Université Pierre et Marie Curie **UFR d'informatique**

Laboratoire d'informatique de Paris 6 **École doctorale EDITE**

Raisonnement sur les incertitudes et apprentissage pour les systèmes de dialogue conventionnels

THÈSE

présentée et soutenue publiquement le 18 mai 2010

pour l'obtention du

Doctorat de l'université Paris 6

(spécialité informatique)

par

Romain Laroche

Composition du jury

Rapporteurs :	Frédérick GARCIA (Directeur de recherche) Fabrice LEFEVRE (Maître de conférence HDR)
Examinateurs :	Patrick GALLINARI (Professeur) Ioannis KANELLOS (Professeur) Olivier PIETQUIN (Directeur de recherche)
Directrice :	Bernadette BOUCHON-MEUNIER (Directrice de recherche)
Encadrant :	Philippe BRETIER (Docteur)

Résumé

De plus en plus de secteurs industriels ont besoin d'applications de dialogue. Les services de relation client sont nombreux : vente à distance, service après vente, suivi des commandes, service de renseignements...Les hotlines de ces services vocaux sont très coûteuses et souvent saturées. Au final, le service rendu est à la fois mauvais et couteux.

Face à ce problème, l'industrie et la recherche peinent à converger. D'un côte, les systèmes de dialogue industriels sont conçus à l'aide d'automates décisionnels décrivant la logique de dialogue. Ces automates sont réputés simplistes, difficiles à concevoir et sous-optimaux. De l'autre côté, les scientifiques se focalisent sur des techniques avancées que seuls des experts du domaine sont capables d'implanter et qui restent difficilement contrôlables.

En partant de l'architecture globale du système, cette thèse s'est efforcée de réconcilier la recherche et l'industrie en inscrivant les avancées technologiques dans le processus industriel. Ce travail a mené à la définition d'un nouveau formalisme de raisonnement sur les incertitudes, à la définition d'un nouveau processus de décision non Markovien et aux implantations et optimisations d'algorithmes plug-and-play dédiés au problème posé.

Les fonctionnalités avancées développées dans la thèse permettent d'améliorer la robustesse, de garantir l'optimalité des choix de conceptions et d'obtenir un retour d'usage directement interprétable par les chefs de projet. Ces résultats ont motivé l'implémentation de la première mondiale d'application de dialogue commerciale embarquant de l'apprentissage par renforcement en ligne.

Mots-clés: systèmes de dialogue, apprentissage par renforcement, raisonnement sur les incertitudes.

Abstract

More and more industries need dialogue applications. In customer relationship management, they range from home shopping to after-sales service, order tracking, directory enquiries. . .The hotlines of these vocal services are very costly and often overcrowded. In the end, the proposed service is expensive and has a low quality level.

Confronted with this problem, industry and research struggle to converge. On the one hand, industrial dialogue systems are designed with decision-making automata describing the dialogue logics. These automata are reputed simplistic, hard to design and suboptimal. On the other hand, scientists focus on advanced techniques that only experts are able to implement and that remain sorely monitorable.

Grounded in the system global architecture, this PhD thesis endeavoured to reconcile research with industry by enclosing the scientific advances into the industrial process. This work led to the definition of a new model for reasoning on uncertainties, to the definition of a new non-Markovian decision process and to the implementations and optimisations of plug-and-play algorithms dedicated to the problem.

The advanced functionalities developed in this thesis enable to improve robustness, to guarantee the optimality of design choices and to have the project managers receive an easy-to-comprehend usage feedback. These results have motivated the implementation of the world première of commercial dialogue application incorporating online reinforcement learning.

Keywords: dialogue systems, reinforcement learning, uncertainty management

Table des matières

Introduction

1 Contexte

Le travail de thèse porte sur le domaine des systèmes de dialogue. Ceux-ci incluent notamment les systèmes vocaux interactifs, mais aussi les chatbots ou les interactions que certains jeux vidéos proposent entre le joueur et certains personnages. De manière plus globale, on peut généraliser ces systèmes à des interactions homme machine se faisant à travers une interface utilisant un sous-ensemble du langage naturel.

France Télécom R&D s'intéresse depuis longtemps aux systèmes de dialogues, notamment pour tout ce qui ressort du domaine des services vocaux. Aujourd'hui, la technologie Disserto est fortement axée sur le dialogue en langue naturelle. Elle a déjà servi à développer des services de qualification d'appel (Générali, Crédit Agricole Centre Ouest, la Banque Postale), l'annuaire téléphonique interne, les services téléphoniques automatisés de France Télécom (le 3000, le 3900, le 1013), ...

La solution de dialogue Disserto est une chaîne d'outils complète destinée à la conception, au développement et à la production de services vocaux interactifs. L'utilisateur peut interagir avec l'application soit en pressant les touches du téléphone (DTMF), soit en prononçant des commandes vocales. Ces commandes vocales peuvent être soit en mots isolés, en mots connectés, ou alors en parole continue associée à un analyseur sémantique. Cette dernière modalité permet des applications dites en langage naturel. L'application rétroagit vers l'utilisateur en émettant des messages sonores pré-enregistrés ou en synthèse de la parole. La figure 1 présente l'architecture générale de la solution Disserto.

La chaîne Disserto est composée des applications suivantes pour le cycle de conception-développement :

- Dialog Design Studio (DDS), application graphique de conception utilisant le paradigme de l'automate à états finis, où chaque état est appelé phase ;
- Dialog Analyser Studio (DAS), application permettant la mise au point d'un ensemble de règles sémantiques destinées à l'Analyseur sémantique, pour les applications utilisant le langage naturel ;
- Dialog Code Generator (DCG) est un générateur de code qui prend en entrée une

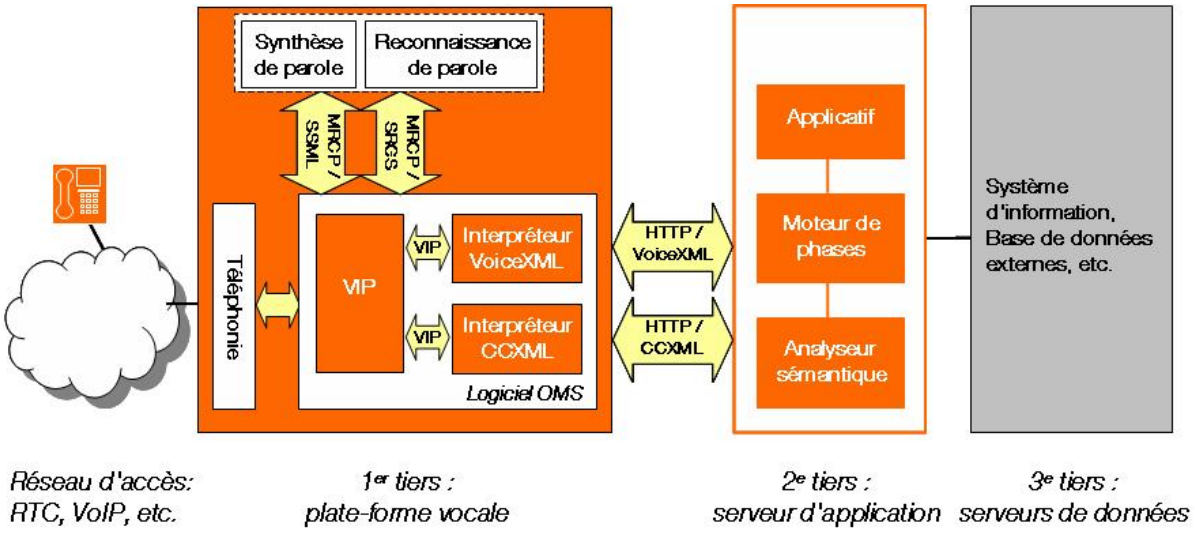


FIGURE 1 – Architecture générale de la solution Disserto

application décrite dans DDS, ainsi qu'une base d'exceptions de code java, et génére en sortie une description de service XML qui décrit la cinématique des phases, et une collection de servlets java qui implémentent la logique métier de l'application vocale.

De plus, Disserto est composée des applications suivantes pour le cycle de production :

- un Analyseur Sémantique (AS), qui prend en entrée une phrase ou expression prononcée par un utilisateur, telle que le moteur de reconnaissance de parole continue l'a capturée, et produit en sortie une ou plusieurs étiquettes sémantiques qui vont permettre d'orienter la suite du dialogue ;
- un Moteur de Phases (MP), qui utilise les fichiers produits par DCG pour faire tourner l'application vocale et produire à la volée les pages VoiceXML décrivant l'interaction vocale.

Enfin, Disserto s'appuie sur les composants suivants :

- un serveur d'application JEE qui permet de faire tourner le Moteur de Phases, l'Analyseur Sémantique, et l'application vocale ; à France Télécom, nous utilisons JOnAS du consortium Objectweb ;
- un media serveur vocal conçu à base de technologies maîtrisées et éditées par France Télécom/ROSI/DPS, avec les sous-composants majeurs suivants : VIP, le coeur du media serveur ; CVMX, l'interpréteur VoiceXML ; Baratinoo, la synthèse de la parole en unités longues éditée par France Télécom R&D ; Philsoft 3.2 ou Telispeech, les moteurs de reconnaissance de la parole de Telisma.

Mais Disserto peut bien sûr fonctionner avec d'autres composants JEE et media serveur vocal.

2 Enjeux

De plus en plus de secteurs industriels ont besoin d'applications de dialogue. Les services de relation client qui peuvent être exécutés à distance sont nombreux : commande à distance, service après vente, suivi des commandes, service de renseignements...Les hotlines de ces services vocaux sont très coûteuses et souvent saturées. Au final, le coût est partiellement reporté sur le client qui est à la fois mécontent par le mauvais service et par son prix. Et pourtant, la plupart des appels concernent des problèmes dont la résolution pourrait facilement être automatisée. Ce constat explique l'expansion rapide des services vocaux ces dernières années, comme en atteste la citation suivante du Datamonitor d'août 2006 :

> Spending on speech recognition solutions and services eclipsed the $1 billion mark in 2005 [...] By 2009, spending is expected to more than double to reach $2.7 billion. [1]

Malgré ces chiffres encourageants, il apparaît que les avancées scientifiques dans le domaine du dialogue ont échoué leur passage industriel. En effet, dès qu'une nouveauté scientifique voit le jour, le temps de développement augmente et de manière encore plus problématique, le développeur doit manipuler des concepts de plus en plus complexes. Un autre obstacle rencontré est le peu de réutilisabilité des implémentations entre une application et une autre, même si celles-ci traitent d'un sujet proche.

Du point de vue théorique, les travaux sur l'insertion de fonctionnalité avancées dans les interfaces de développement est un domaine qui intéresse de plus en plus la communauté informatique, notamment pour l'optimisation de site web à travers les récents travaux sur les tests multivariables (Kohavi et al., 2009) (A/B testing généralisé), comme l'apparition récente de Google Analytics en atteste [2]. Nous considérons les travaux de cette thèse facilement transposable dans le contexte du web, notamment en ce qui concerne les théories et les algorithmes qui y sont développées. Ceci élargit d'autant plus les enjeux de cette thèse.

3 Objectifs

Malgré l'expansion décrite plus haut, l'expérience de l'utilisateur est restée frustrante, coûteuse en terme de temps, et inefficace. Les améliorations sont limitées par le fait que les technologies actuelles de système de dialogue sont intrinsèquement inflexibles et

1. Les dépenses sur les solutions et services de reconnaissance vocale ont dépassé le seuil du milliard de dollars en 2005. D'ici 2009, ces dépenses devraient plus que doubler et atteindre 2.7 milliards de dollars.
2. http ://www.google.com/intl/fr_ALL/analytics/

assujetties aux erreurs de reconnaissance vocale, qui sont inévitables dans ce type de systèmes. Cette thèse se donne l'ambition d'effectuer un bond qualitatif en matière de robustesse, flexibilité, efficience et naturel des systèmes de dialogue, en dotant le système de capacités de raisonnement sur l'incertitude et en lui permettant d'apprendre à mieux interagir sur la base de son expérience.

Le premier apport de cette thèse concerne une nouvelle approche pour le traitement des incertitudes, particulièrement adaptée au dialogue et aux contraintes liées au développement de ce type d'applications. La théorie défendue a pour objectif de pouvoir mesurer les probabilités des paradigmes qui lui importent, de façon mathématiquement exacte sur la base des observations du système.

La seconde contribution traite de l'apprentissage dans les systèmes de dialogue dans un objectif d'assistance à la conception des systèmes de dialogue. Les solutions d'apprentissage dans les systèmes de dialogue actuels cherchent à améliorer directement le système. Dans cette thèse, nous nous efforcerons d'optimiser automatiquement la conception du système à l'aide de l'expérience en ligne, et, en conséquence le système lui-même. Nous atteignons ainsi la convergence entre la conception, l'exécution et l'évaluation du système. Pour ce faire, nous définissons un nouveau cadre formel pour les systèmes de décision à base d'automate et nous y implantons un algorithme d'apprentissage par renforcement original, adapté à ce modèle de décision.

Ces deux nouveautés convergent et s'alimentent mutuellement dans une architecture système de dialogue unificatrice. De même le processus de développement d'un système de dialogue a particulièrement été pris en compte et les nouveautés impacteront très peu le niveau d'expertise requis pour les développeurs, tout en fournissant de manière transparente ou non des nouvelles fonctionnalités allant de la robustesse à l'optimisation en ligne des capacités de dialogue du système.

4 Organisation du document

Le chapitre 1 présente les architectures actuelles des systèmes de dialogue. Il en montre les failles, dès que l'on souhaite y insérer de l'apprentissage et/ou de la gestion des incertitudes. Nous proposons ensuite de ne plus baser cette architecture sur les concepts de tours de dialogue et d'actes de dialogue, mais au contraire d'avoir une approche système à une notion d'input/output pour définir les notions de tour et d'actes.

Le chapitre 2 présente un nouveau cadre formel appelé le Logical Framework for Probabilitic Reasoning (LFPR) permettant le raisonnement sur les incertitudes dans le contexte d'un système de dialogue conventionnel (c'est-à-dire implanté manuellement, par opposition aux systèmes de dialogue générés automatiquement). Dans un premier temps, nous

montrons que les règles manuelles ne permettent pas de rendre compte des dépendances complètes nécessaires aux méthodes probabilistes, alors que les approches de la famille de la théorie de l'Évidence n'ont pas cette contrainte. Dans un second temps, nous introduisons le LFPR qui est une extension de la théorie de l'Évidence. Enfin, nous faisons un tour de la littérature pour montrer les forces du LFPR et justifier son utilisation dans les systèmes de dialogue.

Le chapitre 3 s'intéresse plus particulièrement aux outils de conception et de monitoring pour les développeurs d'applications de dialogue. En nous reposant sur la "synergistic convergence"[3] entre la recherche et l'industrie et la "VUI-completeness"[4] indispensable à la méthodologie industrielle, nous présentons une nouvelle approche intégrée des outils de conception et de monitoring. Nous insistons ensuite sur les fait que ces outils servent non seulement à faciliter le travail de conception et de maintenance des développeurs, mais également à introduire des capacités d'apprentissage dans les systèmes de dialogue conventionnels.

Le chapitre 4 suit le raisonnement logique de la thèse en établissant la problématique de l'apprentissage dans les systèmes de dialogue conventionnels. Nous y affirmons que la première nécessité est de lever l'hypothèse markovienne et ainsi de se priver des algorithmes d'apprentissage classique que l'on applique aux Markov Decision Process (MDP). De manière à remplacer ce modèle, nous définissons un nouveau processus de décision non markovien : le Module-Variable Decision Process (MVDP). Puis, nous constatons que la plupart des algorithmes de la littérature ne s'appliquent plus, parce qu'ils ont recours à du bootstrapping, qui lui même fait l'hypothèse markovienne. Seul l'algorithme Monte Carlo ne fait pas l'hypothèse markovienne. La suite du chapitre est consacrée à la définition et la comparaison théorique de deux algorithmes. Le premier algorithme est une amélioration de l'algorithme Monte Carlo : le Compliance-Based Reinforcement Learning (CBRL). Le second algorithme apprend un module "critique" lui permettant d'utiliser des méthodes de bootstrapping telles que le Q-learning ; il est nommé le Module-Variable Temporal Difference Learning.

Le chapitre 5 montre que l'hypothèse non markovienne permet de réduire la complexité du système et d'accélérer considérablement les temps de convergence à l'aide d'un exercice se rapprochant de la problématique du dialogue. Ensuite, nous continuons la comparaison des deux algorithmes CBRL et MVTDL, pour montrer que les deux algorithmes atteignent des performances comparables en terme de convergence. Mais la nécessité pour le MVTDL de concevoir du module "critique" le rend difficile à utiliser pour une réelle application de dialogue. C'est la raison pour laquelle la suite des tests, notamment en ce qui concerne

3. Convergence synergique.
4. Complétude de l'interface vocale utilisateur (Voice User Interface).

la méthode de compromis entre l'exploration et l'exploitation, ne sera réalisée qu'avec le CBRL.

Le chapitre 6 présente l'expérimentation laboratoire sur une application de dialogue destinée à aider les personnes à installer leur Livebox. Les objectifs de cette expérimentation sont de montrer que le CBRL peut converger très vite dans un système de dialogue réel avec des testeurs humains et de montrer qu'une application conventionnelle peut être énormément améliorée à l'aide d'une optimisation, même lorsque toutes les alternatives semblent raisonnables. Malgré le faible volume de testeurs (40 pour un total de 160 dialogue), nous avons observé une forte amélioration des récompenses reçues par le système qui s'est traduit par une réduction de plus d'une erreur de reconnaissance vocale par dialogue.

Pour finir, un dernier chapitre conclut cette thèse en rappelant ses contributions majeures et en posant les futures étapes à franchir pour les systèmes de dialogues conventionnels ayant des capacités d'apprentissage et de gestion de l'incertitude.

Cette thèse contribue au projet européen CLASSiC (n°216594 EC FP7, Call 1, voir section 1.1) et a reçu à ce titre des fonds de la Communauté Européenne.

Chapitre 1

Architecture

1.1 Introduction

Ce chapitre a pour but de réunir dans une même architecture les différentes approches du dialogue, en particulier celles des contributeurs au projet CLASSiC [5] (Computational Learning in Adaptive Systems for Spoken Conversation). CLASSiC est un projet européen (n°216594 EC FP7, Call 1) qui a débuté le 1er mars 2008 pour une durée de 36 mois. Nous retrouvons dans ce projet exactement les mêmes thématiques que celles développées dans cette thèse : incertitude, apprentissage, traitement statistique de bout en bout du système de dialogue. J'ai activement participé à la construction de ce projet et continué à y contribuer depuis son lancement. Le fait qu'un projet de cette envergure consacre l'intégralité de ses efforts sur ces thématiques démontre à lui seul l'intérêt de la problématique de la thèse.

Le but général du projet CLASSiC est de faciliter le déploiement rapide de systèmes de dialogue robustes et précis qui peuvent apprendre de leur expérience. L'approche est basée sur des méthodes d'apprentissage statistique avec un traitement des incertitudes unifié à travers le système dans son intégralité (Automatic Speech Recognition, Spoken Language Understanding, Dialogue Management, Natural Language generation et Speech Synthesis). Ceci aura pour résultat un cadre d'exécution modulaire avec une représentation explicite de l'incertitude partagée entre ses différentes sources (erreurs de compréhension, ambiguïtés, etc...) pour les restituer aux entités capables de les utiliser (stratégie de dialogue, génération de la réponse, etc...).

L'effort a porté sur les axes suivants, par ordre décroissant d'importance :

1. L'architecture est construite dans un objectif industriel, c'est-à-dire que les contraintes opérationnelles sont prises en compte.

5. http ://www.classic-project.org/

2. L'architecture proposée permet l'implémentation des axes de recherche du projet CLASSiC (gestion de l'incertitude et apprentissage en ligne ou hors ligne, ce qui correspond également au sujet de la thèse).
3. Les modules existant dans le consortium CLASSiC ne devront pas être réimplantés.
4. L'architecture ne requiert pas d'extension de la part des systèmes existants, toujours à l'échelle du consortium CLASSiC.

Les points (1) et (2), qui sont ceux qui nous intéressent dans le cadre de la thèse, sont garantis.

La section 1.2 présente la chaîne de décision du dialogue et ses modules. Cela fournit au lecteur les connaissances nécessaires au sujet des systèmes de dialogue et les concepts qui seront utilisés pour justifier les choix qui ont été réalisés.
La section 1.3 introduit la nécessité de systèmes de contrôle pour les systèmes de dialogue. Ils sont au nombre de trois : le Log Manager (LM) qui contrôle les enregistrements du système de dialogue, de manière à pouvoir rejouer les dialogues ; le Context Manager (CM) qui contrôle les connaissances du système de dialogue, principalement pour la gestion de l'incertitude ; et enfin le Scheduler qui contrôle la dépense des ressources par le système de dialogue.

1.2 La chaîne de dialogue

La chaîne de dialogue classique (voir la figure 1.1) et présente dans tous les systèmes de dialogue contient cinq modules : l'Automatic Speech Recognition (ASR), le Spoken Language Understanding (SLU), le Dialogue Management (DM), le Natural Language Generation (NLG) et le Text To Speech (TTS) [6]. Dans les prochaines figures, nous y ajoutons un sixième module, que l'on considère généralement comme une entité externe au système de dialogue : le Back-Office (BO). De la même manière, l'utilisateur fait partie intégrante de ces architectures.

1.2.1 Modules

Automatic Speech Recognition

Le module d'Automatic Speech Recognition (ASR) a pour objectif de transformer la parole de l'utilisateur en une ou plusieurs séquences de mots notées selon leur vraisemblance. En addition des mots prononcés par l'utilisateur, l'ASR peut reconnaître des connaissances à partir de marqueurs non-verbaux (Shafran et al., 2003), par exemple :

6. La reconnaissance vocale automatique, la compréhension du langage parlé, la gestion de dialogue, la génération en langue naturelle et la synthèse vocale.

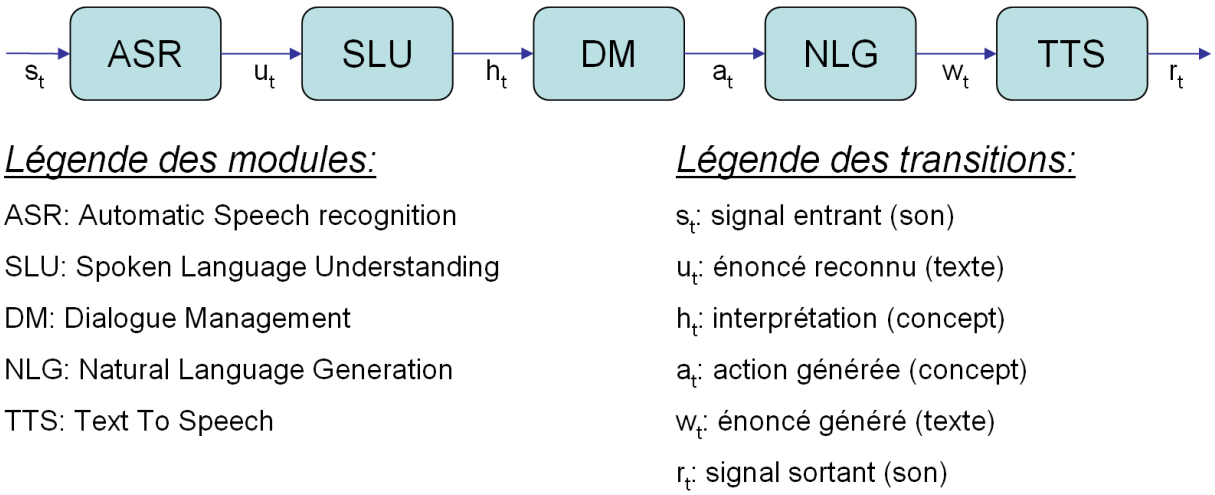


FIGURE 1.1 – Architecture classique d'un système de dialogue.

- Le locuteur dans le cas d'une application de domotique (Furui, 1997),
- Le sexe de l'utilisateur détecté souvent dans les systèmes de dialogue industriels pour pouvoir utiliser des grammaires de reconnaissance vocale adaptées au sexe du locuteur (Tran & Sharma, 2003; Sigmund & Dostál, 2005),
- L'âge de l'utilisateur (Minematsu et al., 2002),
- L'émotion dans la voix (Shafran et al., 2003),
- ...

Spoken Language Understanding

Le module de Spoken Language Understanding (SLU) transforme les séquences de mots en Dialogue Acts (DAs, voir la section 1.2.2), c'est-à-dire une transcription de l'action réalisée par le locuteur en langage symbolique. De plus, le SLU peut inférer des connaissances qui ne sont pas de l'ordre des DAs, par exemple :

- Le niveau de langage de l'utilisateur : "Ma DNS a l'air okay, mais j'arrive toujours pas à pinger google." → langage familier,
- Le niveau technique de l'utilisateur : "Ma DNS a l'air bonne, mais je n'arrive toujours pas à pinger google." → haut niveau,
- La confiance en lui-même de l'utilisateur : "Je crois que ma box internet est bien connectée." → pas très sûr,
- Le sexe de l'utilisateur : "je suis satisfait de mes services" → c'est un homme qui parle,
- Le dialecte (Shafran et al., 2003),
- ...

Dialogue Management

Le module de Dialogue Management (DM) a pour mission de transformer les DAs en entrée fournis par le SLU et DAs en sortie pour la génération du NLG. Ce module définit les stratégies d'interaction du système de dialogue. En plus des DAs en sortie, il peut renseigner le NLG et le TTS avec des informations additionnelles (voir les deux parties suivantes).

Natural Language Generation

Le module de Natural Language Generation (NLG) a pour mission de transformer les DAs fournis par le DM en phrases pour le TTS. Le NLG peut aussi prendre en compte des informations additionnelles que le DM pourrait lui fournir telles que :

- Le niveau de langage : "Ya pas" versus "Il n'y a pas",
- Le niveau technique : "drag and drop" versus "cliquer et sans relâcher le bouton déplacez ...",
- Les marqueurs d'incertitude : "peut-être", "sans doute", "généralement", utilisation du conditionnel, ...
- Les effets d'amorçage (priming effect Pickering & Garrod, 2004; Janarthanam & Lemon, 2009) : répéter, ou non, les expressions utilisées par l'utilisateur,
- ...

Text-To-Speech

Le module de Text To Speech (TTS) transforme les phrases fournies par le NLG en signal audio. Le TTS peut également prendre en compte des informations additionnelles que le DM pourrait lui fournir telles que :

- Les marqueurs d'incertitude (Cadic & Segalen, 2008),
- L'émotion dans la voix (Hirose et al., 2006),
- Les styles de parole : question, affirmatif neutre, affirmatif positif, affirmatif négatif, emphase contrastée (Eide et al., 2004),
- Insertion d'éléments paralinguistiques : "euh", "ah", ...(Eide et al., 2004; Cadic & Segalen, 2008),
- ...

Back Office

Les modules du Back Office (BO) sont les logiciels qui n'interagissent pas directement avec l'utilisateur, c'est-à-dire que les interactions entre le système de dialogue et le BO ne sont pas observables directement par l'utilisateur. En conséquence, cela ne peut pas être

un canal de communication entre le système de dialogue et l'utilisateur. Généralement les modules de BO sont (mais ne sont pas réduits à) des bases de données ou des commandes de test système. Et les commandes de base des BO sont : chercher une information (c'est-à-dire faire le test ou chercher dans la base de donnée) ou modifier la base de donnée. Comme expliqué dans la sous-section 1.2.2, nous classons en deux catégories les accès au BO :

- Actes d'écriture : actes qui changent l'état interne de l'application de BO.
- Actes de lecture : actes qui ne changent pas l'état interne de l'application de BO.

1.2.2 Concepts

Maintenant que l'on a rapidement abordé l'état de l'art des différents modules, cette section introduit des concepts, que l'on manipule fréquemment sans pour autant en avoir une définition communément admise. La première contribution de cette thèse est justement l'établissement de ces définitions et classification, dans le but, notamment, de permettre une gestion de l'incertitude du début jusqu'à la fin de la chaîne de dialogue. Il est tout d'abord nécessaire de faire la distinction entre le traitement correspondant à la consommation et celui correspondant à la production, de manière à donner un sens différent aux probabilités dénotant des incertitudes manipulées lors de chacun de ces traitements. Ensuite, la gestion de l'incertitude crée un besoin en gestion des ressources : le fait que le système doive prendre en compte plusieurs alternatives multiplie par autant la complexité du système et en conséquence son utilisation des ressources. Enfin, une formalisation poussée des actes est indispensable à partir du moment où le BO est ajouté à l'architecture de dialogue et où des chemins incertains peuvent être empruntés. Les actes ne se réduisent pas forcément aux DAs, mais peuvent prendre d'autres formes, par exemple une requête vers le BO. De même, le concept de tour est remis en cause, puisque la notion d'acte s'est diversifiée.

Consommation-production

Du point de vue du système, la chaîne de dialogue peut être séparée en deux parties fonctionnelles : l'ASR et le SLU sont utilisés pour générer un ensemble d'interprétations de ce que l'utilisateur souhaite dire, alors que le NLG et le TTS travaillent sur des simulations de stratégies de dialogue. La frontière entre ces deux parties se situe quelque part au milieu dans le DM, qui prend des interprétations en entrée et renvoie des actions en sortie.

La partie de consommation dans la chaîne de dialogue correspond au processus d'acquisition de connaissance. La pertinence de la connaissance est binaire. Elle est soit juste, soit fausse. Il n'y a pas de position intermédiaire. Le rôle de la gestion de l'incertitude

dans le processus de consommation sera de conserver les chemins des différentes interprétations possibles, qui sont exclusives entre elles (même si elles peuvent être redondantes) puisqu'un énoncé ne peut avoir deux significations (et si l'énoncé est à double sens, on peut de toute façon normaliser en disant que l'unique signification correcte est celle qui agrège ces deux sens). D'un côté plus conceptuel, le rôle de la gestion de l'incertitude est de définir l'état du monde s où le système évolue, ou plus précisément de réduire l'ensemble des états possibles, en définissant une distribution de probabilité $P(s)$ sur ces états. Comme nous faisons l'hypothèse que tous les états possibles sont représentés, la somme de cette distribution sur l'ensemble des états est égale à 1 :

$$\sum_{s} (P(s)) = 1 \tag{1.1}$$

La partie de production dans la chaîne de dialogue correspond au processus de génération de plan, ou planification (Kambhampati, 1997; Boutilier & Brafman, 1997; Louis, 2002; Ghallab et al., 2004). Durant ce processus, le système de dialogue envisage les possibilités d'action qui lui sont offertes et il s'efforce de déterminer le meilleur plan d'action. C'est en quelque sorte une simulation des différentes stratégies qui peuvent être adoptées (stratégies de dialogue pour le DM, stratégies rhétoriques pour le NLG et stratégies d'expression pour le TTS). Idéalement, chaque plan généré π devrait être calculé en confrontant chaque état du monde s possible selon la probabilité $P(s)$ calculée lors de la consommation et son utilité $u_\pi(s)$ (von Neumann & Morgenstern, 1944) à l'aide de la formule suivante :

$$U(\pi) = \sum_{s} P(s) u_\pi(s) \tag{1.2}$$

Malheureusement, dans le cas précis des systèmes de dialogue, il est difficile, voire impossible d'anticiper les utilités d'un plan pour chaque état. En effet, il est difficile de mesurer les différences d'utilité entre deux hors-sujets, de même, pour les différences d'utilité entre deux énoncés qui atteignent le même but immédiat (communiquer un acte de langage) mais en mettant l'utilisateur dans un état émotionnel différent (calme versus stressé par exemple). En conséquence de ces difficultés, nous considérons dans cette thèse une fonction d'utilité $u_\pi(s)$ binaire :

$$u_\pi(s) = \begin{cases} 0, \text{ si le plan d'action } \pi \text{ n'est pas pertinent dans } s \\ 1, \text{ si le plan d'action } \pi \text{ est pertinent dans } s \end{cases} \tag{1.3}$$

Dans un état du monde donné, plusieurs stratégies peuvent être pertinentes, contrairement à la partie de consommation qui n'accepte qu'une seule interprétation pour un

énoncé utilisateur donné. Ce processus de génération de plan permet au système de comparer plusieurs alternatives sur la base des combinaisons des connaissances qui sont distribuées dans les différents modules de la production (DM, NLG et TTS). L'annexe A développe le raisonnement dont les grandes lignes ont été défendues dans cette partie sur des exemples concrets.

L'idée importante à retenir est que le même module de gestion de l'incertitude est classiquement utilisé pour les deux parties : consommation et production. Pourtant, la première de ces deux parties est de nature probabiliste alors que la seconde manipule des utilités. Nous avons réuni ces deux approches sous un même concept : la probabilité de pertinence. La probabilité de pertinence d'une déduction de type consommation est égale à sa probabilité de véracité. Quant à la probabilité de pertinence d'une déduction de type production, elle est égale à la probabilité que ce plan d'action en construction soit pertinent dans le contexte de dialogue, selon la formule d'utilité 1.3.

Ressources

Pour accomplir sa tâche le système de dialogue a des ressources. Ces ressources sont limitées et cette limitation implique des contraintes. La première ressource est le temps : le système de dialogue doit répondre à un énoncé de l'utilisateur dans un laps de temps très court (entre 1 et 3 secondes suivant le contexte). La seconde ressource est la puissance processeur disponible. Cette ressource est conditionnée par la plate-forme et le flux d'appel. Ainsi, on peut imaginer un système de dialogue qui est plus performant la nuit parce que les lignes et donc les plate-formes vocales sont moins surchargées. Ces deux ressources sont profondément connectées. La capacité de stockage de données est une troisième ressource qui peut être ignorée parce qu'elle n'implique pas vraiment de contrainte pour les systèmes de dialogue.

Actes

Qu'est-ce qu'un Acte ? Pour un système de dialogue, le concept d'acte est souvent confondu avec le concept d'acte de dialogue (DA). Dans cette thèse, la notion d'acte est bien plus large. La première chose à remarquer, quand le système décide d'un traitement à exécuter, est qu'il est rarement simple, mais plutôt composé. Définissons un acte comme une action atomique exécutée par un système ou un humain. A un instant donné, le système devra décider d'exécuter une séquence d'actes atomiques. Énumérons les types d'actes et présentons leurs particularités.

Les actes peuvent être classés selon leur portée :

– Un acte peut-être un acte de dialogue (Dialogue Act : DA). C'est un acte qui joue un

rôle dans la communication entre le système de dialogue et l'utilisateur. Le dialogue est rythmé par ces actes. Les actes de dialogue doivent être envoyés en un tempo bien défini. Les DAs sont les porteurs de la contrainte temporelle existant entre le système et l'utilisateur. Selon la complexité et le contenu des actes en entrée, le temps de réponse disponible pour le système de dialogue ne doit pas dépasser un certain seuil.

- Un acte peut être exécuté par le BO (Back-Office Act : BOA). Conceptuellement, le BOA est une tâche exécutée par une entité externe au système de dialogue, mais comme il n'est pas observable par l'utilisateur (au moins pendant le dialogue), ce n'est pas un acte de communication avec lui, et il est distinct d'un DA. Il y a deux types de BOA :
  - Un BOA peut être une routine d'écriture (Write Back-Office Act : WBOA), par exemple une transaction dans une application boursière.
  - Un BOA peut être une routine de lecture (Read Back-Office Act : RBOA), par exemple une recherche de restaurant selon des contraintes exprimées par l'utilisateur.
- Un acte peut être une routine interne (Internal Act : IA), c'est-à-dire un acte qui n'utilise aucune ressource externe au système de dialogue.

Sous un autre aspect, les actes peuvent être classés selon leur impact sur le dialogue :

- Un acte peut avoir une influence directe sur le dialogue. Les DAs ont tous forcément un impact sur le dialogue, mais les WBOAs aussi, puisque ces actions impliquent un changement externe au système. Ces actes sont appelés des actes d'engagement (Committing Act : CA). Le système de dialogue s'engage en exécutant ces actes. Un acte de transaction boursière est un CA WBOA, c'est-à-dire que lors du reste du dialogue, le système ne pourra pas ignorer d'avoir exécuté cette action. La première conséquence est qu'il doit informer l'utilisateur que cette transaction boursière a été réalisée.
- A un niveau moindre d'implication, un acte peut avoir une influence indirecte sur le dialogue par sa consommation de ressources, et notamment du temps, puisque c'est la seule ressource partagée avec l'utilisateur. Nous les appelons des actes consommateurs de temps (Time Consuming Act : TCA). Les RBOAs et IAs peuvent être longs à exécuter. Dans ce cas, ils doivent être contrôlés de manière à planifier les dépenses des ressources du système au mieux.
- Certains actes n'ont finalement aucun impact sur le dialogue. Ce sont des actes instantanés qui n'engagent pas le dialogue. Ils n'ont pas besoin d'être contrôlés (No Impact Act : NIA).

Enumération des Possibilités Il y a plusieurs types d'actes. Nous allons recenser ici toutes les possibilités à partir des catégories que nous venons de définir (portée/impact).

- Les actes vers l'utilisateur : DAs/CAs.
- La plupart des exécutions dans le système de dialogue : IAs/NIAs.
- Certaines exécutions sont lourdes et doivent être contrôlées : IAs/TCAs.
- Les requêtes sur des petites bases de données qui ne demandent pas beaucoup de temps pour fournir la réponse : RBOAs/NIAs.
- Les requêtes sur des grosses bases de données qui demandent un temps non négligeable pour fournir une réponse et qui nécessitent d'être contrôlées en conséquence : RBOAs/TCAs.
- N'importe quelle écriture sur une base de donnée est un acte qui engage le système de dialogue puisqu'il implique une modification du monde extérieur : WBOAs/CAs.

Le choix de l'exécution d'une séquence d'actes doit prendre en compte chaque acte ayant un impact direct ou indirect sur le dialogue.

Typage des Actes Nous typons les actes de la manière suivante :

- Portée : DA, BOA ou IA.
- Niveau d'engagement : CA, TCA ou NIA.
- Niveau de la chaîne de dialogue : est-on dans la partie de consommation ou de production ?
- Durée prévue : durée de l'exécution de l'acte.
- Quantité d'unités processeur nécessaires pour la réalisation de la tâche a priori.
- Identifiant de l'acte : identifiant unique réservé à l'acte.

champ	Acte n°1	Acte n°2	Acte n°3
Portée	DA	BOA	IA
Niveau d'engagement	CA	CA	TCA
Production / consommation	null	Production	Consommation
Durée prévue	null	20ms	200ms
Ressources nécessaires	null	null	2Ginstructions
Identifiants	n°1	n°2	n°3
Incompatibilités	aucune	aucune	aucune
Tâche	Speaker.play(DA.wav)	BO.insert(trans(CB1, CB2))	SLU.resume(id=3)

TABLE 1.1 – Exemples d'actes avec leurs typages

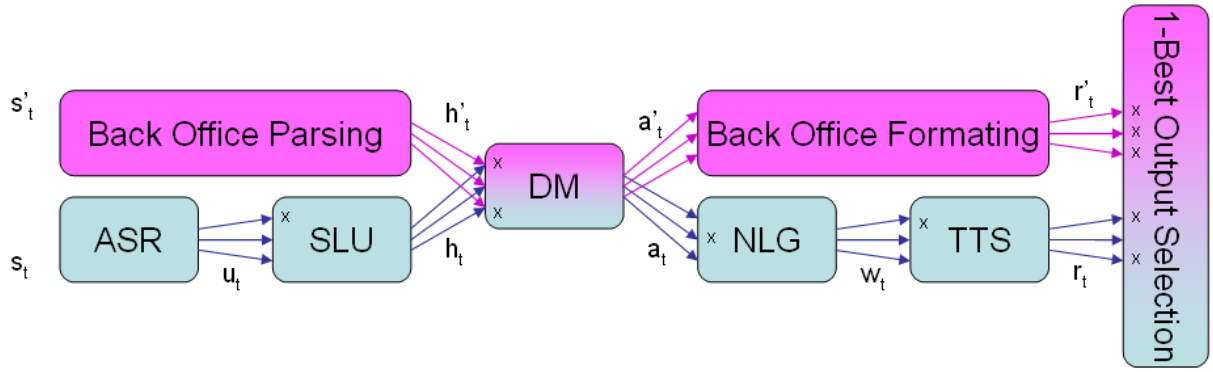


FIGURE 1.2 – La vue architecturale d'un tour du système (ST)

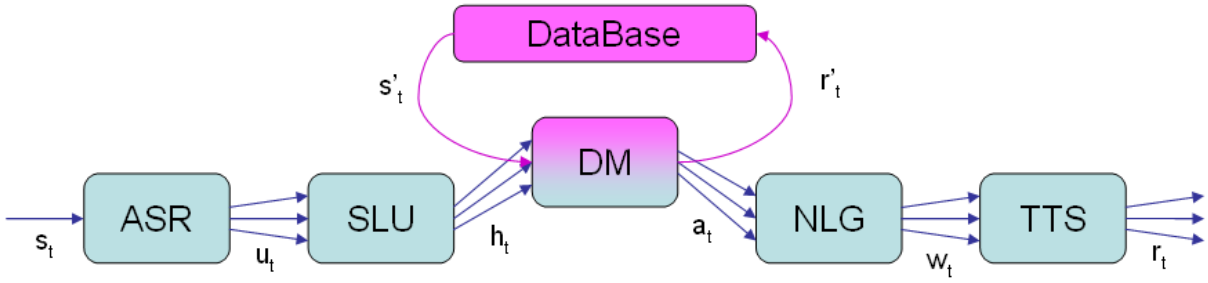


FIGURE 1.3 – La vue architecturale d'un tour de dialogue (DT)

- Incompatibilités avec d'autres actes : identifiant des actes qui ne peuvent pas être exécutés simultanément (dans le cas où le système de dialogue est multiprocesseur), pour des raisons d'accès mémoire par exemple.
- Description de la tâche : "Module.methode(args)".

On peut ajouter à ce typage les utilités qui caractérisent chaque output (ensemble d'actes).

Exemples d'actes L'acte numéro 1 est un DA qui a été généré à travers le NLG et le TTS. L'acte numéro 2 est le WBOA décrit plus tôt pour la transaction dans l'application boursière. L'acte numéro 3 est un IA TCA : le système met en pause l'un de ses processus parce qu'il sait que son traitement risque d'être coûteux en matière de ressources. Ce dernier exemple a été généré par le module SLU qui ne peut traiter toute la liste N-best [7] fournie par l'ASR en un temps raisonnable. Le tableau 1.1 décrit les actes.

7. Une liste N-best comprenant les N meilleurs résultats associés à un score provenant de connaissances statistiques. On utilise ce type de liste notamment pour la reconnaissance vocale et pour l'interprétation sémantique.

Tours

Nous manipulons plusieurs concepts de tours :

- Le tour système (System Turn : ST) est défini par un cycle entre deux choix d'actes externes.
- Le tour de dialogue (Dialogue Turn : DT) est défini par le cycle qui comprend une interaction utilisateur et une interaction système, le silence pouvant être considéré comme une interaction. Un DT contient au moins un ST. Comme les actes du système ne sont pas forcément dialogiques, il se peut qu'un DT contienne plusieurs STs.
- Un tour de service est défini par le cycle entre deux étapes de complétion de la tâche. Cela comprend toujours au moins un DT. Ce concept n'est pas nécessaire pour l'architecture, il correspond simplement à des problématiques d'implémentation de service.

Les figures 1.2 et 1.3 donnent respectivement une représentation des concepts de ST et DT. Pour une simplicité de représentation ces figures ne représentent les accès au BO que depuis le DM, mais le même genre de comportement est possible pour les autres modules de la chaîne de dialogue. En réalité, le concept de tour le plus pertinent est celui qui permet de rendre compte des contraintes imposées au système. Le DT est intéressant dans le sens où il rend compte de la contrainte temporelle du dialogue.

D'un point de vue fonctionnel, il est primordial de considérer également le tour de dialogue marqué par les CAs du système. En effet, dans un système où les observations sont incertaines, plusieurs plans d'actes sont générés et lorsque l'un des plans d'actes est exécuté et qu'au moins l'un de ses actes est engageant (CA), alors le système doit prendre en compte le fait d'avoir réalisé ce CA. A contrario, si aucun acte n'est engageant, peu importe pour le reste du traitement si cet acte a été exécuté ou non. Plus loin, dans la section 1.3, nous verrons que l'exécution d'un CA impose un traitement particulier de la part du système que nous appelons la prise d'engagement.

1.2.3 Interface

Dans cette sous-section, les canaux de communication entre les différents composants sont débattus. Dans un premier temps, nous clarifierons l'information qui doit être échangée, puis dans un second temps de quelle manière véhiculer l'information.

Interface ASR-SLU

Le module d'ASR renvoie une liste N-best avec la notation des séquences de mots. Ces scores sont dans le cas général des probabilités qui peuvent avoir deux sens différents :

1. La probabilité que cette séquence de mots précise soit la séquence qui a été dite. Cela signifie que la liste N-best des séquence de mots propose des alternatives qui sont mutuellement exclusives, et que l'on peut définir une distribution de probabilité sur la liste complète. Du point de vue du dialogue, la logique est de se dire qu'une seule séquence de mots a été prononcée par l'utilisateur, et qu'il ne peut pas avoir dit à la fois "je veux un restaurant italien" et "j'veux un restaurant italien". Comme les deux ne peuvent pas avoir été dits, ces deux phrases ne peuvent être justes à la fois, et donc nous avons la garantie de l'inégalité suivante de leurs probabilités respectives : $p_1 + p_2 \leq 1$.
2. La probabilité que cette séquence de mot corresponde au signal reçu, indépendamment des autres séquences de mots possibles. Cela signifie que plusieurs séquences de mots peuvent correspondre à ce qu'a dit l'utilisateur. Cette vision reflète mieux la façon dont les modules d'ASR fonctionnent en majorité aujourd'hui. Chaque séquence est évaluée indépendamment, c'est-à-dire que le signal sonore peut être en adéquation avec "je veux un restaurant italien" à une probabilité p_1 et en adéquation avec "j'veux un restaurant italien" à une probabilité p_2 et que l'on ait $p_1 + p_2 > 1$. Dans ce cas, les séquences de mots ont des probabilités dont on ne connaît pas les dépendances. L'estimation moyenne de leur dépendance revient à les considérer indépendantes (maximisation de l'entropie dans la théorie de l'information de Shannon, 1948).

Le problème ici est que l'ASR fournit une probabilité du type (2) et que la chaîne de dialogue a besoin de probabilités du type (1) : le nombre n d'alternatives dans la liste N-best est souvent assez important et il est bien plus facile de raisonner sur des informations exclusives qu'indépendantes. En effet avec n informations exclusives, il y a $n+1$ portions d'espaces des possibles à considérer : chaque portion où une de ces informations est vraie et la portion d'espace ou aucune de ces informations n'est vraie. A contrario, en présence de n informations indépendantes, chaque information divise l'espace des possibles en deux, ce qui fait que le nombre de portions d'espaces est 2^n.

La façon classique de résoudre ce problème est d'opérer une normalisation approximative : $p_k^* = \frac{p_k}{\sum_i p_i}$. Mais, elle n'est pas exacte et son inexactitude est problématique à plusieurs égards. Trois exemples sont fournis :

– **Non absorbance de la probabilité** 1 **:** Lorsqu'une source d'information est certaine de l'information qu'elle produit, cette information ne doit pas pouvoir être remise en cause. En effet, lorsqu'une probabilité égale à 1 est affectée, alors elle doit absorber toutes les autres informations concernant la même proposition. Cette absorbance n'est pas garantie par la normalisation approximative. Par exemple, si le résultat de reconnaissance vocale renvoie $p_1 = 1$ pour la phrase s_1 et $p_2 = 0.2$ pour

la phrase s_2 (probabilités du type (2)). Cela signifie que l'algorithme est sûr que s_1 a été dit. En conséquence, la distribution de probabilité (de type (1)) devrait être $p_1^* = 1$ et $p_2^* = 0$. Mais avec cette normalisation, on trouve : $p_1^* = 0.83$ et $p_2^* = 0.16$.

- **Suppression des marqueurs d'ignorance :** Imaginons que l'utilisateur répond à une question oui/non et que le système ne sait reconnaître que les réponses "oui" et "non". Si l'utilisateur répond "nouais", alors l'algorithme renverra des probabilités de ce type : $p_1 = 0.7$ pour la séquence "oui" et $p_2 = 0.7$ pour la séquence "non". La normalisation trouvera alors les probabilités suivantes : $p_1^* = 0.5$ et $p_2^* = 0.5$. Si on recommence le même processus avec $p_1 = 0.01$ et $p_2 = 0.01$ (pour la réponse "Je suis en plein accord avec ce que vous venez de dire" à la même question), on trouvera le même résultat. Il en résulte que l'on perd de l'information à propos de l'ignorance. Dans le premier cas, on est pratiquement sûr que l'un des deux a été dit alors que dans le second, on est pratiquement sûr que ni l'un ni l'autre ne l'a été. Le système devrait pouvoir prendre en compte ce genre d'information, par exemple pour décider d'essayer un autre modèle d'ASR (à large vocabulaire) dans le second cas.
- **Perte de l'incertitude dans le cas où une seule séquence n'est possible :** Si seule la séquence s avec probabilité p non nulle est reconnue, alors avec la normalisation, on trouve le résultat suivant $p^* = p/p = 1$. Pourtant, ce qui a été dit peut être Out-Of-Vocabulary [8] et la valeur de p est porteuse d'information.

