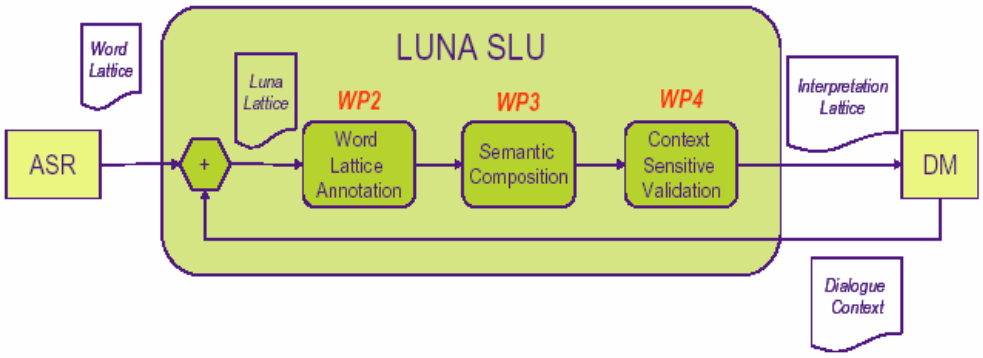

FIGURE 3.3 – Diagramme des modules du système de dialogue LUNA

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

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

3.4 Evaluation au niveau interprétation

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

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

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

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

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

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

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

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

3.5 Gestionnaire de dialogue

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

3.6 Conclusions

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