La façon d'utiliser les scores de confiance de l'ASR et du SLU est une question complexe qui a déjà été posée dans la littérature (Thomson et al., 2008b). Le cadre formel LFPR décrit dans le chapitre 2 permet de résoudre le problème de façon théorique (voir la section 2.6).

Interface SLU-DM

Les modules de SLU renvoient une liste N-best des représentations sémantiques des actes de dialogues avec une distribution de probabilité au DM. Le SLU peut fournir au DM des informations supplémentaires telles que le niveau de langage de l'utilisateur, son niveau technique, ...

Interface DM-NLG

Le DM contient la logique de dialogue et renvoie en sortie des alternatives de séquences actes en fonction de l'état du dialogue et du contexte actuel. La sortie du DM est donc une liste N-best associée à des utilités, dans le cas général, puisque l'on se trouve dans la partie

8. Hors du vocabulaire.

production de la chaîne de traitement. Il se peut que plusieurs alternatives proposées par le DM soient pertinentes dans un même contexte.

Interface NLG-TTS

Le NLG retourne une liste N-best d'énoncés textuels associés à des utilités. Au final, le TTS proposera à son tour une liste N-best de synthèses vocales associées à des utilités. C'est alors au rôle du Scheduler de choisir la séquence d'actes à exécuter.

Interface avec le Back Office

L'interface du BO est fortement dépendante de l'application.

1.3 Les contrôleurs

Les contrôleurs que nous avons introduits sont au nombre de 3 : le Log Manager, le Context Manager et le Scheduler. Leur spécifications fonctionnelles sont définie dans cette section. Pour une API succinte, vous pouvez vous référer à l'annexe B.

1.3.1 Le Log Manager

Le Log Manager est responsable de l'enregistrement automatique des logs. Idéalement, le service de dialogue devrait être capable de rejouer les dialogues sur la seule base de ses logs. Son rôle est de fournir les informations nécessaires à l'algorithme d'apprentissage et aux outils d'annotation et de monitoring des logs.

1.3.2 Context Manager

Le Context Manager est le module utilisé pour le stockage temporaire des informations sur la session en cours. Il existe différentes façon de représenter ces informations.

Les Context Manager à base d'état

Les Context Managers à base d'état rassemblent les "Belief States" et leurs probabilités dans le Context Manager. Son rôle est limité à celui de base de données où tous les états possibles du monde sont énumérés. L'"Information State" (Bohlin et al., 1999; Larsson & Traum, 2000; Bos et al., 2003) est un exemple de structure utilisée dans les systèmes de dialogue pour répertorier les contextes. Il est constitué d'une partie statique (SIS) où sont sauvegardées entre autres les règles, les actions, le typage de la partie dynamique et les plans de l'agent et une partie dynamique (DIS) où sont conservées les connaissances qui

peuvent évoluer au cours du dialogue. Notre définition du Context Manager ne concerne que le DIS. Le DIS est fortement typé et contient une structure globale permettant de décrire exactement l'état d'un dialogue. L'utilisation des Partially Observable Markov Decision Process (Sondik, 1971; Lovejoy, 1991) dans les systèmes de dialogue (Williams & Young, 2005) s'est naturellement tournée vers cette représentation à base d'état pour le transformer en Hidden Information State (Young et al., 2005) (HIS). Par définition, l'utilisation de modèles dérivés des Markov Decision Process impose un Context Manager à base d'états.

La force des approches à base d'états est leur caractère fortement typé qui structure la connaissance et permet d'utiliser de façon plus ou moins simplifiée les théories de l'interaction. Lorsque l'on commence à considérer les contextes probabilisés, un autre de leurs atouts est la rapidité de traitement de plusieurs contextes en parallèle, à la manière d'une base de données. En effet, pour chaque nouvelle information, il suffit de faire un raffinement [9] de la partition des états considérés. Cependant, les faiblesses de ce type de représentation du contexte sont les contrecoups de leurs forces : d'une part le fort typage imposé par l'"Information State" impose une méthode stricte au développeur qui est souvent plus restrictive que structurante, et d'autre part le nombre d'HIS à traiter en parallèle explose exponentiellement en fonction du nombre d'informations probabilisées reçues.

Les Context Manager à base de connaissances

Contrairement à l'approche à base d'état, ce type de Context Manager conserve les connaissances indépendamment les unes des autres. Ces connaissances sont les objets manipulés. Dans la littérature des systèmes de dialogue, les systèmes fondés sur la logique modale BDI (Belief-Desire-Intention, Bretier, 1995; Sadek et al., 1997) sont des exemples de systèmes de dialogues dont le Context Manager est basé sur des connaissances. Le Context Manager rassemble les connaissances inférées par chacun des modules de la chaîne de dialogue et les met à disposition. Le rôle du Context Manager est alors plus complexe : il sert également à déduire de nouvelles connaissances à partir de celles qui constituent la base, grâce à des règles dépendantes du formalisme utilisé et des propriétés liées au domaine applicatif de l'application de dialogue. Par exemple, la théorie formelle de l'interaction de Sadek (1991; 1997) définit une quarantaine de règles qui sont plus ou moins universelles. Mais chaque application de dialogue nécessite un complément de règles applicatives.

Les choses se complexifient considérablement lorsque l'on s'intéresse aux connaissances incertaines :

9. Refinement en anglais

- Renforcement : si deux informations distinctes démontrent la même conclusion, alors le score de confiance de cette proposition doit croître.
- Contradiction : si deux informations distinctes sont contradictoires, alors les scores de confiance accordés à chacune de ces informations doivent décroître.
- Réfutation : si une information a été contredite, alors les scores de toutes les déductions faites à partir de cette première information doivent décroître.
- ...

Au moment de l'écriture cette thèse, aucun système de dialogue ne contient de Context Manager à base de connaissances incertaines à notre connaissance. Le chapitre 2 présente un cadre formel original : le Logical Framework for Probabilistic Reasoning.

1.3.3 Le Scheduler

Le Scheduler est responsable du déroulement des différentes tâches dans le système. Sa mission est multiple :

- Le Scheduler doit assurer les contraintes temps réel du système, c'est-à-dire de s'assurer que des actes de dialogue sont envoyés à l'utilisateur dans un temps raisonnable.
- Dans une application de dialogue avec gestion de l'incertitude, le traitement de toutes les tâches ne peut être accompli dans les contraintes temps réel. Le Scheduler doit ordonner les différentes tâches assignées par le système, en fonction de leurs caractéristiques (voir la partie 1.2.2), notamment leurs complexités computationnelles et leurs utilités (ou la probabilité de sa pertinence pour le bon déroulement du dialogue).
- Le Scheduler stocke les alternatives de séquences d'actes à exécuter (actes qui ont été planifiés, voir la partie 1.2.2).
- Le Scheduler choisit la séquence d'actes à exécuter. Lorsque cette séquence comprend au moins un Committing Act (CA), le Scheduler a pour tâche de renseigner les différents modules des choix qu'il a faits. Cette tâche est ce que l'on appelle l'engagement du système vis-à-vis de l'utilisateur. Elle comprend :
  - la remise à zéro de l'ensemble des séquences d'actes considérées : les séquences d'actes qui étaient en compétition avant la décision qui entraîne l'engagement du système, ne sont plus pertinentes a priori, puisque l'état d'engagement du dialogue a changé.
  - la mise à jour et le nettoyage du Context Manager : le CM a besoin d'être informé du choix qui a été fait, ne serait-ce que pour stocker la décision. Pour des raisons de performance computationnelle, il est également indispensable que le CM soit allégé des connaissances qui deviendront obsolètes pour les tours de dialogue suivants : par exemple les déductions faites suite aux différents plans qui ont été générés

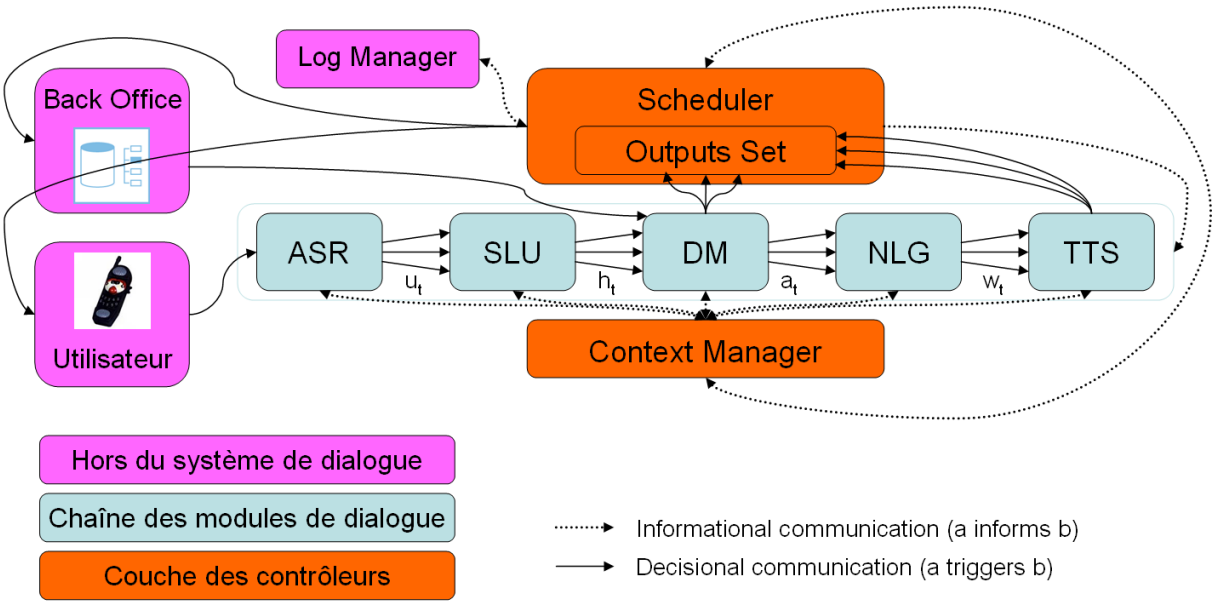


FIGURE 1.4 – L'architecture complète du système

pour les alternatives d'actes peuvent être supprimées.

- le compte-rendu des actes exécutés auprès des modules de la chaîne de dialogue : par exemple le module ASR doit être renseigné du fait que le système pose une question "oui"/"non", pour qu'il puisse charger la grammaire adéquate.

– Le Scheduler est enfin le module qui envoie au Log Manager la plupart des traces à enregistrer.

1.3.4 Vue générale des interactions entre les différents contrôleurs

Maintenant que nous disposons des modules de contrôle requis, nous considérons que l'horloge naturelle du système est gouvernée par les actes engageants, que la figure 1.2 illustre partiellement. La figure 1.4 la reprend pour y ajouter les éléments de contrôle. Les lignes pleines représentent les processus décisionnels, tandis que les lignes en pointillé montrent les échanges informationnels entre les différents composants.

Pour aider le lecteur à appréhender les diverses subtilités de cette architecture, nous explicitons de façon conceptuelle ce qui se passe à partir du moment où le système reçoit un énoncé de l'utilisateur.

Contrairement à ce qui est représenté sur la figure, le module qui reçoit l'énoncé de l'utilisateur est le Scheduler. Le rôle du Scheduler est d'organiser les différentes tâches entre les modules de la chaîne de dialogue. La première tâche du tour de dialogue sera donc de demander au module de reconnaissance vocale (ASR) de générer plusieurs alternatives de texte reconnu. Le Scheduler recevra alors en résultat de cette tâche la demande que

l'interpréteur (SLU) traite le N-best. Le Scheduler décidera ensuite que le système exécute la tâche en partie (ou en totalité, ou au contraire ne permettra jamais l'exécution de la tâche), et enverra un N'-best au SLU et au Context Manager pour garder la trace des informations traitées. Il a ainsi le pouvoir et la responsabilité de limiter la taille des N'-best alternatives traitées par chaque module. Il gardera alors en mémoire dans son "Output Set" le $(N - N')$-best restant non traité. Cette tâche restante pourra alors être exécutée plus tard si le Scheduler considère que cela est nécessaire et que les contraintes temps réel seront respectées.

En résumé, le Scheduler a le pouvoir de demander à chaque module de la chaîne de dialogue d'exécuter des tâches et ceux-ci ont le pouvoir d'ajouter des tâches à la file d'attente du Scheduler. Cette file d'attente est appelée "Output Set" et elle est plus généralement une liste de séquences d'actes atomiques. Les contraintes temps réel du dialogue homme-machine imposent au système d'exécuter un acte (le meilleur étant donné les connaissances, déductions et planifications actuelles) de dialogue au bout d'un certain temps. Mais pendant ce temps, le Scheduler orchestre les tâches internes comme bon lui semble.

La chaîne de dialogue n'est alors plus que le bon usage classique des modules. L'ASR demandera typiquement à ce que sa sortie soit traitée par le SLU, mais cette correspondance ne figure plus en dur dans l'architecture. Chaque Committing Act et Time Consuming Act sont donc ainsi contrôlés par le Scheduler.

De la même manière que le Scheduler fait la correspondance entre les tâches et les processus étant donné les ressources dont il dispose, le Context Manager fait la correspondance entre les diverses connaissances incertaines du système. Pour ce faire, chaque module de la chaîne de dialogue a accès au Context Manager en lecture et en écriture pour s'informer de l'état incertain courant et ajouter des faits incertains à la base de connaissance.

Une fois que le Scheduler décide d'effectuer un Committing Act, il effectue ce que l'on a appelé plus tôt l'engagement. Il stoppe toutes les tâches en cours d'exécution, il vide son "Output Set", il informe les divers modules de la chaîne de dialogue des différents Committing Acts qui sont en train d'être exécutés et il nettoie le Context Manager, en supprimant les informations obsolètes et en synthétisant les informations à conserver, de manière à ne garder que le strict nécessaire.

Pour aider le lecteur à comprendre plus en profondeur le fonctionnement général du système, nous avons déroulé le même scénario du point de vue du Context Manager dans l'annexe C et du point de vue du Scheduler dans l'annexe D. Ces annexes utilisent l'API décrite dans la section suivante.

Chapitre 2

Gestion de l'incertitude dans les systèmes de dialogue

Nous venons de définir dans le chapitre précédent une architecture générale pour l'insertion de traitements statistiques dans les systèmes de dialogue. Une de ces fonctionnalité est la gestion des incertitudes. Ce chapitre cherche à résoudre la problématique de prise en compte du caractère incertain de la plupart des informations que le système reçoit, ou déduit.

L'objectif affiché de cette fonctionnalité est d'améliorer la robustesse du système de dialogue en le rendant capable de rattraper plus facilement ses erreurs, voire d'anticiper la possibilité d'une erreur et ainsi préférer une stratégie de désambiguïsation.

La section 2.1 introduit l'existant en matière de gestion de l'incertitude dans les systèmes de dialogue industriels. La section 2.2 expose les diverses techniques de raisonnement sur des faits incertains de la littérature et les confronte à notre problème. La section 2.3 présente un nouveau formalisme logique adapté à l'architecture présentée dans le chapitre 1. Et enfin, la section 2.5 fait la positionnement de cette théorie par rapport à la littérature du domaine.

2.1 Contexte

En conformité avec les standards industriels actuels, le Dialogue Manager (DM) du système de dialogue de France Télécom R&D est basé sur un automate. Un état de l'automate ne correspond pas exactement à un état du système mais plutôt à une famille d'états du système. En effet, des variables globales ou locales viennent enrichir la description par état de l'automate. Ces variables sont conservées dans le module Context Manager (CM). La plupart des choix de transition entre un état et un autre se font après une requête vers

le CM. La figure 2.1 donne une idée de l'architecture du système actuel.

La première remarque est que le système est incomplet en terme de gestion de l'incertitude. En effet, une fois la meilleure interprétation fournie au DM par le Speech Language Understanding (SLU), le DM n'a plus de gestion de l'incertitude. Il ne lui reste que la possibilité de refuser l'interprétation que le SLU lui a fournie pour recommencer le traitement sur la deuxième meilleure interprétation.

La seconde remarque concerne le manque de symétrie : le SLU n'a pas la capacité d'enrichir le CM et une fois que le DM a déterminé une action à exécuter, le reste du traitement concerne la transformation de cette action conceptuelle en chaîne de caractères dans un premier temps puis en un flux audio dans un second temps.

La figure 2.2 est une simplification de la figure 1.4. Le contexte est accessible en lecture et en écriture depuis n'importe quelle partie de la chaîne de dialogue. L'automate du DM reste identique à sa version sans gestion de l'incertitude, mais il est maintenant parcouru de manière stochastique et les incertitudes ainsi générées sont gérées par le CM. Par exemple, dans un état de l'automate, si la transition est déterminée par le test suivant : $X < 0$, l'état envoie une requête au CM qui délivre en réponse une estimation des probabilités concernant le résultat du test. Ensuite, le parcours continue dans chacune des branches considérées selon les probabilités mesurées par le CM.

2.2 Gestion de l'incertitude

2.2.1 Les logiques défaisables

La motivation initiale de la gestion de l'incertitude porte sur l'intégration de méthodes statistiques, mais par soucis de complétude, nous allons expliquer en préambule en quoi les méthodes non numériques semblent moins adaptées au problème du dialogue vocal.

En effet, il parait intuitif qu'un système de dialogue n'a le droit qu'à un nombre

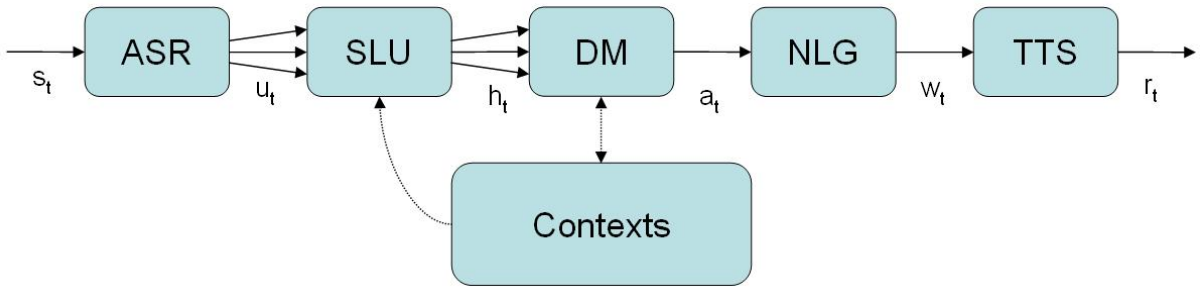


FIGURE 2.1 – L'architecture actuelle.

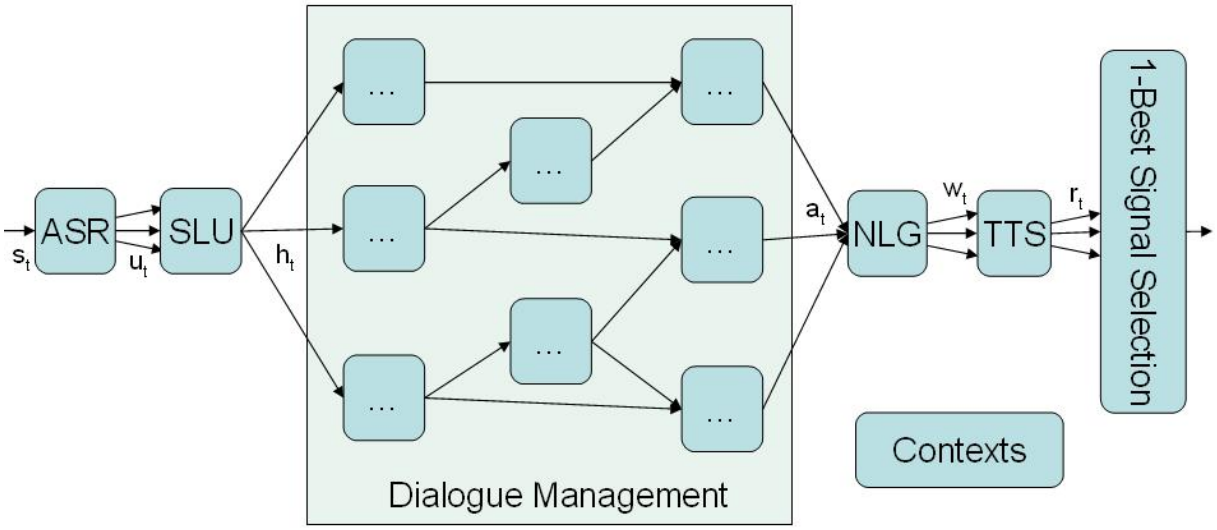


FIGURE 2.2 – L'architecture cible.

restreint d'erreurs par dialogue. Et en conséquence, les logiques du défaisable [10] (Pollock, 1987; Garcia, 1993) ou le domaine plus vaste de la théorie de l'argumentation (Dung, 1993; Governatori & Maher, 2000) pourraient convenir. Cependant, il faut garder à l'esprit que la nature des informations dont nous disposons sont d'ordre statistiques sous la forme d'une liste N-best où chaque résultat de reconnaissance vocale ou d'interprétation est associé à une probabilité. Le système reçoit une nouvelle liste à chaque tour de parole. D'une part, utiliser des logiques non-numériques forcerait à réduire chaque liste N-best à ses deux ou trois meilleurs éléments et à simplement garder une relation d'ordre les liant. Énormément d'information peut-être perdue. D'autre part, ces logiques s'intègrent généralement plus difficilement à l'apprentissage par renforcement (ou en tout cas, c'est un domaine qui reste inexploré à notre connaissance). Il n'est donc pas possible de retenir ce type de traitement des incertitudes dans l'optique d'une intégration finale dans l'architecture définie dans le chapitre 1.

2.2.2 L'incertitude et les systèmes de dialogue

Les sources principales d'incertitude dans un système de dialogue sont les modèles statistiques utilisés dans l'ASR et le SLU. Même si cela pose encore beaucoup de problèmes aujourd'hui (Litman et al., 2000; Gabsdil & Lemon, 2004; Jonson, 2006; Crook & Lemon, 2009), il est raisonnable de penser que ces modules sauront estimer les probabilités de leurs résultats dans un futur proche (Li & Huerta, 2007; Huet et al., 2007; Thomson et al., 2008b; Williams & Balakrishnan, 2009). D'autres modules peuvent également

10. Defeasible logics en anglais.

fournir des règles probabilisées. Par exemple les règles apprises par apprentissage sont de nature statistique et peuvent être exprimées sous forme de probabilités. Quand on parle de couplage incertitude/apprentissage, le premier réflexe est souvent de penser aux réseaux probabilistes (Pearl, 1988; Sutton & Barto, 1998) et c'est ce vers quoi se sont tournés les premiers systèmes de dialogue traitant les incertitudes (Roy et al., 2000; Williams & Young, 2005; Young, 2006). Mais dans le cas d'un DM conventionnel à base de règles ou de transitions dans un automate, ils ne conviennent pas pour les raisons suivantes :

D'une part, le système de dialogue échange des signaux avec l'utilisateur. Le traitement de ces signaux renseigne le système avec des informations dérivées, sur la base d'hypothèses. Les règles sont de façon générale à sens unique et ne permettent pas de définir une matrice de dépendance complète. Par exemple, le résultat d'une interprétation peut être la probabilité de cette proposition : "l'utilisateur a dit qu'il avait un problème avec internet". Mais la matrice de probabilité conditionnelle permettant de passer à la proposition suivante : "l'utilisateur a un problème avec internet" ne peut être obtenue directement parce que l'utilisateur peut ne pas avoir dit qu'il avait un problème et pourtant en avoir un. De manière plus formelle, les règles de déduction du DM donnent des probabilités du type $p(a|b)$ quand la règle est $b \Rightarrow a$. Pour pouvoir utilise des réseaux probabilistes, il faudrait que l'on ait aussi la connaissance de $p(a|\neg b)$, c'est-à-dire une règle de la forme $\neg b \Rightarrow a$.

D'autre part, le dialogue nécessite un cadre logique qui permette de déduire des inférences non triviales ou des incohérences telles que "il est peu probable que l'utilisateur souhaite s'abonner à la fois à l'offre ADSL et à une connexion internet via le câble" et les réseaux probabilistes sont inadaptés pour le raisonnement logique.

2.2.3 La Théorie de l'Évidence

La problématique décrite dans cette partie renvoie directement à la problématique d'agrégation de connaissances peu fiables. Dempster (1968) a défini une typologie des croyances qui pour la première fois se définissait selon un intervalle, avec une borne inférieure et une borne supérieure. Cette typologie a permis à Shafer (1976) d'asseoir les bases de la théorie des fonctions de croyance. Nous allons en rappeler les fondements.

Soit un cadre de discernement $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$, une fonction de croyance est une capacité de Choquet (Choquet, 1953) monotone d'ordre fini, c'est-à-dire une fonction de 2^Ω vers $[0, 1]$ obéissant aux contraintes suivantes :

$$\begin{cases} Bel(\emptyset) = 0 \\ \forall n \geq 1, \forall i \in \{1, \ldots, n\}, A_i \subseteq \Omega \\ Bel(\bigcup\limits_{i \in \{1,\ldots,n\}} A_i) \geq \sum\limits_{\substack{I \subseteq \{1,\ldots,n\} \\ I \neq \emptyset}} (-1)^{|I|+1} Bel(\bigcap\limits_{i \in I} A_i) \end{cases} \tag{2.1}$$

Les fonctions de probabilité classiques sont un cas particulier de fonctions de croyance où les sous-ensembles A_i sont disjoints deux à deux et où l'inégalité est réduite à une égalité.

Les fonctions de croyance sont généralement définies à l'aide d'une allocation de masse m définie de 2^Ω vers $[0, 1]$ telle que :

$$\sum_{A \in \Omega} m(A) = 1 \tag{2.2}$$

$$\forall A \subseteq \Omega, Bel(A) = \sum_{\substack{B \subseteq A \\ B \neq \emptyset}} m(B) \tag{2.3}$$

Les allocations de masse et les fonctions de croyance sont deux écritures pour une même information. La règle de combinaison de Dempster (1968), notée $\otimes$, permet d'obtenir les allocations de masse de la fusion de deux informations correspondant aux allocations de masse m_1 et m_2. Cette combinaison n'est applicable que pour les informations indépendantes, mais elle peut être utilisée de manière plus générale pour les informations dont on n'a pas connaissance de dépendance.

$$\forall A \subseteq \Omega, (m_1 \otimes m_2)(A) = \sum_{B \cap C = A} m_1(B) m_2(C) \tag{2.4}$$

Il est d'usage de normaliser la masse de telle manière que $m(\emptyset) = 0$. $\kappa = (m_1 \otimes m_2)(\emptyset)$ est appelé le degré de conflit entre les allocations de masse m_1 et m_2. Ce processus de normalisation a été longuement défendu et critiqué (Smets 1990; 1992), mais il est finalement difficile de remettre en cause ce processus dans un contexte logique. En effet, dans un système logique, si l'on rencontre une inconsistance, la solution qui s'impose à nous est de défausser cette possibilité par la normalisation. La normalisation devient alors entièrement liée au fait que l'on a rencontré une inconsistance et cette inconsistance n'est due qu'à la description logique sous-jacente. Comme le cadre de discernement est infini dans un système logique, critiquer la normalisation revient alors à critiquer la base de connaissances.

Ce formalisme est très précieux dans la mesure où il permet de combiner des informations disjointes, à la manière dont un système logique déduit de nouveaux faits à partir

de connaissances de sources différentes. Il permet également de gérer les contradictions ou incompatibilités présents dans les informations, et que l'on découvre au fur et à mesure de leurs combinaisons, grâce à l'indicateur κ. Il ne reste plus qu'à définir comment calculer à partir de l'allocation de masse les fonctions de probabilité inférieure et supérieure, quelquefois appelées respectivement fonction de croyance et fonction de plausibilité. La croyance d'un ensemble A correspond à la somme des masses que l'on sait appartenir à l'ensemble A. La plausibilité d'un ensemble A est la somme des masses pouvant potentiellement appartenir à l'ensemble A, c'est-à-dire la somme des masses que l'on ne sait pas appartenir au complémentaire de A dans Ω.

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2.5}$$

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \tag{2.6}$$

La fonction de plausibilité est duale de la fonction de croyance. Il en suit une grande quantité de propriétés parmi lesquelles les règles de monotonie (implications 2.7 et 2.8), les règles de sous-additivité (inégalités 2.9 et 2.10), les règles de sur-additivité (inégalités 2.11, 2.12 et 2.13) et les conditions aux limites (égalités 2.14 et 2.15). Toutes ces règles sont des conséquences directes des propriétés déjà énoncées.

$$A \subseteq B \Rightarrow Bel(B) \geq Bel(A) \tag{2.7}$$

$$A \subseteq B \Rightarrow Pl(B) \geq Pl(A) \tag{2.8}$$

$$Bel(A) + Bel(\overline{A}) \leq 1 \tag{2.9}$$

$$Bel(A \cup B) \geq Bel(A) + Bel(B) - Bel(A \cap B) \tag{2.10}$$

$$Pl(A) + Pl(\overline{A}) \geq 1 \tag{2.11}$$

$$Pl(A \cup B) \leq Pl(A) + Pl(B) - Pl(A \cap B) \tag{2.12}$$

$$Bel(A) \leq Pl(A) \tag{2.13}$$

$$Bel(\emptyset) = 0 \text{ et } Bel(\Omega) = 1 \tag{2.14}$$

$$Pl(\emptyset) = 0 \text{ et } Pl(\Omega) = 1 \tag{2.15}$$

Différents travaux ont permis de porter ce formalisme dans un cadre logique (de Kleer, 1986a; de Kleer, 1986b; de Kleer, 1986c; Kohlas & Monney, 1995), mais ces travaux restent des approches trop symboliques pour être utilisés directement dans un système de dialogue, qui fonctionne plus de manière procédurale, à l'aide de règles ad hoc.

En ce qui concerne les autres efforts pour faire cohabiter scores de confiance et logique, nous pouvons citer rapidement Nilsson (1986) qui a défini la logique probabiliste,

mais sa volonté de lever toutes les présuppositions généralement faites, notamment sur les indépendances conditionnelles, rend cette théorie inapplicable en pratique parce que trop complexe computationnellement et pas assez précise (elle définit des probabilités inférieures et supérieures très éloignées). A l'opposé, bien que très performante en matière de complexité, la logique floue (Zadeh, 1978; Bouchon-Meunier, 1995; Bouchon-Meunier & Nguyen, 1996; Bouchon-Meunier, 2007) a été abandonnée étant donné le fait que les systèmes de dialogue manipulent des probabilités et que les ressources computationnelles sont suffisantes pour les traiter à l'aide de théories de la famille de la théorie de l'Évidence.

Cette théorie de l'Évidence appliquée à la mise en œuvre de la théorie de l'interaction rationnelle (Sadek, 1991; Bretier, 1995) a donné naissance à d'autres travaux de raisonnement sur les incertitudes pour les systèmes de dialogue comme dans la thèse de Mallat Desmortiers (2003). Malheureusement, ce type d'approche ne permet pas le raisonnement par défaut, si utile dans notre contexte applicatif.

C'est la raison pour laquelle la section suivante décrit un formalisme logique original appelé le Logical Framework for Probabilistic Reasoning (LFPR) (Laroche et al., 2008). Cette modélisation est fortement inspirée de la théorie de Dempster-Shafer adapté à l'architecture présentée dans le chapitre 1.

2.3 Logical Framework for Probabilistic Reasoning

Cette section présente un nouveau cadre formel pour le raisonnement sur l'incertitude dans un cadre logique. Il permet de calculer des probabilités inférieure et supérieure d'un proposition p à la manière de la théorie de Dempster (Dempster, 1968).

2.3.1 Les ensembles de mondes

La plupart des représentations de l'incertitude démarrent par la définition d'un ensemble des *mondes possibles* (Halpern, 2005), que l'on appelle certaines fois des *états* ou des *résultats élémentaires*. Intuitivement, ce sont les mondes qu'un agent considère possible. Par exemple, lorsque l'on lance un dé à six faces, il est raisonnable de considérer six mondes possibles : un pour chaque face. Ceci est représenté par un ensemble $W* = \{w_1, w_2, w_3, w_4, w_5, w_6\}$.

Dans ce chapitre, les objets qui sont connus sont appelés *évènements* ou *propositions*. Formellement, un évènement ou une proposition est un ensemble de mondes possibles. Par exemple, un évènement tel que "le résultat est pair" correspond à l'ensemble $\{w_2, w_4, w_6\}$. De façon classique, l'incertitude d'un agent est représentée par un ensemble $W' \subseteq W*$. L'agent considère alors un évènement U possible si $U \cap W' \neq \emptyset$; c'est-à-dire s'il considère

un monde possible qui appartienne à U. Si l'agent a observé que le résultat du lancer de dé était pair alors il considère que l'évènement "le résultat est inférieur ou égal à 3" possible parce que $U \cap W' = \{w_2\} \neq \emptyset$. De même, un agent sait U si $W' \subseteq U$, ce qui est équivallent à considérer que $\overline{U}$ (le complémentaire de U dans $W*$) est impossible.

Dans notre cas, nous ne chercherons pas à définir $W*$ précisément parce que celui-ci de façon inhérente au problème du dialogue de profondeur infinie. En effet, un dialogue n'a pas une durée bornée. Il est donc possible de recevoir N énoncés utilisateur pour N arbitrairement grand. Chacune de ces réceptions d'énoncé utilisateur est un évènement qui divise $W*$ au minimum en deux : la possibilité d'avoir reçu ou non cet énoncé. En conclusion, la cardinalité de $W*$ est supérieure à 2^N avec N arbitrairement grand. Dans les faits, il est donc plus commode de considérer $W*$ de taille infinie, chaque énoncé utilisateur apportant un raffinement [11] supplémentaire.

L'objectif de la modélisation par mondes possibles est de restreindre l'ensemble des mondes que l'agent considère possible de manière à connaitre le plus précisément possible, le monde réel w_r.

2.3.2 Mesure de probabilité

Nous faisons une distinction forte entre la probabilité d'une proposition et la probabilité que cette proposition soit *démontrée*. Pour bien saisir cette distinction, clarifions le processus d'assimilation d'une connaissance. La connaissance est véhiculée par une *information*. Cette information peut provenir des règles de la base de connaissances et être ainsi être le fruit d'un raisonnement logique. Elle peut également provenir d'une entité extérieure à la base de connaissances, par exemple d'un module de la chaîne de dialogue, ou encore de l'environnement. Cette information porte un sens. Nous dirons que donc que l'information i véhicule une *connaissance*. Cette connaissance est représentée sous forme logique par une *proposition* p avec une certaine fiabilité numérique f. La proposition p est démontrée par l'information i avec la probabilité $P(p \text{ est démontrée}) = f$. Cependant, celle-ci n'est pas nécessairement la probabilité de la proposition. Pour reprendre l'exemple de la section précédente, si l'information fournie par l'interprétation "l'utilisateur a dit qu'il avait eu un problème avec internet", alors le système a une démonstration que "l'utilisateur a un problème avec internet". Cette démonstration repose sur le bien-fondé de l'interprétation. Cependant même si cette hypothèse est fausse, c'est-à-dire même si l'interprétation n'est pas bien fondée, il se peut que l'utilisateur ait un problème avec internet.

Comme un système de dialogue reçoit des informations incertaines, il est probable, ou

11. "refinement" dans la littérature anglosaxone

en tout cas possible, qu'il reçoive des informations contradictoire. Il sera donc amené à déduire l'inconsistance $\perp$. Cette inconsistance, par définition, ne peut appartenir à aucun monde possible. De même, par définition, $\perp$ permet de déduire toutes les propositions de la logique utilisée [12]. Il en résulte qu'il n'existe qu'un monde impossible $w_\perp$: le monde où tout est vrai – tout et son contraire. Nous allons donc travailler sur l'ensemble des mondes $W = W* \cup \{w_\perp\}$.

Un monde possible w est un monde consistent, c'est-à-dire un monde où $\perp \notin w$. Nous allons définir une mesure μ sur l'ensemble des mondes W. Une mesure est fonction ensembliste respectant une inégalité et deux égalités : la positivité (équation 2.16), la nullité de l'ensemble vide (équation 2.17) et la σ additivité (équation 2.18). Une mesure de probabilité vérifie en plus que la somme des mesures est égale à 1 (équation 2.19). Nous considérons donc la mesure de probabilité $\mu(S)$ sur un ensemble de mondes S, ou de manière équivalente la mesure $\mu(p)$ d'une proposition p qui est définie comme la mesure de son support $S_p \subseteq W$:

$$\forall E \subseteq W,\ \mu(E) \geq 0 \tag{2.16}$$

$$\mu(\emptyset) = 0 \tag{2.17}$$

$$\forall E_i \subseteq W \text{ disjoints deux à deux, } \mu\left(\bigcup_{i\in I} E_i\right) = \sum_{i\in I} \mu\left(E_i\right) \tag{2.18}$$

$$\sum_{w\in W} \mu(w) = 1 \tag{2.19}$$

Les résultats suivants concernant la tautologie $\top$ sont directs :

$$\forall w \in W,\ \top \in w \tag{2.20}$$

$$\mu(\top) = 1 \tag{2.21}$$

2.3.3 Démonstration

Dans notre formalisme un agent raisonne à partir de démonstrations. Une *démonstration* d est constituée de sa proposition p et de l'ensemble de mondes D où la proposition est démontrée. Une démonstration peut être une hypothèse elle-même, c'est-à-dire directement dépendante d'une information générée par un module. Mais plus généralement, une démonstration peut être impliquée par un ensemble de démonstrations $\{d_k\}$ grâce à une règle. Dans ce dernier cas, D est l'intersection des supports D_k des démonstrations d_k.

12. Le travail présenté dans ce chapitre ne fait aucune présupposition sur la logique sous-jacente. Il fait simplement l'hypothèse de l'existence de règles d'inférence permettant de déduire des propositions à partir d'une ou plusieurs autre propositions.

Les démonstrations appartiennent à un agent. En effet, deux agents auront des démonstrations différentes et donc des incertitudes différentes. A bien des égards, cette approche ensembliste rapporte à la logique modale. Introduisons le concept de démonstration dans la logique épistémique (Hintikka, 1962) basé sur l'accessibilité aux monde. Dans le LFPR, un monde est la composition d'instanciations de variables aléatoires, c'est-à-dire qu'un monde donne une valeur à chaque variable aléatoire. Appelons D_a l'opérateur modal pour les démonstrations de l'agent a. Ainsi $w \models D_a(p)$ se lit de la manière suivante : l'agent a a la démonstration de p dans le monde w. Cela signifie que dans le monde w, il sait que p est vrai et que dans ce monde, p est vrai. L'agent a ne peut pas faire d'erreur tant qu'il a une démonstration de ce qu'il pense. Cet opérateur modal suit le système axiomatique $S5$ (Lewis & Langford, 1932) :

(D)	$D_a(p) \Rightarrow \neg D_a(\neg p)$	Consitency
(T)	$D_a(p) \Rightarrow p$	Truth
(K)	$D_a(p \Rightarrow q) \Rightarrow (D_a(p) \Rightarrow D_a(q))$	Deductive Cogency
(4)	$D_a(p) \Rightarrow D_a(D_a(p))$	Self-awareness
(5)	$\neg D_a(p) \Rightarrow D_a(\neg D_a(p))$	Negative Introspection

Ces axiomes sont dits épistémiques car ils sont en général utilisés pour modéliser les connaissances et les croyances d'un agent. Ici, l'opérateur modal D_a représente les démonstrations qui permettront à l'agent a d'obtenir une estimation de la véracité de chaque proposition et D_a n'a pas du tout la même sémantique que les opérateurs de Belief ou de Knowledge présents dans la bibliographie des logiques BDI classiques (Belief-Desire-Intention, Bratman, 1987; Sadek, 1991). D_a est avant tout une modélisation du raisonnement de l'agent a. Ce raisonnement se matérialise naturellement en connaissances, mais l'objectif premier reste de pouvoir partitionner l'ensemble des mondes entre les mondes où l'agent a démontré p, l'ensemble des mondes où l'agent a démontré $\neg p$ et enfin l'ensemble des mondes où ni p ni $\neg p$ n'ont pu être démontrés.

Il est important de justifier l'axiome **(T)**, appelé "Truth". On pourrait penser qu'un agent puisse faire des erreurs quant à ses règles, à propos d'une mauvaise connaissance statistique de la règle dans l'ensemble des mondes. Mais dans ce cas, l'erreur ne provient pas du fait que dans les mondes où $D_a(p)$ est vrai, p serait faux, mais d'une mauvaise mesure d'ensemble des mondes où $D_a(p)$ est vrai. C'est donc une erreur potentielle provenant de l'instanciation de la modélisation et non du formalisme de représentation des démonstrations. L'axiome **(T)** est la base même du LFPR.

L'axiome **(D)** signifie que les démonstrations faites par l'agent sont cohérentes. **(D)** est en fait une conséquence directe de **(T)** et du postulat que le monde réel est un monde consistent.

L'axiome **(K)** est très important dans la mesure où c'est lui qui permet à l'agent d'inférer de nouvelles démonstrations à partir d'hypothèses. Retirer l'axiome **(K)** est équivalent à retirer à l'agent toutes ses capacités de déduction. En effet, l'agent n'a connaissance que de ce qui est "sous" D_a.

Enfin, les axiomes **(4)** et **(5)** sont très similaires et stipulent que l'agent sait exactement quelles démonstrations il a pu réaliser. Ce n'est pas vraiment nécessaire pour l'opérateur de démonstration mais ce sont des axiomes communément acceptés pour le raisonnement épistémique et il semble raisonnable de les considérer vrais pour les démonstrations.

Dans la suite de ce chapitre, nous nous concentrons sur le cas mono-agent et nous simplifions en conséquence la notation D_a de la démonstration par la suppression du paramètre a. De même, nous utiliserons régulièrement la notation D_p pour référer à l'ensemble des mondes où la proposition p a été démontrée. Il est nécessaire de distinguer cette notation de la notation de la démonstration $d_i = (D_i, p_i)$ qui réfère à une démonstration en particulier résultant de l'information i. Nous avons alors de façon directe l'inclusion suivante : $D_i \subseteq D_p$.

2.3.4 Le contexte et la mesure contextuelle

Chaque fois que le système reçoit une nouvelle information i, le système est conditionné par le contexte impliqué par l'évènement e_i de l'arrivée de cette information. Ce contexte C_{e_i} est l'ensemble des mondes où la proposition p_{e_i} [13] "l'évènement e_i a eu lieu" est vraie :

$$C_{e_i} = S(p_{e_i}) \tag{2.22}$$

Où $S(p_{e_i})$ est le support de la proposition p_{e_i}. Le contexte global C_{glob} est le contexte impliqué par tous les évènements reçus par le système et par le fait que l'on considère que le monde réel n'est pas le monde impossible, ce qui implique le développement suivant de C_{glob}, où $D_\perp$ est l'ensemble des mondes où $\perp$ a été démontré :

$$C_{glob} = \overline{D_\perp} \cap \bigcap_e C_e \tag{2.23}$$

Nous nous intéressons à la probabilité des mondes étant donné le contexte actuel, c'est-à-dire, étant donné les informations que nous avons reçues. En conséquence, la mesure de probabilité qui nous intéresse pour la démonstration est la mesure μ_{glob} contextuelle conditionnée au contexte global C_{glob} :

$$\begin{cases} \mu_{glob} : 2^W \to [0,1] \\ \mu_{glob}(S) = \dfrac{\mu(S \cap C_{glob})}{C_{glob}} \end{cases} \tag{2.24}$$

13. p_{e_i} doit être distingué de p_i qui est le contenu propositionnel de l'information i.

2.3.5 Subsomption

Si le cadre propositionnel est doté d'une relation de subsomption $\preccurlyeq$, alors la relation d'ordre partiel $\preccurlyeq_d$ est construite pour les démonstrations. Une démonstration d_1 est subsumée par une seconde d_2, si la proposition p_1 qu'elle démontre est subsumée par la proposition p_2 que démontre d_2 et si son support de démonstration D_1 est inclus dans le support de démonstration D_2 de d_2 :

$$d_1 \preccurlyeq_d d_2 \Leftrightarrow \begin{cases} p_1 \preccurlyeq p_2 \\ D_1 \subseteq D_2 \end{cases} \tag{2.25}$$

2.3.6 Fiabilité et applicabilité

Jusqu'à maintenant, nous avons un système qui peut sembler très proche des approches ensemblistes d'Halpern (2005) pour un contexte inférentiel et donc logique. Ce n'est pas le cas. En effet, Halpern (2005) parle de famille de mesures pour décrire l'incertitude de l'agent concernant les ensembles de monde. Étant donné une famille de mesures de probabilité $\mathscr{P}$, définies sur une algèbre $\mathscr{F}$ de sous-ensembles de W et $U \in \mathscr{F}$, il pose les probabilités inférieure et supérieure :

$$\mathscr{P}_*(U) = inf\,\{\mu(U) : \mu \in \mathscr{P}\} \tag{2.26}$$

$$\mathscr{P}^*(U) = sup\,\{\mu(U) : \mu \in \mathscr{P}\} \tag{2.27}$$

Contrairement à Halpern (2005), nous ne considérons qu'une mesure de probabilité, qui est en fait la mesure de probabilité *a priori* sur l'ensemble des mondes possibles. La mesure de probabilité *a posteriori* est fourni par la contextualisation introduite par l'équation 2.24. La mesure de probabilité a priori est inconnue. Elle est fournie valeur par valeur par les sources d'information externe, par exemple dans le cas du dialogue, le module de reconnaissance vocale affirmera qu'il est sûr à 80% que la phrase prononcée est "J'ai un problème avec ma ligne de téléphone". Le contexte de dialogue dans lequel le système se trouve permettra d'assimiler cette mesure de probabilité a priori pour calculer les probabilités inférieure et supérieure a posteriori que l'utilisateur ait bien dit cela.

Les familles de mesure de probabilité utilisées par Halpern (2005) servent à utiliser des mesures partielles telles que dans l'exemple suivant. Imaginez un sac rempli de dix balles. On sait que trois d'entre elles sont rouges et que les sept restantes sont chacune soit jaune, soit bleue. Imaginons maintenant un tirage unique dans ce sac. On a bien sûr une mesure de 30% de chance de tirer une balle rouge. Mais en ce qui concerne les balles jaunes ou bleues, cela n'est pas aussi simple. C'est une mesure de probabilité inconnue dont la valeur est entre 0 et 70%. Pour cette exemple la famille de mesures $\mathscr{P}$ contiendrait

les huit possibilités de la contenance du sac allant de 0 balle jaune et 7 balles bleues à 7 balles jaunes et 0 balle bleue. Nous aurions donc :

$$\mathscr{P}_*(jaune) = inf\{\mu(jaune) : \mu \in \mathscr{P}\} \tag{2.28}$$

$$\mathscr{P}_*(jaune) = 0 = \mu(jaune) : \mu \begin{cases} \mu(rouge) = 0.3 \\ \mu(jaune) = 0 \\ \mu(bleu) = 0.7 \end{cases} \tag{2.29}$$

$$\mathscr{P}^*(jaune) = sup\{\mu(jaune) : \mu \in \mathscr{P}\} \tag{2.30}$$

$$\mathscr{P}^*(jaune) = 0.7 = \mu(jaune) : \mu \begin{cases} \mu(rouge) = 0.3 \\ \mu(jaune) = 0.7 \\ \mu(bleu) = 0 \end{cases} \tag{2.31}$$

$$\mathscr{P}_*(bleu) = inf\{\mu(bleu) : \mu \in \mathscr{P}\} \tag{2.32}$$

$$\mathscr{P}_*(bleu) = 0 = \mu(bleu) : \mu \begin{cases} \mu(rouge) = 0.3 \\ \mu(jaune) = 0.7 \\ \mu(bleu) = 0 \end{cases} \tag{2.33}$$

$$\mathscr{P}^*(bleu) = sup\{\mu(bleu) : \mu \in \mathscr{P}\} \tag{2.34}$$

$$\mathscr{P}^*(bleu) = 0.7 = \mu(bleu) : \mu \begin{cases} \mu(rouge) = 0.3 \\ \mu(jaune) = 0 \\ \mu(bleu) = 0.7 \end{cases} \tag{2.35}$$

Pour le LFPR, nous n'avons pas encore donné la sémantique associée à cette notion de support de démonstration. En mathématiques, une démonstration permet d'établir une proposition à partir de propositions initiales, précédemment démontrées, en s'appuyant sur un ensemble de règles de déduction. La proposition une fois démontrée peut ensuite être elle-même utilisée dans d'autres démonstrations. Dans toute situation où les démonstrations initiales sont vraies, la proposition démontrée devrait être vraie ; on ne pourrait la remettre en cause qu'en remettant en cause une ou plusieurs des démonstrations initiales ou le système de règles de déduction lui-même, que nous représentons également sous la forme de démonstration, de manière à s'affranchir de la définition d'un tel système d'une part pour conserver la généralité du LFPR et d'autre part pour permettre l'utilisation de règles de déductions incertaines.

Le support D_d d'une démonstration d est par définition l'ensemble de mondes où d est fiable et applicable. Si F_d est le support de la fiabilité de d et A_d le support de l'applicabilité de d, alors l'égalité suivante est vérifiée par définition :

$$D_d = A_d \cap F_d \tag{2.36}$$

Une démonstration d d'une proposition p est dite *fiable* dans un monde w si d est valide

dans w, ce qui implique nécessairement que p est vraie dans w. Les règles statistiques du système sont capables de fournir une probabilité, que nous avons déjà évoqué plus tôt sous le nom de fiabilité numérique et qui peut être identifié à la mesure probabiliste du support de la fiabilité $f = \mu(F)$ de la démonstration qu'elle génère. Pour le problème des balles dans le sac, les informations dont nous disposerions sur la mesure de probabilité serait la suivante :

$$\mu \begin{cases} \mu(F_{rouge}) = 0.3 \\ \mu(F_{jaune \vee bleu}) = 0.7 \end{cases} \tag{2.37}$$

$$F_{rouge} \cap F_{jaune \vee bleu} = \emptyset \tag{2.38}$$

La contrainte d'exclusion entre les balles rouges et les autres ajoute à la connaissance le fait que les balles ne peuvent être jaunes (ou bleues) et rouges à la fois.

La plupart des théories de l'incertain s'arrêtent à la définition des probabilités inférieure et supérieure à l'aide de concepts équivalents au concept de fiabilité que nous venons de définir. Ces théories ont alors des difficultés à raisonner sur des hypothèses de travail (de Kleer, 1986a; de Kleer, 1986b; de Kleer, 1986c; Kohlas & Monney, 1995), ou à faire du raisonnement par défaut [14]. Le LFPR permet ces types de raisonnement grâce au concept d'applicabilité.

Une démonstration d est dite *applicable* dans un monde w si la sémantique permet d'appliquer la démonstration d dans w. Ces considérations sémantiques incluent toutes les conditions qui ne font pas partie de la fiabilité. Généralement, le fait même qu'un règle d'inférence puisse syntaxiquement être appliquée, suppose que les démonstrations utilisées soient applicables. C'est-à-dire, que le contexte impliqué par la présence même des démonstrations hypothèses de l'inférence, implique lui-même les applicabilités de ces démonstrations et donc de la démonstration inférée. Mais il peut arriver que des connaissances supplémentaire viennent contraindre cette applicabilité syntaxique.

Le cas le plus courant intervient lors de l'utilisation d'une règle fréquentielle de diagnostic : si p est observée alors c'est en général q qui l'a causée. Ces règles fréquentielles de diagnostic sont en fait duales avec d'une part un sens $p \Rightarrow q$ incertain et statistique, et d'autre part, un sens $q \Rightarrow p$ certain. L'exemple type est p : "le gazon est mouillé" et q : "il a plu". En effet, l'explication la plus probable à l'observation de p est q, mais cela reste incertain. Et inversement, lorsque q est vérifié, alors on en déduire de façon certaine p. Ce type de diagnostic est très fréquemment utilisé dans les systèmes de dialogue (voir la sous-section 2.3.7). On souhaite éviter le comportement suivant : si l'on sait r : "j'ai arrosé mon jardin" et sa conséquence directe $r \Rightarrow p$, on ne veut pas en déduire q. De

14. En fait ces deux types de raisonnement sont similaire, l'hypothèse de travail correspondant au raisonnement par défaut "en l'absence de connaissance venant la contredire".

manière équivalente, par contraposée, lorsque $\neg q$ est connu, on ne veut en pas déduire $\neg r$. Il est à noter pourtant que ces conclusions sont obtenues par l'application d'une seule règle incertaine : $p \Rightarrow q$.

La proposition r est en fait un évènement qui sort de l'ordinaire ou en tout cas de la règle statistique $p \Rightarrow q$. r est une autre explication de q. Les hypothèses de l'Assumption-based Truth Maintainance System (de Kleer, 1986a; de Kleer, 1986b; de Kleer, 1986c) et de la théorie des Hints (Kohlas & Monney, 1995) utilisent le concept d'anomalie dans la logique de la manière suivante, à l'aide de deux règles certaines :

- $p \Rightarrow q \vee a$: "si le gazon est mouillé, alors soit il a plu, soit nous sommes en présence d'une anomalie."
- $r \Rightarrow a$: "le fait que j'ai arrosé le jardin est une anomalie."

Nous entrons plus profondément dans les détails et la correspondance entre la théorie des Hints et le LFPR dans la sous-section 2.5.2.

Plutôt que d'insérer ces connaissances dans la logique en y ajoutant le concept mal défini d'anomalie [15], nous utilisons notre approche ensembliste sur les mondes possibles en introduisant le concept d'applicabilité.

On modélisera le problème précédent ainsi : certes la règle $r \Rightarrow p$ est toujours valide et donc fiable, mais elle n'est pas toujours applicable. En particulier, elle n'est pas applicable quand la règle $p \Rightarrow q$ est fiable :

$$A_{r \Rightarrow p} \cap F_{p \Rightarrow q} = \emptyset \tag{2.39}$$

De cette manière, le fait même d'avoir appliqué la règle $r \Rightarrow p$ nous informe sur le fait que le contexte global actuel exclut la fiabilité de la règle $p \Rightarrow q$. Et réciproquement, si $\neg q$ est constaté, on aura toujours la conclusion qu'il est probable que $\neg p$, mais on s'interdira de démontrer $\neg r$. Ce dernier point est complexe à bien saisir. En effet, la conclusion $\neg r$ sera fiable et donc valide mais elle n'apporte rien à la connaissance statistique a priori de la fréquence de r. Même si la conclusion $\neg r$ serait valide sur l'ensemble des mondes possibles, considérer ceci comme une démonstration nous amènerait à biaiser les calculs des probabilités inférieure et supérieure concernant la question r ou $\neg r$. Ce cas est l'exemple parfait où certaines règles peuvent être fiables sans pour autant être applicables. La sous-section suivante illustre encore un peu mieux le fonctionnement général de l'applicabilité sur un exemple concret.

15. A partir de quand décide-t-on qu'un évènement est une anomalie ? On peut également avoir envie d'utiliser deux règles statistiques $p \Rightarrow q$ et $p \Rightarrow r$. La théorie des Hints gère assez mal ce genre de règles en utilisant une seule règle : $p \Rightarrow q \vee r \vee a$ et en donnant des probabilité a priori pour q et r, ce qui n'est pas tout à fait équivalent.

2.3.7 Illustration

Pour illustrer le raisonnement par défaut, l'exemple de Tweety étant un oiseau qui ne vole pas parce que c'est un manchot est le plus connu. Mais pour nous replacer dans un contexte de dialogue, nous allons présenter un cas concret d'application de dialogue équivalent au problème Tweety.

Dans un service après-vente de connexion internet, on suppose qu'une règle statistique modélise le fait que les problèmes de connexion sont la plupart du temps consécutifs à un mauvais branchement des fils de la box internet. On suppose également que, dès que le système reçoit un appel, il teste automatiquement la ligne et que ce processus peut éventuellement produire une démonstration d'une autre explication du problème de connexion internet. Si le modèle était basé seulement sur des fiabilités (en omettant l'applicabilité), le système infèrerait que la box internet n'a probablement pas été correctement connectée, simplement sur la base d'un test de ligne qui aurait révélé un problème.

(I)	$\neg ligne \Rightarrow \neg internet$	$F_{\mathbf{I}}$	$A_{\mathbf{I}}$
(II)	$\neg internet \Rightarrow \neg boxconnectée$	$F_{\mathbf{II}}$	$A_{\mathbf{II}}$
(III)	$\neg ligne \Rightarrow \neg boxconnectée$	$F_{\mathbf{III}}$	$A_{\mathbf{III}}$

La règle **(I)** établit qu'un problème de ligne téléphonique entraîne un problème de connexion à internet. C'est une règle de définition qui est toujours fiable : $F_{\mathbf{I}} = W$. La règle **(II)** stipule que la majorité des problèmes de connexion sont causés par une erreur dans le branchement de la Livebox. C'est une règle fréquentielle, qui est fiable sur une partie non nulle des mondes, mais pas sur sa totalité. La démonstration **(III)** conclut qu'un problème de ligne téléphonique entraîne généralement que la box a été mal connectée. Elle a été inférée à partir des deux premières règles, et elle est en conséquence fiable sur un ensemble de mondes non vide :

$$F_{\mathbf{III}} = F_{\mathbf{I}} \cap F_{\mathbf{II}} \neq \emptyset \tag{2.40}$$

Cependant, chacun admettra que la démonstration **(III)** n'a aucune valeur dans un tel contexte. Simplement, le fait que la règle **(I)** ait été appliquée conditionne l'ensemble des mondes possibles. Si l'information de métaconnaissance $A_{\mathbf{I}} \cap F_{\mathbf{II}} = \emptyset$ est ajoutée, alors le support de la démonstration **(III)** est réduite à l'ensemble vide :

$$D_{\mathbf{III}} = F_{\mathbf{I}} \cap F_{\mathbf{II}} \cap A_{\mathbf{I}} \cap A_{\mathbf{II}} = \emptyset \tag{2.41}$$

Prenons le temps de préciser qu'il se peut très bien qu'il y ait une double raison à la panne de connexion internet et que l'utilisateur n'ait pas non plus bien connecté sa

box internet, mais le système logique n'a aucune raison de penser que c'est le cas, ou, autrement dit, il n'a aucune démonstration de cette proposition.

En tant que conclusion à l'exemple précédent, l'égalité $\mu_{glob}(D_{\mathbf{III}}) = 0$ est obtenue.

Cette section a transmis au lecteur les concepts et les outils nécessaires au calcul de la mesure de la probabilité contextuelle des démonstrations. Ces outils sont requis pour générer une évaluation numérique des probabilités contextuelles des propositions. Ceci est le sujet de la prochaine section.

2.4 Évaluation

A ce niveau, le système est supposé avoir une base de démonstrations saturée logiquement. Cependant, la véracité des propositions ne peut être directement calculée par le système. Comme la base de démonstration sera probablement plus ou moins contradictoire, suivant que le système a reçu ou non des informations contradictoires, le système nécessite un procédé d'évaluation pour mesurer la part de vrai et de faux et construire son opinion à propos d'une proposition p.

2.4.1 Séparation de l'ensemble des mondes en quatre parties

La figure 2.3 montre à l'aide de deux vues que l'ensemble des mondes à considérer se distribuent en quatre classes à partir du moment où la question p se pose :

- La zone d'ignorance : zone où ni p ni $\neg p$ n'ont pu être démontrés.
- Les deux zones de détermination :
  - La zone où seule p (ou une proposition subsumant p) a été démontrée. Dans ce cas, on peut en conclure que p est vrai.
  - La zone où seule $\neg p$ (ou une proposition subsumant $\neg p$) a été démontrée. Dans ce cas, on peut en conclure que p est faux.
- La zone de confusion : zone où p et $\neg p$ ont tous les deux été démontrés. Dans ce cas, on peut en déduire l'inconsistance $\perp$ et donc que l'on se trouve dans le monde impossible.

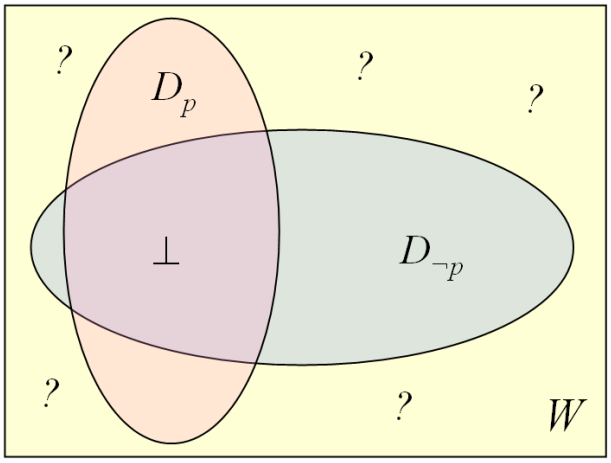
La probabilité inférieure d'une proposition p est la mesure de l'ensemble des mondes où p a été démontré et où $\neg p$ n'a pas été démontrée dans le contexte global C_{glob}. C'est équivalent à la mesure contextuelle (par rapport au contexte C_{glob}) des mondes où p a été démontré :

$$P_{inf}(p) = \mu_{glob}(D_p) = \frac{\mu(D_p \cap C_{glob})}{\mu(C_{glob})} \tag{2.42}$$

La probabilité supérieure d'une proposition p est la mesure de l'ensemble des mondes où $\neg p$ n'a pas été démontrée dans le contexte global C_{glob}. C'est équivalent à la mesure

	D_p	$\overline{D_p}$
$D_{\neg p}$	$\bot$	$\neg p$
$\overline{D_{\neg p}}$	p	???

(a) Vue à l'aide du tableau des démonstrations possibles.



(b) Vue graphique de l'ensemble des mondes.

FIGURE 2.3 – La séparation des ensembles de monde en quatre.

contextuelle (par rapport au contexte C_{glob}) des mondes où $\neg p$ n'a pas été démontré :

$$P_{sup}(p) = \mu_{glob}(\overline{D_{\neg p}}) = \frac{\mu(\overline{D_{\neg p}} \cap C_{glob})}{\mu(C_{glob})} \tag{2.43}$$

Comme $\overline{D_{\neg p}} \cap \overline{D_\bot} \supseteq D_p \cap \overline{D_\bot}$, $P_{sup} \geq P_{inf}$ est valide. Il peut aussi être démontré que P_{inf} et P_{sup} sont duaux :

$$P_{inf}(p) + P_{sup}(\neg p) = \frac{\mu(D_p \cap C_{glob}) + \mu(\overline{D_p} \cap C_{glob})}{\mu(C_{glob})} = \frac{\mu(C_{glob})}{\mu(C_{glob})} = 1 \tag{2.44}$$

2.4.2 Simplifications

Malheureusement, il n'y a pas à disposition suffisamment de moyens pour instancier les différents ensembles qui sont manipulés dans les démonstrations, et même si cela était possible, nous rencontrerions de grosses difficultés à calculer les intersections et les unions. Généralement, le mieux qui puisse être fait est d'estimer les mesures objectives (par opposition à contextuelles) des fiabilités des démonstrations : $\mu(F_i) = f_i$, et de déclarer certaines relations entre les ensembles caractéristiques des démonstrations : par exemple, $F_i \cap A_j = \emptyset$. De manière à rendre les probabilités inférieure et supérieure calculables, les relations suivantes sont considérées par défaut :

– Sans information complémentaire, si le contexte global C_{glob} implique l'utilisation d'une démonstration d avec une applicabilité A_d, alors $C_{glob} \subset A_d$ est vérifié. En d'autre mots, si le contexte implique l'application syntaxique d'une règle, alors, par défaut, l'applicabilité de la démonstration est garantie.

– Les ensembles atomiques sont les ensembles qui ne sont pas composés. Typiquement, les contextes impliqués par les évènements et les fiabilités et applicabilités des hypothèses sont les ensembles atomiques que nous considérons. Sans information complémentaire, les ensembles atomiques sont considérés indépendants entre eux. En conséquence, si les G_i sont des ensembles atomiques sans relation connue entre eux, alors la relation suivante est supposée :

$$\mu\Big(\bigcap_i G_i\Big) = \prod_i \mu(G_i) \tag{2.45}$$

Le premier de ces deux points est indispensable, dans le mesure où l'applicabilité correspond aux contraintes sur le support de démonstration qui ne peut être observée. La fiabilité est à l'inverse la partie "visible" ou plutôt mesurable du support de démonstration.

Au contraire, la considération d'indépendance entre les fiabilités n'est pas obligatoire. En effet, de façon alternative, on peut considérer que les fiabilités sont le plus chevauchant possible, c'est à dire que si deux fiabilités sont comparées, il y en a toujours une qui est incluse dans l'autre. Dans ce cas, on se retrouve avec les résultats de la logique floue (Zadeh, 1978; Bouchon-Meunier, 1995; Bouchon-Meunier & Nguyen, 1996; Bouchon-Meunier, 2007). A l'inverse, si l'on considère que les différentes fiabilités s'excluent au maximum, on retrouve la logique probabiliste de Nilsson (1986). L'indépendance que nous considérons nous amène, quant à elle, aux résultats de la théorie de l'Évidence de Dempster (1968).

2.4.3 Algorithme d'évaluation

La formule 2.42 doit être calculée numériquement. Le but de l'algorithme 1 est de simplifier les numérateur et dénominateur de manière à rendre ce calcul numérique possible.

La première étape de l'algorithme est le développement du numérateur et du dénominateur en ensembles atomiques. Les relations connues entre les ensembles sont suffisantes pour mener cela à bien.

$$A \cup B = \overline{\overline{A} \cap \overline{B}} \tag{2.46}$$

$$\mu(\overline{A} \cap B) = \mu(B) - \mu(A \cap B) \tag{2.47}$$

Pour la probabilité inférieure, le calcul se présente de cette façon :

$$P_{inf}(p) = \frac{\mu(D_p \cap C_{glob})}{\mu(C_{glob})} \tag{2.48}$$

$$P_{inf}(p) = \frac{\mu(D_p \cap \overline{D_\perp} \cap \bigcap_e C_e)}{\mu(\overline{D_\perp} \cap \bigcap_e C_e)} \tag{2.49}$$

Grâce à la relation 2.47, on obtient :

$$P_{inf}(p) = \frac{\mu(D_p \cap \bigcap_e C_e) - \mu(D_\perp \cap \bigcap_e C_e)}{\mu(\bigcap_e C_e) - \mu(D_\perp \cap \bigcap_e C_e)} \tag{2.50}$$

Intéressons-nous maintenant au calcul de $D_\bullet$ ($\bullet$ prenant la valeur de p ou $\perp$). C'est le support de l'ensemble des démonstrations $\mathscr{D}_\bullet = \{d_i\}_{i\in[1,n]}$ de la proposition $\bullet$. Pour avoir une démonstration de $\bullet$, il suffit d'en avoir au moins une. En conséquence, $D_\bullet$ est l'union des supports de démonstration des éléments de $\mathscr{D}_\bullet$. De plus, chaque démonstration d_i n'est démontrée que si tous ses hypothèses [16] $\mathscr{H}_{d_i} = \{d_{ij}\}_{j\in[1,n_i]}$ le sont. Grâce à la relation 2.46, nous arrivons à la formule suivante :

$$D_\bullet = \bigcup_{d_i\in\mathscr{D}_\bullet} \bigcap_{d_{ij}\in\mathscr{H}_{d_i}} D_{ij} \tag{2.51}$$

$$D_\bullet = \overline{\bigcap_{d_i\in\mathscr{D}_\bullet} \overline{\bigcap_{d_{ij}\in\mathscr{H}_{d_i}} D_{ij}}} \tag{2.52}$$

En passant à la formule de la mesure et en utilisant de façon répétée la relation 2.47, nous arrivons au résultat suivant :

$$\mu(D_\bullet \cap \bigcap_e C_e) = \mu(\overline{\bigcap_{d_i\in\mathscr{D}_\bullet} \overline{\bigcap_{d_{ij}\in\mathscr{H}_{d_i}} D_{ij}}} \cap \bigcap_e C_e) \tag{2.53}$$

$$\mu(D_\bullet \cap \bigcap_e C_e) = \mu(\bigcap_e C_e) - \mu(\bigcap_{d_i\in\mathscr{D}_\bullet} \overline{\bigcap_{d_{ij}\in\mathscr{H}_{d_i}} D_{ij}} \cap \bigcap_e C_e) \tag{2.54}$$

$$\mu(D_\bullet \cap \bigcap_e C_e) = \mu(\bigcap_e C_e) - \sum_{\mathscr{D}\subseteq\mathscr{D}_\bullet} (-1)^{|\mathscr{D}|} \mu(\bigcap_{d_i\in\mathscr{D}} \bigcap_{d_{ij}\in\mathscr{H}_{d_i}} D_{ij} \cap \bigcap_e C_e) \tag{2.55}$$

$$\mu(D_\bullet \cap \bigcap_e C_e) = \mu(\bigcap_e C_e) - \sum_{\cap_i\in\mathscr{I}_\bullet} \pm\mu(\cap_i \cap \bigcap_e C_e) \tag{2.56}$$

Où $\mathscr{I}_\bullet$ est l'ensemble d'intersections d'ensembles atomiques définis par la formule suivante :

$$\sum_{\cap_i\in\mathscr{I}_\bullet} \pm\mu(\cap_i \cap \bigcap_e C_e) = \sum_{\mathscr{D}\subseteq\mathscr{D}_\bullet} (-1)^{|\mathscr{D}|} \mu(\bigcap_{d_i\in\mathscr{D}} \bigcap_{d_{ij}\in\mathscr{H}_{d_i}} A_{ij} \cap F_{ij} \cap \bigcap_e C_e) \tag{2.57}$$

Nous obtenons ainsi des numérateur et dénominateur sous une forme de somme (ou différences) de mesures d'intersection d'ensembles atomiques que nous notons de façon

16. Nous appelons "hypothèses" d'une démonstration d ses ancêtres qui ont permis de déduire d qui n'ont pas de parents. C'est à dire l'ensemble des démonstrations nécessaires à l'obtention de d qui ne sont pas le fruit d'une inférence, mais d'informations brutes : règles incertaines définies lors des spécifications ou observations de l'environnement par des capteurs. Ce sont les hypothèses qui ont été effectuées pour arriver à la démonstration d en question

générique $\cap_i$:

$$P_{inf}(p) = \frac{\sum_{\cap_i \in \mathscr{I}_\perp} \pm\mu(\cap_i \cap \bigcap_e C_e) - \sum_{\cap_i \in \mathscr{I}_p} \pm\mu(\cap_i \cap \bigcap_e C_e)}{\sum_{\cap_i \in \mathscr{I}_\perp} \pm\mu(\cap_i \cap \bigcap_e C_e)} \tag{2.58}$$

La seconde étape traite les incompatibilité qui apparaissent dans les intersections $\cap_i$. Par exemple si l'on sait que deux ensembles A et B sont disjoints, alors $\cap_i = \emptyset$ pour toute intersection de la forme : $\cap_i = A \cap B \cap \ldots$. On en déduit directement que $\mu(\cap_i \cap \bigcap_e C_e) = 0$.

La formule 2.58 devient donc :

$$P_{inf}(p) = \frac{\sum_{\cap_i \in \mathscr{I}^*_\perp} \pm\mu(\cap_i \cap \bigcap_e C_e) - \sum_{\cap_i \in \mathscr{I}^*_p} \pm\mu(\cap_i \cap \bigcap_e C_e)}{\sum_{\cap_i \in \mathscr{I}^*_\perp} \pm\mu(\cap_i \cap \bigcap_e C_e)} \tag{2.59}$$

Où $\mathscr{I}^*_p$ et $\mathscr{I}^*_\perp$ sont, respectivement pour p et $\perp$, les ensembles d'intersections restant à la fin de la seconde étape.

La troisième étape fait la simplification des applicabilités restantes, sur lesquelles aucune information à propos de la mesure n'est connue. Le fait que $C_{glob} \subseteq A_d$ est utilisé pour les retirer de chaque intersection $\cap_i \cap \bigcap_e C_e$, comme expliqué dans la sous-section précédente. Les intersections $\cap_i$ sont composées maintenant exclusivement de fiabilités. De même les redondances de fiabilités (intersection d'un ensemble par lui-même) sont supprimées :

$$\cap_i = \bigcap_d F_d \tag{2.60}$$

En nommant I l'itérateur des intersections de $\mathscr{I}^*_\bullet$, nous réécrivons la formule 2.59 de la manière suivante :

$$P_{inf}(p) = \frac{\sum_{I \in \mathscr{I}^*_\perp} \pm\mu(\bigcap_{d \in I} F_d \cap \bigcap_e C_e) - \sum_{I \in \mathscr{I}^*_p} \pm\mu(\bigcap_{d \in I} F_d \cap \bigcap_e C_e)}{\sum_{I \in \mathscr{I}^*_\perp} \pm\mu(\bigcap_{d \in I} F_d \cap \bigcap_e C_e)} \tag{2.61}$$

Dans la quatrième étape, comme expliqué dans la sous-section 2.4.2, les ensembles atomiques dans chaque intersection $\cap_i$ sont considérés comme indépendants [17]. Ceci nous

17. De façon alternative, comme nous l'avons vu également dans la sous-section 2.4.2, on peut choisir d'autres modes de résolution que l'indépendance pour retrouver la théorie de combinaison des informations souhaitée. Quoi qu'il en soit l'objectif de cette quatrième étape est de sortir l'intersection de la mesure de probabilité pour simplifier la fraction par le facteur $\prod_e \mu(C_e)$ et obtenir une formule basée sur des mesures de fiabilités de démonstration atomiques.

Algorithm 1 Algorithme du calcul de la probabilité inférieure d'une proposition p dans le cadre formel LFPR

Le moteur d'inférence est saturé.
{Extraction des diverses informations de la base}
Extraction de l'ensemble $\mathscr{D}_p$ des démonstrations dont le contenu propositionnel subsume p.
Extraction de l'ensemble $\mathscr{D}_\perp$ des démonstrations dont le contenu propositionnel est $\perp$.
for all Démonstrations $d_i \in \mathscr{D}_p \cup \mathscr{D}_\perp$ **do**
  Extraction de toutes ses démonstrations hypothèses $\mathscr{H}_{d_i} = \{d_{ij}\}_{j\in[1,n_i]}$
end for
{Développement et réduction de la disjonction des démonstrations de $\mathscr{D}_p$ et $\mathscr{D}_\perp$}
Génération des $2^{\mathscr{D}_p}$ intersections $\cap_i$ pour $\mathscr{D}_p$: $\mathscr{I}_p$
Génération des $2^{\mathscr{D}_\perp}$ intersections $\cap_i$ pour $\mathscr{D}_\perp$: $\mathscr{I}_\perp$
for all Intersections $I \in \mathscr{I}_p \cup \mathscr{I}_\perp$ **do**
  Suppression des doublons de démonstration dans I
  Suppression des éléments de l'intersection I s'il comprend deux ensembles disjoints
  Suppression des applicabilités de l'intersection I
end for
Les ensembles d'intersections ainsi obtenus sont renommés $\mathscr{I}_p^*$ et $\mathscr{I}_\perp^*$
{Calcul numérique à l'aide de la formule 2.64}
for all Intersections $I \in \mathscr{I}_p^* \cup \mathscr{I}_\perp^*$ **do**
  Calcul du produit des fiabilités
end for
Calculs de $\sum_{I\in\mathscr{I}_p^*} \pm \prod_{d\in I} f_d$ et $\sum_{I\in\mathscr{I}_\perp^*} \pm \prod_{d\in I} f_d$
Calcul de P_{inf}

permet d'exécuter une simplification de la fraction par les facteurs contextuels qui apparaissent au dénominateur et au numérateur :

$$P_{inf}(p) = \frac{\sum_{I\in\mathscr{I}_\perp^*} \pm \prod_{d\in I} \mu(F_d) \times \prod_e \mu(C_e) - \sum_{I\in\mathscr{I}_p^*} \pm \prod_{d\in I} \mu(F_d) \times \prod_e \mu(C_e)}{\sum_{I\in\mathscr{I}_\perp^*} \pm \prod_{d\in I} \mu(F_d) \times \prod_e \mu(C_e)} \tag{2.62}$$

$$P_{inf}(p) = \frac{\sum_{I\in\mathscr{I}_\perp^*} \pm \prod_{d\in I} \mu(F_d) - \sum_{I\in\mathscr{I}_p^*} \pm \prod_{d\in I} \mu(F_d)}{\sum_{I\in\mathscr{I}_\perp^*} \pm \prod_{d\in I} \mu(F_d)} \tag{2.63}$$

Pour la cinquième étape, seules des mesures atomiques de fiabilité restent et leurs

mesures sont données par les règles ou observations qui les ont générées, ce qui fait que l'on peut calculer numériquement la probabilité :

$$P_{inf}(p) = \frac{\sum\limits_{I \in \mathscr{I}^*_\perp} \pm \prod\limits_{d \in I} f_d - \sum\limits_{I \in \mathscr{I}^*_p} \pm \prod\limits_{d \in I} f_d}{\sum\limits_{I \in \mathscr{I}^*_\perp} \pm \prod\limits_{d \in I} f_d} \tag{2.64}$$

L'algorithme 1 résume les diverses étapes du calcul de P_{inf} avec une factorisation des calculs et simplifications un peu différentes. Le calcul de P_{sup} est similaire en partant de la formule 2.43 :

$$P_{sup}(p) = \frac{\mu(\overline{D_{\neg p}} \cap C_{glob})}{\mu(C_{glob})} \tag{2.65}$$

$$P_{sup}(p) = \frac{\mu(\overline{D_{\neg p}} \cap \overline{D_\perp} \cap \bigcap_e C_e)}{\mu(\bigcap_e C_e) - \mu(D_\perp \cap \bigcap_e C_e))} \tag{2.66}$$

$$P_{sup}(p) = \frac{\mu(\overline{D_{\neg p}} \cap \bigcap_e C_e)}{\mu(\bigcap_e C_e) - \mu(D_\perp \cap \bigcap_e C_e))} \tag{2.67}$$

$$P_{sup}(p) = \frac{\mu(\bigcap_e C_e) - \mu(D_{\neg p} \cap \bigcap_e C_e)}{\mu(\bigcap_e C_e) - \mu(D_\perp \cap \bigcap_e C_e))} \tag{2.68}$$

$$P_{sup}(p) = \frac{\sum\limits_{\cap_i \in \mathscr{I}^*_{\neg p}} \pm \mu(\cap_i \cap \bigcap\limits_e C_e)}{\sum\limits_{\cap_i \in \mathscr{I}^*_\perp} \pm \mu(\cap_i \cap \bigcap\limits_e C_e)} \tag{2.69}$$

$$P_{sup}(p) = \frac{\sum\limits_{I \in \mathscr{I}^*_{\neg p}} \pm \mu(\bigcap\limits_{d \in I} F_d \cap \bigcap\limits_e C_e)}{\sum\limits_{I \in \mathscr{I}^*_\perp} \pm \mu(\bigcap\limits_{d \in I} F_d \cap \bigcap\limits_e C_e)} \tag{2.70}$$

$$P_{sup}(p) = \frac{\sum\limits_{I \in \mathscr{I}^*_{\neg p}} \pm \prod\limits_{d \in I} \mu(F_d)}{\sum\limits_{I \in \mathscr{I}^*_\perp} \pm \prod\limits_{d \in I} \mu(F_d)} \tag{2.71}$$

$$P_{sup}(p) = \frac{\sum\limits_{I \in \mathscr{I}^*_{\neg p}} \pm \prod\limits_{d \in I} f_d}{\sum\limits_{I \in \mathscr{I}^*_\perp} \pm \prod\limits_{d \in I} f_d} \tag{2.72}$$

Les transitions entre chaque ligne du calcul sont directes une fois que le calcul de P_{inf} bien maîtrisé. De même, l'algorithme 1 pour le calcul de P_{inf} est très facilement adapté

au calcul de P_sup en remplaçant p par $\neg p$ et P_{inf} par P_{sup}, et bien sûr, en utilisant la formule 2.72 à la place de la formule 2.64.

Ces algorithmes ont été implantés dans un module de raisonnement probabiliste capable d'effectuer des inférences logiques telles que le modus ponens ou la conception de règles ad hoc, mais également d'accepter des déductions de sources externes.

2.4.4 Exemple

L'exemple introduit au début de la section est repris maintenant. Pour montrer la simplicité du calcul, nous ajoutons même une nouvelle information au système : il y a un problème avec la ligne (obtenu grâce au test du Back-Office) et la box pourrait ne pas être connectée (l'utilisateur a dit qu'il n'était pas sûr de ses connexions). Nous obtenons donc les hypothèses suivantes :

(I)	$\neg ligne \Rightarrow \neg internet$	$F_{\mathbf{I}} = W$	$A_{\mathbf{I}}$
(II)	$\neg internet \Rightarrow \neg boxconnectée$	$F_{\mathbf{II}}$	$A_{\mathbf{II}}$
(III)	$\neg ligne$	$F_{\mathbf{III}}$	$A_{\mathbf{III}}$
(IV)	$\neg boxconnectée$	$F_{\mathbf{IV}}$	$A_{\mathbf{IV}}$

La métaconnaissance suivante est toujours considérée : $A_{\mathbf{I}} \cap F_{\mathbf{II}} = \emptyset$. Les inférences suivantes sont générées :

(V) = **(I)** + **(II)**	$\neg ligne \Rightarrow \neg boxconnectée$	$F_{\mathbf{V}}$	$A_{\mathbf{V}}$
(VI) = **(I)** + **(III)**	$\neg internet$	$F_{\mathbf{VI}}$	$A_{\mathbf{VI}}$
(VII) = **(VI)** + **(II)**	$\neg boxconnectée$	$F_{\mathbf{VII}}$	$A_{\mathbf{VII}}$
(VIII) = **(V)** + **(III)**	$\neg boxconnectée$	$F_{\mathbf{VIII}}$	$A_{\mathbf{VIII}}$

Le but de l'exemple est de montrer l'évaluation des probabilités inférieure et supérieure de la proposition *boxconnectée*, qui sera notée *cb* dans la suite de la section dans un but de concision. Maintenant que le moteur d'inférence est saturé, nous pouvons passer aux extractions des diverses informations de la base, notamment les ensembles de démonstration de *cb*, $\neg cb$ et $\perp$ que nous notons respectivement $\mathscr{D}_{cb}$, $\mathscr{D}_{\neg cb}$ et $\mathscr{D}_{\perp}$:

$$\mathscr{D}_{cb} = \emptyset \tag{2.73}$$

$$\mathscr{D}_{\neg cb} = \{\mathbf{(IV)}, \mathbf{(VII)}, \mathbf{(VIII)}\} \tag{2.74}$$

$$\mathscr{D}_{\perp} = \emptyset \tag{2.75}$$

Comme il n'y a aucune démonstration de *cb*, le calcul de $P_{inf}(cb)$ est direct :

$$P_{inf}(cb) = \frac{\mu(D_{cb} \cap C_{glob})}{\mu(C_{glob})} = \frac{\mu(\emptyset)}{\mu(C_{glob})} = 0 \tag{2.76}$$

Le calcul de la probabilité supérieure est plus intéressante. Il existe trois démonstrations différentes de $\neg cb$: **(IV)**, **(VII)** et **(VIII)**. Intuitivement, les démonstrations **(VII)** et **(VIII)** sont identiques et invalides à cause de l'incompatibilité entre $A_{\mathbf{I}}$ et $F_{\mathbf{II}}$. C'est ce que nous allons retrouver en appliquant l'algorithme.

Comme il n'y a aucune démonstration de $\bot$, nous nous intéresserons exclusivement aux ensembles $\mathscr{D}_{\neg cb}$, $\mathscr{I}_{\neg cb}$ et $\mathscr{I}^*_{\neg cb}$. Extrayons maintenant pour chaque $d_i \in \mathscr{D}_{\neg cb}$ l'ensemble $\mathscr{H}_{d_i}$ de ses démonstrations hypothèses :

$$\mathscr{D}_{\neg cb} \begin{cases} \mathbf{(IV)} : \mathscr{H}_{\mathbf{(IV)}} = \{\mathbf{(IV)}\} \\ \mathbf{(VII)} : \mathscr{H}_{\mathbf{(VII)}} = \{\mathbf{(I)}, \mathbf{(II)}, \mathbf{(III)}\} \\ \mathbf{(VIII)} : \mathscr{H}_{\mathbf{(VIII)}} = \{\mathbf{(I)}, \mathbf{(II)}, \mathbf{(III)}\} \end{cases} \tag{2.77}$$

Les démonstrations **(VII)** et **(VIII)** étant équivalentes (elles ont été obtenues en appliquant les mêmes règles dans un ordre différent). L'une des deux peut être supprimée [18] : **(VIII)**. $\mathscr{I}_{\neg cb}$ est donc le produit croisé simple de deux démonstrations :

$$\mathscr{I}_{\neg cb} \begin{cases} \pm\cap_1 = +W \\ \pm\cap_2 = -A_{\mathbf{(IV)}} \cap F_{\mathbf{(IV)}} \\ \pm\cap_3 = -A_{\mathbf{(I)}} \cap F_{\mathbf{(I)}} \cap A_{\mathbf{(II)}} \cap F_{\mathbf{(II)}} \cap A_{\mathbf{(III)}} \cap F_{\mathbf{(III)}} \\ \pm\cap_4 = +A_{\mathbf{(I)}} \cap F_{\mathbf{(I)}} \cap A_{\mathbf{(II)}} \cap F_{\mathbf{(II)}} \cap A_{\mathbf{(III)}} \cap F_{\mathbf{(III)}} \cap A_{\mathbf{(IV)}} \cap F_{\mathbf{(IV)}} \end{cases} \tag{2.78}$$

A cause de la métaconnaissance $A_{\mathbf{I}} \cap F_{\mathbf{II}} = \emptyset$, $\cap_3$ et $\cap_4$ sont vides. On obtient donc pour $\mathscr{I}^*_{\neg cb}$:

$$\mathscr{I}^*_{\neg cb} \begin{cases} \pm\cap_1 = +W \\ \pm\cap_2 = -F_{\mathbf{(IV)}} \end{cases} \tag{2.79}$$

On obtient donc au final :

$$\sum_{I \in \mathscr{I}^*_{\neg p}} \pm \prod_{d \in I} f_d = 1 - f_{\mathbf{(IV)}} \tag{2.80}$$

$$\sum_{I \in \mathscr{I}^*_{\bot}} \pm \prod_{d \in I} f_d = 1 \tag{2.81}$$

$$P_{sup}(p) = \frac{\displaystyle\sum_{I \in \mathscr{I}^*_{\neg p}} \pm \prod_{d \in I} f_d}{\displaystyle\sum_{I \in \mathscr{I}^*_{\bot}} \pm \prod_{d \in I} f_d} \tag{2.82}$$

$$P_{sup}(p) = 1 - f_{\mathbf{(IV)}} \tag{2.83}$$

18. L'algorithme 1 ne prévoit pas ce genre de simplification parce que ce cas est traité dans le moteur d'inférence à l'aide de la règle de subsomption 2.25.

2.5 Comparaison avec l'état-de-l'art

2.5.1 La Théorie de l'Évidence

Le problème traité par la Théorie de l'Évidence (voir la sous-section 2.2.3) consiste à évaluer la véracité de propositions logiques étant donné un ensemble d'observations incertaines (capteurs, avis d'experts, règles probabilistes, ...). La Théorie de l'Évidence (Dempster, 1968; Shafer, 1976) n'a aucun équivalent du concept d'applicabilité. C'est la raison pour laquelle Pearl (1988) a fait la critique du cadre formel en montrant qu'il était incapable de traiter des règles par défaut. Si l'on omet le concept d'applicabilité, il y a une équivalence entre la Théorie de l'Évidence et le LFPR.

Le concept de démonstration du LFPR n'est rien d'autre qu'une répartition de masse où l'ensemble $E \subseteq \Omega$ correspond à la proposition p (la proposition p décrit un sous-ensemble de Ω), et où la masse $m(E)$ correspond à la mesure de la fiabilité de la démonstration $\mu(F_d)$. De façon analogue, chaque répartition de masse m peut être émulée dans le LFPR en suivant la procédure suivante : pour chaque ensemble $E \subseteq W$ tel que $m(E) \neq 0$, une démonstration d_E doit être créée dans le LFPR, de telle façon que la proposition p_E qu'il démontre soit la description logique de E, que sa fiabilité F_E ait une mesure égale à $m(E)$, que les fiabilités F_E générées de cette manière avec la répartition de masse m soient toutes disjointes deux à deux et enfin que les applicabilités A_E des démonstrations soient toutes égales à W.

L'approche de la Théorie de l'Évidence se concentre sur la répartition des masses alors que l'approche du LFPR est plus centrée sur la logique et les inférences. Ainsi, alors que la mesure μ du LFPR est constante et ne dépend pas des différentes observations, la répartition des masses m de la Théorie de l'Évidence est mise à jour suite à chaque nouvelle observation et la fonction de croyance en résultant n'est ni plus ni moins que la mesure contextuelle μ_{glob}. De plus, la Théorie de l'Évidence ne permet pas de conserver symboliquement les dépendances entre les différentes connaissances, et c'est ce qui nous a empêché de l'appliquer directement.

2.5.2 La Théorie des Hints

L'idée principale de Kleer quand il a inventé l'Assumption-based Truth Maintainance System (ATMS, de Kleer 1986a; 1986b; 1986c) était de permettre de raisonner sur la base de suppositions. Ces suppositions de l'ATMS correspondent à la notion d'hypothèse dans le LFPR. Le rôle de l'ATMS est de calculer l'ensemble de suppositions nécessaires pour obtenir un nœud donné, qui est libellé par une proposition. Cet ensemble est en fait l'union des nœuds libellé par la proposition. Et pour satisfaire un nœud, il est nécessaire de

satisfaire tous les suppositions qui ont permis d'obtenir le nœud. En conclusion, l'ensemble des suppositions prend la forme d'une union d'intersection. La partie union montre que la proposition du nœud peut être inférée à partir de raisonnements différents et la partie intersection provient de la combinaison des suppositions nécessaire pour un raisonnement. On retrouve beaucoup de similarité avec l'approche du LFPR, mais nous allons plutôt nous intéresser à l'analogie entre le LFPR et la Théorie des Hints de Kohlas et Monney qui est en fait une extension de l'ATMS de Kleer.

Le sujet de la Théorie des Hints (Kohlas & Monney, 1995) est de fusionner la Théorie de l'Évidence avec l'ATMS. L'idée principale a été d'insérer les probabilités dans le concept de supposition de l'ATMS. Dans la Théorie des Hints, l'exemple de la section précédente devient :

(I)	$\neg ligne \Rightarrow \neg internet$
(II)	$\neg internet \Rightarrow \neg boxconnectée \vee a_1$
(III)	$\neg ligne \vee a_2$
(IV)	$\neg boxconnectée \vee a_3$
(V)	$\neg ligne \Rightarrow a_1$

La règle **(V)** dit simplement que $\neg ligne$ fait partie des exceptions de la règle **(II)**. Les règles générées sont alors les suivantes :

(VI) = **(I)** + **(II)**	$\neg ligne \Rightarrow \neg boxconnectée \vee a_1$
(VII) = **(VI)** + **(III)**	$\neg boxconnectée \vee a_1 \vee a_2$
(VIII) = **(III)** + **(V)**	$a_1 \vee a_2$
(IX) = **(I)** + **(III)**	$\neg internet \vee a_2$
(X) = **(IX)** + **(II)**	$\neg boxconnectée \vee a_1 \vee a_2$

Nous retrouvons finalement que **(VIII)** subsume **(VII)** et **(X)**. Le rôle de l'applicabilité est le même que celui porté par la règle **(V)** : modéliser le contexte qui est impliqué par l'utilisation de la règle **(I)**. Quand le LFPR contraint le contexte avec de la méta-connaissance, la Théorie des Hints le fait à travers la logique, et finalement les deux arrivent au même résultat. La différence entre les deux théories concerne principalement l'approche et la complexité : dans la Théorie des Hints, la logique est surchargée par les anomalies, alors que dans le LFPR, il suffit de mémoriser les hypothèses qui ont été utilisées pour aboutir à chaque démonstration. En conséquence, la complexité à la requête est plus grande dans le LFPR, parce que le calcul des dépendances n'a pas été réalisée pendant l'inférence logique, contrairement à la Théorie des Hints. L'intérêt de l'approche

du LFPR est de ne calculer les dépendances complexes que quand cela est requis. Le résultat est qu'il n'y a presque aucune complexité additionnelle entre un moteur d'inférence classique et le moteur du LFPR pendant la génération de démonstrations. Au final, le seul surcoût de complexité provient des tests de subsomption qui sont un peu plus complexes dans le LFPR que dans un moteur basique sans capacités de gestion des incertitudes, mais ce surcoût est largement contrebalancé par la simplification de la description logique.

2.6 Transformation de connaissances indépendantes en connaissances exclusives

Cette section utilise le Logical Framework for Probabilistic Reasoning pour résoudre un premier problème théorique qui se pose dans les systèmes de dialogue. Ce problème est le calcul de la formule de transformation entre un ensemble de faits indépendants et incompatible en un ensemble de faits exclusifs. L'idée est que la reconnaissance vocale produit des scores de confiance sous forme de probabilité. Mais un score de confiance pour un énoncé s_1 et un score de confiance pour un énoncé s_2 sont obtenus de façon indépendante par le module de reconnaissance vocale, ce qui implique entre autres que l'on n'a aucune garantie que la somme des probabilités affectées aux deux énoncés soit inférieur à 1. Pourtant, l'énoncé ne peut être à la fois s_1 et s_2. L'intérêt de cette étude est de recalculer les probabilités exclusives (c'est à dire dont la somme vaut au maximum 1) des énoncés étant donné les scores obtenus de façon indépendante. La sous-section 1.2.3 fournit des détails supplémentaires sur le contexte de cette étude.

En l'absence de connaissance sur les applicabilité des différentes connaissances, nous avons fait la seconde hypothèse de la sous-section 2.4.2 : $C_{glob} \in A_d$. Cette considération permet de simplifier énormément les calculs et s'abstraire des notions de contexte et d'applicabilité dans la suite de cette section.

2.6.1 Les notations pour la problématique de la reconnaissance vocale

Pour la problématique de la reconnaissance vocale, nous supposons que ce module fournit au système une liste N-best dont les probabilités sont indépendantes, ce qui signifie que les règles qui ont permis d'inférer ces différents énoncés sont toutes différentes et indépendantes elles-même. C'est une hypothèse forte, mais la plus raisonnable. En effet, s'il existe une dépendance quelconque dans le N-best, ce qui est plus que probable, il est impossible de la connaître. En conséquence, comme dans la théorie de Shannon (1948), la meilleure approximation correspond au maximum de l'entropie, ce qui corres-

pond également à l'hypothèse d'indépendance entre les informations. Ainsi, le résultat de la reconnaissance vocale est une liste de longueur N de binômes constitués de l'énoncé et de sa probabilité :

$$ASRresult = \{s_k, F_k\}_{k\in[0,n-1]} \tag{2.84}$$

Où F_k est la fiabilité du formalisme LFPR et $\mu(F_k) = p_k$ la probabilité que l'énoncé s_k soit la transcription du signal. Nous connaissons donc la relation d'indépendance entre les différents F_k :

$$\forall i \neq j, \mu(F_i \cap F_j) = \mu(F_i)\,\mu(F_j) \tag{2.85}$$

Le but de cette étude est d'obtenir l'état de connaissance, c'est-à-dire la même partition de probabilité, avec des informations exclusives, c'est-à-dire dont les fiabilités F_k^* sont disjointes :

$$\forall i \neq j, F_i^* \cap F_j^* = \emptyset \tag{2.86}$$

Nous associons le terme p_i^* à la probabilité de l'information exclusive liée à l'énoncé s_i (que nous appellerons information i). De même pour chaque notation a_i dans le système indépendant, il y a une notation équivalente dans le système exclusif : a_i^*. La notation a_i° se rapporte à une notation ou l'autre. Le but du calcul est de définir l'ensemble $p^*_{i\,k\in[0,n-1]}$ tel que les relations suivantes soient respectées :

$$\forall i, P_{inf}(i) = P^*_{inf}(i) \tag{2.87}$$

$$\forall i, P_{sup}(i) = P^*_{sup}(i) \tag{2.88}$$

En fait, la formule 2.44 démontre que les égalités 2.88 sont une conséquence directe des égalités 2.87 (et inversement). Ainsi, dans la suite, nous ne nous intéresserons qu'à $P^\circ_{sup}(i)$.

2.6.2 Résolution de la problématique

Le calcul de la probabilité est réalisé selon la formule 2.42 :

$$P^\circ_{inf}(i) = \frac{\mu\left(F_i^\circ \cap \overline{\bigcup_{j\neq i} F_j^\circ}\right)}{\mu\left(\overline{F_\perp^\circ}\right)} \tag{2.89}$$

Où $F_\perp^\circ$ est l'ensemble des mondes inconsistants où le terme $\perp$ a été démontré, c'est-à-dire où au moins deux énoncés s_i et s_j sont démontrés. Le calcul de $F_\perp$ est complexe, et il n'est pas nécessaire, puisqu'on peut également le calculer par normalisation. $F_\perp^*$, en revanche est très simple à calculer, puisque tous les énoncés sont exclusifs dans ce système,

il ne peut jamais y avoir deux énoncés s_i et s_j démontrés dans le même monde, et nous arrivons en conséquence à la formule suivante :

$$F_{\perp}^{*} = \emptyset \tag{2.90}$$

$$\mu\left(\overline{F_{\perp}^{*}}\right) = 1 \tag{2.91}$$

De même, $P_{inf}^{*}(i)$ est très simple à calculer, toujours grâce à sa propriété d'exclusivité :

$$P_{inf}^{*}(i) = \frac{\mu\left(F_i^{*} \cap \overline{\bigcup_{j \neq i} F_j^{*}}\right)}{\mu\left(\overline{F_{\perp}^{*}}\right)} \tag{2.92}$$

$$= \frac{\mu\left(F_i^{*} \cap \bigcap_{j \neq i} \overline{F_j^{*}}\right)}{1} \tag{2.93}$$

$$= \mu\left(F_i^{*}\right) \tag{2.94}$$

$$= p_i^{*} \tag{2.95}$$

Il ne reste donc qu'à déterminer $P_{inf}(i)$, si nous combinons les formules 2.95 et 2.87 :

$$p_i^{*} = P_{inf}(i) \tag{2.96}$$

Concentrons-nous sur le numérateur de la formule 2.89 pour le système indépendant :

$$\mu\left(F_i \cap \overline{\bigcup_{j \neq i} F_j}\right) = \mu\left(F_i \cap \bigcap_{j \neq i} \overline{F_j}\right) \tag{2.97}$$

$$= \mu\left(F_i\right) \prod_{j \neq i} \mu\left(\overline{F_j^{*}}\right) \tag{2.98}$$

$$= p_i \prod_{j \neq i} 1 - p_j \tag{2.99}$$

$$= \frac{p_i}{1 - p_i} \prod_{j} \left(1 - p_j\right) \tag{2.100}$$

$$= \kappa \frac{p_i}{\left(1 - p_i\right)} \tag{2.101}$$

$$\text{With } \kappa = \prod_{j} 1 - p_j \tag{2.102}$$

L'égalité 2.97 est obtenue par transformation de l'union en intersection à l'aide d'une réécriture de la formule 2.46 : $\overline{A \cup B} = \overline{A} \cap \overline{B}$. L'égalité 2.98 provient d'une simplification liée à l'hypothèse d'indépendance. L'égalité 2.99 transforme les ensembles en probabilités

à l'aide des mesures unitaires. L'égalité 2.100 s'efforce d'extraire les parties de la formule dépendantes de i. Et enfin, les équations 2.101 et 2.102 rendent abstraites les parties indépendantes de i.

Maintenant, nous avons besoin de normaliser et il faut pour cela prendre en compte non seulement les numérateurs de tout le N-best, mais également le cas où aucun des éléments du N-best n'est la bonne transcription. Ce dernier cas correspond au cas où tous les p_i sont faux :

$$\mu\left(\overline{F_\perp}\right) = \sum_i \kappa \frac{p_i}{1-p_i} + \prod_j (1-p_j) \tag{2.103}$$

$$= \kappa \sum_i \frac{p_i}{1-p_i} + \kappa \tag{2.104}$$

L'équation 2.103 est obtenue par ajout de la partie d'ignorance à la distribution connue des poids. L'équation 2.104 utilise la formule 2.102 pour substituer le cas où tous les p_i sont faux par κ. Maintenant, nous pouvons déterminer p_i^* :

$$p_i^* = P_{inf}(i) \tag{2.105}$$

$$p_i^* = \frac{\mu\left(F_i \cap \overline{\bigcup_{j \neq i} F_j}\right)}{\mu\left(\overline{F_\perp}\right)} \tag{2.106}$$

$$p_i^* = \frac{\kappa \dfrac{p_i}{1-p_i}}{\kappa \sum_j \dfrac{p_j}{1-p_j} + \kappa} \tag{2.107}$$

$$p_i^* = \frac{\dfrac{p_i}{1-p_i}}{\sum_j \dfrac{p_j}{1-p_j} + 1} \tag{2.108}$$

L'équation 2.105 découle directement de la formule 2.96. L'équation 2.106 est également une réécriture de l'équation 2.89. L'équation 2.107 incorpore les résultats obtenus dans les équations 2.101 et 2.104. La formule finale est obtenue par la simplification des termes κ. κ n'a donc pas besoin d'être calculé.

2.6.3 Remarques sur la transformation

D'un point de vue computationnel, le calcul du système exclusif à partir du système indépendant est en $O(N)$ où N est la taille du N-best, ce qui est très peu coûteux.

De plus, ce calcul est mathématiquement exact. Et d'ailleurs, si l'on analyse la formule, nous nous rendons compte que nous retrouvons tous les comportements souhaités :

- La formule est une moyenne équivalente à une normalisation simple, mais la référence de la normalisation est la vraisemblance [19] : $L_k = \frac{p_k}{1-p_k}$. L'utilisation de la vraisemblance garantit l'absorbance de $p_k = 1$.
- Le terme +1 dans le dénominateur a également un effet très intéressant.
  - Le terme +1 correspond à l'ignorance du système.
  - Il empêche p_k^* d'être supérieur à p_k et nous avons l'égalité $p_k^* = p_k$ seulement dans deux cas : $p_k = 1$ ou $\forall j \neq k$, $p_j = 0$. Ceci est intuitif dans la mesure où la compétition entre deux énoncés doivent affaiblir p_k sauf quand celui-ci est absorbant.
  - Il garantit également que $\sum_k p_k^* = 1$ si et seulement si $\exists k$, tel que $p_k^* = p_k = 1$. Ceci est également souhaitable, puisque, tant qu'il y a de l'incertitude, il y a un risque qu'un mot soit hors du vocabulaire et que et la bonne séquence de mots ne fasse pas partie du N-best.
- Si le N-best ne comprend qu'un élément, alors $p^* = p$.

Un autre exemple pour bien comprendre le fonctionnement général de la transformation : l'ASR peut reconnaître le sexe de l'utilisateur en plus de la séquence de mots. Il produirait ainsi une liste n-best de séquences de mots et une liste m-best de sexes (avec $m = 0$, 1 ou 2). Après normalisation, les différentes séquences de mot reconnues seraient exclusives entre elles, de même pour les différents sexes reconnus, mais les séquences et les sexes ne seraient pas inter-exclusifs.

2.7 Conclusion

Dans ce chapitre, nous avons présenté un cadre formel original pour le raisonnement sur les incertitudes qui a toutes les propriétés souhaitées dans le cadre des systèmes de dialogue : le Logical Framework for Probabilistic Reasoning (LFPR). A l'heure où cette thèse est écrite, une prototype de raisonnement sur les incertitudes basé sur le LFPR a été implémenté. Il doit être intégré dans une application de dialogue pour le prototype d'application de dialogue final du projet CLASSiC. Ces avancées théoriques permettront entre autres de rendre le système moins sensible aux erreurs de reconnaissance de reconnaissance vocale et ainsi d'en améliorer la robustesse.

Le prochain chapitre introduit la problématique de l'apprentissage dans les systèmes de dialogue conventionnels. Ce volet "apprentissage" est complémentaire à au volet "gestion des incertitudes" traité dans ce chapitre dans la mesure où le premier permet d'exploiter

19. Likelihood

de façon optimale la précision probabiliste apporté par le second.

Chapitre 3

Retour d'usage

3.1 Introduction

Ce chapitre a pour objectif de montrer comment le leitmotiv de la satisfaction des contraintes industrielles a pu entraîner une réflexion nous menant directement à l'utilisation d'outils d'apprentissage. A la fin de ce chapitre, nous arriverons donc à réfuter le lieu commun affirmant que les contraintes industrielles sont un obstacle à l'avancée scientifique dans un domaine. Au contraire, elles en définissent le cadre applicatif et lui garantissent une structure tangible. La philosophie poursuivie dans cette thèse est oulipienne, des contraintes naissent les idées, l'originalité et la création. De nombreuses citations littéraires viennent corroborer cet état d'esprit :

> Tout pouvoir vient d'une discipline et se corrompt dès qu'on en néglige les contraintes. (Roger Caillois)

> L'art est toujours le résultat d'une contrainte. Croire qu'il s'élève d'autant plus haut qu'il est plus libre, c'est croire que ce qui retient le cerf-volant de monter, c'est sa corde. (André Gide)

> La contrainte n'est pas un désir de limiter son univers, mais bien l'inverse, strictement. (Régine Detambel)

La recherche récente dans les systèmes de dialogue a fait un appel vers une "synergistic convergence"[20] entre la recherche et l'industrie (Pieraccini & Huerta, 2005). Cet appel concerne la convergence des architectures, abstractions et méthodes des deux communautés. En suivant cette motivation, plusieurs orientations de recherche ont été proposées. Ce chapitre discute trois d'entre elles : la conception de dialogue, la gestion du dialogue et l'évaluation du dialogue. La conception et la gestion du dialogue reflètent les chemins respectifs empruntés par l'industrie et la recherche lors de la construction de leurs systèmes

20. Convergence synergique.

de dialogue. L'évaluation du dialogue est un sujet intéressant les deux communautés, mais qui reste difficile à placer dans une perspective opérationnelle.

L'industrie du dialogue vocal –en particulier l'organisation verticale des vendeurs technologiques et des intégrateurs de plates-formes– est structurée autour de l'architecture bien connue de la norme industrielle VoiceXML (Ferrans et al., 2004). Le modèle de dialogue sous-jacent du VoiceXML est une simple transposition du modèle linguistique à base de tours de dialogue sur l'architecture Web basée sur les échanges navigateur-serveur (McTear, 2004). Le navigateur contrôle les moteurs de parole (Automatic Speech Recognition et Text-To-Speech) qui sont intégrés dans la plate-forme vocale selon le document VoiceXML fourni par le serveur d'application. Le document VoiceXML contient un ensemble d'énoncés à jouer et la liste des interactions possibles que l'utilisateur est supposé avoir en chaque point du dialogue. Les développeurs du système de dialogue [21], en réutilisant les normes et technologies Web (par exemple, J2EE, JSP, XML, ...), ont l'habitude de concevoir directement les dialogues sous la forme d'automate à états fini. Un tel procédé de développement déterministe et contrôlé permet à l'industrie du dialogue vocal d'atteindre un équilibre entre l'utilisabilité et le coût (Paek, 2007). Cet équilibre est fortement influencé par les contraintes imposées par les applications et l'avancée technologique actuelles. Jusqu'à maintenant, les progrès de l'industrie ont principalement porté sur les méthodologies et les outils. Ce chapitre soutient que les outils sont des facilitateurs qui améliorent à la fois le compromis utilisabilité-coût et la fiabilité des nouvelles technologies.

La recherche en dialogue vocal a développé de nombreux modèles et abstractions pour la gestion du dialogue : les agents rationnels (Sadek, 1991; Bretier, 1995; Sadek et al., 1997; Louis, 2002), l'Information State Update (Bohlin et al., 1999; Larsson & Traum, 2000; Bos et al., 2003) et les modèles fonctionnels (Pieraccini et al., 1997; Constantinides et al., 1998; Shriver et al., 2001; Pieraccini et al., 2001) entre autres. Seuls un faible nombre de ces concepts ont été transférés à l'industrie. Depuis la fin des années 90, la recherche s'est attaqué à la problématique ambitieuse de l'automatisation de la conception des systèmes de dialogue (Levin et al., 1998; Singh et al., 1999; Young, 2000; Lemon & Pietquin, 2007), dans le but de conjointement réduire les coûts de développement et optimiser l'efficience et la robustesse du dialogue. Récemment, des critiques (Paek 2006; 2008) ont été formulées et des approches novatrices (Singh et al., 2002; Baudoin, 2008; Williams, 2008) ont été proposées, cherchant à la fois à combler le vide entre le recherche –focalisée par les systèmes de dialogue basés sur les Markov Decision Process (MDP, Bellman, 1957b; Sutton & Barto, 1998)– et l'industrie –focalisée sur les outils, modèles et

21. Dans ce chapitre, le terme "développeur" renvoie sans distinction aux concepteurs de services vocaux, aux développeurs d'applications, et aux ingénieurs impliqués dans la construction du système de dialogue.

procédés de conception de systèmes de dialogue. Ce chapitre et plus généralement cette thèse contribuent à l'extension de cet effort.

A propos de l'évaluation du dialogue, Paek (2007) a souligné que, pendant que la recherche se posait la question de "comment évaluer au mieux un système de dialogue" (Walker et al. 1997; 2000), l'industrie s'est concentrée sur "comment concevoir optimalement les systèmes de dialogue". Le premier essaie de définir une métrique d'évaluation des systèmes de dialogue commensurable, permettant ainsi la comparaison entre systèmes disparates. Le second essaie d'optimiser le retour sur investissement des services aux clients. Cette enquête critique de l'évaluation du dialogue montre qu'il n'existe aucune réponse exhaustive aux questions sous-jacentes : "comment le système de dialogue se comporte-t-il en comparaison des autres systèmes de dialogue ?", "comment le système de dialogue se comporte-t-il par rapport aux utilisateurs ?", "comment le système de dialogue peut-il s'améliorer ?", ... Ce chapitre propose un point de vue original sur ces questions, en introduisant une nouvelle base pour l'évaluation du dialogue.

Ce chapitre soulève la question de la convergence entre les trois sujets du dialogue que nous avons évoqués. Il affirme la nécessité de s'atteler aux trois simultanément pour réconcilier la recherche et l'industrie et atteindre une rupture technologique. La section 3.2 introduit l'état de l'art des outils de conception des systèmes de dialogue industriels et montre comment ils s'intègrent aux contraintes industrielles : la "VUI-completeness" [22], l'accessibilité technologique et la confiance. La section 3.3 démontre que le travail des développeurs de systèmes de dialogue est facilité par l'intégration de fonctionnalités d'évaluation dans l'outil de conception du dialogue, lorsque ces développeurs appliquent une méthode de conception itérative/expérimentale. Cette convergence entre la conception du dialogue et l'évaluation bâtit les fondations pour traiter l'évaluation du dialogue. La section 3.4 conclut ce chapitre en confirmant la portée de la problématique de la thèse.

3.2 Etat de l'art des outils de conception

L'industrie poursuit le principe de VUI-completeness (Pieraccini & Huerta, 2005) : "*the behaviour of an application needs to be completely specified with respect to every possible situation that may arise during the interaction. No unpredictable user input should ever lead to unforeseeable behaviour*" [23]. Les développeurs de systèmes de dialogue considèrent qu'une technologie, un outil ou une méthodologie est fiable, quand elle ou il les aide à

22. Complétude VUI (Voice User Interface).

23. le comportement d'une application se doit d'être complètement spécifié selon chaque situation qui puisse advenir durant l'interaction. Aucun énoncé de l'utilisateur, aussi imprédictible soit-il, ne doit jamais pousser le système dans un comportement inattendu.

atteindre la VUI-completeness et à la contrôler.

3.2.1 Un outil graphique permettant de générer dynamiquement du VoiceXML

Les machines à états finies sont une technologie fiable pour les développeurs de systèmes de dialogue. Quand on les applique à la gestion du dialogue, elles permettent aux développeurs de se concentrer sur la définition exhaustive de la logique de dialogue, sans aborder la question de son implémentation. Pour être complètement fiable, cette approche de la gestion du dialogue requiert une transition directe entre la logique de dialogue spécifiée et son implémentation. De plus, cette implémentation doit être conforme aux normes industrielles.

L'outil de conception du dialogue de France Télécom R&D satisfait ces contraintes. Il fournit aux développeurs une abstraction graphique de la logique de dialogue, que l'on appelle dans la littérature parfois "call flow"[24]. Grâce à la fonctionnalité de génération dynamique de VoiceXML, cet outil de conception apporte aux développeurs une garantie totale que la VUI-completeness au niveau de la conception de la logique de dialogue se traduit dans la complétude similaire de l'implémentation. Cette propriété rencontrée durant la conception se prolonge lors de la maintenance : si les développeurs ne modifient qu'une partie limitée de la logique de dialogue, l'outil garantit que seul le code correspondant en subit l'impact. Cette garantie influence positivement la VUI-completeness, la fiabilité et le coût de développement.

L'abstraction graphique proposée par cet outil de conception est conforme aux représentations habituelles des automates finis. Les nœuds sont les états du dialogue. Les transitions sont soit déclenchées par des conditions sur des variables de session soit par des entrées de l'utilisateur ou d'applications tierces. A l'intérieur même de cette abstraction graphique, un état du dialogue inclut à la fois l'état informationnel et l'action suivante associée du système. En ce sens, pour l'industrie, la conception de dialogue couvre généralement plus que la gestion du dialogue stricto sensu. Tout d'abord, les énoncés utilisateur attendus sont représentés dans la logique de dialogue comme des transitions entre les états. Cette spécification peut indiquer le type d'énoncés parlés attendus de la part de l'utilisateur à chaque étape du dialogue, jusqu'à préciser le modèle de reconnaissance vocale et les valeurs de ses paramètres. Les interprétations sémantiques possibles correspondantes sont des labels de transition d'un état du dialogue à un autre. Ensuite, la conception du dialogue inclut généralement la formulation précise de chaque action du système. Les synthèses audio correspondantes sont alors générées à l'aide d'enregistrements humains,

24. Flux de dialogue.

de moteurs de Text-To-Speech (TTS) ou d'outils de conception de la parole basés sur des technologies TTS nouvelles, contrôlées manuellement (Cadic & Segalen, 2008; Cadic et al., 2009; Boidin et al., 2009a).

L'abstraction graphique pour la conception de dialogue couvre exactement les mêmes éléments que le document VoiceXML. Les développeurs peuvent alors complètement contrôler le comportement du système depuis l'outil de conception graphique. De plus, ces abstractions graphiques sont très simples à partager, à comprendre et à mettre à jour.

3.2.2 Un outil de conception graphique qui oriente la façon d'appréhender la problématique du dialogue

Selon le périmètre de la logique de dialogue et l'abstraction graphique que l'outil fournit aux développeurs de systèmes de dialogue, deux paradigmes de conception opposés sont possibles. Le premier paradigme propose d'anticiper les énoncés utilisateur et leurs sens, puis de définir ce que le système de dialogue est censé faire dans chaque cas. Au contraire, le second paradigme propose de définir initialement les actions et formulations du système de dialogue pour spécifier ensuite les réactions attendues de la part de l'utilisateur. Les deux paradigmes contiennent une partie de conception concernant les actions système et une partie de classification des entrées utilisateur avec leurs significations possibles. La différence entre les deux repose sur l'ordre dans lequel ces deux parties sont traitées.

A ce niveau du chapitre, l'opposition entre les deux paradigmes peut sembler insignifiant. Mais la section 3.3 montre que le choix de l'une des deux conditionne fortement la façon dont les développeurs conduisent leurs travaux de conception et d'évaluation.

L'outil de conception du dialogue présenté dans cette section est conforme au choix de paradigme fait par la norme VoiceXML : un document VoiceXML définit en premier, pour chaque état de dialogue, un ensemble d'énoncés à générer, puis il anticipe les interactions possibles avec l'utilisateur.

La figure 3.1 montre un aperçu de l'outil de conception. C'est un extrait de la conception graphique du prototype de système de dialogue portant sur l'installation de la Livebox (voir le chapitre 6). Il illustre la logique de dialogue appliquée lorsque le système vérifie si l'utilisateur appelle de chez lui pour faire l'installation de la Livebox. L'abstraction graphique se lit de gauche à droite. Tout d'abord, il y a un rectangle avec les angles arrondis représentant l'énoncé système "Êtes-vous actuellement chez vous ?". Ensuite, une réponse est attendue de la part de l'utilisateur, soit une entrée vocale correspondant à une des significations attendues, soit une entrée vocale rejetée, ou aucune entrée vocale dans une fenêtre temporelle prédéfinie (ici un time-out $T0$ défini quelque part dans les fichiers de

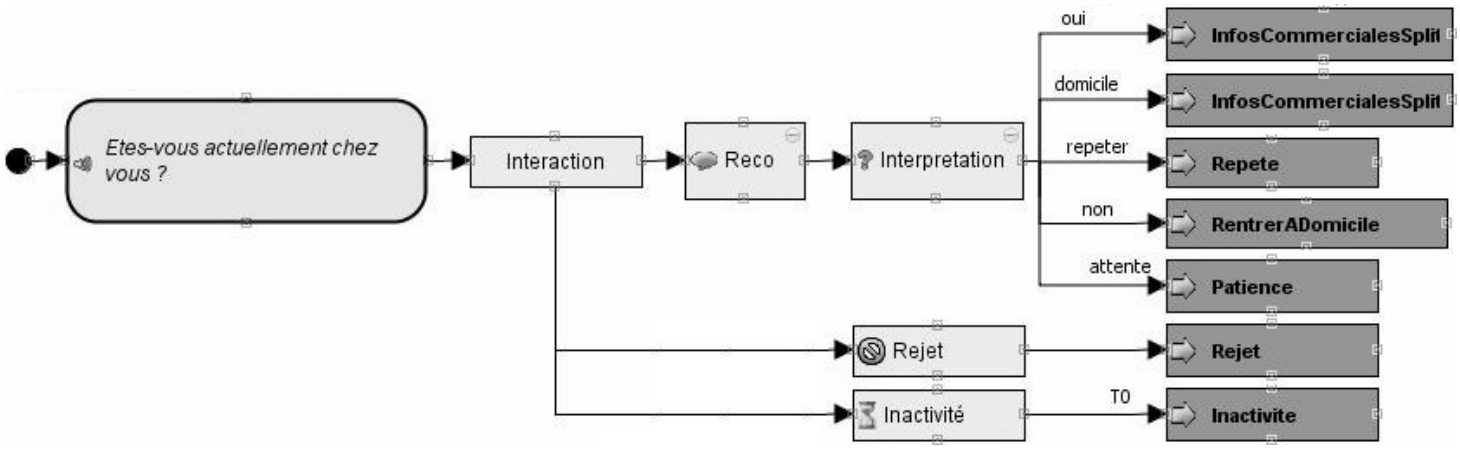


FIGURE 3.1 – L'abstraction graphique d'un état de dialogue d'un prototype.

configuration). La conception prévoit trois réactions spécifiques à cet état du dialogue de la part de l'utilisateur : "oui", "non" ou "je suis à mon domicile" et deux réactions plus générales : "répétez" ou "patientez". Chacun de ces scénarii mène à un nouvel état du dialogue. Par exemple, si la signification reconnue de l'énoncé utilisateur est la réponse "non" alors l'état de dialogue "*RentrerADomicile*" est atteint.

L'outil de conception est interactif : le double-clic sur un état de dialogue ouvre la vue graphique correspondante. Cette fonctionnalité procure aux développeurs un moyen de suivre les chemins empruntés lors de dialogues, tout en spécifiant le système. En addition de la vérification automatique de la complétude de l'automate accomplie par l'outil, cette possibilité de suivre les différents chemins renforce la confiance des développeurs en la VUI-completeness du système de dialogue.

3.2.3 Contrôle vs. automatisation : le seuil de confiance

Comme le soutiennent Pieraccini et Huerta (2005), même si cela n'est pas encore communément admis dans la communauté scientifique, les machines à états finies appliquées à la gestion du dialogue ne restreignent pas la modélisation du dialogue à des scénarii entièrement dirigés. Les machines à états finies, par leur simplicité intrinsèques, sont facilement extensibles à des modèles puissants et flexibles. L'outil de conception du dialogue présenté dans les sous-sections précédentes permet les extensions suivantes : modules de dialogue, conception hiérarchique, invocation de fonction arbitraire à n'importe quel point de la conception, tests sous forme de condition pour séparer le flux en plusieurs chemins. Toutes ces extensions rendent possible la conception de n'importe quelle topologie de machine à états finie requise pour faire face aux modèles de dialogue complexes tels que les interactions à initiative mixte. Au final, le modèle de dialogue n'est pas le point où la

recherche et l'industrie n'arrivent pas à converger.

La divergence de point de vue concerne le compromis entre la VUI-completeness et l'automatisation de la conception du dialogue. Comme certains travaux récents le soulignent (Paek 2007; 2008), la gestion du dialogue à base de MDP visant à l'automatisation complète de la conception du dialogue est rejetée par les développeurs de systèmes de dialogue industriels. Même si ces systèmes sont plus adaptatifs, ils sont considérés comme des boîtes noires incontrôlables qui restent très sensibles aux différents paramétrages. Les développeurs ne veulent pas se reposer sur des systèmes ayant généré automatiquement leur logique de dialogue sans un certain degré de contrôle et de supervision.

Comme nous le développerons plus tard dans la sous-section 4.2.3, Singh et al. (2002), puis plus profondément Williams (2008) ont fait un effort substantiel pour atteindre ce prérequis industriel. Ce dernier a développé un système de dialogue hybride regroupant un système conventionnel à base d'automate suivant les contraintes industrielles et un module de décision basé sur les Partially Observable Markov Decision Process (POMDP Sondik, 1971; Lovejoy, 1991). Les POMDP sont des MDP auxquels on a ajouté des capacités de gestion des incertitudes liées au caractère non-observable des états du dialogue. Les responsabilités du système hybride sont partagées entre ces modules de la façon suivante : l'automate conventionnel sélectionne un ensemble de mouvements de dialogue candidats pour la génération et le module POMDP prend la décision finale de celui qui sera effectivement joué à l'utilisateur. Outre les résultats qui montrent une gain en vitesse de convergence et en performance du système, ce travail constitue une grande avancée de la recherche académique vers la réalité industrielle parce que tous les mouvements de dialogue que le module POMDP peut choisir ont été validés au préalable par l'automate conventionnel. Néanmoins, ce système ne satisfait pas toutes les contraintes industrielles.

Premièrement, même pour des applications très simples, cette approche force les développeurs à paramétrer très précisément l'automate conventionnel et le module de décision POMDP. Au final, les développeurs sont amenés à utiliser des compétences scientifiques pointues rares. En conséquence, la fluidité du processus industriel s'en retrouve fortement abîmée.

Deuxièmement, et nous reviendrons dessus dans la sous-section 4.2.4 et la section 4.3, le fait d'appliquer l'apprentissage au niveau des mouvements de dialogue est une contrainte forte. Les développeurs ont l'habitude de concevoir des conditions sur des fonctions et des variables arbitraires pour séparer le flux en plusieurs chemins. Pour avoir une confiance totale dans le module de décision automatique, les développeurs ont besoin qu'il prenne ses décisions sur la même dimension que leur espace de conception, de manière à pouvoir conserver la flexibilité et le pouvoir d'expression. Tel que Williams a hybridé son système, les développeurs de la partie conventionnelle restent très contraints dans la conception

par les limites théoriques de l'apprentissage des POMDP.

Troisièmement, un module apprenant tel que celui de Williams n'explique pas ses choix. Il dit quoi faire, mais pas pourquoi. Bien que les développeurs aient le contrôle des comportements proposés au module de décision POMDP, ils n'ont aucun moyen de comprendre pourquoi telle alternative a été préférée à une autre. En conséquence, ce ne sera jamais un outil permettant de généraliser l'ergonomie du dialogue ou les pratiques des développeurs.

Ces considérations sont les principales contraintes que la thèse s'est imposée. Le parti-pris de la thèse est de dire que le processus de décision doit se faire au niveau des décisions locales de conception, c'est-à-dire au niveau des transitions de l'automate, et non au niveau des mouvements de dialogue. Si l'on dispose d'un tel processus de décision et d'un algorithme d'apprentissage adapté, il est alors possible d'intégrer des fonctionnalités d'apprentissage automatique aux systèmes de dialogue industriels.

Cette sous-section a montré que la confiance n'est pas encore suffisante pour que les développeurs soient prêts à adopter de telles technologies d'automatisation ou d'optimisation de la logique de dialogue. La section suivante de ce chapitre montre comment l'outil de conception présenté dans cette section peut être couplé à des fonctionnalités de supervision qui contribuent à améliorer la confiance que les développeurs peuvent porter sur les nouvelles technologies d'apprentissage.

3.3 Couplage de l'outil de conception avec des fonctionnalités de supervision de l'évaluation

Alors que les chercheurs s'intéressent à la mesure du progrès que leurs algorithmes et méthodes permettent d'obtenir, les ingénieurs de l'industrie s'occupent des paramétrages et optimisations des systèmes de dialogue. Leur premier critère d'évaluation est la complétion de tâche, appelée certaines fois le taux d'automatisation, parce que le déploiement d'un service de dialogue est toujours réalisé dans l'objectif de réaliser une tâche spécifique. Ces tâches sont choisies pour leur potentiel à être traitées automatiquement, ce qui est également lié à la capacité de mesurer la complétion de cette tâche. La plupart du temps, la complétion de la tâche est estimée à l'aide de "Key Performance Indicators"[25] (KPI). Pour atteindre un maximum de "oui" à une question oui/non, il a été démontré que le schéma de formulation "si vous voulez [...], dites oui" est le plus performant. Les cas aussi simples ne sont pas fréquents. Quand l'estimation de la complétion de tâche requiert plusieurs KPI et une interprétation difficile à formaliser, l'évaluation du dialogue devient un

25. Indicateurs clefs de performance.

problème majeur pour les développeurs.

Dans cette section, nous étudions la convergence entre la conception et l'évaluation, et comment leur synergie peut offrir de nouvelles fondations au domaine de l'évaluation dans une perspective opérationnelle.

3.3.1 Outils de visualisation de flux de dialogues

Quelques papiers (Abella et al., 2004; Wright et al., 2005) se sont penchés sur la question des outils de supervision. Dans leurs travaux, le corpus à visualiser est un ensemble de logs de dialogues. L'outil cherche à révéler la façon dont le système évolue entre ses états possibles. Comme un système de dialogue est trop complexe pour énumérer tous ses états, les logs du dialogue sont considérés comme des ensembles de variables (représentant l'état du système) qui évoluent au court du temps et des interactions et l'outil propose de faire une projection sur un sous-ensemble de ces variables pour étudier les flux entre les différentes valeurs de ces variables. De cette manière, les graphes générés peuvent soit montrer le flux d'appel, c'est-à-dire comment les différentes étapes du dialogues sont atteintes et où elles mènent, ou révéler comment certaines variables, telles que le nombre d'erreurs depuis le début du dialogue, évoluent. Leur outil est principalement un outil servant à comprendre comment les utilisateurs se comportent, parce qu'aucune connexion n'existe avec la façon dont le système a été construit. En conséquence à cela, l'outil fonctionne au niveau du tour de dialogue, et même dans un grand nombre de cas, ils ne sont pas tous visibles. Cet outil sert donc à comprendre si le système remplit le besoin utilisateur, mais il n'est d'aucune utilité pour comprendre comment le rendre meilleur. En d'autres mots, il permet une évaluation du système, sans atteindre le but initial que les industriels se fixent : comment rendre le système meilleur.

L'outil original que nous proposons dans la suite de cette section sert à ces deux objectifs, grâce à la convergence entre la visualisation de la conception et la visualisation de l'évaluation.

3.3.2 La projection des retours de l'expérience utilisateur dans l'outil de conception

Les KPI sont difficiles à lister ou classifier exhaustivement. Certains sont liés à des mesures du système et d'autres requièrent des information additionnelles collectées soit par les annotations manuelles des logs, soit à l'aide de questionnaires remplis par les utilisateurs.

Les questionnaires servent à mesurer l'utilisabilité perçue par l'utilisateur. En marge des difficultés (coût en argent et en temps) inhérentes à ce processus d'évaluation des

systèmes de dialogue, Paek (2007) a souligné que "*because usability factors are subjective, they can be erratic and highly dependent on the complex interplay of user interface attributes*"[26]. De plus, la méthodologie basée sur les questionnaires utilisateurs ne s'intéressent pas à l'évaluation du système de dialogue, mais à la perception utilisateur de l'interaction, qui est elle-même bruitée par son expérience individuelle.

Parmi les mesures du système, certains KPI établissent des propriétés centrées sur la tâche, par exemple la complétion de tâche ou la durée de la tâche. D'autres KPI sont des propriétés liées au dialogue. Elles sont soit définissables automatiquement –par exemple le taux de raccroché, le nombre de tours de dialogue, la durée moyenne des tours, le taux de barge-in[27] ou les délais de réponse– ou elles nécessitent des transcriptions et annotations manuelles –par exemple le taux d'erreurs de reconnaissance vocale, ou d'interprétation sémantique. La performance du système peut aussi être mesurée, par exemple la charge, la disponibilité et le temps de réponse du système.

Les KPI peuvent être mesurés à plusieurs niveaux de granularité : leur portée s'échelonne de globale (sur tout le dialogue) à locale (lié à l'état du dialogue courant). Les KPI locaux sont donc associées à des états de l'automate de conception. Quand les développeurs conçoivent une transition entre deux états du dialogue, ils peuvent estimer les ordres de grandeur pour leurs KPI locaux. Quand ils conçoivent des états de récupération d'erreurs de dialogue, ils espèrent que l'utilisateur les évitera et restera dans le flux d'appel principal. De même, quand ils définissent les transitions partant d'un état de dialogue, ils savent lesquelles font partie du déroulement normal du dialogue et lesquelles sont des exceptions que l'on souhaiterait éviter. Ainsi, la conception du dialogue devient très fortement liée à la conception des KPI locaux.

L'outil de conception de France Télécom, décrit dans la section 3.2, procure une fonctionnalité innovante : la projection des KPI locaux dans la conception de dialogue originelle. Pour mener cela à bien, il est nécessaire que le système produise des logs extensifs. Ensuite, l'outil est capable de charger les corpus de dialogue, construit sur ces logs, et de calculer automatiquement les KPI correspondants. Les KPI locaux sont ainsi projetés sur la vue graphique de la conception du dialogue. La plupart de ces KPI peuvent ainsi être automatiquement calculés et affichés.

La figure 3.2 montre le même exemple de conception que la figure 3.1 mais depuis la vue d'évaluation où les KPI locaux sont projetés sur la visualisation graphique de la

26. Parce que les indicateurs d'utilisabilité sont subjectifs, ils peuvent être imprévisibles et hautement dépendants des interactions complexes des attributs de l'interface utilisateur.

27. Définition de "barge-in" : Capacité d'un humain à parler par dessus un prompt du système. Le barge-in est particulièrement fort dans le cas d'utilisateurs fréquents de systèmes de dialogue. Deux types de barge-in sont en général pris en compte, celui où l'humain parle sans que le système ne comprenne, ni modifie son comportement ; et celui où l'humain interrompt le système par le fait même de parler.

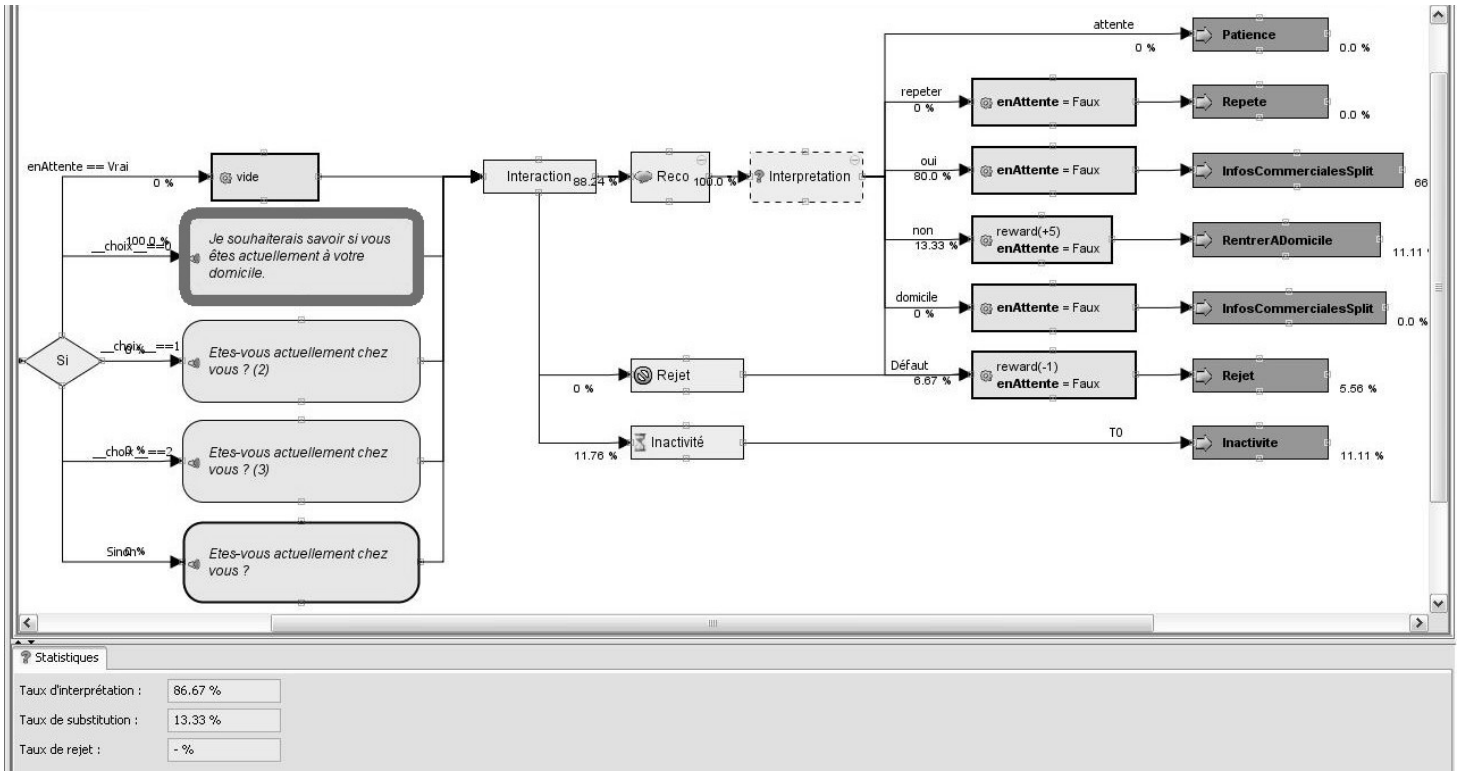


FIGURE 3.2 – Quelques retours d'usage liés à la sélection d'une alternative d'énoncé.

logique de dialogue. Les fréquentations locales sont affichées en pourcentage dans chaque branche. Par exemple, 11.76% des utilisateurs n'ont pas réagi dans le délai du time-out, et 66% des utilisateurs pour lesquels la reconnaissance vocale donne un résultat ont suivi le flux d'appel vers l'état de dialogue "*InfoCommercialeSplit*". D'autres indicateurs de performance sont affichés en bas de la fenêtre. Ils sont corrélés à l'objet graphique sélectionné. Quand l'objet rectangulaire "*reco*" est sélectionné, le taux d'erreur par mot, le taux de rejet de la parole et les autres KPI liés à l'ASR sont affichées. La figure 3.2 montre la vue quand l'objet rectangulaire "*interprétation*" est sélectionné. On peut voir en bas de la page le taux de bonne interprétation, le taux de substitution et le taux de rejet.

En référence au retour de force, cette fonctionnalité est appelée "retour d'usage" parce qu'il rappelle le sentiment que les développeurs peuvent éprouver en comparant l'expérience utilisateur au sein même de leur logique de dialogue. C'est un moyen très efficace pour détecter et délimiter les problèmes majeurs de conception, parce qu'il permet aux développeurs de vérifier si leurs prédictions sont conformes aux expériences mesurées des utilisateurs.

Lors de l'évaluation d'un système de dialogue, une des questions pour les développeurs est de décider quoi faire étant données des valeurs de KPI. Face à un fort taux d'erreur de

reconnaissance, les développeurs peuvent décider s'il faut augmenter la taille du corpus d'apprentissage pour mieux entraîner le module de reconnaissance vocale ou s'il faut au contraire reformuler la question pour obtenir un ensemble plus restreint de réponses possibles de la part des utilisateurs.

Cet outil propose également une fonctionnalité de test multivariable [28] (Kohavi et al., 2009). Cette méthode, très répandue dans la conception de sites Web, consiste à tester plusieurs alternatives et de sélectionner la meilleure sur la base d'un ensemble de critères prédéfini. A propos de la VUI-completeness, la présentation de l'automate complet aux développeurs est acceptable, tant qu'ils peuvent inspecter et contrôler chaque branche de l'automate. En général, ils viennent même généralement avec plusieurs alternatives en compétition dans plusieurs points de choix différents, lesquels ne peuvent être validés que d'une manière statistique.

Cette capacité à comparer toutes les alternatives de stratégie de dialogue dans un unique champ de test est un élément majeur pour accélérer le processus d'amélioration du service de dialogue, en réduisant de beaucoup la boucle de rétroaction. De plus, elle homogénéise les conditions dans lesquelles chaque alternative est testée, réduisant ainsi les interférences potentielles telles que l'habitude d'un utilisateur à une version donnée du système. Elle augmente également la pertinence statistique du lien causal entre les alternatives testées et les différences entre les performances mesurées.

La projection graphique des KPI dans la logique de dialogue couvre les alternatives de dialogue : on a juste besoin que le calcul de KPI soit conditionné par le choix d'alternative. La figure 3.2 illustre la fusion de plusieurs alternatives d'énonciations dans un seul point de choix : trois styles de parole (calme, neutre et expressive) pour une même formulation et une variante de formulation. Les KPI qui sont affichés sont conditionnés par l'alternative qui a été sélectionnée manuellement, ici la première alternative est entourée d'une ligne gris épais. D'une certaine mesure, le couplage du retour d'expérience dans la logique de dialogue constitue une nouvelle expérience de conception de service : le concepteur peut maintenant sculpter son service et ressentir directement les effets de ses modifications sur les mesures de KPI.

Par l'observation des effets de chaque alternative sur les KPI en sélectionnant successivement les différentes alternatives, les développeurs ont accès à un moyen fiable de comprendre comment améliorer leur service. Comme Paek (2007), nous incitons la communauté à se concentrer plus particulièrement sur la question de savoir comment améliorer un système plutôt que sur le calcul d'une performance relative inter-systèmes.

28. Quand l'ensemble des variables est un singleton appliqué à deux alternatives, la méthodes est connue sous le nom d'A/B testing.

3.3.3 Réexamination de l'évaluation du dialogue

Les approches traditionnelles de l'évaluation du dialogue cherchent à mesurer à quel point le système est adapté aux utilisateurs. Nous souhaitons rappeler que chaque interaction entre l'utilisateur et le système est une expérience unique et non reproductible. Tout d'abord, parce que chaque nouveau dialogue est construit conjointement entre le système et l'utilisateur de manière unique selon leurs compétences respectives : l'expérience de l'utilisateur et les limites du système. Ensuite, parce qu'une personne s'adapte très rapidement et change de comportement aussi vite face à un système de dialogue. Les approches traditionnelles de l'évaluation du dialogue sont basées sur le cadre fragile de référence utilisateur, qui n'est pas suffisamment fiable pour une approche scientifique ou industrielle même s'il reste malheureusement utilisé de manière courante dans la littérature (Rieser & Lemon, 2008a).

Comme nous l'avons montré dans les sections précédentes, le paradigme de conception imposé par l'outil de conception de France Télécom permet aux développeurs de spécifier la logique de dialogue, puis de construire des classifications locales et les KPI incluant les ordres de valeur attendus. Ceci est une suggestion de changement dans le cadre de référence utilisé pour l'évaluation du dialogue : au lieu d'essayer de mesurer l'adéquation entre le système et l'utilisateur selon la subjectivité de l'utilisateur, on peut essayer de mesurer l'adéquation entre le système et l'utilisateur, selon la subjectivité de la logique de dialogue, les KPI et leurs valeurs attendues.

Prendre la conception logique de dialogue comme référence permet de réexaminer la question de l'évaluation du dialogue. La base proposée pour l'évaluation de dialogue est fiable pour les développeurs, puisqu'elle est à la fois stable et sous contrôle. Les déviations par rapport aux situations prédéfinies sont directement traduites en des valeurs anormales de KPI qui sonnent des alertes. Ces alertes, qui sont calculables automatiquement, avertissent les développeurs de la présence de problèmes dans leur logique de dialogue.

Cette approche pour l'évaluation de dialogue a prouvé être utile pour l'amélioration des systèmes de dialogue. Pour savoir comment un système se comporte par rapport à d'autres systèmes, on a besoin de métriques commensurables. Faire des mesures commensurables de déviation requiert une calibration initiale. La conception de la logique de dialogue et les KPI associés sont dépendants de l'implémentation, et il en est de même pour les mesures de déviation. La calibration de ces mesures hétérogènes entre deux systèmes différents reste une question ouverte qui n'est pas approfondie dans cette thèse.

3.4 Conclusion

La conception du dialogue et la gestion du dialogue reflettent les différents chemins que l'industrie et la recherche ont suivis pour la construction de leurs systèmes de dialogue. L'évaluation du dialogue est connectée aux deux sujets, mais a généralement été étudié à part.

Ce chapitre a présenté un outil de conception du dialogue original, qui intègre dans son interface graphique des fonctionnalités de supervision de l'évaluation. Le paradigme de conception imposé par l'outil incite le développeur à prédire les ordres de valeurs des KPI locaux lorsqu'il implémente la logique de dialogue. Cela résulte en un nouveau paradigme d'évaluation utilisant la conception du dialogue comme référence et essayant de mesurer les déviations entre les valeurs mesurées et prédites des KPI locaux.

Les développeurs ont confiance en l'outil pour satisfaire le principe de VUI-completeness. Alors que ce principe est généralement appliqué à la conception du dialogue, l'outil permet d'en élargir son application à l'évaluation du dialogue, rendant ainsi possible la comparaison entre plusieurs alternatives de conception.

Ceci place les développeurs de système de dialogue dans un cycle vertueux d'amélioration du service, assez proche de la philosophie des algorithmes d'apprentissage par renforcement. De plus, l'inspectabilité garantie par la fonctionnalité de retour d'usage permet au développeur de comprendre, analyser et généraliser les différences observées de comportement des utilisateurs entre les alternatives testées.

La suite de cette document est consacrée à l'implémentation d'algorithmes d'apprentissage permettant d'automatiser l'adaptation du système en ligne. L'implémentation de tels algorithmes est entièrement compatible avec la synergie entre la conception et l'évaluation du dialogue que nous avons défendue dans ce chapitre. La combinaison de l'apprentissage et de l'outil permet une optimisation plus rapide et contrôlée algorithmiquement tout en garantissant un contrôle optimal des décisions du système par les développeurs. Cette approche est la première de la littérature qui propose de l'apprentissage pour le dialogue tout en préservant la VUI-completeness, indispensable à l'industrie.

Chapitre 4

Apprentissage pour les systèmes de dialogue

4.1 Introduction

Dans l'industrie, les services aux clients jouent un rôle de plus en plus crucial dans le markéting et pour l'image de marque. Ces services peuvent être incarnés physiquement (dans un magasin par exemple), à distance et humains (les hotlines ou les échanges par mail) ou à distance et automatisés (répondeurs automatiques ou agents conversationnels sur les portails web des entreprises). Cependant, en termes de coûts, l'incarnation physique requiert un local placé stratégiquement (et donc à fort coût) et des ressources humaines. Les relations à distance avec des opérateurs humains requièrent un local à faible coût et des ressources humaines en moindre nombre (le temps de traitement d'un client est plus rapide par téléphone). Les relations à distance automatisées ne requièrent qu'un coût de machines, dont le coût est sans doute équivalent aux machines utilisées par les opérateurs humains et le coût de développement initial, c'est-à-dire plusieurs homme-mois d'ingénieurs, dépendant de la complexité du développement du service. La course au low-cost a conduit beaucoup de compagnies industrielles à automatiser leurs services. Ceci explique le développement rapide des Systèmes de Dialogue (SD) ces dernières années.

Cependant, les SD restent difficiles à développer : la boucle essai-erreur (boucle de conception/test dans le processus de développement des SD) peut être longue à converger. Au final, la boucle est stoppée de façon arbitraire pendant la convergence, plutôt par manque de ressources (temps ou argent) que parce que l'on considère être arrivé à un système optimal. L'utilisation d'outils évolués tels que celui présenté dans le chapitre 3 est une première étape indispensable pour garantir la VUI-completeness de la conception et de l'évaluation et ainsi raccourcir le temps de réponse de la boucle essai-erreur. Ce chapitre a

pour objectif de présenter des cadres formels et des algorithmes permettant l'apprentissage dans le système de dialogue, tout en continuant à garantir la VUI-completeness.

Les algorithmes d'apprentissage et les développeurs de la logique de dialogue sont deux acteurs sur le même objet : le système de dialogue. Mais ils opèrent dans des espaces temporels différents. L'algorithme d'apprentissage met à jour sa politique à la fin de chaque nouveau dialogue alors que les développeurs ne font leurs contrôles que sporadiquement. Le même genre d'opposition entre ces deux acteurs peut être fait à propos de l'espace d'actions. L'algorithme d'apprentissage ne peut changer sa politique que parmi une quantité limitée d'alternatives, alors que le développeur peut faire des changements plus profonds tels que l'implémentation de nouvelles branches de dialogue, des ajouts d'alternatives, des ajouts de nouveaux points d'alternatives, des suppressions d'alternatives, ...De plus, la portée de leur vue varie énormément également. L'algorithme d'apprentissage se concentre sur les ensembles d'alternatives, l'évaluation automatique et ignore le reste, alors que les développeurs peuvent appréhender l'application de dialogue dans sa globalité, en tant que système ou service. Ils ont aussi accès à d'autres évaluations apportées par les annotations, ou les évaluation subjectives.

Ces différences en fonctionnalité rendent leurs rôles respectifs complémentaires. Les développeurs ont la responsabilité de la vue d'ensemble de l'application et des changements de macro-stratégies et l'algorithme d'apprentissage a un rôle d'adaptation rapide pour offrir une maintenance et une optimisation en temps réel.

Tout d'abord, la section 4.2 fait l'analyse de l'état de l'art de l'apprentissage dans les systèmes de dialogue et le confronte aux besoins actuels de l'industrie. Ensuite, la section 4.3 présente et motive notre méthode d'optimisation locale, qui suit l'approche "KPI locale" déjà introduite dans le chapitre 3. Puis, la section 4.4 définit le cadre formel Module Variable Decision Process (MVDP) dans lequel nous nous plaçons et la section 4.5 présente deux algorithmes le Compliance Based Reinforcement Learning (CBRL) et le Module Variable Temporal Difference Learning (MVTDL) fonctionnant dans le MVDP. Enfin, la section 4.6 conclut le chapitre.

4.2 Analyse de l'état de l'art

Après avoir identifié le besoin de l'apprentissage dans les systèmes de dialogue, cette section fait un survol de son état de l'art, dans un premier temps en introduisant les méthodes d'apprentissage hors ligne dont l'objectif est d'automatiser la conception des systèmes de dialogue, puis dans un second temps en présentant les méthodes hybrides avec une conception manuelle et optimisation automatique en ligne. En se basant sur les besoins actuels de l'industrie, cette section analyse ensuite l'état de l'art et pose la

problématique de l'apprentissage dans les systèmes de dialogue.

4.2.1 Le Besoin en Apprentissage

Cette sous-section identifie trois valeurs ajoutées potentielles de l'apprentissage dans les SD.

Automatisation du processus de design

Comme expliqué au début du chapitre, la réduction de coût est l'argument principal pour justifier l'emploi de SD en lieu et place de services rendus par des humains. Mais le développement reste un investissement significatif, en terme de ressources humaines et en infrastructure. Suivant la technologie utilisée, le développeur de SD est souvent contraint de maîtriser des outils scientifiques ou technologiques très complexes. De plus, le travail effectué pour un application de dialogue est très peu réutilisable pour une autre application, même quand le domaine reste proche. Par exemple, Lecœuche (2001) s'est efforcé de trouver un ensemble d'actes de dialogue qui sont communs à tous les services que l'on peut imaginer et d'en extraire des stratégies générales d'interaction. Mais le fait que ces actes soient tous utilisés dans des SD différents ne signifie pas qu'ils soient utilisés de la même façon.

La première valeur ajoutée que des techniques d'apprentissage pourrait proposer est l'automatisation du processus de conception. Si, dans le futur, nous arrivons à générer automatiquement les SD à partir de données récoltées sur les services à implémenter, alors la conception du SD aura un coût beaucoup moindre.

Cette automatisation de la conception des SD a constitué la première motivation pour l'insertion de l'apprentissage dans les systèmes de dialogue (Levin et al., 1998; Singh et al., 1999; Young, 2000; Lemon & Pietquin, 2007). Ils utilisent des Processus de Décision Markoviens (MDP Bellman, 1957b; Sutton & Barto, 1998) pour apprendre selon l'état courant du dialogue le meilleur mouvement de dialogue (dialogue move) à exécuter. Cette approche a conduit à des résultats prometteurs mais, comme Paek (2006; 2008) le rappelle, certains obstacles restent infranchissables (voir sous-section 4.2.2).

Optimisation

Dans le processus de conception de SD, le développeur d'application vocale fait face à beaucoup de décisions difficiles à prendre : "à partir de quel niveau de confiance, le système doit-il considérer que la reconnaissance vocale a bien fonctionné ?" ou "doit-il demander à l'utilisateur une confirmation explicite ?", "dans un contexte donné, le système doit-il utiliser des questions ouvertes, fermées ou à choix multiples ?", "doit-il utiliser le mot

Wifi ou connexion sans fil étant donné ce qu'il sait de l'utilisateur ?", ou encore pour la synthèse vocale, "quel type d'expression est optimale pour faire passer un message donné ?". Ce type de questions apparaît dans chaque application sous des formes diverses et avec des réponses variées. Ces décisions ne sont pas évidentes, mais elles ont un impact fort sur le succès du dialogue. Comme nous l'avons déjà mentionné dans la section 3.3, il a été démontré qu'il était possible de gagner un nombre significatif de transferts dans le service des renseignements français simplement en changeant la génération d'une question classique oui-non pour une plus directive du style : "si vous voulez être transféré, dites oui". Des études en psychologie cognitive ont ensuite montré que de manière générale, les questions formulées de la manière suivante "si vous voulez ..., dites oui" reçoivent plus de réponses positives que les question oui/non plus directes. Si ces choix ne sont pas évidents, la meilleure façon de les faire est de laisser le système s'adapter au service par l'expérience. En effet, l'apprentissage en ligne peut être utilisé comme un procédé d'optimisation automatique.

Suite à l'échec partiel de l'objectif d'automatisation, beaucoup de travaux ont cherché à s'en défaire, comme Williams et Young (2005); Williams et Young (2007) et Young et al. (2005); Young (2006) qui se sont intéressés à l'optimisation de la robustesse des systèmes de dialogue, à l'aide des MDP partiellement observables (POMDP Sondik, 1971; Lovejoy, 1991). Moyennant un contrôle manuel encore plus fort de la complexité du système de dialogue, ces travaux ont implémenté une gestion de l'incertitude générée par la reconnaissance vocale et l'interprétation, qui leur permet d'être moins sensibles à leurs erreurs et de plus facilement les rattraper.

Un second courant de travaux sur l'apprentissage dans les SD a relâché sensiblement l'objectif d'automatisation pour optimiser les systèmes de dialogues automatiquement générés à base de corpus. Cuayáhuitl et al. (2006; 2007) a cherché à améliorer la convergence des algorithmes d'apprentissage en échange d'un effort porté sur la conception du MDP et de manière à rendre possible l'utilisation des SD à base de MDP sur des applications plus complexes. Il propose d'utiliser des algorithmes d'apprentissage par renforcement hiérarchiques (tels que MAXQ Dietterich, 1998 ou HAM Parr & Russell, 1997) dans le but d'aider l'algorithme d'apprentissage en définissant une hiérarchie entre les différentes stratégies de dialogue : les macro-stratégies (c'est-à-dire une stratégie qui est suivie durant plusieurs tours de dialogue) et des stratégies courtes définissant simplement le prochain mouvement de dialogue. Les séquences de stratégies courtes sont optimisées de manière à accomplir une macro-tâche et l'algorithme d'apprentissage optimise aussi les transitions entre les macro-stratégies. Alors que Cuayáhuitl a correctement mis l'accent sur le besoin de suivre des macro-stratégies dans un dialogue, nous insistons plutôt sur le fait qu'un mouvement de dialogue est déjà la plupart du temps une composition de plusieurs

décisions stratégiques plus fines (voir partie 4.2.4).

Singh et al. (2002) et Williams (2008; 2009) sont encore allés un peu plus loin dans l'objectif d'optimisation en abandonnant complètement l'automatisation de la conception du système. En effet, ils ont proposé de construire un SD hybride, moitié implémenté manuellement et moitié basé sur des MDP (ou POMDP). Le but recherché est vraiment nouveau : optimiser les fonctionnalités de dialogue d'un SD conventionnel (c'est-à-dire conçu manuellement). Rapidement, l'idée consiste à concevoir manuellement la plupart des stratégies de dialogue mais de proposer plusieurs mouvements de dialogue au lieu d'un seul. La partie MDP du SD hybride choisit ensuite celui qui sera restitué à l'utilisateur. Notre positionnement vis-à-vis de ces travaux a déjà été évoqué dans la section 3.2.3 et il sera encore longuement détaillé dans la partie 4.2.3.

Adaptation au cours du temps

Les utilisateurs évoluent au cours du temps. Ils s'habituent aux technologies et on peut s'attendre à ce que, dans quelques années, les gens n'aient plus peur de parler à un SD et qu'en conséquence, certains messages d'aide deviendront complètement obsolètes. Il est difficile d'anticiper dès maintenant les usages que les SD rendront dans une dizaine d'années. Un SD qui a besoin d'être mis à jour, voire développé à nouveau chaque année a aussi un coût en termes de maintenance. L'adaptation au cours du temps rendue possible par l'apprentissage en ligne permettra au SD et à l'utilisateur de s'inter-adapter l'un l'autre jusqu'à atteindre un optimal commun dans leur communication. Cette fonctionnalité pourrait aussi rendre possible l'extension d'une application, ou même de commencer à utiliser des objets de dialogue pour le développement des SD. Ainsi, deux services hétérogènes pourrait être fusionnés automatiquement en un seul, parce que l'apprentissage en ligne apprendrait à alterner entre les services.

Comme l'adaptation peut être considéré comme une dimension de l'optimisation, dans la suite de ce chapitre, la fonctionnalité d'optimisation, par opposition à la fonctionnalité d'automatisation, inclut la propriété d'adaptation. Cependant, jusqu'à aujourd'hui, cette fonctionnalité apportée par l'apprentissage n'a été étudiée dans aucun travail de recherche. L'apprentissage en ligne est d'ailleurs très en retrait dans la littérature, comme Paek (2006) le regrette. Certains travaux peuvent néanmoins être facilement adaptés à cette problématique (Singh et al., 2002; Williams, 2008; Li et al., 2009; Baudoin, 2008).

4.2.2 Apprentissage hors ligne à base de MDP

Ces dernières années, beaucoup de chercheurs se sont intéressés aux mécanismes d'apprentissage pour implémenter les stratégies de dialogues des SD. La plupart d'entre eux,

ont concentré leurs efforts sur les techniques d'apprentissage par renforcement (Sutton & Barto, 1998) à base de Markov Decision Process (MDP). Le processus de conception automatique de stratégies de dialogue pour les SD à base de MDP est le suivant (voir l'architecture globale sur la figure 4.1, Pietquin, 2005) : collecte d'un corpus de dialogues client / opérateur humain, annotation du corpus, apprentissage d'un modèle utilisateur (Levin et al., 2000; Young, 2000; Georgila et al., 2005), utilisation du modèle utilisateur pour générer des millions de dialogues artificiels, apprentissage des stratégies de dialogue à partir de ces millions de dialogues (Levin et al., 1998; Singh et al., 1999; Young, 2000; Lemon & Pietquin, 2007). Dans le but d'avoir un modèle adapté au caractère non-observable du dialogue, certains chercheurs (Williams & Young, 2005; Young, 2006; Williams & Young, 2007; Henderson & Lemon, 2008) ont remplacé les MDP par des Partially Observable MDP. Des travaux plus récents se sont efforcés de résoudre des problèmes spécifiques au dialogue (Rieser & Lemon, 2006), de résoudre les problèmes spécifiques aux MDP dans les SD (Henderson et al., 2005) ou encore d'appliquer de nouveaux algorithmes d'apprentissage par renforcement pour l'apprentissage du modèle utilisateur (Geist et al., 2009) ou pour l'apprentissage du Dialogue Management lui-même (Gašić et al., 2009; Lefèvre et al., 2009). Ces travaux sur les MDP constituent une avancée indéniable en terme de réduction de coût de développement des SD. Cependant, ils font toujours face aux limitations suivantes soulignées par Pieraccini et Huerta (2005) et Paek (2006; 2007; 2008).

Fonctions objectifs. Les fonctions de récompense que le système cherche à maximiser sont difficiles à définir, parce que plusieurs dimensions hétérogènes sont en compétition : la satisfaction de l'utilisateur, le coût des ressources, l'accomplissement de la tâche, ...

Modélisation de l'espace des états Dans les SD à base de MDP actuels, beaucoup d'énergie est encore dépensée sur la construction de l'espace d'états. Il n'est donc pas question d'automatiser complètement le processus. Encore plus problématique, ajouter

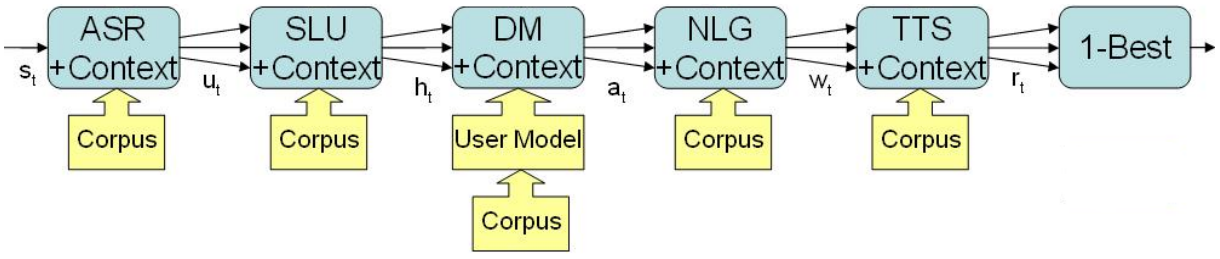


FIGURE 4.1 – L'architecture pour l'automatisation de la conception des SD, où le dialogue satisfait les contraintes des chaînes de Markov (Pietquin, 2005).

ou supprimer une variable d'état n'est pas simple et rend l'apprentissage du système précédent complètement obsolète. C'est une limitation grave à l'automatisation du processus et à la mise à jour ou la maintenance des SD.

Apprentissage en ligne de la politique L'apprentissage hors ligne à base de MDP ne semble pas compatible avec l'apprentissage en ligne, à cause de la différence d'ordre de grandeur des corpus entre les deux techniques d'apprentissage (on passe du million au millier) et à cause de la différence de nature de l'évaluation (l'évaluation en ligne est automatisée alors que l'évaluation pour apprentissage hors ligne est fournie par le modèle utilisateur, c'est en quelque sorte une généralisation des évaluations annotées dans le corpus initial).

Manque de lisibilité Les MDP sont durs à interpréter. Ceci implique que le comportement des SD ne peut pas être anticipé et que rien ne peut garantir la sûreté globale du SD au concepteur du service.

Manque de contrôle Les développeurs de service manquent de contrôle sur les SD à base de MDP. Un développeur peut être gêné par une décision que le système prendrait dans un état donné et il pourrait vouloir en forcer un autre. Cependant, ce genre de "tuning" de l'application de dialogue, qui devrait être l'opération de base, est très complexe dans les MDP et impose au développeur d'avoir de très bonnes connaissances en apprentissage par renforcement.

Pas une réponse pour l'optimisation Théoriquement, l'apprentissage hors ligne pourrait également servir à l'objectif d'optimisation présenté dans la sous-section précédente ; mais en fait, il apparait que les paradigmes appris sont très simples et auraient pu être spécifiés à la main par n'importe quelle personne très facilement. La raison principale pour cela est que les modèles utilisateur sont construits sur quelques centaines de dialogues dans les meilleurs cas, ce qui signifie que ces modèles sont très loin de pouvoir exprimer la variabilité des dialogues entre humains. Ensuite, l'apprentissage à partir de ces modèles sont par définition limités et probablement même sujet à du sur-apprentissage sur les dialogues originaux. La seconde raison est que la plus grosse partie de l'apprentissage concerne les mécanismes du dialogue qui paraissent évidents aux humains et que le reste ne peut pas être optimisé de la même manière. En d'autre mots, l'espace de stratégies de dialogue est trop vaste comparé à la taille du corpus initial et l'apprentissage ne peut aboutir qu'à une spécification très approximative de la politique optimale. Le résultat est que les SD automatiquement générés n'arrivent pas au niveau de performance des SD implémentés à la main (Thomson et al., 2008a).

4.2.3 Les systèmes de dialogue hybrides connaissance experte/apprentissage par renforcement

Jusqu'à maintenant, très peu de travaux en apprentissage dans les SD ont pris en compte les contraintes opérationnelles industrielles. Parmi ces rares publications, deux chemins ont été empruntés : une méthode d'anticipation rationnelle d'actions communicatives permettant de répondre aux besoins que l'utilisateur n'a pas encore exprimés (Baudoin, 2008) et une juxtaposition d'un processus de décision à base de MDP et d'un automate conventionnel dans un même SD (Singh et al., 2002; Williams, 2008; Li et al., 2009).

Une méthode d'anticipation rationnelle d'actions communicatives

Ce travail (Baudoin, 2008) constitue une expérience unique d'apprentissage supervisé (c'est-à-dire sans renforcement) dans la littérature. L'idée est de prédire l'action suivante de l'utilisateur à l'aide d'un prédicteur. Quand la prédiction est jugée fiable a priori, le système se permet de l'anticiper, par exemple en fournissant directement à l'utilisateur la réponse à la question qu'il allait poser, ou encore en proposant pro-activement une aide à l'utilisateur, quand le système pressent une erreur. Cependant, l'application de cette méthode est restreinte à une anticipation tour par tour et ne garantit en rien la maximisation de la fonction objectif. De plus, il ne traite qu'un problème particulier de l'optimisation de dialogue : l'anticipation d'une action utilisateur qui se traduit souvent dans le comportement du système par l'insertion de messages d'aide ou sur-information dans le périmètre de l'application.

La méthodologie d'apprentissage en ligne de Singh

Singh et al. (2002) a monté la première expérience d'optimisation en ligne d'un système de dialogue. Pour des raisons évidentes, il n'a pas pu laisser les utilisateurs interagir avec un système complètement aléatoire. La méthode retenue est donc, dans un premier temps, d'implémenter manuellement le système en lui laissant un petit ensemble exploratoire. Ensuite, le système apprend sur la base des dialogues avec des utilisateurs réels. Enfin, il peut exploiter la politique de dialogue, optimale selon ce que le système a pu observer jusqu'alors. La méthodologie complète est rappelée :

1. Choisir une mesure de récompense appropriée aux dialogues, une représentation correcte des états du dialogue et définir une politique de dialogue qui associe chaque état possible du dialogue à un ensemble d'actions raisonnables.

2. Construire un système initial d'entraînement basé sur des états qui crée un ensemble exploratoire d'états, qui essaie, plusieurs fois depuis chaque état, chacune des actions de l'ensemble exploratoire. Bien qu'exploratoire, ce système devra toujours fournir la fonctionnalité recherchée à la base.
3. Utiliser ces dialogues d'entraînement pour construire un modèle empirique à l'aide de MDP sur l'ensemble des états. Les transitions de ce MDP modéliseront les réactions de la population d'utilisateurs et les récompenses obtenues suites à chacune des actions du système.
4. Calculer la politique de dialogue optimale selon ce MDP.
5. Réimplanter le système en utilisant la politique de dialogue apprise.

L'utilisation de MDP n'est pas anodine, dans la mesure où cela impacte fortement le SD hybride :

- cela implique la définition d'une représentation à base d'états du dialogue. La citation suivante de Singh et al. (2002) exprime bien la limitation d'industrialisation imposée par l'utilisation de MDP : "We view the design of an appropriate state space as application-dependent and a task for a skilled system designer."[29]
- l'utilisation de MDP impose d'avoir un cadre formel Markovien, et à ce jour seul un modèle d'action au niveau du tour existe (Pietquin, 2005), alors que l'apprentissage se ferait idéalement plutôt au niveau des décisions lors de la conception du système de dialogue (quel feedback, quelle aide insérer, quelle question poser, quelles générations et quelles synthèses de tous ces actes, ...). Ceci implique également plusieurs limitations :
  - Difficultés à inclure les choix pris après le Dialogue Management (DM) dans l'apprentissage par renforcement. En effet les DM, NLG et TTS sont des modules développés par des groupes de recherche distincts, et même s'il est techniquement possible d'étendre le MDP pour couvrir la chaîne complète de production (voir section 1.2.2), à notre connaissance, à la date où nous écrivons ces lignes, seule Rieser a fait cet effort pour le NLG (Rieser et Lemon 2008a; 2008b; 2008c; 2009a; 2009b), sans toutefois aller jusqu'à la TTS (même si certains travaux semblent aller dans ce sens : Boidin et al., 2009b), ni faire l'intégration avec l'apprentissage au niveau du DM.
  - Difficultés liées à l'interprétation contextuelle. En effet, même s'il existe des POMDP capables de gérer les incertitudes, il n'existe aujourd'hui aucune technique d'interprétation contextuelle capable d'utiliser un POMDP. Il reste toutefois possible de résoudre les inconnues contextuelles dans le MDP également.

29. Nous voyons la conception d'un espace d'états approprié comme étant une tâche applicative assignée à un concepteur de système hautement qualifié.

– De façon plus importante, le problème principal impliqué par cette modélisation d'action au niveau du tour de dialogue est que les décisions du DM deviennent les décisions de la totalité du SD. En effet, comme les deux derniers points l'illustrent, ce type de DM a tendance à phagocyter les modules voisins, de manière à résoudre les différents problèmes liés à cette modélisation d'action. Cette expansion artificielle du DM a pour conséquence dangereuse de complexifier davantage l'apprentissage et de ne plus séparer les métiers relatifs à chaque module.

En conclusion, les principaux arguments à l'encontre de la solution proposée par Singh et al. (2002) sont la complexité additionnelle requise de conception d'une application de dialogue et la nécessité d'apprendre au niveau des mouvements de dialogue au lieu d'apprendre au niveau des alternatives elles-mêmes.

Jusqu'ici, seules trois publications ont proposé une implémentation de cette méthodologie. La première implémentation fait partie du même papier (Singh et al., 2002), tandis que les deux dernières (Williams 2008; 2009), plus récentes, proposent une adaptation de cette méthodologie aux POMDP .

Le système NJFun

NJFun(Singh et al., 2002) est une application vocale de dialogue qui fournit des informations sur les choses à faire dans le New Jersey. De façon informelle, le DM de NJFun demande successivement à l'utilisateur de renseigner les attributs d'activité, de localisation et de temps. NJFun demande d'abord à l'utilisateur de renseigner l'attribut courant (et possiblement les autres attributs, selon l'initiative). Si l'attribut courant n'est pas obtenu, NJFun demande une nouvelle fois de renseigner l'attribut (et possiblement les attributs suivants). Si NJFun ne reçoit toujours pas de valeur pour l'attribut courant, NJFun passe à l'attribut suivant. Quand NJFun reçoit avec succès une valeur, il peut soit confirmer que la valeur est bonne, ou demander directement l'attribut suivant. Les décisions prises par le MDP sont donc restreintes aux choix d'action suivants :

– le type d'initiative à utiliser quand on demande ou redemande la valeur d'un attribut,
– et s'il faut ou non confirmer la valeur d'un attribut.

Il y a théoriquement 960 états (mais seulement 62 qui peuvent vraiment être accédés dans un dialogue) au MDP que Singh définit selon sept variables :

– si le système a accueilli l'utilisateur,
– quel est l'attribut courant,
– la confiance du module de reconnaissance vocale sur la valeur de l'attribut (faible, moyenne ou forte) ou l'état de confirmation de l'attribut (confirmé ou non confirmé),
– si la valeur a été obtenue pour l'attribut courant,

– combien de fois l'attribut courant a été demandé,
– si une grammaire restrictive ou non-restrictive a été utilisée,
– s'il y a eu des problèmes avec un quelconque attribut précédent.

Parmi ces 62 états, pour seuls 42, plusieurs actions ont été définies. Le système explore aléatoirement les différentes actions sur ces états de dialogue, et il calcule une politique optimale selon son expérience.

La méthodologie de Singh adaptée aux POMDPs

L'implémentation de la méthodologie a été formalisé plus profondément dans les papiers de Williams (2008; 2009), et en conséquence, elle est moins dépendante de l'application de dialogue. Un DM conventionnel est utilisé pour générer un ensemble d'actes de dialogue alternatifs. En parallèle, un POMDP maintient un état des croyances (ou un état des connaissances probabilisées, c'est-à-dire une distribution probabiliste sur les états possibles du dialogue). A la fin de chaque tour de dialogue, le POMDP choisit la meilleure séquence d'acte parmi les alternatives présélectionnées par le DM conventionnel, et le met à jour, en fonction du choix pris et de son état courant. Cette mise à jour n'est pas très claire, particulièrement la façon dont le concepteur du DM conventionnel peut évaluer la dépendance entre le prochain état du DM conventionnel et l'état des croyances du POMDP.

Face à la complexité des SD, l'ensemble d'états est trop grand pour garantir une convergence avec des POMDP. C'est pourquoi une compression de ces états est effectuée. Par exemple, la distribution des croyances portant sur un champ donné sont résumées en la probabilité de la valeur la plus probable.

Trois systèmes sont ensuite comparés : un système conventionnel, un système à base de POMDP et un système hybride suivant la méthodologie de Singh. Les résultats montrent que le système hybride devient meilleur que le système conventionnel dix fois plus vite que le système à base de POMDP.

4.2.4 Analyse de la problématique et positionnement de nos travaux

Le problème du dialogue est défini sur un domaine vaste : les variables sont nombreuses, souvent continues et potentiellement toutes discriminatives. Pareillement, les actes de dialogue ont une forte combinatoire : idéalement, nous voudrions optimiser le feedback (pas de feedback, feedback implicite, feedback explicite), le type de question (question ouverte, question à choix multiple, question oui/non), la stratégie de dialogue (quelle question poser, quand continuer ou arrêter de poser une question, comment rattraper un incident de

dialogue, ...), l'insertion de messages d'aide (quand fournir de l'aide à utilisation, quelle aide, sous quelle forme), la génération (quels mots, quelle syntaxe), la synthèse (quelle expressivité, quelle voix, quel style de parole). Une manière de réduire la complexité inhérente au problème est d'aider les algorithmes d'apprentissage avec des règles expertes (conçues manuellement par le concepteur de l'application). Ces règles peuvent être la définition d'un sous-ensemble d'actes de dialogue possibles dans un contexte donné, comme vu dans la section 4.2.3 avec la méthodologie de Singh. Ces règles peuvent également être la définition des variables pertinentes pour un apprentissage localisé comme dans l'approche de Baudoin ou comme par exemple, en définissant que le choix des mots dans le NLG n'est dépendant que des mots qui ont été reconnus par l'ASR, de manière à profiter - ou non - l'effet d'amorçage (priming effect, Zurif, 1995).

Une des idées défendues dans cette thèse est que le choix d'une séquence d'actes dans un contexte donné est la conjonction de plusieurs décisions plus restreintes et plus ou moins indépendantes entre elles apparaissant à des niveaux différents de la chaîne de dialogue (voir chapitre 1). Dire que ces décisions sont indépendantes est de façon sûre une approximation, puisqu'il est impossible de connaître leurs impacts collatéraux. Cependant, nous considérons qu'il est raisonnable de faire cette approximation et que l'optimisation indépendante des décisions locales est équivalente à l'optimisation globale du système. Précisons toutefois que le concept d'indépendance utilisé ici concerne simplement les prises de décision locales. Ils se peut tout à fait que leurs états locaux se recouvrent et donc que deux décisions soient influencées par une même caractéristique du dialogue. Cependant, ces deux décisions sont prises indépendemment l'une de l'autre dans la mesure où elles ne sont pas prises simultanément en une décision globale. Nous illustrons cette prise de position par une analogie de la tâche du dialogue en une tâche simple d'apprentissage par renforcement sur un problème d'alignement entre le système et l'environnement :

- L'état du dialogue est une grille m-dimensionnelle $G = \bigotimes_{1 \leq k \leq m} [1..n_k]$, chaque dimension correspondant à un état local défini par les variables locales prises en compte pour les décisions locales. Un état local est donc décrit par la dimension G_k de la grille G. Cette dimension contient n_k états possibles.
- L'ensemble des actes de dialogue A correspondant à chacune des dimensions fait partie du simplexe de dimension m, et il est équivalent à m décisions d'actes A_k d'ordre 1 selon chaque dimension : $A = \bigotimes_{1 \leq k \leq m} A_k$.
- Un dialogue est un chemin emprunté dans cette grille. Il se termine, soit quand une action amène le chemin hors de la grille (position associée avec une récompense négative), soit quand la position atteinte est une des cases cibles (position associée à une récompense positive).
- La variabilité de l'utilisateur est modélisée par une déviation aléatoire du chemin à

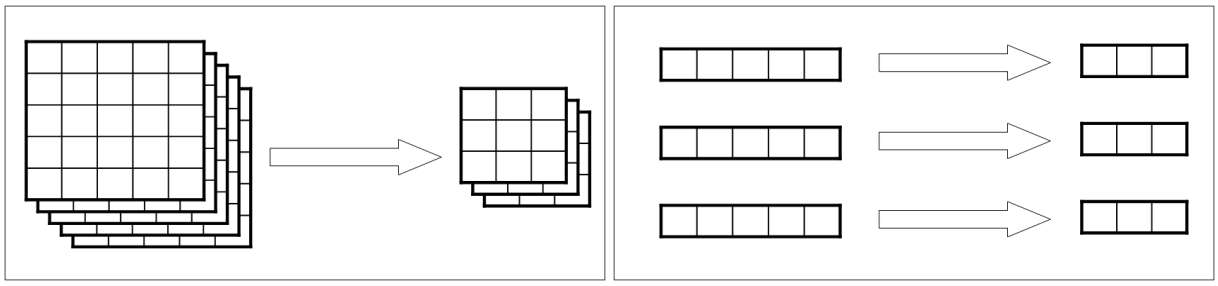


FIGURE 4.2 – Comparaison de la complexité entre l'apprentissage de $(y_1, y_2, y_3) = f(x_1, x_2, x_3)$ (sur la gauche) l'apprentissage de $y_1 = f_1(x_1)$, $y_2 = f_2(x_2)$ et $y_3 = f_3(x_3)$ (sur la droite).

chaque étape.

Ce problème exagère les préceptes d'indépendance entre les dimensions. En effet, une décision locale sur une dimension k n'affectera que la dimension k, ce qui n'est évidemment pas le cas pour le dialogue où chaque décision a un effet global sur l'interaction. La condition d'indépendance du dialogue ne fait que stipuler que la décision locale de dimension k ne dépend que que de l'état local de dimension k. Cette exagération de l'indépendance rend possible l'utilisation d'apprentissage markovien à l'aide d'un MDP dédié à chaque dimension avec l'analogie présentée ci-dessus. Nous nous interdisons l'exploitation de cette propriété pour traiter cette analogie, de manière à nous placer dans les conditions du dialogue.

Dans un problème tel que celui-ci, il est bien plus rapide d'apprendre sur chaque dimension k, la micro-police $\pi_k(x_k) \in A_k$ selon sa position actuelle x_k sur la dimension k que d'apprendre la politique générale $\pi(x) \in A = \bigotimes_{1 \leq k \leq m} A_k$ sur sa position $x = [x_1, \ldots, x_m]$. Le gain en complexité est le suivant : $\sum_{1 \leq k \leq m} n_k |A_k|$ au lieu de $\prod_{1 \leq k \leq m} n_k |A_k|$. Même sur un exemple simpliste de grille tridimensionnelle avec cinq états sur chaque dimension et trois actions possibles (les deux cases adjacentes et la case actuelle), on trouve une simplification de complexité de l'ordre de 75 : $\frac{5^3 * 3^3}{3*5+3*5+3*5}$ (voir la figure 4.2). En vérité, ce gain de complexité est atténué par le fait que les processus d'apprentissage interfèrent et ajoutent du bruit au système, mais ce gain est réel. Cette capacité à paralléliser l'apprentissage a également pour avantage de donner la possibilité de spécifier certaines dimensions à l'aide de règles expertes.

4.2.5 Conclusion de l'état de l'art

Cette section a abordé les différentes motivations pour l'apprentissage, puis elle a présenté les tentatives de l'état de l'art pour l'apprentissage dans les systèmes de dialogue, soit dans une optique d'automatisation, soit dans un but d'optimisation. Sur la base de

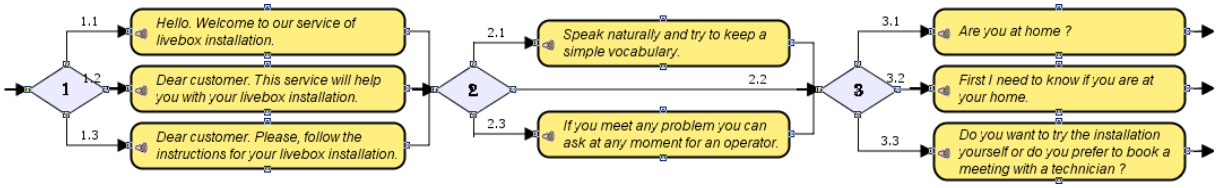


FIGURE 4.3 – Exemple de mouvement de dialogue composé de plusieurs décisions locales

ce qui a été présenté, une approche pour l'apprentissage à base de décisions locales a été introduite. Avant de passer à la définition du cadre formel, nous allons préciser cette approche et faire un tour d'horizon des différentes difficultés imposées par l'apprentissage dans un système de dialogue en général et pas cette approche en particulier. Nous expliquerons également quelles sont les forces et valeurs ajoutées de cette approche.

4.3 La méthode d'optimisation locale

Après avoir présenté notre approche pour l'apprentissage dans le chapitre 3, cette section la confronte aux critiques que Paek et Pieraccini ont pu adresser aux techniques d'apprentissage de la littérature actuelle. Il ressort de cette confrontation que la majorité des limitations sont levées. Il reste néanmoins de nombreuses difficultés techniques et scientifiques liées à notre approche, mais cette section conclut en montrant que ses valeurs ajoutées justifient l'effort réalisé sur ce point dans cette thèse.

4.3.1 Collecte des décisions et récompenses

Nous allons avoir besoin des divers concepts d'actes présentés dans la section 1.2.2. Les décisions prises par le système sont des actes internes (IA). Ils sont collectés au fur et à mesure de leurs utilisations, de manière plus ou moins asynchrone. En revanche les récompenses sont obtenues après chaque évènement externe et sa consommation (voir section 1.2.2) par le système. Ces deux types d'information, absolument nécessaires pour l'optimisation du système ne sont pas de manière générale synchronisés. Quand les décisions sont au niveau du choix du mouvement de dialogue comme dans le reste de la littérature, et lorsque le système de dialogue est relativement simple, la correspondance est facile : à une action correspond une récompense directe. Dans le cas général que nous explorons ici, ce n'est pas le cas. Il peut y avoir plusieurs tours de dialogue sans rencontrer aucune décision locale à optimiser et il peut y avoir plusieurs décision locales dans un même mouvement de dialogue, tel que dans l'exemple de la figure 4.3.

L'exemple de la figure 4.3 fait partie d'une application de dialogue en anglais pour l'aide à l'installation de la Livebox. Il correspond aux premiers mouvements de dialogue

	Strengths	Weaknesses
Objective function	• Optimization framework • Explicitly defined and adjustable • Can serve as evaluation metric	• Unclear what dialogues can and cannot be modelled using a specifiable objective function • Not easily adjusted
Reward function	• Overall behavior very sensitive to changes in reward	• Mostly hand-crafted and tuned • Not easily adjusted
State space and transition function	• Once modelled as MDP or POMDP, well-studied algorithms exist for deriving policy	• State space small due to algorithmic limitations • Selection is still mostly manual • No best practices • No domain-independent state variables • Markov assumption • Not easily adjusted
Policy	• Guaranteed to be optimal with respect to the data	• Removes control away from developers • Not easily adjusted
Evaluation	• Can be rapidly trained and tested with user models	• Real user behavior may differ • Creating user model may be more work than deploying prototype • Hand-crafted policy should be tuned to same objective function

FIGURE 4.4 – Un résumé des forces et faiblesses de l'apprentissage par renforcement dans la littérature (Paek & Pieraccini, 2008).

possibles pour le système de dialogue. Il est constitué de trois parties : le message d'accueil où le système se présente et salue l'utilisateur, un message d'aide pour faciliter l'utilisation du service vocal et une relance vers l'utilisateur pour guider l'interaction. Ces trois parties sont sujettes à incertitude de la part du concepteur du service et il souhaite utiliser plusieurs alternatives à chaque niveau, de manière à ce que le système découvre par l'expérience, les alternatives les plus performantes.

Il y a la nécessité d'associer les décisions et les récompenses à une même métrique temporelle, pour pouvoir les associer ensemble. L'idée retenue est que les décisions internes ne sont effectivement prises qu'à partir du moment où le mouvement de dialogue est effectué vers l'extérieur du système (c'est-à-dire l'utilisateur dans la plupart des cas), et que cela fait partie du processus d'engagement décrit dans la section 1.3.3. Avant cela, les décisions ne sont que des "bouts" de plans. Pour trouver une métrique commune, il suffit donc d'associer un "time stamp" à chacune des décisions. Le tour de dialogue est le time stamp qui sera utilisé dans la thèse, mais il est également envisageable de considérer un time stamp universel mesuré en secondes.

4.3.2 Positionnement par rapport aux critiques adressées par Paek et Pieraccini

Nous allons positionner la méthode par rapport aux diverses critiques qui ont pu être faites sur l'apprentissage à base de MDP classique, notamment par Pieraccini et Huerta (2005) et Paek (2006; 2007; 2008). La figure 4.4 rappelle ces critiques.

La définition des fonctions d'objectif est un point crucial pour l'apprentissage par renforcement, et c'est l'une des parties les plus problématiques pour le dialogue. En effet,

les systèmes de dialogue sont des objets complexes qui entraînent des interactions dont les indicateurs de performance (KPI pour Key Performance Indicators) sont multiples : les KPIs objectifs (durée du dialogue en seconde et en nombre d'interactions, erreurs de reconnaissance vocale, d'interprétation contextuelle, résolution de la tâche, etc...) et les KPIs subjectifs (satisfaction utilisateur, utilisabilité du service, qualité des renseignement, qualité perçue de la compréhension du système, etc...).

Il est intéressant de voir dans notre méthode que les fonctions d'objectifs ne sont pas forcément les mêmes pour tous les points de choix. Ainsi, le point de choix concernant la question posée pourra être évalué par la pertinence de la réponse vis-à-vis de la question, mais ce KPI n'aura probablement pas de valeur pour le message d'accueil par exemple. Il est toutefois dangereux de conditionner les décisions à des récompenses trop locales, car on risque de perdre l'objective global du service. Cependant, cela reste un levier très intéressant pour accélérer la convergence en fournissant aux différentes décisions des récompenses plus directes. L'étude des récompenses locales ne fait pas l'objet de cette thèse et nous considèrerons par la suite que les récompenses sont globales et qu'elles concernent tous les points de choix.

De plus, l'outil de retour d'usage décrit dans la section 3 est une aide importante pour donner au développeur un moyen de supervision de sa fonction objectif. Il peut en effet visualiser simultanément différents KPI dans l'interface et la fonction objectif définie pour l'évaluation automatique est révisable. C'est une nouvelle démonstration de la synergie entre les algorithmes d'apprentissage par renforcement et le travail de maintenance, monitoring ou supervision des développeurs dont nous avons déjà parlé en introduction de ce chapitre.

A propos de l'ensemble d'états et de la fonction de transition, le formalisme décrit dans la présentation de NJFun (Singh et al., 2002) constitue déjà une avancée, puisqu'il permet de réduit de façon critique le nombre d'états et de sélectionner seulement les variables qui sont pertinentes pour ce que l'on cherche à apprendre. La description de l'implémentation de Williams (2008; 2009) est moins précise et ne le fait pas explicitement, mais nous pouvons supposer que ce travail a été réalisé dans la fonction de compression et que c'est la raison première du gain de rapidité observé par rapport au système POMDP. Notre approche va encore un peu plus loin en permettant à un concepteur sans qualification en apprentissage par renforcement de faire ce travail. La contrainte Markovienne a été relaxée mais c'est plus un coût qu'une avancée, dans la mesure où nous ne disposons plus des algorithmes classiques d'apprentissage par renforcement (voir la sous-section 4.3.3).

Alors que les SD à base de MDP se comportent comme une boîte noire, la politique apprise via notre méthode se situe au niveau local, ce qui garantit un pouvoir explicatif des choix que l'algorithme fait. En plus de cela, une convergence entre la conception et

le retour d'usage est possible en visualisant dans l'interface graphique de conception les différentes performances obtenues par chacune des alternatives (voir le chapitre 3). Plus que de laisser le contrôle aux développeurs, notre approche fournit un nouvel outil de retour d'usage à l'aide duquel les développeurs pourront apprendre de nouvelles pratiques et les ergonomes tester leurs théories.

Comme notre approche est une approche d'apprentissage en ligne, les faiblesses liées à l'évaluation ne sont plus vraiment pertinentes. Mais permettons nous d'ajouter un nouvel avantage à notre approche qui consiste à permettre l'évaluation du système dans sa totalité, c'est-à-dire la chaîne de dialogue complète (voir chapitre 1), alors que les techniques à base de MDP ne font qu'optimiser la partie Dialogue Management.

4.3.3 Difficultés techniques

L'estimation automatique de la fonction d'objectif

De manière à apprendre en ligne, il faut être capable d'évaluer automatiquement le succès ou l'échec d'un dialogue. Le SD doit être capable de calculer une estimation des fonctions objectives. La complétion de la tâche ne peut être évaluée que si le SD connaît le but utilisateur, mais la prise de connaissance de ce but est en général le principal prérequis à l'accomplissement de la tâche, sinon le seul. Il est donc en général impossible d'évaluer la complétion de la tâche. De même, la satisfaction utilisateur est souvent indétectable par le système. En bref, l'automatisation du calcul de la fonction d'objectif est incomplète et imprécise, et ceci vient s'ajouter au fait que la fonction d'objectif est aussi souvent mal définie. Néanmoins, cette situation complexe peut être améliorée par de l'évaluation locale sur des objectifs précis, en orientant la conception du système pour rendre possible l'évaluation.

On peut imaginer, par exemple, dans un service de prise de rendez-vous que le système demande explicitement de valider le rendez-vous qui vient d'être fixé. L'utilisateur doit faire cette validation s'il souhaite avoir ce rendez-vous, et le système utilise cette validation comme l'évaluation de l'accomplissement de la tâche à laquelle il a été assigné. Encore mieux, l'objectif de la question dans la figure 4.5 est de transférer l'appel. Le SD a une évaluation précise et complète de cette tâche, indépendamment de l'intention de l'utilisateur ou de sa satisfaction pour le service.

Équilibre entre la conception initiale, l'exploration et l'exploitation

Du point de vue utilisateur, il n'est pas acceptable que le système de dialogue ne soit pas en mesure de fournir la fonctionnalité de base du service. C'est la raison pour laquelle le SD ne peut pas faire une exploration totale en ligne et que la conception du

système de dialogue doit garantir que le système ne fera pas des mouvements de dialogue non pertinents. En conséquence, nous avons une nouvelle contrainte forte interdisant l'apprentissage sans conception préalable. Ainsi, l'automatisation de la conception du SD par apprentissage en ligne est à proscrire. Et il ne peut être qu'un outil pour l'optimisation actuellement (voir section 4.2.1).

Il est difficile de savoir a priori comment doser la part de conception initiale et d'optimisation. C'est un dosage qui est propre à chaque application de dialogue et qui est dépendant de la fréquentation du service et de la variabilité des performances à chaque point de choix. Ce dernier point est le plus difficile à estimer.

Il est également important de savoir équilibrer les phases d'exploration et d'exploitation dans la mise en service du système de dialogue. L'objectif est de maximiser l'espérance de récompense totale. Pour ce faire, il est nécessaire d'explorer pour maximiser l'apprentissage, mais aussi d'exploiter ce que le système a déjà appris. C'est un sujet de recherche assez fourni et nous allons comparer plusieurs techniques dans la phase d'expérimentation de la section 5.4.

Évaluation des algorithmes d'apprentissage en ligne pour les systèmes de dialogue

Les algorithmes d'apprentissage en ligne ne peuvent pas être comparés dans des conditions réelles, parce que ces conditions réelles ne sont pas reproductibles. En effet, un dialogue ne peut servir à l'évaluation de plusieurs algorithmes, puisque le fait de suivre la politique d'un algorithme ou celle d'un autre, fait déjà partie du processus d'apprentissage. Pour résoudre partiellement ce problème, nous avons implémenté une simulation du problème que nous traitons dans le dialogue, similaire à la description de l'analogie de la section 4.2.4. L'étude des algorithmes est réalisée sur la base de ce problème dans le chapitre 5.

Un formalisme non markovien

L'idée de notre approche est d'apprendre au niveau des décisions locales de conception. Les principaux objectifs sont d'une part de limiter les variables sur lequel l'algorithme apprend et d'autre part d'établir une convergence entre les paradigmes de conception, d'exécution et d'évaluation des systèmes de dialogue.

La figure 4.3 est un cas pratique de limitation des variables de dépendance. Si on utilisait un MDP classique au niveau des mouvements de dialogue, l'ensemble d'états serait le produit des variables de dépendances définies pour chaque décision locale et l'ensemble des actions serait le produit des alternatives définies à chaque point de choix. La

complexité rendrait alors le problème intraitable avec un nombre de dialogue raisonnable. De manière à résoudre cette "curse of dimensionality"[30], nous considérons l'apprentissage au niveau local. Mais cela rompt l'hypothèse markovienne. En effet, l'état local atteint lors du point de choix concernant l'insertion du message d'aide n'est pas dépendant que de la paire état-action de la décision prise lors du choix du message d'accueil. Comme nous l'avons vu dans la section 4.2.4, la figure 4.2 illustre bien l'amélioration de complexité apportée par notre approche.

Il est bien sûr possible de formaliser les décisions locales dans un processus markovien (en augmentant la complexité du processus markovien, soit en augmentant la taille de son espace d'états, soit en rendant le processus d'ordre n, ce qui revient finalement au même). Cependant, ce n'est pas la voie que nous avons retenue. En effet, l'une de nos premières motivations est justement de réduire la complexité du problème en s'affranchissant des contraintes markoviennes. Les résultats obtenus dans la section 5.2 apportent la preuve du bien fondé d'une telle position.

Comme le processus de décision n'est pas markovien, dans un état donné, on ne peut plus faire l'hypothèse que l'on est capable d'anticiper le prochain état atteint. En conséquence, il n'est pas possible d'utiliser des méthodes de bootstrapping classique (Sutton & Barto, 1998), c'est-à-dire les méthodes qui mettent à jour les estimations d'espérance sur la base des estimations des états suivants. Les meilleurs algorithmes d'apprentissage par renforcement utilisent du bootstrapping, tels que Dynamic Programming (Bellman, 1957a; Si et al., 2004) ou Temporal Difference Learning (Sutton, 1988). Seules les méthodes Monte Carlo (Metropolis & Ulam, 1949; Fishman, 1996; Sutton & Barto, 1998) ne font pas de bootstrapping. Nous proposons dans cette thèse deux algorithmes distincts. Le premier est une amélioration de l'algorithme de Monte Carlo (voir sous-section 4.5.1). Le second calcule une estimation de la performance de l'état de dialogue courant pour permettre l'utilisation de techniques de bootstrapping (voir sous-section 4.5.2).

Dans la même idée, les algorithmes d'apprentissage par renforcement hiérarchiques (HRL Barto & Mahadevan, 2003), parmi lesquels l'exploitation de structure (Boutilier et al. 1995; 1996), la minimisation du MDP (Dean & Givan, 1997), les hiérarchies de machines HAM (Parr & Russell, 1997), l'algorithme MAXQ (Dietterich, 1998), les Options (Sutton et al., 1999) ou les actions concurrentes (Rohanimanesh et Mahadevan 2001; 2003), ne permettent pas de modéliser le problème que nous soulevons, puisqu'ils n'ont jamais été conçus pour faire face à la limite de bootstrapping que nous rencontrons. Comme notre formalisme, les Semi-MDP (Sutton et al., 1999) ne sont pas markoviens, mais ils continuent de reposer sur une structure markovienne (d'où leur nom "semi" markovien) qui n'existe plus du tout dans notre cas. Cette structure markovienne sous-jacente

30. Malédiction de la dimensionnalité.

permet aux Semi-MDP d'utiliser des techniques de bootstrapping classiques.

Il est donc nécessaire de développer de nouveaux algorithmes d'apprentissage par renforcement non markoviens pour notre problème. Ceux-ci ont une convergence plus lente que les algorithmes utilisant du bootstrapping, mais le gain en complexité apporté par la séparation des variables la compense très largement comme les sections 5.2 et 5.3 en attestent.

4.3.4 Valeurs ajoutées

La section précédente a évoqué les différentes difficultés imposées par nos choix. Cette section montre les atouts apportés par notre approche. Ceux-ci seront évalués dans la suite de la thèse.

Séparation des dépendances

La séparation des dépendances est une des justifications principales de notre approche : traiter l'apprentissage à un niveau local permet de limiter les dépendances et ainsi de réduire la complexité de l'optimisation. Les sections 5.2 et 5.3 démontrent que le gain en complexité compense largement la perte des outils d'apprentissage basés sur l'hypothèse markovienne.

Facile à implémenter

Lors de la totalité de la thèse, nous gardons à l'esprit les contraintes industrielles. L'une d'elles consiste à faire en sorte que les applications de dialogue puissent être développées par des ingénieurs sans spécialisation particulière. En d'autres mots, le concepteur d'une application de dialogue n'est pas un expert en Machine Learning et les outils que nous proposons ne doivent requérir aucune compétence particulière dans ce domaine. Notre approche permet cela dans la mesure où les questions qui sont posées au concepteur sont intuitives : "quel est l'ensemble des alternatives qu'il souhaite tester ?", "quel est l'ensemble des variables à partir desquelles la décision doit être prise ?" et "comment puis-je évaluer automatiquement la qualité du dialogue ?". La sous-section 6.1.2 montre sur l'exemple de l'expérimentation Livebox que la conception des alternatives reste simple, contrairement aux paramétrages des techniques d'apprentissage sur les mouvements de dialogue avec des MDP.

Retour d'usage

Une autre préoccupation industrielle concerne les outils de monitoring et de reporting offerts aux concepteurs des applications. Le fait d'apprendre au niveau des choix de

conception, permet une visualisation de l'apprentissage intégrée à l'interface de conception. De cette manière, le retour d'usage est directement interprétable par les concepteurs et les chefs de projet. Le chapitre 3 décrit cet outil de supervision.

4.4 Un formalisme de décision adapté au problème

Fidèlement à l'architecture pour l'automatisation de la conception des SD (voir la figure 4.1), la nouveauté proposée dans cette section est de tirer avantage de l'expertise du concepteur de SD. Le concepteur spécifie la plupart des stratégies de dialogue et laisse certains points de décision non spécifiés, ce qui n'est ni plus ni moins qu'une condition "switch/case" dont la condition de test est inconnue telle que celle de la figure 4.5. Ensuite, le SD pourra apprendre comment se comporter dans ces cas d'indécision. Un cadre formel tel que celui-ci nous contraint à définir un nouvel algorithme d'apprentissage par renforcement compatible avec l'apprentissage en ligne dans un SD partiellement spécifié.

4.4.1 Une nouvelle approche pour l'apprentissage dans les SD

Il a été observé dans les travaux précédents sur l'apprentissage de stratégie de dialogue, que la plupart des paradigmes appris paraissaient évidents pour un humain, et encore plus pour un téléconseiller. L'idée principale développée dans cette section est que les comportements évidents doivent être spécifiées à la main par le concepteur de l'application et que les décisions non-évidentes doivent être prises sur la base de l'expérience du système. Utiliser l'apprentissage dans le but de satisfaire les besoins de l'utilisateur est une bonne chose, mais quel est l'intérêt d'apprendre des choses qu'un concepteur humain pourrait spécifier en très peu de temps ? Cette partie fait le postulat que, dans un SD, il y a certaines décisions qui ont un fort impact sur la qualité de dialogue et qui ne sont pas évidentes à prendre pour le concepteur. La figure 4.5 reprend l'exemple que nous avons déjà évoqué précédemment. Si certaines décisions ne sont pas évidentes, la meilleure façon de les prendre est de laisser le système s'adapter au service grâce à son expérience. L'automatisation de cette optimisation serait une grande avancée pour les SD, parce que les défauts de spécification sont la limitation principale de performance des SD conventionnels. La figure 4.5 fait suite à une réalité observée avec une application de dialogue commerciale : un gain de transfert significatif a été réalisé simplement en changeant la génération du message "do you want to be transferred ?" [31] à "if you want to be transferred, say : yes" [32]. Cet exemple concret montre que le postulat est bien fondé

31. Voulez-vous être transféré ?
32. Si vous voulez être transféré, dites oui.

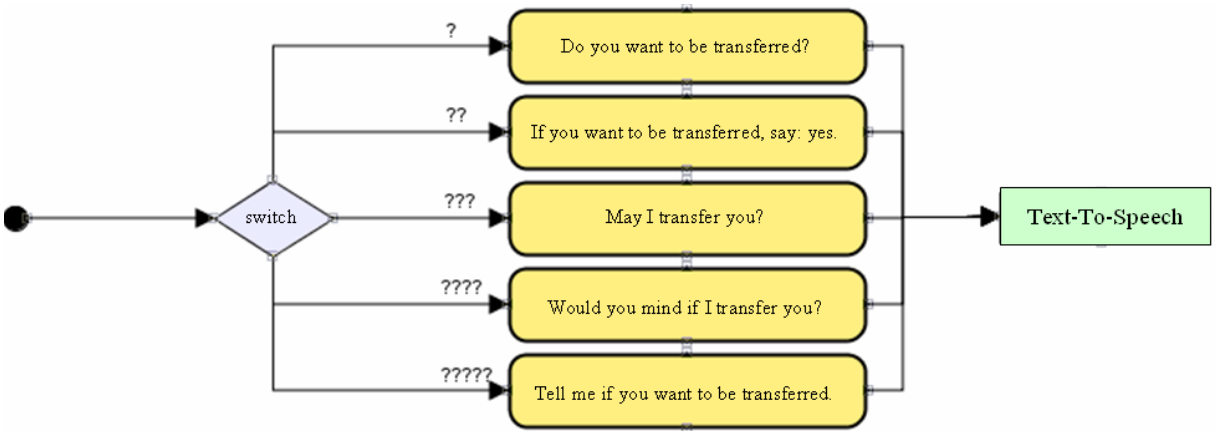


FIGURE 4.5 – Exemple de point de décision non spécifié : le concepteur ne sait pas quelle formulation est la plus efficace pour avoir l'accord de transfert d'appel.

et que des améliorations significatives peuvent être réalisées automatiquement grâce à de l'apprentissage en ligne.

L'apprentissage par renforcement est la famille d'algorithmes qui s'impose d'elle-même pour l'optimisation du dialogue. En effet, la performance d'un dialogue correspond à une collecte de récompenses obtenues lors d'une interaction avec un environnement. Les performances des actions sont non seulement dépendantes de l'environnement, mais également des actions que le système effectuera par la suite.

4.4.2 Rappels sur les Processus de Décision Markoviens

Cette sous-section introduit les Processus de Décision Markoviens (MDP Bellman, 1957b) comme modèle pour l'apprentissage par renforcement. Un MDP est un quadruplet (S, A, P, R) où :

- S est l'ensemble des états
- A est l'ensemble des actions
- $P : S^2 \times A \mapsto [0, 1]$ est une fonction qui définit la probabilité qu'une action a réalisée dans un état s cause une transition vers l'état s' où $a \in A$ et $s, s' \in S$
- $R : S \mapsto \mathbb{R}$ est une fonction qui définit la récompense immédiate associée avec l'état s.

Dans chaque état, le but est de maximiser la récompense à long-terme. Celle-ci est calculée de la manière suivante :

$$r = \sum_{t=0}^{\infty} \gamma^t R\left(s_t\right) \tag{4.1}$$

De manière à ne pas planifier des récompenses dans un temps infini, il y a généralement un facteur $0 \leq \gamma \leq 1$ appelé le coefficient de réduction. $\gamma = 0$ et $\gamma = 1$ sont les cas extrêmes où l'on cherche à maximiser respectivement la récompense immédiate et la récompense à horizon infini. L'objectif de l'apprentissage est d'apprendre à maximiser l'espérance de récompense à long-terme.

Le MDP peut être modélisé en évaluant la fonction d'état $V : S \mapsto \mathbb{R}$ qui mesure l'espérance de récompense quand un état s est atteint. La fonction d'état-action $Q : S \times A \mapsto \mathbb{R}$ peut également être calculée :

$$V(s) = R(s) + \gamma \max_a \sum_{s'} P_a(s, s') V(s') \tag{4.2}$$

$$Q(s, a) = R(s) + \gamma \sum_{s'} P_a(s, s') V(s') \tag{4.3}$$

Ensuite, la politique d'action exploitant le MDP $\pi : S \mapsto A$ est celle qui maximise la fonction Q :

$$\pi(s) = \operatorname*{argmax}_a Q(s, a) \tag{4.4}$$

4.4.3 L'Architecture de Conception

Pour illustrer les différents concepts, nous considérons que l'architecture du SD est un automate augmenté d'un contexte local et global, c'est-à-dire respectivement un ensemble de variables locales et de variables globales. Chaque état de l'automate correspond à un module de décision. Un état du monde, comme dans la définition d'état des MDP est un état de l'automate et des contextes. Chaque décision mène à un autre état et possiblement à une mise à jour du contexte. Cette décision peut faire suite à un simple test, par exemple une variable x qui serait positive ou négative, ou après un traitement lourd, par exemple par l'Automatic Speech Recognition (ASR).

Cependant la théorie n'est pas limitée à ce type d'architecture : elle est facilement transposable au systèmes logiques à base de règles.

4.4.4 Le Module-Variable Decision Process

Comme nous l'avons vu dans la sous-section 4.3.3, l'approche que nous avons du problème est non-Markovienne. Les MDP ne peuvent donc être utilisés tels quels. Cette sous-section décrit un nouveau formalisme de processus de décision : Module-Variable Decision Process (MVDP). Ce formalisme a été publié (Laroche et al. 2009c; 2009a).

Un *module* est une unité de calcul qui peut choisir une *alternative* en fonction de son *contexte* (défini par un ensemble de variables). Le module est équivalent à un état de l'automate de conception. Le couple module/contexte n'est pas équivalent à l'état du

système de dialogue, parce que seules certaines variables sont visibles dans un module donné. En effet le contexte que l'on considère est local.

Chaque module suit une *politique*. Cette politique gouverne les choix qui sont faits à l'intérieur du module. Une politique est une fonction à partir de l'espace des contextes vers l'espace des alternatives : $\pi_{module}\,(contexte) \mapsto alternative$. Un module est dit *apprenant* quand sa politique, ou de manière équivalente sa fonction d'état-action varie au court de sa vie.

Une politique prend des *décisions*. Une décision est constituées des champs suivants : le module qui a pris la décision, le contexte qui a permis de prendre la décision, l'alternative choisie lors de la prise de décision et l'instant auquel la décision a été prise.

$$decision = (module, contexte, alternative, instant) \tag{4.5}$$

Un *épisode* est une chaîne de décisions. Chaque épisode commence avec la décision de démarrer le dialogue et finit par la décision de terminer le dialogue. Un *sous-épisode* est une sous-chaîne dde décisions commençant par une décision quelconque et terminant avec la décision de terminer le dialogue. Formellement, les besoins de spécification experte imposent de définir le cadre Module-Variable Decision Process (MVDP) qui est un quadruplet (M, V_M, A_M, T) où :

- M est l'ensemble des modules
- V_M est l'ensemble des variables pertinentes (ou autrement appelé le contexte local) d'un module donné $m \in M$
- A_M est l'ensemble des actions possibles (ou alternatives) dans un module donné $m \in M$
- T est l'échelle de temps. Il sert à synchroniser les décisions prises avec les récompenses obtenues. Généralement, il est soit exprimé en numéro de tour de dialogue soit en temps universel (en secondes). Nous considérons dans toute la thèse que $T \subseteq \mathbb{R}$.

Dans l'exemple de la figure 4.5, le module apprenant est le bloc "switch-case". Les contextes V_m peuvent être par exemple l'âge et/ou le niveau de langage de l'utilisateur. Les cinq possibles alternatives A_m sont les cinq transitions.

Contrairement aux MDP, la fonction de transition P n'est pas explicitée dans les MVDP. En effet, comme le MVDP n'est pas Markovien, la fonction P n'est d'aucune utilité. De même, la fonction de récompense R ne fait pas partie du modèle. En fait, comme nous l'avons vu dans la sous-section 4.3.3, les récompenses ne sont pas reçues lorsque les états décisionnels sont atteints. Bien qu'une récompense immédiate (même nulle) soit reçue à chaque tour de dialogue, certains mouvements de dialogue peuvent être composés de plusieurs décisions de modules apprenants alors que d'autres mouvements de dialogue

sont le fruit d'une stratégie entièrement implémentée manuellement sans traverser aucun module apprenant. C'est la raison pour laquelle, il est nécessaire de définir un modèle de récompense qui soit un couple (R, T) :

- T est l'échelle de temps partagée avec le MVDP.
- $R_t \in \mathbb{R}$ est la récompense immédiate reçue à l'instant $t \in T$

De la même manière que pour les MDP, à chaque décision, le but est de maximiser la récompense à long-terme. La formule 4.1 est légèrement modifiée pour prendre en compte la spécificité temporelle imposée par les MVDP. Pour chaque décision d, on définit la récompense à long terme :

$$r_d = \sum_{t_k > t_d}^{\infty} \gamma^{t_k - t_d} R_k \tag{4.6}$$

De même, dans le but de construire sa politique, le module peut (mais n'y est pas contraint) générer une *fonction d'état-action* qui prédit la récompense à horizon infini étant donné le contexte courant et le choix d'une alternative $Q_\pi(v, a) \mapsto \mathbb{R}$. Q_π est dépendante de π parce que de manière générale, l'espérance de récompense à long terme est dépendante des décisions qui seront prises plus tard. La politique d'exploitation cherche à maximiser l'espérance de récompense :

$$\pi_m(v) = \underset{a}{\operatorname{argmax}} \left(Q_{m,\pi}(v, a) \right) \tag{4.7}$$

Dans le reste du chapitre, la politique considérée est toujours la politique d'exploitation, c'est-à-dire que le système cherchera toujours à optimiser les récompenses du dialogue courant, sans prendre en compte l'intérêt que peut susciter l'exploration de nouvelles décisions. Ainsi, le compromis exploration/exploitation n'est pas discuté pour l'instant, mais il le sera plus tard dans la section 5.4.

4.5 Algorithmes d'apprentissage pour le MVDP

Deux algorithmes d'apprentissage par renforcement ont été implémentés pour le MVDP :

- Le Compliance-Based Reinforcement Learning (CBRL) qui est une amélioration de l'algorithme de Monte-Carlo (Metropolis & Ulam, 1949; Fishman, 1996; Sutton & Barto, 1998) qui prend en compte lors de l'apprentissage par renforcement, le fait que les décisions sont évaluées à travers le bruit des décisions prises plus tard lors de l'épisode. Pour réduire ce bruit, nous évaluons les distances entre ces décisions et la politique actuelle de l'agent. Cet algorithme a été publié (Laroche et al. 2009c; 2009a).
- Le Module-Variable Temporal Difference Learning (MVTDL) qui compense l'impossibilité de faire directement du bootstrapping (voir section 4.3.3) en implémentant

un prédicteur de la performance de l'état courant du dialogue. Cet état courant est alors utilisé pour faire du bootstrapping de manière indirecte dans le formalisme MVDP.

4.5.1 Le Compliance-Based Reinforcement Learning

Cadre formel

Comme expliqué dans la section 4.3.3, il n'est pas possible de prédire le couple module / contexte suivant une décision, parce que le MVDP ne rend compte que de l'information locale dont dispose le module qui prend la décision : $d = (m, v_m, a_m, t)$. Néanmoins, il est possible d'implémenter un algorithme qui calcule pour un épisode ou un sous-épisode donné s'il referait les mêmes choix sur la base de sa politique courante. L'algorithme proposé dans cette section note et organise les épisodes e en provenance du corpus de dialogue selon leur compatibilité $c_\pi(e)$ avec la politique actuelle π. La compatibilité (ou "compliance") est un réel négatif qui vaut 0 quand la politique actuelle est identique à celle utilisée lors de la génération de l'épisode. Le calcul de la compatibilité $c_\pi(e)$ d'un sous-épisode $e = \left(d_0, d_1, ..., d_{|e|-1}\right)$ avec une politique π se déroule comme ceci : pour chaque décision $d_k = (m_k, v_k, a_k, t_k)$ dans le sous-épisode (excepté d_0), l'algorithme demande au module m_k, sa compatibilité locale $c_{\pi,m_k}(d_k)$ avec la politique courante π. Ensuite, une formule similaire à celle de la récompense à long terme (voir l'équation (4.1)) est proposée :

$$c_\pi(e) = \sum_{t_k > t_0}^{\infty} \beta^{t_k - t_0} c_{\pi,m_k}(d_k) \tag{4.8}$$

Le calcul de la compatibilité locale ne fait pas partie de l'algorithme d'apprentissage par renforcement : il est distribué dans les modules qui sont supposés être capables d'évaluer la qualité d'une décision étant donné un politique locale, c'est-à-dire de fournir à l'algorithme d'apprentissage par renforcement la compatibilité locale $c_{\pi,m_k}(d_k)$. La fonction d'état-action Q_{m_k} est celle qui est utilisée de façon classique :

$$c_{\pi,m_k}(d_k) = Q_{m_k}(v_k, a_k) - \sup_{a \in A_{m_k}} Q_{m_k}(v_k, a) \tag{4.9}$$

Cependant, les modules qui ne disposent pas de fonction d'état-action peuvent aussi évaluer la compatibilité d'une décision. Par exemple, la compatibilité canonique :

$$\pi(m_k, v_k) = a_k \rightarrow c_{\pi,m_k}(d_k) = 0 \tag{4.10}$$

$$\pi(m_k, v_k) \neq a_k \rightarrow c_{\pi,m_k}(d_k) = -1 \tag{4.11}$$

Comme ces compatibilités sont supposées se combiner à un niveau global, il faut garder à l'esprit que ces compatibilités ont besoin d'être normalisées d'un module à un

autre et que cela peut être très complexe. Les espérances de récompense délivrées par les fonctions Q sont une mesure objective de la compatibilité et c'est une nouvelle raison pour recommander l'utilisation de ces fonctions Q dans les modules.

Néanmoins, le CBRL n'impose pas ce calcul de fonction d'état-action. La sélection d'exemples est simplement fondée sur la politique et l'utilisation de la fonction Q est pratiquement restreinte aux calculs de la politique et de la compatibilité grâce aux équations (4.4) et (4.9). En conséquence, la politique peut être aussi bien calculée avec un algorithme de classification.

L'algorithme est donc capable de fournir à chaque module m le corpus *pour apprentissage supervisé* $s_k = (m, v_k, a_k, r_k, c_\pi(e_k))$ où :

- e_k est un sous-épisode qui commence avec la décision $d_k = (m, v_k, a_k, t_k)$
- v_k est le contexte du module m ($v_k \in V_m$)
- a_k est l'alternative choisie lors de la décision d_k ($a_k \in A_m$)
- r_k est la récompense obtenue à la fin du dialogue associée au sous-épisode e_k, calculé avec l'équation (4.1).
- $c_\pi(e_k)$ est la compatibilité à la politique π du sous-épisode e_k

Chaque module est ensuite capable d'apprendre sur la base du corpus qui lui est fourni. Il est notamment de la charge de chaque module de sélectionner le niveau de compatibilité nécessaire à chaque individu du corpus pour apprentissage supervisé pour qu'il soit pris en compte. De même, le module peut associer des poids aux individus en relation avec leur compatibilité. Cela fait partie des décisions qui sont prises au niveau du module. Ces modules n'ont pas besoin d'avoir un fonctionnement homogène. L'implémentation générique est développée dans la prochaine partie et l'implémentation basique sur le schéma de l'algorithme Monte Carlo sera également proposée dans la sectionsuivante.

L'algorithme générique

Le but du Compliance-Based Reinforcement Learning (CBRL) est de ne faire aucune supposition sur le fonctionnement interne des modules. L'algorithme aura la tâche basique de fournir aux modules les individus du corpus pour apprentissage supervisé de ce type : $s_k = (m, v_k, a_k, r_k, c_\pi(e_k))$ que nous réécrivons pour plus de simplicité $s = (m, v, a, r, c)$. De manière à rendre ceci possible, quatre fonctionnalités sont supposées exister au niveau de chaque module :

- Le module doit avoir une politique, c'est-à-dire qu'il doit être capable de décider une alternative quand on le place dans un contexte donné.
- Un module apprenant doit être capable d'apprendre à partir d'individus du type (v, a, r, c).
- La fonctionnalité la plus contraignante est que les modules sont censés être capables

Algorithm 2 Algorithme générique Compliance-Based Reinforcement Learning

Initialisation de l'ensemble des épisodes $E = \emptyset$
Initialisation de la fonction d'état-action $Q_m(v, a) = 0$
loop
  Générer un ensemble d'épisode E' en utilisant $Q_m(v, a)$
  for all $e_i \in E'$ **do**
    for all $d_{ij} \in e_i$ **do**
      Calcul de la récompense à long terme r_{ij} suivant la décision d_{ij} {Equation 4.6}
    end for
  end for
  $E \leftarrow E \cup E'$
  for all $e_i \in E$ **do**
    for all $d_{ij} = (m_{ij}, v_{ij}, a_{ij}, t_{ij}) \in e_i$ **do**
      Calcul de la compatibilité locale c_{ij} {Equation 4.9 ou équations ou 4.10 et 4.11}
      Calcul de la compatibilité à long terme {Equation 4.8}
    end for
  end for
  for all $m \in M$ **do**
    Mise à jour de la politique π_m grâce à l'apprentissage supervisé sur le corpus $D_m = \{s_k = (m, v_k, a_k, r_k, c_\pi(e_k))\}$, le sous-ensemble des décisions prises par le module m.
  end for
end loop

d'évaluer la compatibilité locale d'un précédente décision d avec leur politique actuelle π_m. Si ce besoin n'est pas satisfait, l'algorithme peut appliquer par défaut la formule de compatibilité canonique (voir les équations (4.10) et (4.11)).

Le fonctionnement algorithmique du CBRL est explicité dans l'algorithme 2. La mise à jour de la politique π_m n'y est pas explicité [33]. En effet, l'algorithme CBRL rassemble toute une famille d'algorithme, dont les algorithmes supervisés de mise à jour diffèrent. Pour cette mise à jour, nous pouvons envisager toutes les techniques d'apprentissage supervisées capables de prendre en compte des pondérations. Cela va de la simple moyenne pondérée de type Monte Carlo à des algorithmes plus complexes tels que les arbres de régression (Breiman et al., 1984).

Plutôt que d'effectuer l'apprentissage sur tous les modules en parallèle tel que l'al-

33. Un choix de mise à jour de π_m sera explicité dans l'algorithme 3.

gorithme 2 le propose, il est également possible de le faire de manière partielle. Comme chaque module a probablement une place fixe dans la stratégie de dialogue, certains modules peuvent être plus terminaux que d'autres (le point de décision de la figure 4.5 est probablement terminal par exemple). Il semble intéressant de commencer l'apprentissage par renforcement avec l'apprentissage supervisé du module le plus final. Ce genre de considération est intéressant dans le cas où les capacités calculatoires sont réduites. Il est également envisageable d'actualiser les politiques de certains modules plus souvent que d'autres, notamment pour prendre en compte le fait que certains modules puissent prendre plus de temps à recalculer leur politique. L'avantage d'effectuer en parallèle les mises à jour des modules, comme dans l'algorithme 2, est de factoriser le calcul des compatibilités locales. Ces considérations d'ordonnancement de l'apprentissage sont un champ de recherche inexploré pour l'instant. Dans la suite de la thèse, l'algorithme que nous considèrerons est la version Monte Carlo du CBRL que nous introduisons dans la partie suivante.

La méthode Monte-Carlo pour le CBRL

L'algorithme 3 a été publié (Laroche et al., 2009a).

On peut appliquer le CBRL en considérant la compatibilité comme un moyen de pondérer les individus pour l'apprentissage supervisé. Si les ensembles d'état et d'action sont discrétisés et que l'on dispose de suffisamment d'individus pour chacun d'entre eux, il est alors possible d'estimer la performance d'un couple état-action en faisant la moyenne pondérée des performances obtenues à chacune des décisions du corpus d'apprentissage. En conséquence, nous effectuons ici une adaptation de la méthode Monte Carlo (Metropolis & Ulam, 1949; Fishman, 1996; Sutton & Barto, 1998) pour y insérer une pondération aux "retours" des expériences passées. La principale contrainte imposée par cette approche concerne la nécessité de discrétiser les ensembles de variables et d'actions, alors que le CBRL ne le requiert pas en général. Plus précisément, une fois le corpus $\{v_k, a_k, r_k, c_k\}_{k\in[1,n]}$ généré, l'apprentissage est réalisé de la manière suivante :

$$Q_m(v,a) = \frac{\sum_{\substack{v_k=v \\ a_k=a}} r_k \omega(c_k)}{\Omega_m(v,a)} \tag{4.12}$$

$$\Omega_m(v,a) = \sum_{\substack{v_k=v \\ a_k=a}} \omega(c_k) \tag{4.13}$$

Où $\omega(\cdot)$ est une fonction strictement croissante sur $]-\infty, 0]$ satisfaisant les égalités suivantes $\lim_{x\to-\infty} \omega(x) = 0$ et $\omega(0) = 1$. Le choix de cette fonction est débattu dans la section 5.2.1. $\omega(\cdot)$ exprime l'impact de la compatibilité sur les pondérations accordées à

Algorithm 3 Algorithme Monte-Carlo du Compliance-Based Reinforcement Learning

Initialisation de l'ensemble des épisodes $E = \emptyset$
Initialisation de la fonction d'état-action $Q_m(v, a) = 0$
loop
  Générer un nouvel épisode e en utilisant $Q_m(v, a)$: $E \leftarrow E \cup \{e\}$
  for all $d_k \in e$ **do**
    Calcul de la récompense à long terme r_k {Equation 4.6}
  end for
  for all $e_i \in E$ **do**
    for all $d_{ij} = (m_{ij}, v_{ij}, a_{ij}, t_{ij}) \in e_i$ **do**
      Calcul de la compatibilité locale c_{ij} {Equation 4.9}
      Calcul de la compatibilité à long terme {Equation 4.8}
    end for
  end for
  Mise à jour de la fonction d'état-action $Q_m(v, a)$ {Equation 4.12}
end loop

chaque individu. $\Omega_m(v, a)$ est le poids total du corpus pour le triplet module-variable-action (m, v, a). Cette valeur dénote en quelque sorte l'expérience du système dans cette configuration.

L'intérêt de cette version de l'algorithme CBRL est sa simplicité et son faible coût computationnel, qui reste néanmoins élevé en comparaison des algorithmes d'apprentissage par renforcement reposant sur l'hypothèse Markovienne. L'ordre de grandeur des systèmes de dialogue actuels nous permet de mettre à jour la politique en temps réel à la fin de chaque dialogue.

De plus, si le coût de cette mise à jour reste trop importante, il est possible de faire des approximations de mise à jour de la politique sans recalculer les compatibilités de toutes les décisions. Le calcul approximatif d'une valeur X est ainsi dénoté par l'écriture $X^{\approx}$. Ainsi pour chaque décision $d = (m_d, v_d, a_d, t_d)$ prise dans le dernier dialogue, si r_d est la récompense à long terme après la prise de décision d, $Q^{\approx}_m(v, a)$ et $\Omega^{\approx}_{v,a}$ sont mis à jour de la façon suivante :

$$Q^{\approx}_{m_d}(v_d, a_d) \leftarrow \frac{\Omega^{\approx}_{m_d,v_d,a_d} Q^{\approx}_{m_d}(v_d, a_d) + r_d}{\Omega^{\approx}_{m_d,v_d,a_d} + 1} \tag{4.14}$$

$$\Omega^{\approx}_{m_d}(v_d, a_d) \leftarrow \Omega^{\approx}_{m_d}(v_d, a_d) + 1 \tag{4.15}$$

L'idée de base est de considérer que la politique évolue peu, et que le dernier épisode généré a suivi la politique courante. Les algorithmes 2 et 3 sont *off-policy*, c'est-à-dire qu'ils

sont conçus pour apprendre la politique optimale même lorsque la politique apprise n'est pas suivie lors de la génération de nouveaux épisodes. Cependant la simplification de la mise à jour que nous proposons ici n'est valable que si la politique de génération d'épisodes est la politique courante. Il est toutefois possible d'adapter cette simplification de mise à jour pour qu'elle soit utilisable *off-policy*. Pour cela, il suffit de remplacer la pondération par 1, par la pondération $\omega(c_k)$ de la compatibilité des dernières décisions. La complexité de ce calcul reste très raisonnable. Voici la formule de mise à jour approximative, dans le cas où la politique suivie n'est pas la politique actuelle (par exemple en cas d'exploration).

$$Q^{\approx}_{m_d}(v_d, a_d) \leftarrow \frac{\Omega^{\approx}_{m_d,v_d,a_d} Q^{\approx}_{m_d}(v_d, a_d) + \omega(c_d)\, r_d}{\Omega^{\approx}_{m_d,v_d,a_d} + \omega(c_d)} \tag{4.16}$$

$$\Omega^{\approx}_{m_d}(v_d, a_d) \leftarrow \Omega^{\approx}_{m_d}(v_d, a_d) + \omega(c_d) \tag{4.17}$$

La compatibilité c_d est calculé à l'aide de la formule 4.8. La différence avec l'algorithme 3 est que les compatibilités des décisions des anciens épisodes ne sont pas recalculées.

4.5.2 Le Module-Variable Temporal Difference Learning

Rappelons que, comme le processus de décision n'est pas markovien stricto sensu, on dispose de trop peu d'information pour anticiper l'état atteint en fonction de l'état précédent et de l'action produite. En conséquence les techniques de bootstrapping ne peuvent pas être directement appliquées au problème.

De manière à surmonter cette difficulté, nous faisons l'hypothèse suivante : il est possible de définir un prédicteur du succès du dialogue. Même si les modules n'ont qu'une vue locale du système et si les prédictions qu'ils peuvent fournir n'ont aucune valeur dans l'absolu, il est toujours possible de construire un algorithme d'apprentissage dont le but est de prédire la performance du système dans un état global donné. Cette hypothèse permet l'utilisation de techniques d'apprentissage par renforcement utilisant du bootstrapping tel que le Temporal Difference Learning (TD-Learning, Sutton, 1988; Garcia & Serre, 2000). Cette technique permet donc d'évaluer des couples état-action sans attendre la fin d'un épisode ou même de recevoir une récompense.

La première partie de cette sous-section rappelle les fondations du TD-Learning et ses implantations les plus connues. La seconde partie fait l'adaptation de ces techniques au cadre formel MVDP (voir section 4.4.4).

Etat de l'art

Le TD-Learning est généralement appliqué aux MDP (voir partie 4.4.2) et il repose sur l'équation 4.1. L'idée de base est de la définir comme relation de récurrence puis de

la décliner en la développant à plusieurs ordres :

$$r(t) = R_{t+1} + \gamma r(t+1) \tag{4.18}$$

$$r(t) = R_{t+1} + \gamma R_{t+2} + \gamma^2 r(t+2) \tag{4.19}$$

$$\dots$$

$$r(t) = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n r(t+n) \tag{4.20}$$

Où γ est le coefficient de réduction, $r(t)$ est la récompense à long terme obtenue après le tour t et R_t est la récompense immédiate reçue au tour t. Ces équations définissent les récompenses à horizon n, mais elles sont équivalentes et elles peuvent toutes être obtenues directement à partir de l'équation de définition 4.1. En conséquence, il est possible d'écrire de nouvelles façons de calculer $r(t)$ en faisant une moyenne pondérée des précédentes équations :

$$r(t) = \sum_{k=1}^{n} (1-\lambda)\lambda^{k-1} r_k(t) + \lambda^n r_n(t) \tag{4.21}$$

Où $0 \leq \lambda \leq 1$ est un paramètre et $r_n(t)$ est le calcul de la récompense à horizon n. La méthode de bootstrapping utilisée dans le TD-Learning consiste à faire une approximation des termes $r(t+k)$ à l'aide d'un prédicteur $V(s_{t+k})$ de l'espérance de récompense $E(r(t+k))$ à long terme dans l'état s_{t+k}. L'apprentissage se fait alors de proche en proche en ajustant le prédicteur état-action que constitue la fonction $Q(s,a)$, à l'aide de l'équation suivante :

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(t) - Q(s,a)) \tag{4.22}$$

Où α est un paramètre exprimant l'impact que l'on souhaite donner à chaque nouvelle expérience. Généralement, α décroît au fur et à mesure du temps. $\alpha = 0$ correspond au cas où le système n'apprend plus des nouvelles expériences qu'il obtient. A l'inverse, $\alpha = 1$ correspond à un système qui oublie tout ce qu'il connaissait auparavant pour considérer que la bonne prédiction pour le futur est celle que l'on vient d'observer. De même, les valeurs extrêmes de λ correspondent à des comportements bien connus : $\lambda = 0$ mène aux algorithmes Q-Learning (Watkins, 1989) ou SARSA (Rummery & Niranjan, 1994), selon la nature du prédicteur $V(s_{t+1})$ (la politique d'exploitation pour le Q-Learning et la politique utilisée pendant la génération pour SARSA) et $\lambda = 1$ (et n maximal) ne nécessite aucun bootstrapping et est équivalent aux retours de Monte Carlo (Metropolis & Ulam, 1949; Fishman, 1996; Sutton & Barto, 1998).

Adaptation du TD-Learning au MVDP

La condition principale pour l'utilisation du TD-Learning est d'avoir un moyen d'estimation de la récompense à long terme dans un état donné, c'est-à-dire d'être capable

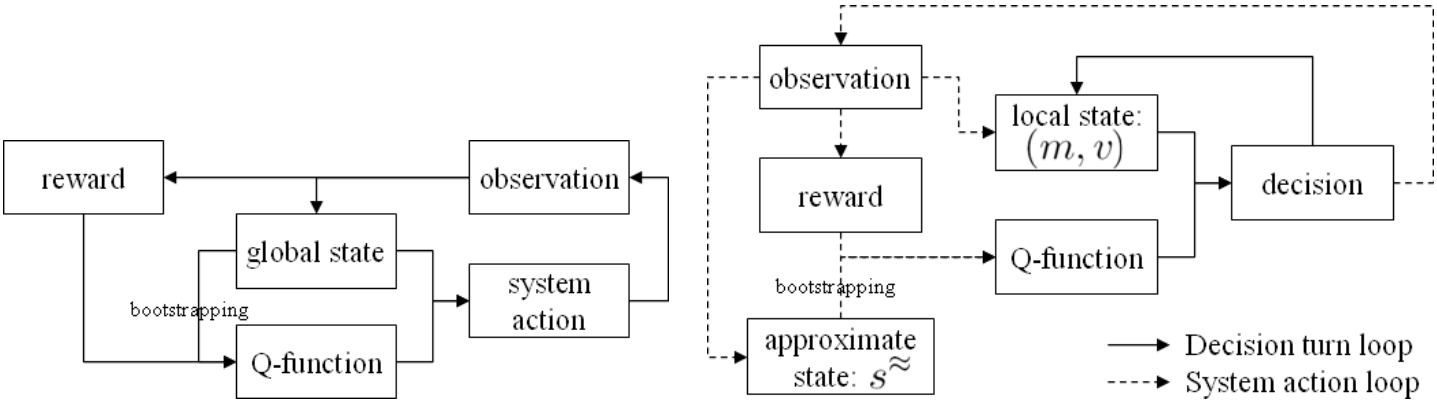
de générer un prédicteur $V(s_{t+k})$. Mais, comme nous l'avons déjà rappelé au début de cette sous-section, le cadre formel imposé par le formalisme MVDP ne contient pas par nature de prédicteur objectif de la récompense à long terme. Les fonction $Q(m, v, a)$ qu'il peut générer ne peut pas fournir ce type de connaissance dans la mesure où v_t et v_{t+1} n'appartiennent pas au même domaine de définition. En conséquence, et contrairement aux algorithmes classiques basées sur des MDP de TD-Learning, l'acteur et le critique ne peuvent pas être la même entité dans le MVDP. L'idée défendu dans cette sous-section consiste à la création d'un module qui accomplit la tâche du critique. Ainsi, le MVDP joue le rôle d'acteur et utilise ce module "critique" pour évaluer ses actions.

Après n'importe quelle action externe choisie par le système, et la réponse de l'environnement, le module "critique" fait l'estimation de la qualité de l'état courant. Ceci constitue le prédicteur de la récompense à long terme $V(s_{t+k})$. Généralement, et encore plus particulièrement pour les applications de dialogue, il est très complexe de concevoir manuellement le module "critique". Nous proposons donc, dans cette sous-section, d'utiliser un algorithme d'apprentissage pour que le module apprenne à évaluer sa performance à l'aide de son expérience. Le rôle du concepteur de l'application de dialogue se résume donc à la construction de l'espace d'état sur lequel le module "critique" apprendra. Étant donné la complexité des systèmes de dialogue, il sera sans doute préférable de faire une sélection parmi les variables pertinentes. Nous appelons cette sélection, le résumé du système. L'état résumé est noté $s^{\approx}$ pour un état global s. L'algorithme que nous utilisons pour l'apprentissage du module "critique" est un simple algorithme Monte Carlo (Metropolis & Ulam, 1949; Fishman, 1996; Sutton & Barto, 1998), qui effectue la moyenne pour un état résumé donné de toutes les récompenses à long terme obtenues lors des passages précédents dans cet état résumé.

La figure 4.6a montre la boucle du TD-Learning dans une modélisation par MDP. Selon l'état global, la politique génère une action vers l'environnement. Cette action mène à une nouvelle observation, c'est-à-dire un mouvement de dialogue utilisateur. Cette observation est traduite en une récompense directe et un nouvel état global qui peut être utilisé pour faire du bootstrapping.

Dans le cadre formel MVDP, l'état local ne peut plus être utilisé pour faire du bootstrapping. En plus de cela, du point de vue du MVDP, il y a deux boucles distinctes : une basée sur les décisions internes impliquant une mise à jour des états locaux et une basée sur les actions émises et reçues par le système, impliquant des mises à jour de l'état global. La figure 4.6b illustre ces deux boucles dont le module "critique" fait partie. C'est ce module "critique" qui est capable d'opérer un bootstrapping à partir de l'état résumé $s^{\approx}$.

Si nous utilisons le Q-Learning (c'est-à-dire quand $\lambda = 0$), après la réception d'un



(a) Schéma du MDP-based TD-Learning.

(b) Schéma du MVDP-based MVTD-Learning.

FIGURE 4.6 – Comparaison entre les structures d'apprentissage du TD-Learning dans les MDP et MVDP.

énoncé utilisateur à l'instant t, plusieurs modules m_i choisissent une actions locale $a_i(t)$ selon leur politique locale :

$$a_i(t) = \pi_i(s_i(t)) = \underset{a}{\operatorname{argmax}} Q_i(s_i(t), a) \tag{4.23}$$

Une fois que toutes ces décisions nécessaires sont prises, le système produit une réponse à l'utilisateur, qui lui-même réagit. L'énoncé utilisateur déclenche une mise à jour de l'état interne du système. Ensuite, l'état interne $s^{\approx}_{t+1}$ est évalué à l'aide de son résumé et du module "critique" qui fait une estimation de l'espérance de récompense à long terme $V\left(s^{\approx}_{t+1}\right)$. Enfin, les fonctions d'état-action Q_i sont mises à jour de la manière suivante :

$$Q_i(s_i(t), a_i(t)) \leftarrow (1-\alpha) Q_i(s_i(t), a_i(t)) + \alpha\left(R_{t+1} + \gamma V\left(s^{\approx}_{t+1}\right)\right) \tag{4.24}$$

L'algorithme 4 donne une implémentation simple de la famille des algorithmes MVTDL, où $\lambda = 0$ et l'algorithme d'apprentissage pour le module "critique" est Monte Carlo.

4.5.3 Comparaison de la complexité des deux algorithmes

Le chapître 5 fera une comparaison plus complète des algorithmes, à la base de tests sur des données artificielles. Mais nous pouvons d'ores et déjà opposer les algorithmes sur des considérations théoriques.

Sur le plan de la complexité des algorithmes, le CBRL est en $o(nb_{dialogues} * nb_{decisions})$, puisqu'à la fin de chaque dialogue, il faut recalculer les compatibilités de chaque décision. Ensuite, en ce qui concerne l'apprentissage supervisé, c'est évidemment dépendant de l'algorithme utilisé, mais il existe des algorithmes en $o(nb_{decision})$ comme la méthode

Algorithm 4 Algorithme Module-Variable avec critique Monte Carlo

Initialisation de la fonction d'état-action $Q_m(v,a) = 0$
Initialisation du module "critique" $V(s^{\approx}) = 0$
Initialisation du décompte des passages par les états résumés $n_{s^{\approx}} = 0$
for all dialogue e **do**
  Initialisation de l'ensemble de décisions initiales $D_0 = \emptyset$
  Initialisation de l'ensemble des états résumés rencontrés $S^{\approx} = \emptyset$
  Initialisation des récompenses immédiates reçues $R = \emptyset$
  for all tour de dialogue t **do**
    Calculer l'état résumé $s_t^{\approx}$ de l'état du système s_t
    $S \leftarrow S \cup \{s_t\}$
    Obtenir la récompense immédiate R_t
    $R \leftarrow R \cup \{R_t\}$
    Estimer l'espérance de gain à long terme : $V(s_t^{\approx})$
    for all les décisions d_i de D_{t-1} **do**
      $Q_i(s_i(t-1), a_i(t-1)) \leftarrow (1-\alpha) Q_i(s_i(t-1), a_i(t-1)) + \alpha(R_t + \gamma V(s_t^{\approx}))$
    end for
    Initialiser l'ensemble des décisions prises dans le tour $D_t = \emptyset$
    for all $d_{t,i}$ de D_t **do**
      $D_t =\leftarrow D_t \cup \{d_{t,i}\}$
    end for
  end for
  Initialiser la récompense à long terme $r = 0$
  for all tour de dialogue t dans l'ordre décroissant **do**
    $n_{s_t^{\approx}} \leftarrow n_{s_t^{\approx}} + 1$
    $V(s_t^{\approx}) \leftarrow \dfrac{n_{s_t^{\approx}} V(s_t^{\approx}) + r}{n_{s_t^{\approx}}}$
    $r \leftarrow \gamma r + R_t$
  end for
end for

Monte Carlo que l'on a présentée dans la partie 4.5.1, ce qui fait que l'ordre total d'apprentissage reste le même. L'utilisation de la mise à jour approximative introduite dans la même partie permet potentiellement de réduire l'ordre de complexité de l'algorithme, mais c'est au détriment de sa performance. La section 5.5 discute la complexité du CBRL plus en profondeur. Le MVTDL comprend deux algorithmes : le TD-Learning (fonction "acteur") qui est une simple mise à jour à chaque décision est de l'ordre $o(nb_{decisions})$; le

module "critique" est quant à lui dépendant de la technique d'apprentissage utilisé, mais la version que nous avons proposé avec la méthode Monte Carlo est d'ordre $o\left(nb_{decisions}\right)$. Au final, le MVTDL est donc moins complexe que le CBRL.

Du point de vue du concepteur de système de dialogue, le CBRL, en comparaison du MVTDL, est moins contraignant : il ne nécessite pas l'implantation d'un module "critique" difficile à bien définir en pratique.

4.6 Conclusion

Ce chapitre a introduit les besoins des SD pour l'apprentissage en ligne. Ensuite il a fait un état de l'art des techniques d'apprentissage qui ont été utilisées dans les systèmes de dialogue et a conclu que l'effort a principalement porté sur l'apprentissage hors ligne dans le but de réduire les coûts de développement plutôt que l'apprentissage en ligne dans le but d'améliorer les fonctionnalité de dialogue du SD. Nous avons adopté la seconde approche. Ensuite, nous avons montré qu'une architecture hybride expertise/apprentissage était appropriée pour tirer avantage des expériences combinées du développeur et du système. Finalement, il a été avancé que les MDP ne correspondaient pas à l'architecture et un nouveau cadre formel : le MVDP a été présenté, pour lequel deux familles d'algorithmes d'apprentissage par renforcement ont été proposées. Le chapitre suivant fait la comparaison entre ces algorithmes à l'aide d'une expérimentation sur des données artificielles.

Chapitre 5

Comparaison et optimisation des algorithmes d'apprentissage sur des données artificielles

Ce chapitre a pour objectif de fournir des optimisations et des tests extensifs de deux implémentations (algorithmes 3 et 4) des méthodes d'apprentissage par renforcement CBRL et MVTDL. Nous commençons par optimiser les algorithmes, puis nous les comparons aux algorithmes de la littérature et enfin nous les étudions les uns par rapport aux autres. Pour finir le chapitre, les sections 5.4 et 5.5 font les études respectives des stratégies d'exploration et de réduction de la complexité du CBRL.

Dans la littérature des systèmes de dialogue, la plupart des articles testent leurs algorithmes à partir de simulations utilisateur (Rieser et Lemon 2008b; 2008d). Cependant, nous préférons faire notre étude sur un exercice à la fois simplificateur et généralisateur de la problématique pour les raisons suivantes : (1) l'exercice est un problème dont l'approche théorique est simple et donc les analyses des comportements des différents algorithmes peuvent être plus facilement réalisées, (2) l'exercice produit une preuve universelle de l'intérêt de l'approche, du modèle MVDP et des algorithmes CBRL et MVTDL, (3) les simulations utilisateurs sont de toute façon trop simples pour garantir une évaluation tangible des algorithmes pour un vrai système de dialogue et (4) les applications de dialogue sont diverses et ne se ressemblent pas, ce qui implique que la démonstration de l'efficacité d'un algorithme dans une application donnée n'est pas une preuve de l'efficacité de cet algorithme pour une autre application.

Nous considérons dans ce chapitre un exercice de parcours d'un pointeur dans une grille m-dimensionnelle. Cet exercice nous permet de tester et comparer les algorithmes rapidement et en grand volume et qui reflète au mieux la problématique des systèmes de dialogue, selon l'analyse faite au préalable dans la sous-section 4.2.4.

5.1 Protocole expérimental

Cette section décrit le protocole expérimental suivi pour les expérimentations réalisées dans ce chapitre. L'environnement de l'exercice est décrit. Ensuite, les paramètres de cet exercice sont spécifiés. Puis, la modélisation de l'exercice dans le MVDP est explicité. Enfin, la section se termine par la description des algorithmes d'apprentissage utilisés pour le benchmark de nos algorithmes (Q-Learning et Monte Carlo pour le MVDP).

5.1.1 Description de l'environnement

L'environnement est une grille m-dimensionnelle bornée. Un agent se déplace librement dans cette grille. Chacun de ses déplacements correspond à une action de l'agent vers l'environnement. Ce déplacement peut l'amener hors de la grille. Dans ce cas, l'épisode est considéré comme ayant échoué et il s'arrête. A l'inverse, une case de la grille est la *case cible* et lorsque cette case est atteinte l'épisode est un succès et il s'arrête également.

Choix d'expérimentations

Pour notre expérimentation, nous considérons une grille tridimensionnelle, chaque dimension comprenant cinq cases dont les indices sont les entiers entre 1 et 5. La case cible est la case de coordonnées $(4, 4, 4)$. Le système a donc $5 * 5 * 5 - 1 = 124$ [34] états non-terminaux possibles. Selon chacune des dimensions, l'agent pourra décider de monter (c'est-à-dire d'ajouter $+1$ à sa coordonnée), descendre -1 ou rester sur place 0. À chaque instant, l'agent possède donc $3 * 3 * 3 = 27$ actions possibles. Un bruit gaussien [35] fort est appliqué sur chacune des dimensions, si bien que l'agent n'a que 32% de chances d'atteindre la case attendue [36].

L'attribution des récompenses est $+1$ en cas de succès et -1 en cas d'échec. De manière à encourager les itinéraires les plus courts, un facteur de réduction de 0.9 a été appliqué (voir les formules 4.1 et 4.6). Chaque épisode démarre avec l'agent placé de façon aléatoire dans la grille.

Comme l'agent peut vite apprendre à ne plus chuter hors de la grille sans pour autant savoir comment atteindre la case cible, chaque épisode est arrêté à l'instant 50 (sans aucune récompense) si l'agent n'a pas atteint un état terminal auparavant. Les différents

34. La case cible est terminale et fait partie de la grille.

35. Pour chaque dimension, une déviation suivant une loi gaussienne centrée en 0 est ajoutée. La valeur ainsi calculée est arrondie de manière à obtenir un entier correspondant à la coordonnée de la case atteinte.

36. Par exemple, si la position est $(1, 2, 5)$ et que l'agent choisit l'action $(+1, +1, -1)$, il n'aura que 32% de chance d'arriver sur la case $(2, 3, 4)$.

algorithmes sont testés sur des *runs* de 500 épisodes, sachant que les meilleurs algorithmes convergent à partir de 250 épisodes.

De manière à ne pas biaiser la comparaison entre les algorithmes d'apprentissage, une méthode d'exploration identique a été appliquée pour chaque algorithme. La probabilité d'exploration à chaque décision est calculée de la façon suivante [37] :

$$P(exploration) = 0.5 * 0.995^{|E|} \tag{5.1}$$

Une fois que l'exploration a été choisie, une décision ou action est prise au hasard sur la base des performances respectives des différentes décisions, de telle manière que l'action optimale a encore plus de chance d'être choisie que les sous-optimales. Ce choix, ne se basant que sur les valeurs Q, est fait de la même façon pour tous les algorithmes.

La performance optimale théorique du problème, sans exploration, est 0.63.

5.1.2 Factorisation du problème selon le MVDP

La modélisation en MDP est très simple : 124 états non terminaux et 27 actions en chacun de ces états. Le MVDP, quant à lui, permet de factoriser les dimensions et de définir des dépendances restreintes entre les décisions de chaque dimension.

A chaque tour, l'agent prend trois décisions locales depuis les trois dimensions de l'exercice. Les décisions sont prises en charge par des modules différents : m_1, m_2 et m_3. Pour chaque module m, le système a cinq états locaux possibles $V_m = \{s_{m,1}, s_{m,2}, s_{m,3}, s_{m,4}, s_{m,5}\}$, correspondants aux coordonnées 1, 2, 3, 4 et 5 de la dimension m. De même, pour chaque dimension, l'agent prend une décision locale parmi les trois choix : monter, descendre et rester sur place et ainsi nous obtenons pour chaque module m, $A_m = \{a_{m,-1}, a_{m,0}, a_{m,+1}\}$.

Cet exercice reflète les caractéristiques du dialogue de la manière suivante :

- Espace de haute dimension : le problème est fortement parallèle et un design des indépendances (ne faire dépendre la décision sur la dimension m que sur la position relative à la même dimension m) permet de réduire fortement la complexité du systèmes.
- Faible nombre d'épisodes/dialogues : le banc de test se focalisera sur la rapidité de convergence plutôt que sur son coût computationnel.
- Problème non markovien quand la parallélisation est appliquée : chaque dimension ne garde qu'un contexte local qui ne lui permet pas de prédire le prochain état atteint. Pour l'exercice que nous proposons, il serait possible d'utiliser un MDP

37. En fait, l'utilisation de deux modèles de processus différents : MDP et MVDP rend impossible la définition d'une méthode d'exploration équivalente. Nous avons fait au mieux pour que le Q-Learning, qui est le seul algorithme testé sur des MDP, ait une méthode d'exploration semblable. Sa mauvaise performance relative n'est pas causée par sa différente méthode d'exploration.

pour chacune des dimensions en parallèle, et ainsi obtenir avec le Q-Learning un apprentissage plus performant que les autres algorithmes. Si cela est possible pour l'exercice, cela ne l'est pas dans un système de dialogue où la plupart des modules ne sont rencontrés qu'une fois par dialogue. C'est la raison pour laquelle ces solutions n'ont pas été envisagées.

– Pas de prédicteur du succès du dialogue : mais il est relativement facile de spécifier le module "critique" utilisé dans le MVTDL.

L'objectif principal de cet exercice est d'étudier la gain obtenu avec la parallélisation du problème tel qu'on l'avait défini dans la figure 4.2.

5.1.3 Algorithmes témoins

De manière à positionner nos algorithmes par rapport aux travaux existants, nous utilisons deux algorithmes. Le premier, l'algorithme Q-Learning, sert de référence pour mesurer l'intérêt de la parallélisation du problème, rendu possible par le formalisme MVDP. Le second, la méthode Monte Carlo, est utilisé sur le formalisme MVDP de manière à mesurer le gain en rapidité d'apprentissage apporté par nos algorithmes, par rapport à ce qui existe dans la littérature en matière d'apprentissage non-Markovien.

Algorithm 5 Q-Learning

Initialisation du paramètre α
Initialisation de la fonction d'état-action $Q(S, A) = 0$
for all épisode **do**
  Initialiser l'état S_0
  for all tour d'action t, jusqu'à ce qu'un état final soit observé **do**
    Choisir une action A_t étant donné l'état S_t selon $Q(S_t, A_t)$ et la politique d'exploration.
    Faire l'action A_t.
    Observer une récompense immédiate R_t et l'état S_{t+1} atteint.
    Mise à jour de la fonction Q :
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t + \gamma Q(S_t + 1, A_t) - Q(S_t, A_t))$
  end for
end for

Algorithm 6 Monte-Carlo classique

Initialisation de l'ensemble des épisodes $E = \emptyset$
Initialisation de la fonction d'état-action $Q_m(v, a) = 0$
loop
  Générer un nouvel épisode e en utilisant $Q_m(v, a)$ et la politique d'exploration :
  $E \leftarrow E \cup \{e\}$
  for all $d_k \in e$ **do**
    Calcul de la récompense à long terme r_k {Equation 4.6}
    Mise à jour du poids total associé au triplet (m_k, v_k, a_k) :

$$\Omega_{m_k}(v_k, a_k) \leftarrow \Omega_{m_k}(v_k, a_k) + 1$$

    Mise à jour de la fonction d'état-action Q :

$$Q_{m_k}(v_k, a_k) \leftarrow \frac{Q_{m_k}(v_k, a_k) + r_k}{\Omega_{m_k}(v_k, a_k)}$$

  end for
end loop

Q-Learning

Cet algorithme ne bénéficie pas de la factorisation des états $S = (v_1, v_2, v_3)$ et des actions $A = (a_1, a_2, a_3)$. En conséquence, il y a 124 états non terminaux disposant chacun de $3 * 3 * 3 = 27$ actions possibles. L'algorithme 5 sert à mesurer la valeur ajoutée de la factorisation des états permis par le MVDP présenté dans la section 4.4.4.

Monte Carlo classique factorisé selon le MVDP

Cet algorithme est équivalent à la méthode Monte Carlo appliquée au CBRL de l'algorithme 3 où toutes les compatibilités sont fixées à 0 (et donc les poids $w(c)$ à 1). De manière plus compacte, l'algorithme 6 est appliqué. Cet algorithme sert à mesurer la valeur ajoutée des algorithmes CBRL et MVTDL présentés dans la section 4.5.

5.2 Optimisation et tests du CBRL

Pour optimiser le CBRL, la principale variable est la fonction de pondération de la moyenne de Monte Carlo en fonction de la compatibilité.

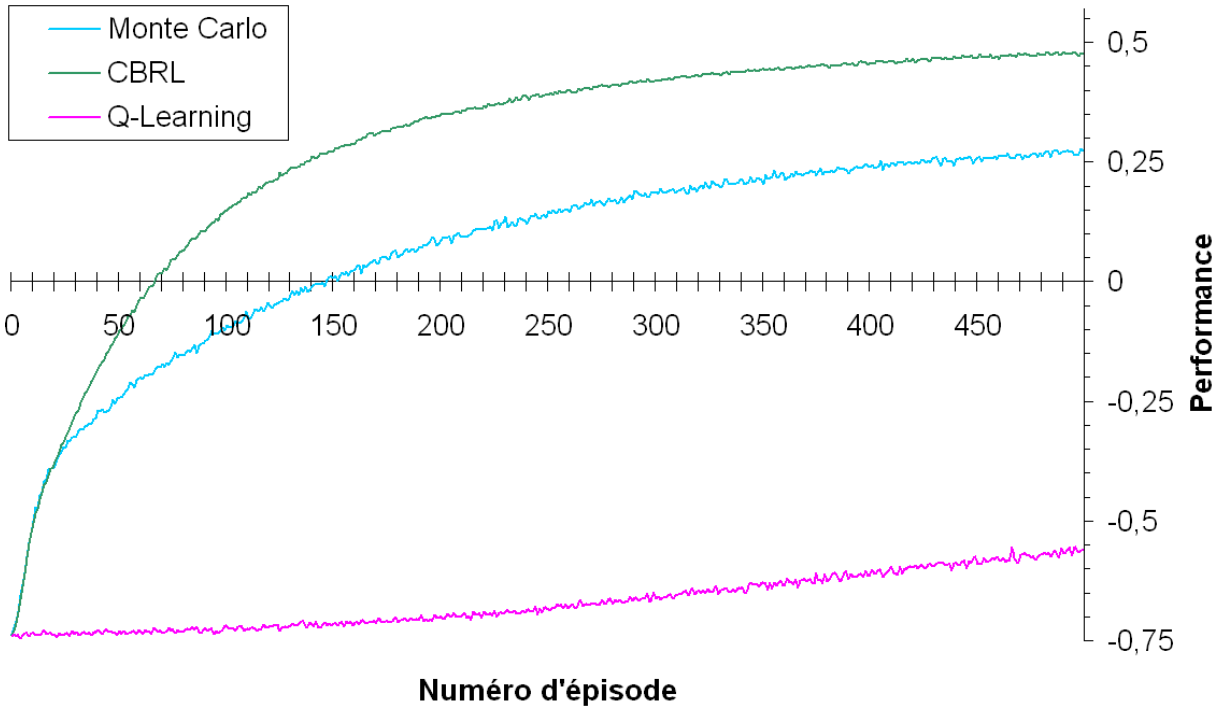


FIGURE 5.1 – Benchmark de l'algorithme CBRL par rapport aux algorithmes témoins.

5.2.1 Optimisation de la pondération en fonction de la compatibilité

Cette sous-section consiste à déterminer la meilleure fonction $\omega(c)$ pour le calcul de la pondération w utilisée dans l'équation 4.13.

Fonction exponentielle

L'idée intuitive est que les différentes incompatibilités devraient multiplier la non-pertinence d'un exemple pour l'apprentissage. Comme les compatibilités sont additionnées, cela nous a amené à l'utilisation de la fonction exponentielle :

$$\omega(c) = \kappa e^{-\tau c} \tag{5.2}$$

Comme le calcul final est normalisé, κ n'a pas d'influence et nous le choisissons égal à 1. La figure 5.1 fait la comparaison entre les algorithmes témoins et l'algorithme CBRL. Ces courbes, ainsi que les suivantes (sauf mention contraires) du chapitre 5 ont été obtenues après 10000 simulations d'apprentissage.

La première remarque à formuler à propos de cette figure est que la modélisation dans le formalisme MVDP a permis d'améliorer de manière brutale la performance de l'apprentissage. En effet, la comparaison entre le Q-Learning (basé sur les MDP) et les

autres méthodes d'apprentissage (basées sur le MVDP) montre une différence d'ordre de grandeur. Comme la figure 4.2 le laissait présager, la complexité du système a été réduite par $\frac{124\times27}{3\times(5\times3)} \approx 74$. L'algorithme CBRL concrétise cet avantage de modélisation et la figure 5.1 révèle qu'après 500 épisodes, ce qui signifie pratiquement 500 échecs pour l'algorithme Q-Learning, ce dernier n'a réussi à apprendre à survivre qu'un tour et demi en plus. Ce n'est pas étonnant quand on fait le calcul du nombre de couples état-action qui amènent l'agent en dehors de la grille : $9 * 6 * 9 + 12 * 3 * 15 + 8 * 1 * 19 = 1178$. Le Q-Learning requiert 10000 épisodes [38] pour atteindre une performance équivalente à la performance atteinte par le CBRL avec la fonction ω exponentielle et $\tau = 1.4$.

La figure démontre également que l'utilisation de la compatibilité améliore grandement la rapidité et la garantie de convergence vers la politique optimale. En effet, même lorsque la durée de l'expérimentation est allongée, la performance de la méthode Monte Carlo classique ne dépasse pas 0.35. La figure 5.2 montre la boucle de contrôle Monte Carlo. En comparaison avec la méthode Monte Carlo classique, le CBRL améliore l'évaluation de la politique actuelle, en portant non seulement son évaluation sur l'épisode qu'il vient de générer, mais également sur la qualité des épisodes précédents pour l'apprentissage.

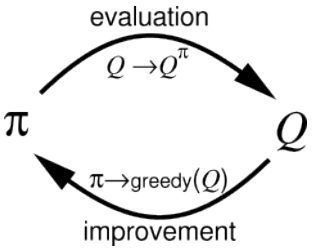


FIGURE 5.2 – La boucle de contrôle Monte Carlo (Sutton & Barto, 1998).

38. Pour l'apprentissage sur 10000 épisodes, la méthode d'exploration a été optimisée, de manière à ce que la performance de l'algorithme Q-Learning soit optimale.

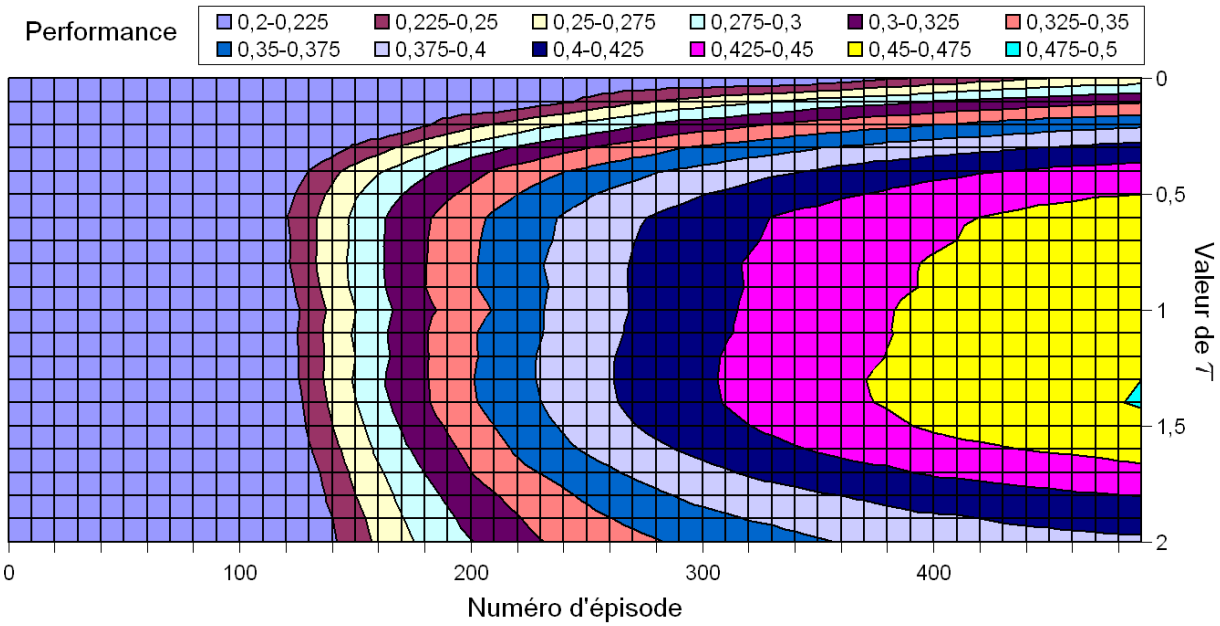


FIGURE 5.3 – Vue colorimétrique de l'impact de τ sur l'apprentissage.

La figure 5.3 donne une vue colorimétrique [39] plus complète, permettant de bien apprécier les remarques suivantes :

- La valeur $\tau = 1.4$ est la valeur qui optimise la fonction exponentielle.
- Les valeurs de τ inférieures à 1.4 prennent moins en compte la compatibilité pour construire leurs politiques. Ceci s'exprime par un démarrage d'apprentissage correcte, mais une difficulté à converger vers la politique optimale.
- Les valeurs de τ supérieures à 1.4 prennent plus en compte la compatibilité pour construire leurs politiques. Ceci s'exprime par des politiques chaotiques, notamment lorsque le nombre d'épidodes est encore réduit. En conséquence, l'apprentissage décroît assez rapidement en performance.

Même lorsque $\tau = 1.4$, la politique est loin d'être optimale à la fin des 500 épisodes. L'analyse du comportement de l'apprentissage pour les différentes valeurs de τ nous indique que l'on souhaiterait avec une valeur de τ forte pour les premiers épisodes et une valeur plus faible pour les épisodes suivants. Nous avons essayé de faire varier τ au cours du temps, sans gain de performance notable. Il y a même un danger à long terme, quand τ devient trop élevé d'obtenir un comportement chaotique où l'algorithme alterne indéfiniment d'une politique à une autre. En fait, le défaut de cette fonction est à la fois d'être trop sévère avec les mauvaises compatibilités, et d'être trop permissive pour les compatibilités qui sont bonnes mais imparfaites. En conséquence, l'algorithme a tendance à rejeter trop vite certaines expériences, alors même que la convergence n'est pas atteinte et à ne pas réussir l'optimisation fine.

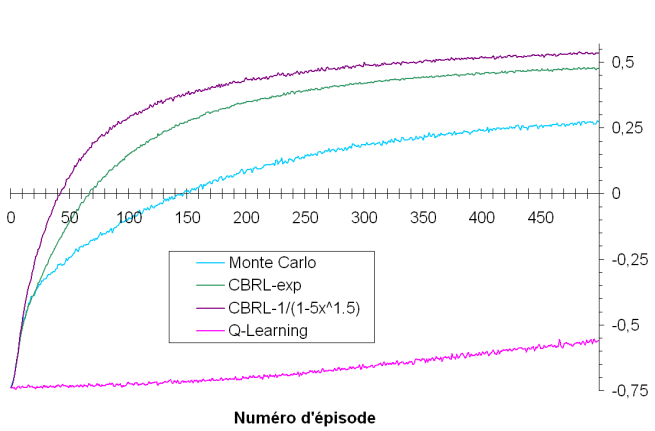
Fonctions de la famille des fractions polynomiales

En conséquence des constatations de la partie précédente, cette partie expérimente des fonctions en fraction polynomiale. La famille de fractions polynomiales les plus simples et satisfaisant les contraintes de la partie 4.5.1 (croissante, $\lim_{x\to-\infty}\omega(x) = 0$ et $\omega(0) = 1$) s'écrit de la façon suivante :

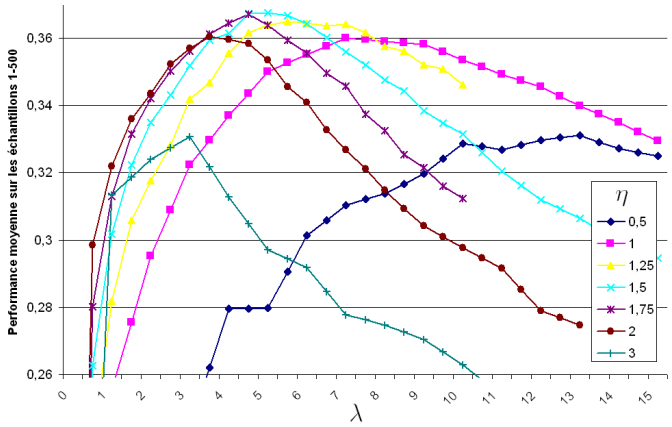
$$\omega(c) = \kappa\frac{1}{1+\lambda c^{\eta}} \tag{5.3}$$

Une nouvelle fois κ disparaît à la normalisation et sa valeur est sans importance. Pour la suite, $\kappa = 1$. Ces fonctions ont donc deux paramètres λ et η. Cette partie s'efforce de les optimiser. Plusieurs valeurs de η ont été expérimentées : 0.5, 1, 1.25, 1.5, 1.75, 2 et 3.

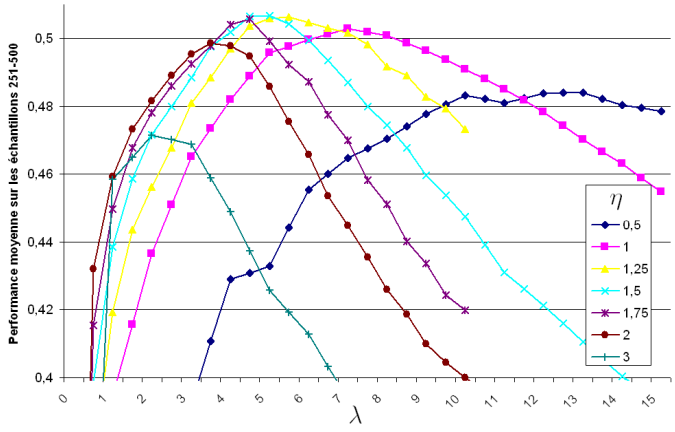
39. Il est à noter que la valeur $\tau = 0$ rend l'algorithme 3 équivalent à la méthode Monte Carlo de l'algorithme 6.



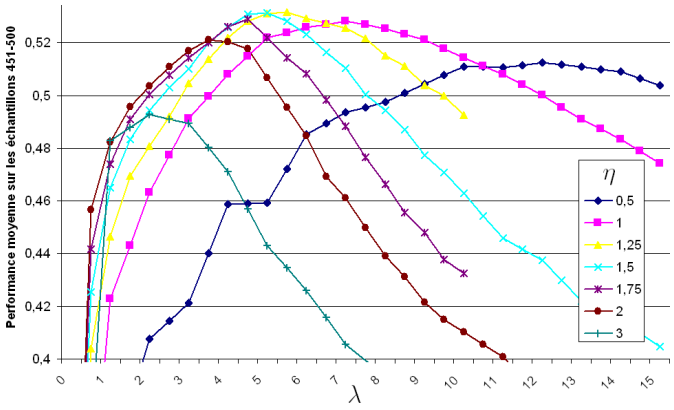
(a) La comparaison de la pondération exponentielle et la pondération en fraction polynomiale.



(b) La moyenne sur les 500 épisodes pour évaluer la performance globale.



(c) La moyenne sur la seconde partie de l'apprentissage, pour évaluer la rapidité de convergence initiale.



(d) La moyenne sur la fin de l'expérimentation, pour évaluer la qualité moyenne de la politique finale.

FIGURE 5.4 – Performances à différents horizons des fractions polynomiales en fonctions de la variation de ses paramètres λ et η.

De manière à pouvoir les comparer grâce à la visualisation de graphique, la moyenne des performances des N derniers épisodes a été réalisé.

La figure 5.4a montre que les fractions polynomiales permettent d'obtenir des performances bien supérieures aux fonctions exponentielles. La figure 5.4b révèle que, même sans optimiser précisément les paramètres, les fractions polynomiales obtiennent des performances très au dessus des exponentielles, puisque celles-ci ne dépassent jamais la performance moyenne de 0.29, alors que les meilleures avec la pondération en fraction polynomiale atteignent 0.36. Les figures 5.4c et 5.4d confirment cette tendance. Les valeurs optimales pour les paramètres λ et η sont respectivement 5 et 1.5.

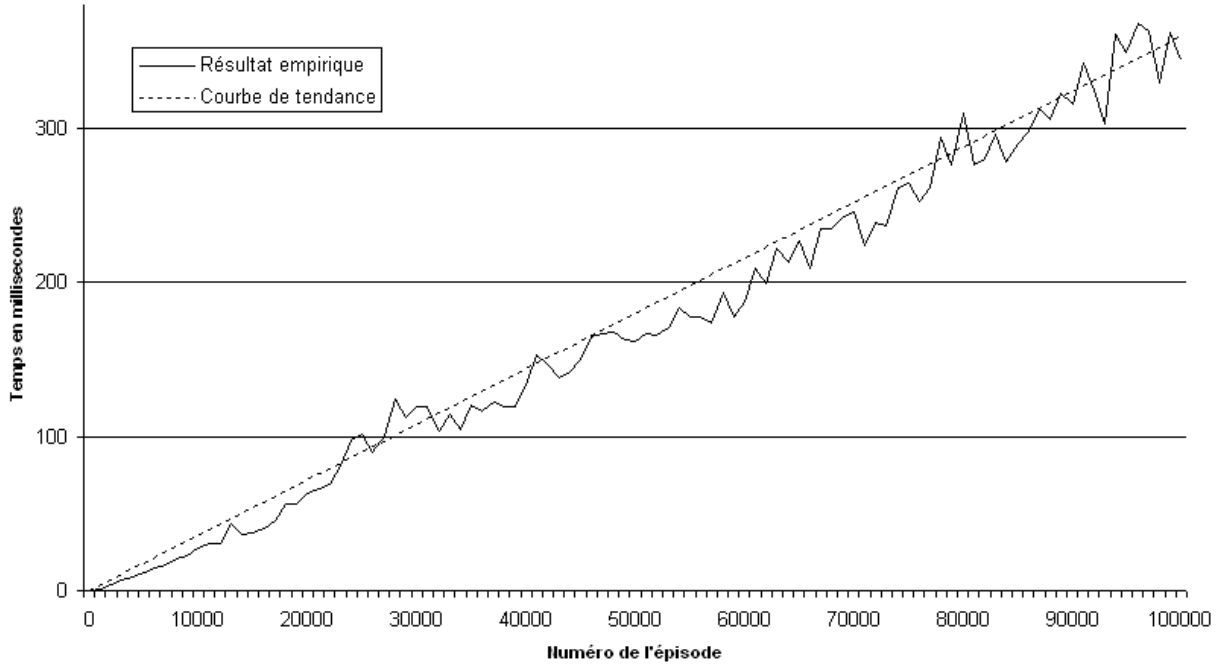


FIGURE 5.5 – Test de charge de l'algorithme CBRL

Ainsi, la fraction rationnelle optimale est $\omega(c) = \dfrac{1}{1+5c\sqrt{c}}$, pour l'exercice en question. Comme ce n'est qu'un résultat empirique, appuyé par aucune considération théorique, il est difficile de la transposer à d'autres applications. Cependant, pour d'autres dimensions de grilles, les mêmes optima ont été observés. Il est donc raisonnable de supposer que ces valeurs de paramètres ont une certaine vérité générale.

5.2.2 Tests de charge

De manière à pouvoir anticiper la charge computationnelle imposée par l'algorithme d'apprentissage sur le serveur vocal, il est important de connaître le temps nécessaire à la mise à jour des compatibilités entre deux dialogues ou épisodes. Dans la sous-section 4.5.3, nous avons déjà introduit cette question de la complexité de l'algorithme CBRL. A la fin de chaque épisode, il est nécessaire de faire la mise à jour des compatibilités et d'appliquer la méthode de calcul des fonction d'état-action Q. La mise à jour des compatibilités est de complexité linéaire par rapport au nombre de décisions prises dans le corpus de dialogue. Dans le cas spécifique de l'algorithme 3, le calcul des fonctions d'état-action Q est également linéaire par rapport au nombre de décisions. Il en résulte que la mise à jour de la politique est linéaire et c'est effectivement ce que le test d'effort révèle, comme la figure 5.5 en atteste.

Ce test montre tout d'abord que l'algorithme tient très bien la charge tant que l'appli-

cation reste d'un volume raisonnable. Mais certaines applications industrielles ont pour vocation d'atteindre quelques millions de dialogues. Dans ce cas, la mise à jour peut requérir plus d'une minute, ce qui est d'autant plus gênant si les serveurs vocaux sont surchargés. C'est la raison pour laquelle, nous nous sommes intéressés à la réduction de cette charge dans la section 5.5.

5.3 Tests du MVTDL

L'optimisation de l'algorithme 4 de la famille MVTDL est plutôt simple, étant donné que l'algorithme ne dispose que d'un paramètre : α. De manière à suivre les recommandations habituelles, nous avons fixé $\alpha = 0.2$ et nous l'avons fait décroître exponentiellement selon la même loi que l'exploration au fur et à mesure des épisodes. Ensuite, nous comparons les performance de cet algorithme avec celles de l'algorithme CBRL vues dans la section précédente. Enfin, deux types d'altération du module "critique" sont considérés : l'imprécision et la dégradation, pour traiter les comportements respectifs de l'algorithme lors de modifications de précision de la mesure des variables et lors de suppression de variables pertinentes ou d'ajout de variables non pertinentes.

5.3.1 Comparaison de l'algorithme MVTDL avec l'algorithme CBRL

Premièrement, la figure 5.6 rappelle les performances des algorithmes témoins. Tout comme le CBRL, le MVTDL les dépasse de beaucoup. Les performances des algorithmes témoins sont analysées plus en profondeur dans la partie 5.2.1.

Deuxièmement, la comparaison de performances entre le CBRL et le MVTDL nécessite un peu plus d'analyse. Le CBRL est plus performant dans les premiers épisodes, mais sa convergence est moins garantie à l'horizon des 500 épisodes. En effet, il est intéressant de noter que le MVTDL requiert une phase d'apprentissage pour le module "critique", ce qui explique son retard en performance dans les premiers épisodes. Une fois que le module critique a convergé, la convergence est très rapide et elle atteint à la fin des 500 épisodes une performance (0.555) relativement proche du maximum théorique (0.63), surtout si on considère le fait que MVTDL continue à faire de l'exploration.

Cependant la modélisation parfaite de la grille de cet exemple ne reflète pas la réalité d'un système de dialogue où les indicateurs de performance de dialogue sont non mesurables ou imprécis dans le meilleur des cas.

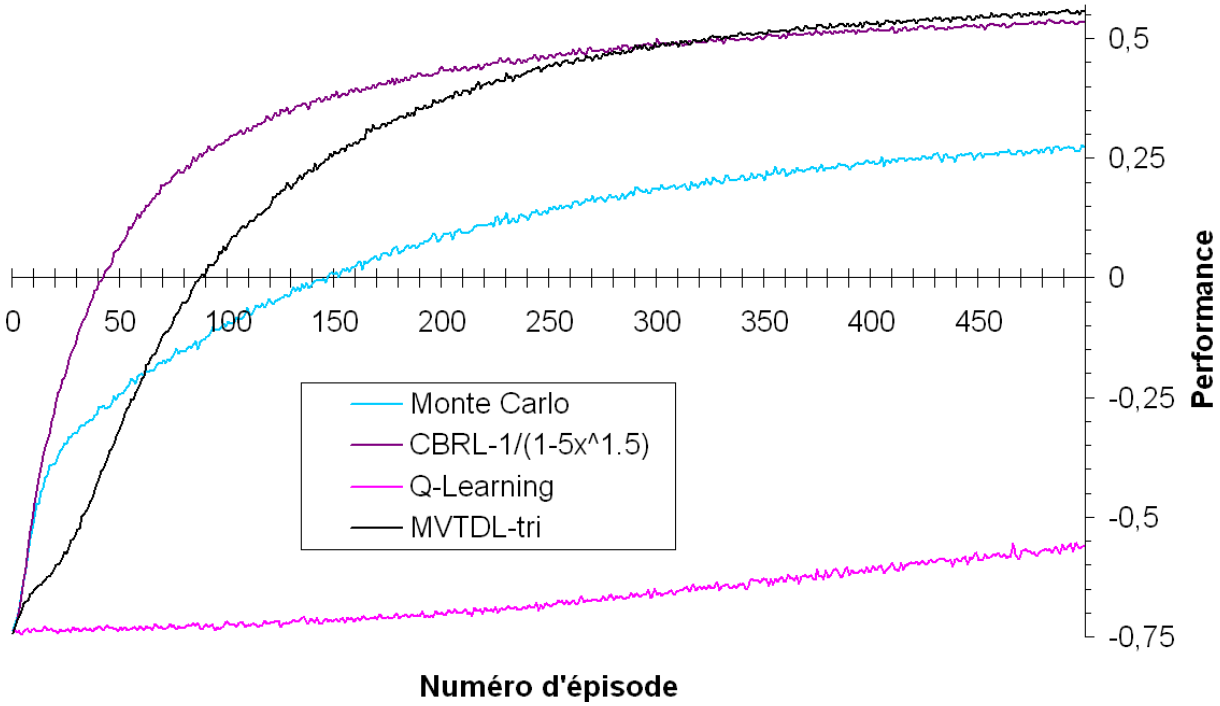


FIGURE 5.6 – Comparaison de performance entre l'algorithme MVTDL, l'algorithme CBRL, et les algorithmes témoins.

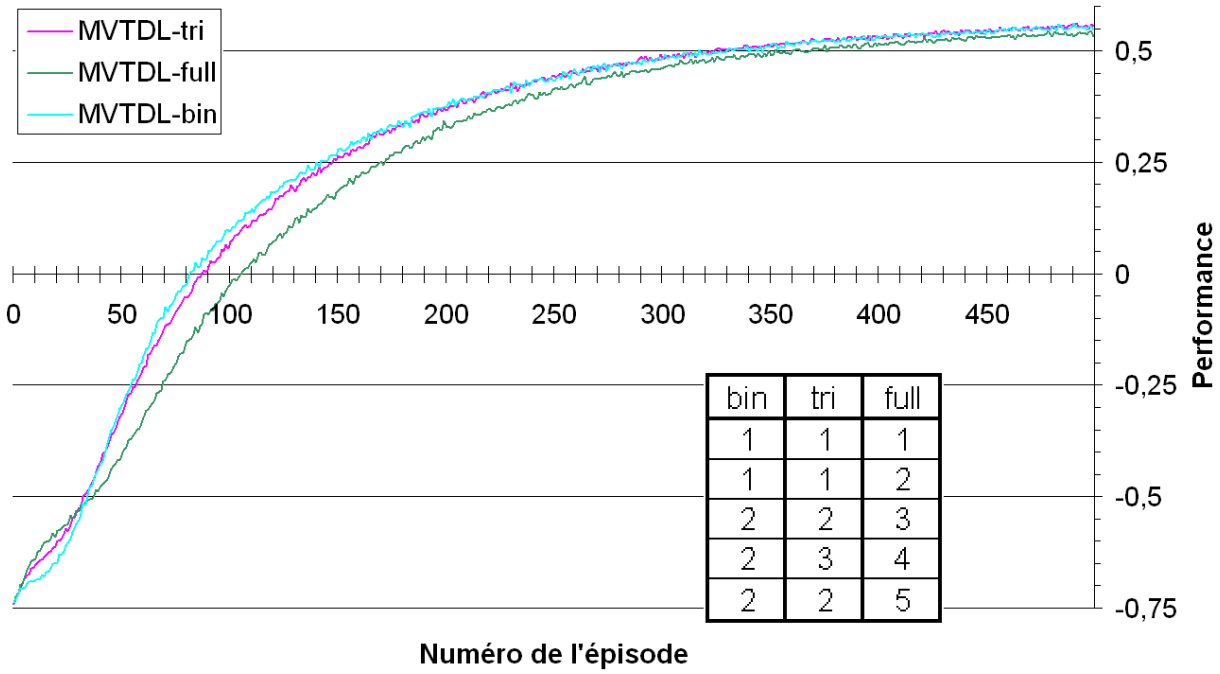


FIGURE 5.7 – Comparaison entre les variations de précision. Les prédicteurs binaire (*bin*), trinaire (*tri*) et complet (*full*) sont comparés.

5.3.2 Précision du module "critique"

Trois niveaux de précision ont été étudiés pour l'implémentation du module "critique" : soit ses états sont exactement les états du système ($5*5*5 = 125$ états, modélisation *full*), soit ses états sont un peu moins précis et sont fournis sur une échelle trinaire pour chaque dimension ($3 * 3 * 3 = 27$ états, modélisation *tri*), soit enfin ses états sont très imprécis et leurs descriptions se font sur une échelle binaire sur chaque dimension ($2 * 2 * 2 = 8$ états, modélisation *bin*). Comme la table dans la figure 5.7 le montre, les imprécisions ont été définies de manière optimale, parce que le but de cette sous-section est d'étudier l'imprécision et non le biais. La sous-section suivante s'intéressera plus particulièrement aux dégradations du module "critique". La figure 5.7 révèle que l'imprécision a un effet positif sur la rapidité de l'apprentissage, parce que la réduction du nombre d'états du module "critique" accélère sa convergence. En revanche, comme la modélisation binaire mène à une optimisation légèrement sous-optimale par rapport à la modélisation trinaire, la modélisation MVTDL-tri est celle retenue pour les autres tests.

5.3.3 Dégradation du module "critique"

Cette sous-section fait l'étude de plusieurs dégradations de la conception du module "critique", de manière à prendre en compte la difficulté d'implémenter un tel module dans un système de dialogue :

- Pour garder la référence de la modélisation parfaite, la figure 5.8 comporte la courbe MVTDL_tri sans dégradation. Elle est labellisée (*0*).
- La suppression de variable pertinente : le système continue à faire des décisions sur la dimension correspondante, mais cette variable n'est plus visible par le critique. L'objectif de cette dégradation est de mesurer l'impact de la situation réelle des systèmes de dialogue où le système ne dispose pas d'une vue complète du monde extérieur. Cette dégradation est labellisée (*-1*).
- L'ajout d'un variable de bruit : le module "critique" est complexifié par l'ajout d'un variable non pertinente. Chaque ajout de variable correspond à la multiplication de la complexité du système par 3, puisque la précision que nous avons choisie pour ces tests est MVTDL-tri. L'objectif de cette dégradation est de mesurer l'impact d'une erreur de conception faite par le développeur. Cette dégradation est labellisée (*+1*).
- La substitution d'une variable pertinente par une variable non-pertinente : cette dégradation est équivalente à la conjonction des dégradations de suppression et d'ajout de variables. Elle est labellisée en conséquence (*-1+1*).
- L'ajout de 5 variables de bruit : cette dégradation est labellisée (*+5*).
- Apprentissage sans le module "critique" : cette expérience permet de mesurer les

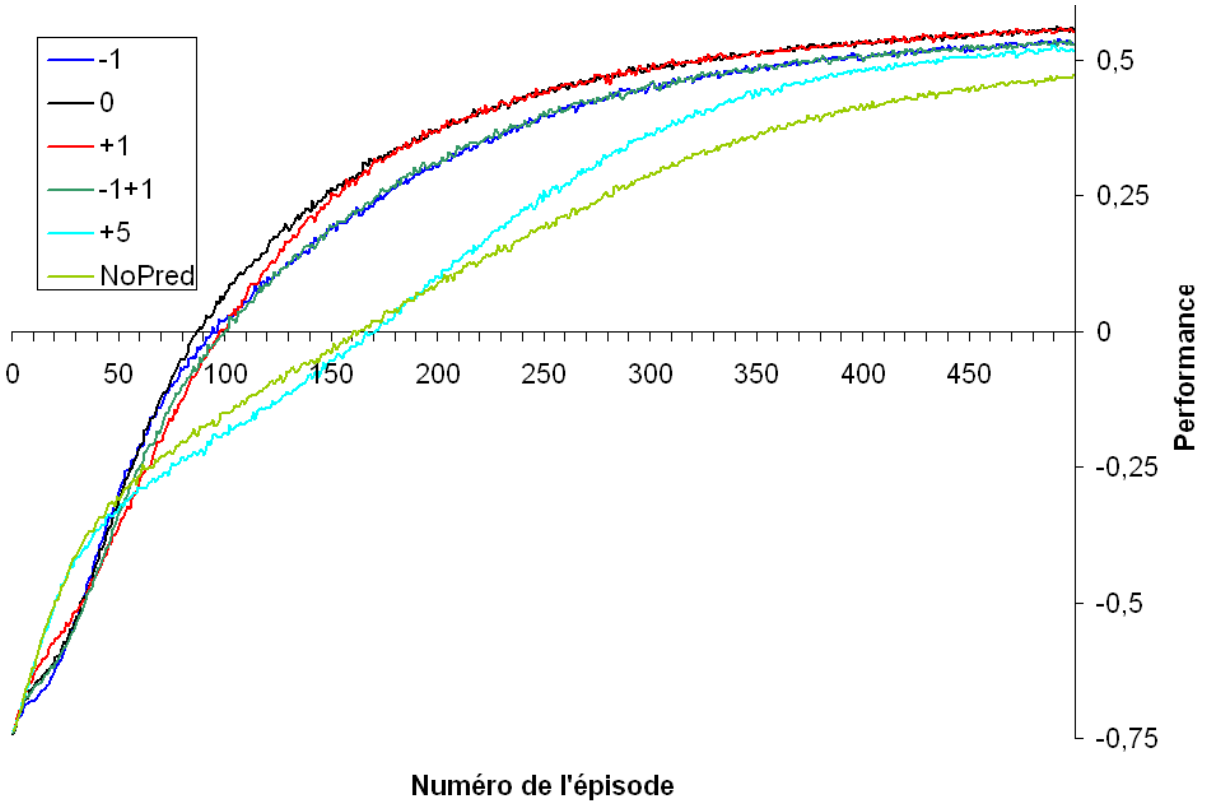


FIGURE 5.8 – Comparaison de MVTDL-tri (*0*) avec l'ajout de variable (*+1*), la suppression de variable (*-1*), la substitution de variable (*-1+1*), de l'ajout de cinq variables (*+5*) et l'apprentissage sans module "critique" (*NoPred*).

gains réalisés par rapport à l'apprentissage sur la base des récompenses directes seules. Cette dégradation est labellisée (*NoPred*)

On peut immédiatement remarquer que l'algorithme se comporte plutôt bien face à la dégradation du module "critique". Il est intéressant d'analyser d'un peu plus près les effets de chaque dégradation, selon les indices d'épisodes :

- 0-35 : Plus la dimension est grande, moins il est probable que le système rencontre un état qu'il a déjà atteint et donc pour lequel le module "critique" estimerait une autre prédiction que la prédiction nulle qui a été définie par défaut. Au début de l'apprentissage, cette prédiction n'a que peu de valeur parce que le système ne fait qu'essayer d'éviter de sortir de la grille, sans vraiment de succès d'ailleurs. En conséquence, la courbe (*+1*) se comporte mieux que la courbe (*0*), qui elle-même est approximativement aussi bonne que la courbe (*-1+1*) et meilleure que la courbe *-1*.
- 35-75 : Sur cette plage, la courbe (*-1*) est aussi bonne que la courbe (*0*), parce que la capacité d'apprentissage est limitée par la quantité d'information reçue et la

suppression de variable n'est pas un obstacle à la quantité d'information reçue. La même remarque est valable pour la comparaison des courbes (*-1+1*) et (*+1*).

- 35-190 : A l'inverse, l'addition d'une variable ralentit l'apprentissage du module "critique". Ceci explique que la courbe (*+1*) soit dépassée par la courbe (*0*) sur cette période. Au final, à partir de l'épisode 190, les modules "critiques" ont convergé, et les deux apprentissages suivent la même courbe.
- 75-190 : La suppression de la variable est maintenant un obstacle à l'apprentissage. Ainsi, la courbe (*-1*) perd du terrain sur la courbe (*0*).
- 190-500 : La courbe (*-1*) ne réussit pas à converger de manière consistante vers la solution optimale à cause de la dimension manquant au module "critique".
- Sur l'ensemble de l'apprentissage, la courbe (*-1+1*) est approximativement la fonction inférieure des courbes (*-1*) et (*+1*). Il se semble y avoir aucune pénalité particulière lié au cumul des deux dégradations.
- La courbe (*NoPred*) est en fait équivalente aux courbes ($+\infty$) et (*-3*). C'est en cela qu'elle est intéressante : elle est la limite des dégradations. De son côté, le module "critique" (*+5*) comporte un nombre d'état $3^5 = 729$ fois plus grand que le module "critique" (*+1*). Après un peu de temps (200 épisodes), le module commence à se démarquer de la courbe *NoPred*.

5.3.4 Conclusions générales du benchmark entre CBRL et MVTDL

La convergence de l'algorithme CBRL est plus rapide que celle de l'algorithme MVTDL, mais il est moins fiable, si l'on considère le cas où le module "critique" est conçu de manière parfaite. C'est sur ce dernier point, que le MVTDL n'est pas fiable à son tour. Si l'on considère en addition que le MVTDL requiert un travail spécifique, et probablement complexe, de la part du concepteur, il est logique de se tourner en priorité vers le CBRL.

La raison de la non-convergence vers la stratégie optimale de l'algorithme CBRL peut être imputée à la stratégie d'exploration qui ne lui est pas adaptée, ou qui est en tout cas sous-optimale. En effet, les algorithmes Monte Carlo sont connus pour avoir une qualité de convergence très dépendante de la stratégie d'exploration (Sutton & Barto, 1998). C'est la raison pour laquelle la prochaine section est dédiée à l'optimisation de la stratégie d'exploration pour le CBRL.

5.4 Optimisation de l'exploration pour le CBRL

Dans cette section, nous faisons le banc de test des quatre techniques de compromis exploration/exploitation de la littérature (Auer et al., 2002). Les méthodes testées sont

l'Upper Confidence Bound (UCB) (Agrawal, 1995), l'Upper Confidence Bound optimisé (UCB-tuned) (Auer et al., 2002), l'ϵ_t-greedy (Sutton & Barto, 1998) avec une meilleure fonction de décroissance d'ϵ_t en fonction de t et enfin une méthode implémentée par nos soin utilisant les intervalles de confiance.

5.4.1 Upper Confidence Bound

L'idée de l'algorithme UCB est de ne plus comparer les différentes alternatives sur la seule base de leurs performances respectives, mais également en comparant le nombre de fois que chacune a été utilisée. Un terme mesurant la fréquentation est alors ajouté à l'espérance :

$$\pi_m(v) = \underset{a \in A_m}{\operatorname{argmax}} \left(Q_m(v,a) + \sqrt{\frac{2 \ln n}{n_a}} \right) \tag{5.4}$$

Où n est le nombre de décisions prises dans l'état $\{m, v\}$ et n_a est le nombre où a a été choisie dans cet état. Il n'y a plus d'exploration à proprement parler puisque l'on cherche toujours à optimiser une même fonction. Cependant, on peut légitimement considérer que le système exploite quand l'optimum concerne une alternative dont le premier terme est élevé et qu'il explore lors que c'est le second terme qui est élevé. Pour le CBRL, il est possible et préférable d'utiliser les fonctions de pondération $\Omega_m(v,a)$ calculées à l'aide de la formule 4.13, en lieu et place de n et n_a. Comme rien ne garantit que cette pondération soit supérieure à 1, un terme $+1$ est ajouté dans le logarithme. Cela ne modifie en rien les convergences quand ces deux valeurs tendent vers l'infini.

$$\pi_m(v) = \underset{a \in A_m}{\operatorname{argmax}} \left(Q_m(v,a) + \sqrt{\frac{2 \ln \left(1 + \sum_{a' \in A_m} \Omega_m(v,a') \right)}{\Omega_m(v,a)}} \right) \tag{5.5}$$

5.4.2 Upper Confidence Bound tuned

La formule de l'UCB a pour gros désavantage d'être fortement dépendant de l'amplitude des récompenses. C'est-à-dire que l'algorithme se comportera de façon très différente si les récompenses sont $+1000$ et -1000 au lieu d'être $+1$ et -1. Pour corriger cette sensibilité automatiquement, Auer et al. (2002) propose d'améliorer le calcul de l'UCB à l'aide d'un calcul de la variance que l'on cherche à majorer également à l'aide d'un UCB.

On utilise donc la formule suivante :

$$V_{m,v}(a) \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{n_a} X_i^2}{n_a} - \left(\frac{\sum_{i=1}^{n_a} X_i}{n_a} \right)^2 + \sqrt{\frac{2 \ln n}{n_a}} \tag{5.6}$$

Où les X_i sont les récompenses à long terme des individus d'apprentissage supervisé qui ont effectué l'action a dans l'état $\{m, v\}$. Les deux premiers termes constituent le calcul de variance empirique. Le troisième et dernier terme est le terme qui permet d'obtenir une majoration de l'écart-type. Elle est analogue à celle utilisée dans la formule 5.4 pour l'espérance. Ensuite la formule 5.4 est mise à jour de la façon suivante :

$$\pi_m(v) = \underset{a \in A_m}{\text{argmax}} \left(Q_m(v,a) + \sqrt{\frac{\ln n \min\left(\frac{1}{4}, V_{m,v}(a)\right)}{n_a}} \right) \tag{5.7}$$

Le facteur $\frac{1}{4}$ est la limite supérieure de la variance d'une variable aléatoire de Bernouilli. Dans le cas général que nous traitons, nous n'avons pas ce type de connaissance et cet élément est abandonné. Après adaptation des différentes formules pour y inclure les pondérations, nous obtenons :

$$V_{m,v}(a) \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{n_a} \omega(c_i) X_i^2}{\Omega_m(v,a)} - \left(\frac{\sum_{i=1}^{n_a} \omega(c_i) X_i}{\Omega_m(v,a)} \right)^2 + \sqrt{\frac{2 \ln \left(1 + \sum_{a' \in A_m} \Omega_m(v,a') \right)}{\Omega_m(v,a)}} \tag{5.8}$$

$$\pi_m(v) = \underset{a \in A_m}{\text{argmax}} \left(Q_m(v,a) + \sqrt{V_{m,v}(a) \frac{\ln \left(1 + \sum_{a' \in A_m} \Omega_m(v,a') \right)}{\Omega_m(v,a)}} \right) \tag{5.9}$$

5.4.3 Exploration ϵ_t-greedy

Une politique bien connue et très simple pour traiter le compromis exploration/exploitation est la règle ϵ-greedy. Cette politique conseille de choisir l'alternative ayant la meilleur espérance de récompense avec une probabilité $1 - \epsilon$ et de choisir une alternative au hasard avec la probabilité ϵ. Clairement, l'exploration constante de probabilité ϵ crée une croissance linéaire du regret (au lieu de logarithmique Lai & Robbins, 1985). La façon évidente de résoudre ce problème est de faire décroître ϵ au cours du temps (et ainsi

le renommer ϵ_t), de manière à ce que l'exploration disparaisse petit à petit au fur et à mesure que les espérances se font de plus en plus précises. Dans les premières sections de ce chapitre, nous avons considéré une évolution exponentielle de la forme : $\epsilon_t = \kappa\gamma^t$. Selon Auer et al. (2002), une formule du type $\epsilon_t = \frac{\kappa}{t}$ est meilleure et permet de prouver la convergence logarithmique théorique.

Cette méthode est pourtant particulièrement inefficace pour notre problème. Une première raison pour cela est que l'exploration ϵ_t-greedy est aléatoire, alors qu'elle est contrôlée dans les autres méthodes. En effet, lorsque le système explore, il choisit une des actions aléatoirement, alors que les autres méthodes présentées dans cette section font leur exploration en fonction des espérances et variances des différentes actions. Il se trouve que notre problème, contrairement aux problèmes traités par Auer et al. (2002), contient des actions dangereuses qu'il faut très vite apprendre à éviter de réaliser (les actions qui mènent hors de la grille). Une seconde raison peut être le fait qu'elle ne prenne pas en compte la compatibilité et que selon les résultats aléatoires d'une expérience, l'algorithme peut avoir besoin de plus ou moins d'exploration. La compatibilité sert également à mesurer ceci : la taille du corpus ayant suivi la politique (supposée) optimale. La réponse à cette question est très importante pour définir le volume d'exploration. Le fait que la méthode ϵ_t-greedy ignore complètement cette indication la pénalise fortement.

L'exploration ϵ_t-greedy est la seule méthode qui utilise un paramètre, qui est de plus assez difficile à fixer a priori. Tous ces arguments font que les résultats de cette méthode ne méritent même pas d'être mentionnés dans le banc de test de la sous-section 5.4.5.

5.4.4 Intervalle de confiance

La dernière méthode que nous testerons ne fait pas partie de la littérature. Elle repose sur les outils statistiques des intervalles de confiance. Rappelons la définition d'un intervalle de confiance [40] :

> En statistiques, et en particulier dans la théorie des sondages, lorsqu'on cherche à estimer la valeur d'un paramètre, on parle d'intervalle de confiance lorsque l'on donne un intervalle qui contient, avec un certain degré de confiance, la valeur à estimer. Le degré de confiance est en principe exprimé sous la forme d'une probabilité. Par exemple, un intervalle de confiance à 95% (ou au seuil de risque de 5%) a une probabilité égale à 0,95 de contenir la valeur du paramètre que l'on cherche à estimer.

Rappelons également le théorème central limite :

40. Source : www.wikipedia.fr

Théorème 1. *Soit $\{X_i\}_i$ une suite de variables aléatoires définies sur un même espace de probabilité, suivant la même loi D et indépendantes. Supposons que l'espérance μ et l'écart-type σ de D existent et soient finis. Soit la somme $S_n = \sum_i X_i$. Alors, l'espérance de S_n est $n\mu$ et son écart-type vaut $\sigma\sqrt{n}$. De plus, la loi de S_n tend vers la loi normale quand n tend vers l'infini.*

La comparaison de K alternatives à un instant donné est en fait la comparaison de leurs corpus $\{X_i = (r_i, c_i)\}_{i\in[1,n_k]}$. A partir de ce corpus, nous pouvons calculer une estimation de l'espérance $\widetilde{\mathbb{E}}(S_{n_a})$ de la somme des récompenses obtenues selon chacune des alternatives. On peut directement reporter cette estimation sur S_n à l'estimation d'espérance $\widetilde{\mathbb{E}}(r)$ des récompenses à long-terme de l'alternative :

$$\widetilde{\mathbb{E}}(r) = \frac{\widetilde{\mathbb{E}}(S_{n_a})}{\sum_{i=1}^{n_a}\omega(c_i)} = \frac{\sum_{i=1}^{n_a}\omega(c_i)r_i}{\sum_{i=1}^{n_a}\omega(c_i)} \tag{5.10}$$

Si l'on considère le déroulement de l'expérience comme un tirage unique de la suite S_{n_a}, alors selon le théorème central limite, ce tirage a un écart-type valant $\sigma(r)\sqrt{\sum\omega(c_i)}$ et on peut faire l'approximation qu'il suit la loi normale si n_a est suffisamment grand. En conséquence, l'approximation de la moyenne $\widetilde{\mathbb{E}}(r)$ suit un écart-type de :

$$\widetilde{\sigma}(r) = \frac{\sigma(r)\sqrt{\sum_{i=1}^{n_a}\omega(c_i)}}{\sum_{i=1}^{n_a}\omega(c_i)} \tag{5.11}$$

$$\widetilde{\sigma}(r) = \frac{\sigma(r)}{\sqrt{\sum_{i=1}^{n_a}\omega(c_i)}} \tag{5.12}$$

Comme $\sigma(r)$ peut être calculé lui aussi empiriquement, nous connaissons une approximation de la loi qui régit le tirage aléatoire de $\widetilde{\mathbb{E}}(r) : \mathcal{N}(\widetilde{\mathbb{E}}(r), \widetilde{\sigma}(r)^2)$. Ainsi lorsqu'il s'agit de définir les dominances entre alternatives, il suffit de déterminer comment des tirages aléatoire de leurs distributions les classent en probabilité. Malheureusement le calcul de ces probabilités est trop lourd pour être réalisé exactement. C'est la raison pour laquelle nous ne ferons que des comparaisons deux à deux. Comme ce calcul est déjà trop complexe pour être réalisé en ligne, une nouvelle simplification est utilisée : la loi ayant l'écart-type le plus faible est considérée comme un point placé en sa moyenne μ fixe. La comparaison entre deux alternatives suivant les lois $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ et $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ avec $\sigma_1 > \sigma_2$ est donc calculée de la façon suivante :

$$P(X_1 \geq X_2) = \frac{1}{\sqrt{2\pi}} \int_{\mu_2}^{+\infty} e^{-\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2}} dx \tag{5.13}$$

$$P(X_1 \geq X_2) = \Phi\left(\frac{\mu_1 - \mu_2}{\sigma_1}\right) \tag{5.14}$$

Où Φ est la fonction de répartition. Pour un nombre d'alternatives supérieur à deux, il est difficile de les classer. La méthode que nous utilisons, est de comparer l'alternative la plus utilisée (c'est-à-dire avec $\sum w(c_i)$ maximal) avec chacune des autres. Pour chaque alternative concurrente, un nombre aléatoire est comparé à la probabilité calculée avec la formule 5.14 et si l'alternative concurrente bat la plus utilisée, alors cette alternative concurrente est choisie. Dans le cas contraire, l'alternative la plus choisie est comparée à l'alternative concurrente suivante. Si l'alternative la plus utilisée a battu toutes les alternatives concurrentes, on la choisit. L'intérêt de ce mode de sélection est que l'alternative la plus utilisée est souvent l'alternative optimale (puisqu'on choisit en général l'alternative avec la meilleure espérance) et également l'alternative avec l'écart-type le plus faible (puisque l'écart-type décroît en fonction de $\sum w(c_i)$).

L'algorithme 7 récapitule la méthode basée sur les intervalles de confiance dans le CBRL-MonteCarlo (algorithme 3).

La méthode IC étant nouvelle, plusieurs tests d'optimisation de la méthode sont réalisés dès cette sous-section, avant de la comparer aux autres méthodes. En effet, étant donné l'incapacité de la méthode à mesurer la probabilité d'avoir chaque alternative dominant les deux autres, nous faisons l'hypothèse que l'alternative la plus utilisée est meilleure que les deux autres séparément. Or, ces deux tests ne sont pas indépendants et cela a tendance à pénaliser le choix de l'alternative la plus performante. De manière à évaluer ce problème, nous avons ajouté un paramètre β à l'équation 5.14, qui correspond à la valeur 0 de β :

$$P(X^* \geq X_i) = \frac{\beta + \Phi\left(\frac{\mu_1 - \mu_2}{\sigma_1}\right)}{\beta + 1} \tag{5.15}$$

La figure 5.9 montre que la courbe $\beta = 0$ est plus longue à converger initialement, mais qu'elle dépasse relativement rapidement (vers l'abscisse 475) les performances de $\beta = 1$. Nous pouvons d'ores et déjà remarquer que la technique d'exploration utilisée lors des tests des sections précédentes n'atteignaient pas de telles performances (0.56) au 500ème épisode. La figure 5.10 inclut une troisième courbe concernant des simulations où nous avons fait varier β au cours du temps, pour bénéficier de la rapidité de β élevé au début de l'apprentissage et de la forte convergence de β nul en fin d'apprentissage. Cela améliore faiblement la courbe $\beta = 1$, mais au final, la courbe $\beta = 0$ est meilleure parce

Algorithm 7 Algorithme Monte-Carlo du Compliance-Based Reinforcement Learning avec exploration basée sur les intervalles de confiance

Initialisation de l'ensemble des épisodes $E = \emptyset$
Initialisation de la fonction d'état-action $Q_m(v, a) = 0$
loop
  Initialisation de l'épisode $e = \emptyset$
  for all k **do**
    Choisir $a^* = \underset{a \in A}{\operatorname{argmax}} \sum_{i=1}^{n_a} \omega(c_i)$
    Calcul de l'espérance μ^* et de l'écart-type σ^* de a^*
    Initialiser l'action choisie : $a_k = a^*$
    for all Choisir aléatoirement $a \in A \setminus \{a^*\}$ **do**
      Calcul de l'espérance μ et de l'écart-type σ de a
      Calcul de $P(X^* \geq X)$ {Equation 5.14}
      Tirage d'un nombre aléatoire $rand$ selon la loi uniforme sur $[0, 1]$
      if $rand > P(X^* \geq X)$ **then**
        Choisir l'action $a_k = a$ et sortir de la boucle for
      end if
    end for
    Exécuter l'action a_k
  end for
  $E \leftarrow E \cup \{e\}$
  for all $d_k \in e$ **do**
    Calcul de la récompense à long terme r_k {Equation 4.6}
  end for
  for all $e_i \in E$ **do**
    for all $d_{ij} = (m_{ij}, v_{ij}, a_{ij}, t_{ij}) \in e_i$ **do**
      Calcul de la compatibilité locale c_{ij} {Equation 4.9}
      Calcul de la compatibilité à long terme {Equation 4.8}
    end for
  end for
  Mise à jour de la fonction d'état-action $Q_m(v, a)$ {Equation 4.12}
end loop

qu'elle offre une meilleure garantie de convergence et c'est celle que nous garderons pour la comparaison avec les méthodes UCB des sous-sections 5.4.1 et 5.4.2.

5.4.5 Test des méthodes d'exploration sur les données artificielles

Dans cette sous-section, les jeux de test des algorithmes durent 500, 2000 et 10000 épisodes. La compatibilité et les paramètres de chaque technique d'exploration ont été optimisés conjointement, de manière à ce que le regret des derniers 20% épisodes soit minimal. Trois familles d'exploration sont comparées dans cette sous-section, UCB (sous-section 5.4.1), UCB-tuned (sous-section 5.4.2) et IC (sous-section 5.4.4).

Jeux de tests sur 500 épisodes

Sur les 500 premiers épisodes, l'exploration ϵ-greedy explore beaucoup moins que les autres méthodes, ce qui explique une grosse différence au début des courbes, entre les abscisses 300 et 400, toutes les méthodes d'exploration testées la dépassent.

L'échelle de la figure 5.11 ne permet pas bien d'en rendre compte, mais toutes les courbes sont encore loin de la politique optimale qui a une espérance de 0.63 et la courbe UCB domine UCB-tuned et IC sur cette période.

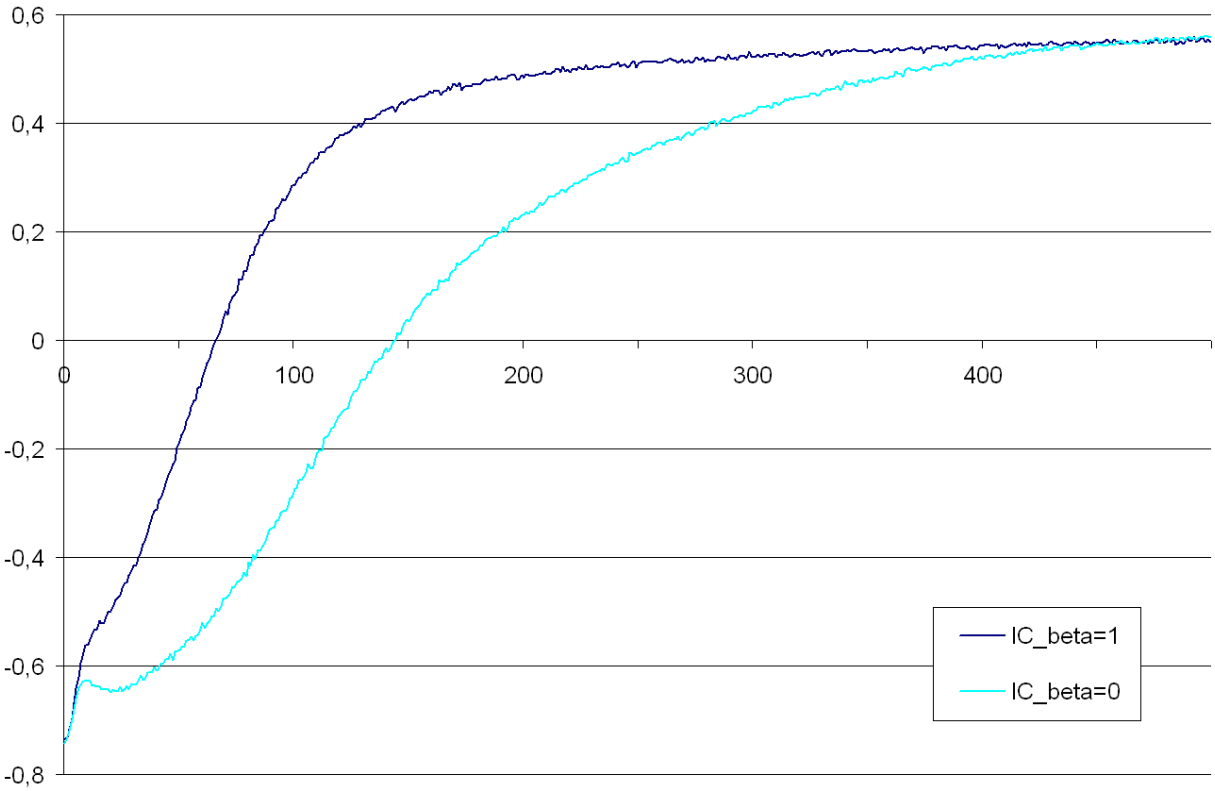


FIGURE 5.9 – Convergence à court terme pour $\beta = 0$ et $\beta = 1$.

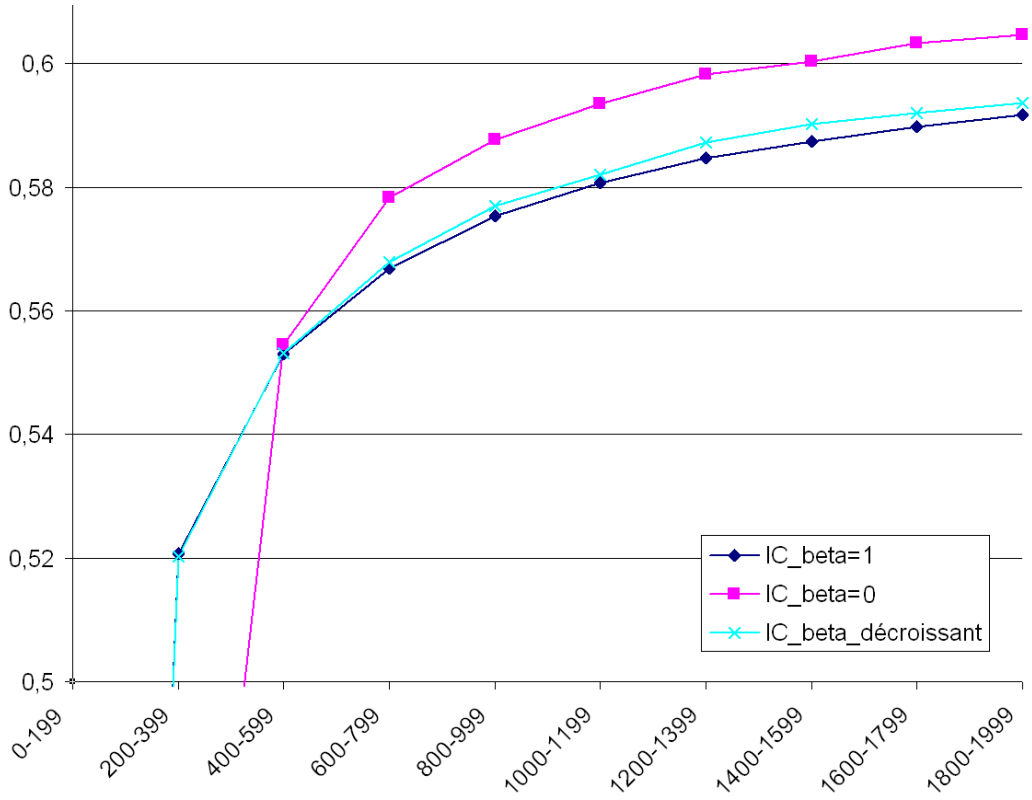


FIGURE 5.10 – Moyenne de performance sur un pas de 200 épisodes des trois valeurs de β considérées.

Jeux de test sur 2000 épisodes

L'exploration ϵ-greedy n'est plus représentée parce qu'elle avait été optimisée pour une durée d'expérience de 500 épisodes. Pour 2000 épisodes, il faudrait redéfinir ses paramètres, ou alors elle ne progresserait pratiquement plus, passé les 500 premiers épisodes et elle resterait en dessous de 0.58. La définition des paramètres de la méthode ϵ-greedy l'exclut automatiquement, étant donné la difficulté à connaître a priori les variances intra et inter alternatives. Nous nous concentrons donc sur les méthodes sans paramètre.

La figure 5.12 représente les performances comparatives des explorations UCB, UCB-tuned et IC. Pour ce faire, nous avons fait les moyennes de chacune selon un pas de 200 épisodes pour gommer le bruit et nous avons fait un zoom entre 0.5 et 0.62 pour mieux rendre compte des différences relatives. On voit que la méthode UCB se fait rapidement dépasser par les deux autres méthodes vers le 500ème épisode. Mais alors qu'UCB-tuned continue à accroître son avance, IC ne reste majorant de UCB que de peu.

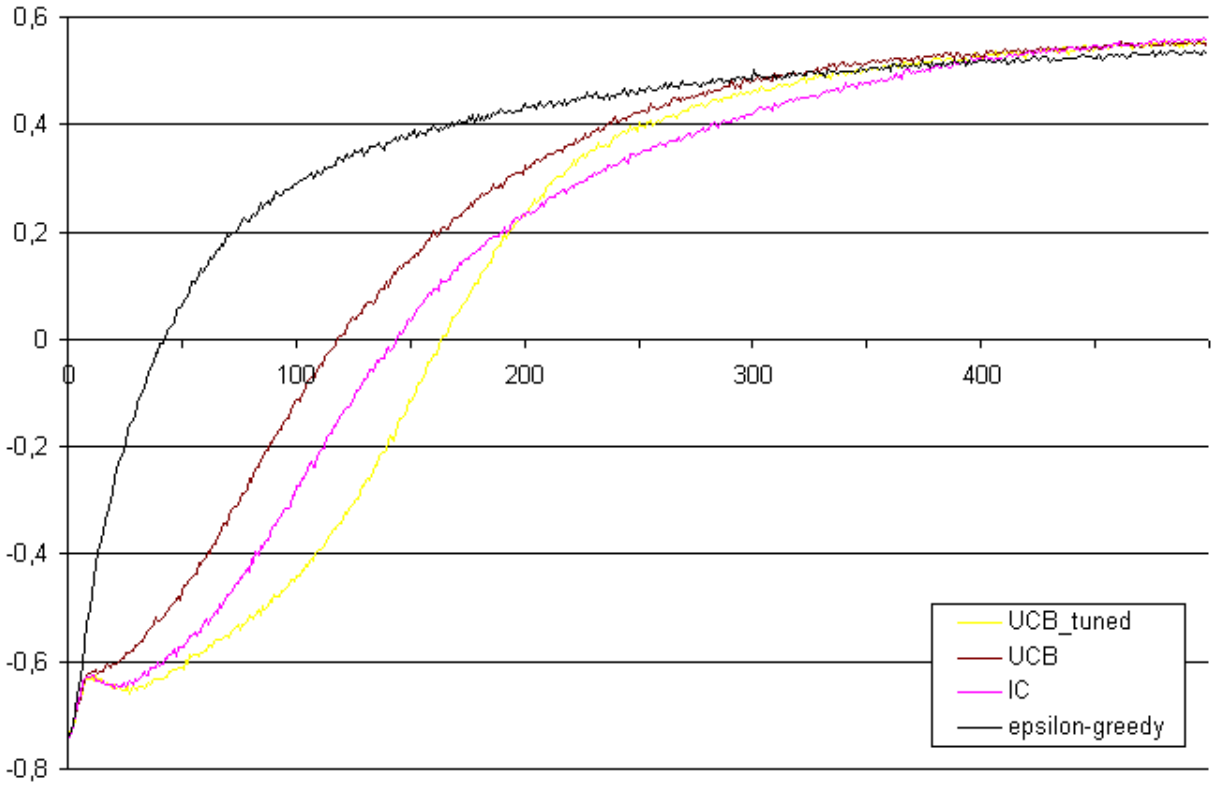


FIGURE 5.11 – Convergence à court terme (500 épisodes) pour les explorations ϵ-greedy, UCB, UCB-tuned et IC.

Jeux de test sur 10000 épisodes

La figure 5.13 présente des courbes plus chaotiques que précédemment. Il y a deux raisons à cela. La première est une raison de zoom : de manière à mieux rendre compte des différences de performance, nous avons réduit la fenêtre de visibilité entre les récompenses 0.6 et 0.63. La seconde raison est une conséquence de la complexité en $O(n^2)$ de l'algorithme CBRL (voir sous-section 4.5.3) qui nous a imposé de réduire le nombre d'exécutions de chaque algorithme à 1000, au lieu de 10000 comme pour les autres courbes.

Cependant la précision est suffisante pour se rendre compte de la supériorité de la méthode UCB-tuned, qui réduit approximativement par deux la distance par rapport à la performance de la politique optimale sans exploration. On se rend compte également que la méthode IC n'est intéressante par rapport à UCB que sur le moyen terme.

5.5 Tests de l'algorithme d'allègement de la charge

La faiblesse de l'algorithme CBRL réside principalement dans le fait qu'il doive réévaluer les compatibilités à chaque nouvel épisode. En conséquence, nous avons étudié le

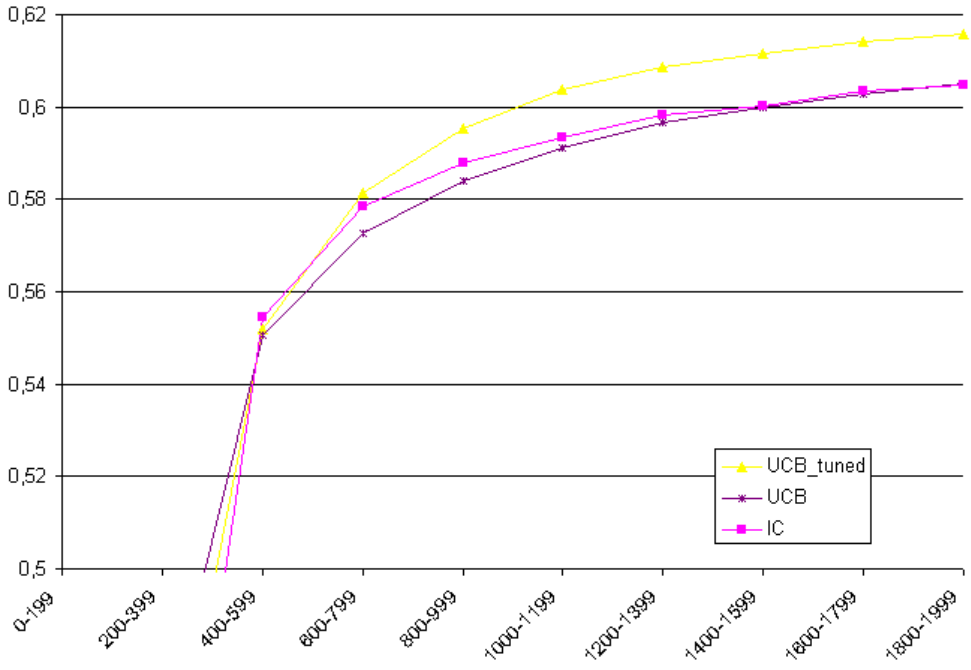


FIGURE 5.12 – Convergence à moyen terme (2000 épisodes) pour les explorations UCB, UCB-tuned et IC.

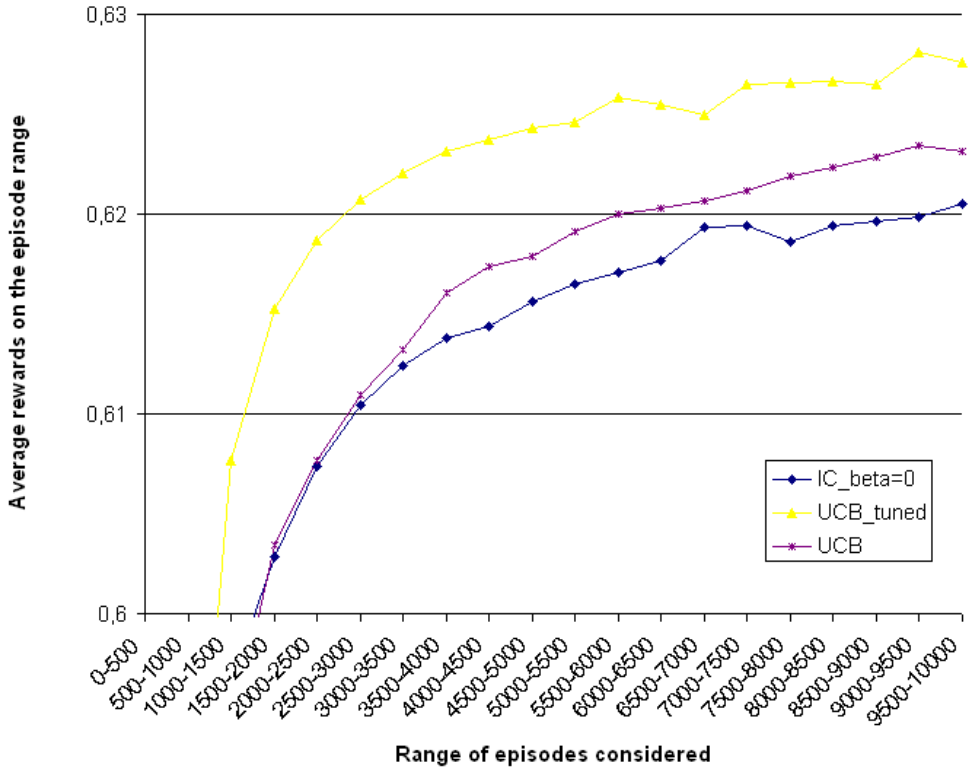


FIGURE 5.13 – Convergence à long terme (10000 épisodes) pour les explorations UCB, UCB-tuned et IC.

δ	$n = 100$	$n = 500$	$n = 10^4$	$n = 10^5$	$n = 10^6$
5	44	105	494	1575	4994
10	75	198	975	3137	9975
20	100	347	1900	6224	19900
50	100	500	4375	15186	49375
100	100	500	7500	29122	97500

TABLE 5.1 – Tableau indiquant le nombre de mises à jours dans les n premiers dialogues pour plusieurs valeurs de δ fixées.

coût impliqué par ce calcul. De même, nous avons étudié des stratégies de réduction de ce temps de calcul par la méthode d'approximation décrite à la fin de la partie 4.5.1.

Le principe de l'allègement de la charge consiste à faire une mise à jour approximative pour une certaine proportion des épisodes, telle que décrite à la fin de la partie 4.5.1. L'idée est de n'effectuer la mise à jour complète qu'un nombre de l'ordre de la racine carrée, pour que la complexité moyenne devienne de l'ordre de la racine carrée du nombre de décisions. Toutefois, il faut garantir que l'apprentissage nécessaire a été réalisé auparavant. Ainsi, la mise à jour complète n'est réalisée que lorsque le terme suivant est non nul :

$$E(\delta\sqrt{n}) - E(\delta\sqrt{n-1}) \tag{5.16}$$

Où $E(x)$ désigne la partie entière de x, n est le numéro de l'épisode que l'on vient de réaliser et δ est un paramètre. Le tableau **??** indique le nombre de mises à jour complètes réalisées après la génération de l'épisode n pour différentes valeurs de δ :

Le nombre de mises à jour est de l'ordre de $O(\sqrt{n})$ quand n tend vers l'infini. Pour faire les tests de dégradation de notre méthode d'allègement de la charge décrit par les formules

Algorithm 8 CBRL Monte Carlo *allègement de charge* **avec** mise à jour rapide

Initialisation de l'ensemble des épisodes $E = \emptyset$
Initialisation de la fonction d'état-action $Q_m(v, a) = 0$
loop
  Générer un nouvel épisode e_{n+1}
  if $E(\delta\sqrt{n+1}) = E(\delta\sqrt{n})$ **then**
    Mise à jour rapide (cf formules 4.16 et 4.17)
  else
    Mise à jour complète (cf algorithme 3).
  end if
end loop

4.16 et 4.17 (algorithme 8), nous avons utilisé comme étalon la méthode où aucune mise à jour de politique n'est effectuée en dehors des mises à jour complètes (algorithme 9).

La figure 5.14 montre les performances de l'algorithme 9 pour les valeurs de δ affichées dans le tableau **??** après 10000 dialogues. On voit une claire dégradation quand $\delta < 20$, mais la qualité de la convergence reste tout de même raisonnablement bonne.

La figure 5.15 montre les performances de l'algorithme 8 pour les valeurs de δ affichées dans le tableau **??** après 10000 dialogues. En comparaison de la figure 5.14, la dégradation est presque invisible.

La figure 5.16 [41] est très intéressante parce qu'elle révèle les différences de comportement entre les algorithmes 8 et 9, et elle permet de souligner un trait important du fonctionnement de la famille d'algorithmes CBRL. On peut remarquer que l'algorithme 9 évolue par palier, ce qui est logique puisque la politique ne change que lors des mises à jour complètes. On voit clairement que l'algorithme 8 avec mise à jour rapide fait de fortes oscillations. En réalité, chaque mise à jour rapide améliore la performance ; et les mises à jour complètes, quant à elles, semblent annuler une bonne partie des améliorations obtenues par les mises à jour rapides. Pour ne pas arriver à la conclusion évidemment fausse que la mise à jour complète est nocive à l'apprentissage, il faut bien prendre en compte ce qui se passe exactement lors de chaque mise à jour, rapide ou complète.

Les mises à jour rapides résolvent localement les problèmes en modifiant certaines Q-fonctions indépendamment du reste. Elles offrent un gain de performance important qui est trompeur parce qu'elle apprend comment résoudre localement les erreurs, mais aussi comment éviter les erreurs qui existent actuellement dans la politique. Cet évitement des erreurs de la politique actuelle ne mène pas à la politique optimale dans le cas général.

41. Les courbes avec mise à jour rapide et sans allègement sont plus chaotiques que celles obtenues dans les sections précédentes parce que les courbes ont été obtenues à partir des jeux de test sur 10000 épisodes et ceux-ci sont très longs à calculer. En conséquence, au lieu d'avoir effectué 10000 simulations de l'apprentissage comme pour la plupart des autres courbes, seuls 1000 simulations ont été utilisées pour le traçage de ces courbes.

Algorithm 9 CBRL Monte Carlo *allègement de charge* **sans** mise à jour rapide

Initialisation de l'ensemble des épisodes $E = \emptyset$
Initialisation de la fonction d'état-action $Q_m(v, a) = 0$
loop
  Générer k nouveaux épisodes $\{e_{n+1}, \ldots, e_{n+k}\}$ où k est le plus petit entier tel que $E(\delta\sqrt{n+k}) - E(\delta\sqrt{n}) > 0$.
  Mise à jour complète (cf algorithme 3)
end loop

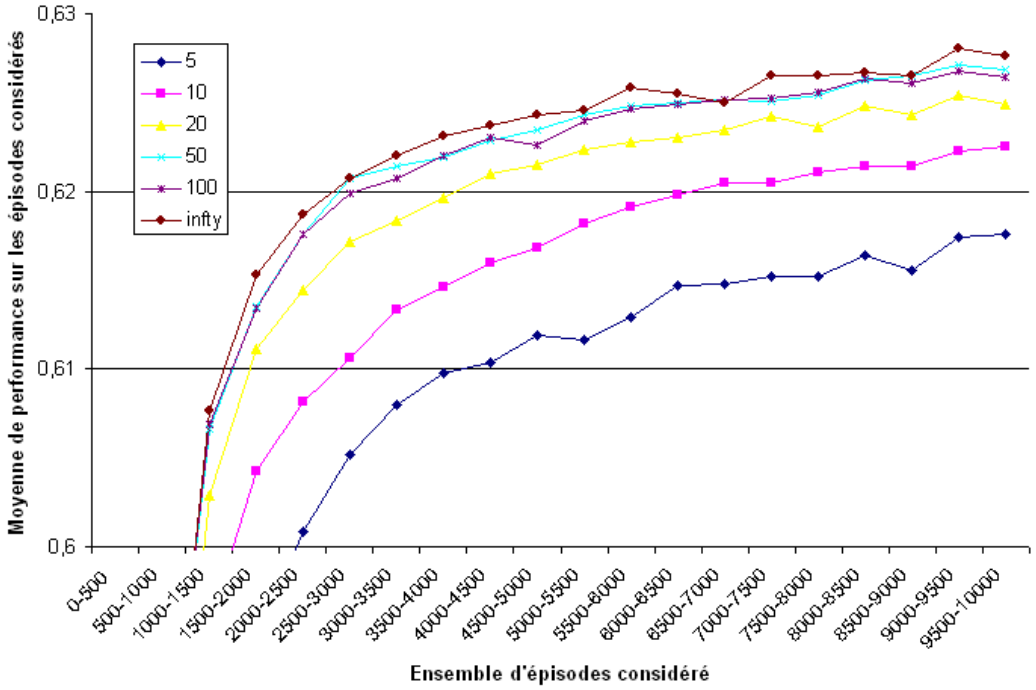


FIGURE 5.14 – Test d'allègement de la charge où la mise à jour de la politique n'est effectuée qu'un nombre de fois de l'ordre de $O(\sqrt{n})$ suivant l'algorithme 9.

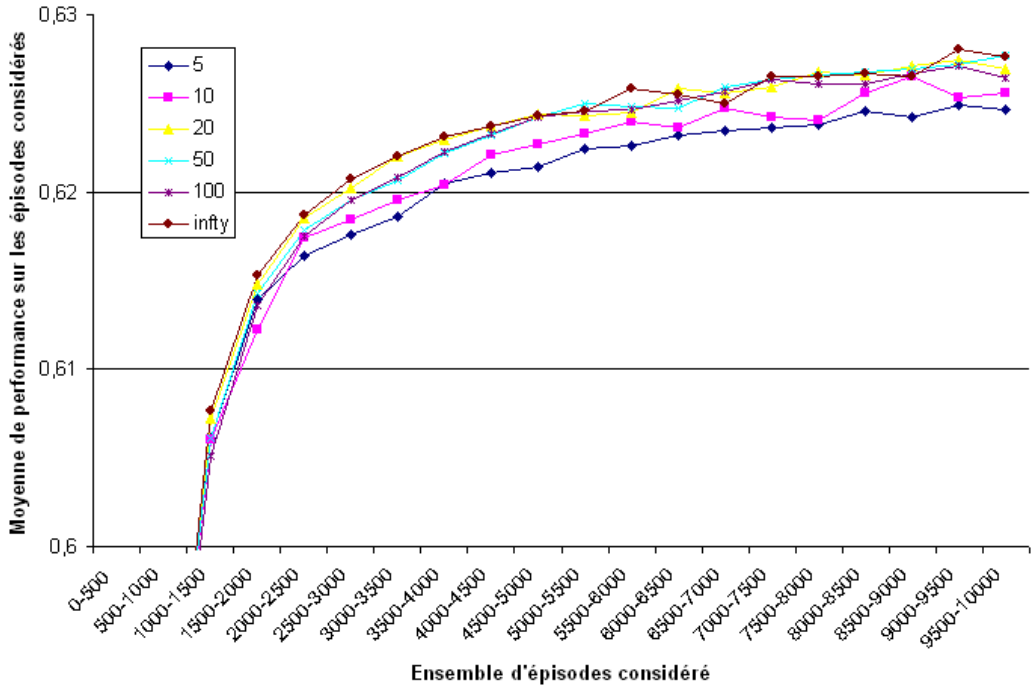


FIGURE 5.15 – Test d'allègement de la charge où la mise à jour de la politique n'est effectuée qu'un nombre de fois de l'ordre de $O(\sqrt{n})$ suivant l'algorithme 8.

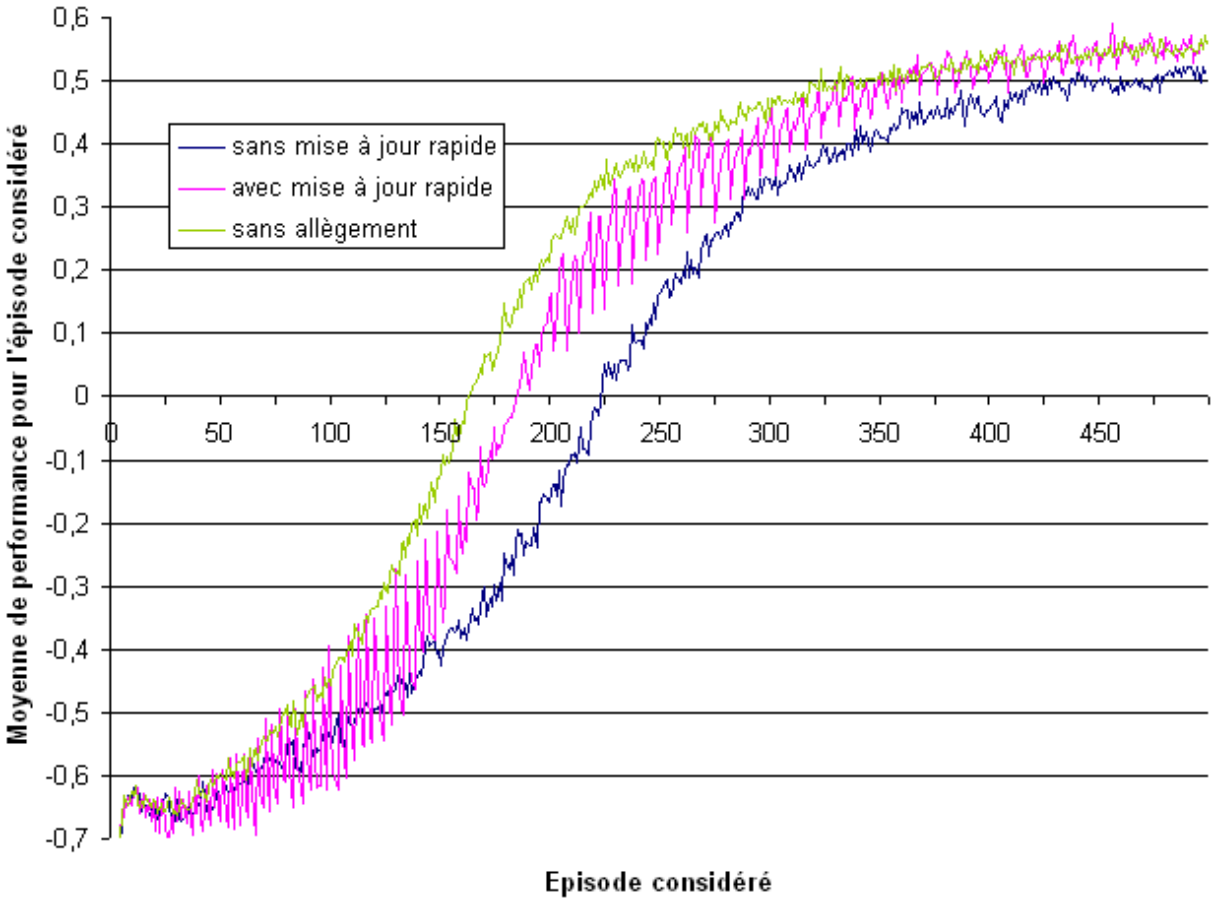


FIGURE 5.16 – Comparaison des algorithmes 9 et 8 avec $\delta = 5$ dans les 500 premiers épisodes.

Par exemple, si la politique actuelle est d'aller en case 5 lorsque l'on se trouve en case 4 et d'aller en case 6 lorsque l'on se trouve en case 5, la mise à jour rapide apprendra non seulement à ne plus aller en case 6 quand on est en case 5, mais aussi à ne plus aller en case 5 quand on est en case 4 et encore de ne plus aller en case 4 quand on est en case 3, ce qui est sous-optimal (la stratégie optimale est de se diriger au maximum vers la case 4).

Les mises à jour complètes permettent de résoudre ces problème par "renforcement". Elles ont une grosse influence négative localement, mais elles évitent de tomber dans des optima locaux. Sur la figure 5.16, la comparaison de la courbe "sans mise à jour rapide" avec la courbe "sans allègement" montre que l'utilisation de la mise à jour rapide systématique n'accélère pas la convergence de l'algorithme, bien au contraire.

Chapitre 6

Expérimentation laboratoire

Ce chapitre présente une expérimentation laboratoire réalisée dans le cadre du projet CLASSiC où l'algorithme CBRL a été implémenté. L'objectif de cette expérimentation est de démontrer l'utilité de notre approche hybride système expert/algorithme d'apprentissage et de montrer que l'adaptation du système a entraîné une amélioration des différents indicateurs de performance (KPI pour Key Performance Indicator) du service.

Le chapitre s'organise de la manière suivante : la section 6.1 décrit précisément l'implémentation du système de dialogue utilisé lors de l'expérimentation, ensuite, la section 6.2 définit les réglages et choix expérimentaux, et enfin la section 6.3 donne les résultats et leur analyse.

6.1 La description du système

Le système servant à l'expérimentation est un système de démonstration qui aide un utilisateur à installer sa box DSL (ici, la Livebox).

6.1.1 Implémentation

Comme procéder à l'installation de la Livebox par téléphone n'est pas très pratique, le système essaie dans un premier temps de trouver un autre moyen pour aider l'utilisateur à installer sa Livebox. Ainsi, pendant la première partie du service, le système pose des questions pour trouver une assistance plus confortable pour l'utilisateur qu'un service de dialogue : le système demande à l'utilisateur s'il souhaite qu'un technicien installe sa Livebox (s'il est d'accord, l'utilisateur est redirigé vers le service de prise de rendez-vous), s'il est actuellement à la maison (s'il n'est pas à la maison, le système demande à l'utilisateur de rappeler quand il le sera) et enfin si l'utilisateur a accès à internet (si c'est le cas, le système fournit à l'utilisateur un URL où il pourra trouver la procédure d'installation de

la Livebox). Une fois que ces questions ont été posées et si les réponses de l'utilisateur n'ont pas permis de court-circuiter la procédure d'installation par téléphone, la procédure d'installation matérielle commence. Elle implique les branchements au secteur, à la prise ADSL, à l'ordinateur (par câble Ethernet) et la mise en place des filtres ADSL.

6.1.2 Points de choix d'alternatives

Au lieu de spécifier un service complètement déterministe, nous avons expérimenté un système flexible, de manière à voir quel type d'information il est possible d'inférer à l'aide de la variabilité du système. Dans cette optique, plusieurs alternatives ont été introduites à plusieurs endroits dans la spécification du service.

Le message d'accueil

Trois variantes de styles de parole ont été testées pour la synthèse du message d'accueil :
- Pas d'accentuation particulière, voix neutre.
- Voix expressive et positive.
- Voix calme et douce.

Ordre des question préliminaires

Comme la question sur l'accès à internet n'est pertinente que si l'utilisateur est chez lui, cette question n'est posée qu'après une réponse positive à la question "êtes-vous chez vous ?". Mais l'ordre entre les questions "êtes-vous chez vous ?" et "souhaitez-vous l'aide d'un technicien ?" sont indépendantes et elles peuvent être interverties. En conséquence, deux stratégies ont été définies pour l'ordre des questions préliminaires.

Generation de la question "êtes-vous chez vous ?"

Deux variantes de génération linguistique ont été testées pour la question "êtes-vous chez vous ?" :
- Une variante directive : "Je souhaiterais savoir si vous êtes actuellement à votre domicile."
- Une variante interrogative : "Êtes-vous actuellement chez vous ?"

Trois variantes de styles de parole ont de nouevau été testées pour la synthèse du message d'accueil :
- Pas d'accentuation particulière, voix neutre.
- Voix expressive et positive.
- Voix calme et douce.

La stratégie d'installation

Durant la procédure d'installation, il y a beaucoup de branchements de matériel à décrire à l'utilisateur : identification de l'adaptateur ToIP (qui n'est pas branché) et les branchements des filtres ADSL et des câbles secteur, ADSL et Ethernet. Il n'y a pas d'ordre évident pour traiter chacune de ces manipulations. En conséquence, nous avons pensé que l'installation pourrait être facilitée si le système expliquait dans un premier temps comment identifier chacun des éléments de la boîte, avant de commencer l'installation en elle-même. Au final, cinq stratégies différentes ont été testées :

- Stratégie 1 : le système vérifie dans un premier temps que chaque élément est bien présent dans la boîte en demandant à l'utilisateur de les identifier dans cet ordre : le câble secteur, le câble ADSL, le câble Ethernet, les filtres ADSL et l'adaptateur ToIP. Ensuite, une fois les identifications terminées, il se charge des branchements des différents éléments dans le même ordre.
- Stratégie 2 : l'ordre est le même que pour la stratégie 1, mais le système ne passe pas par l'étape d'identification/vérification du contenu de la boîte. Il se charge directement des branchements.
- Stratégie 3 : le système vérifie dans un premier temps que chaque élément est bien présent dans la boîte en demandant à l'utilisateur de les identifier dans cet ordre : l'adaptateur ToIP, les filtres ADSM, le câble ADSL, le câble Ethernet et le câble secteur. Ensuite, un fois les identifications terminées, il se charge des branchements des différents éléments dans le même ordre.
- Stratégie 4 : l'ordre est le même que pour la stratégie 3, mais le système ne passe pas par l'étape d'identification/vérification du contenu de la boîte. Il se charge directement des branchements.
- Stratégie 5 : l'ordre est le même que pour la stratégie 3, mais le système demande à l'utilisateur d'effectuer le branchement juste après l'identification de chaque composant.

6.2 Réglages expérimentaux

6.2.1 Tests préliminaires

Une première vague de tests a été menée de manière à déterminer les principaux défauts du système. Dix testeurs externes ont expérimenté le système dans des conditions réelles.

La découverte principale concerne la reconnaissance vocale qui était beaucoup trop sensible au bruit ambiant, particulièrement lorsque l'utilisateur manipule la boîte pour

y chercher les différents éléments. Cette sensibilité au bruit ambiant créait et a continué de créer beaucoup de déclenchements de la reconnaissance vocale sans que le système n'y reconnaisse aucun mot. Ces erreurs de reconnaissance vocale ont entraîné une grande frustration de la part de ces "bêta-testeurs".

Cette pré-évaluation a également permis de détecter un problème dans la logique de dialogue. Après la question "êtes-vous chez vous ?", la demande de confirmation était ambiguë : "pouvez-vous confirmer que vous n'êtes pas chez vous ?". Les utilisateurs se trouvaient partagés sur la réponse à donner : "oui, je confirme" ou "non, je ne suis pas chez moi." ?

Le service a été rectifié en conséquence. La détection d'activité de voix a été réduite, et le prompt de confirmation a été modifié en un prompt plus directif : "j'ai compris que vous n'êtes pas chez vous actuellement. Si cela est correct, dites oui".

6.2.2 Objectifs expérimentaux

L'objectif de l'expérimentation n'est ni de prouver l'utilisabilité du service, ni d'obtenir de la part de nos testeurs des conseils pour l'amélioration du service. Cette expérimentation s'efforce d'illustrer les résultats des approches (voir la sous-section 4.4.1), des modèles (MVDP, voir la sous-section 4.4.4) des algorithmes (CBRL, voir la sous-section 4.5.1) et les outils de l'état de l'art (le retour d'usage, voir le chapitre 3).

Ainsi, le but est de montrer les bénéfices apportés par l'apprentissage dans les systèmes de dialogue conventionnels (c'est-à-dire implémentés manuellement). Dans cette optique, des alternatives raisonnables ont été implémentées (celles présentées dans la section 6.1). Nous attendons de cette expérience une amélioration de son "auto-évaluation" au cours du temps. Ensuite, nous essaierons d'identifier la corrélation entre l'évolution de cette auto-évaluation et les évolutions des différents indicateurs objectifs et subjectifs (KPI).

L'expérimentation n'a pas non plus pour objectif d'accepter les résultats de l'apprentissage comme des connaissances universelles, ni de trouver des explications sur la raison pour laquelle telle alternative a été plus performante qu'une autre. Nous aurions besoin d'une expérimentation plus large avant d'être capables de proposer des résultats de ce type. Nous essayons simplement de montrer que dans un contexte donné –pas forcément le contexte réel, mais un contexte constant au cours de l'expérimentation– certaines alternatives sont plus performantes que d'autres et que le système peut s'améliorer au court du temps en sélectionnant ces alternatives performantes.

En conséquence, l'effort n'a particulièrement porté ni sur la performance du système a priori, ni sur le réalisme des conditions expérimentales. En revanche, nous avons été particulièrement attentifs à la constance du protocole expérimental du début à la fin de l'expérimentation.

L'analyse de l'expérimentation suit quatre axes : l'auto-évaluation du système (voir la sous-section 6.3.1), les performances relatives des différentes alternatives (voir la sous-section 6.3.2), l'évaluation objective (voir la sous-section 6.3.3) et enfin l'évaluation subjective (voir la sous-section 6.3.4).

6.2.3 Protocole expérimental

Chaque testeur fera exactement quatre dialogues avec le système, selon les scénarii suivants : lors du premier dialogue, le superviseur demande au testeur d'interagir naturellement et librement avec le système, selon la situation dans laquelle il s'imaginerait appeler le système. Les scénarii 2, 3 et 4 correspondent aux questions de court-circuitage de l'installation, respectivement l'utilisateur n'est pas chez lui, l'utilisateur a internet et l'utilisateur souhaite qu'un technicien installe sa Livebox. Le scénario 5 est finalement le scénario où le système aide l'utilisateur à installer sa Livebox. Comme le premier dialogue se terminera mécaniquement par l'un des scénarii 2, 3, 4 ou 5, les trois derniers dialogues suivront les trois scénarii restants.

Pour les quatre premiers testeurs (c'est-à-dire pour les 16 premiers dialogues), le système choisit les alternatives de manière complètement aléatoire, de manière à avoir une évaluation précise du système initial. Ensuite, petit à petit le système exploite ce qu'il a appris.

De manière à réduire la variabilité d'un superviseur à un autre, les superviseurs ont reçu des directives (voir l'annexe E).

6.3 Résultats expérimentaux

40 testeurs ont participé à l'expérimentation, ce qui a permis de réaliser 160 dialogues. L'expérimentation a donné des résultats que nous interprétons et analysons selon quatre axes :

- Évolution de l'auto-évaluation au court du temps (sous-section 6.3.1).
- Convergence des points de décision et les confiances à porter sur les écarts observés entre les différentes alternatives (sous-section 6.3.2).
- Évolution des différentes évaluations objectives et leur corrélation avec l'auto-évaluation (sous-section 6.3.3).
- Évolution des différentes évaluations subjectives et leur corrélation avec l'auto-évaluation (sous-section 6.3.4).

6.3.1 Auto-évaluation du système

Réglages

L'auto-évaluation du système est définie à partir des spécifications fonctionnelles du service. De bonnes spécifications fonctionnelles doivent expliciter les niveaux de performance requis et les métriques associées. Elle cherchent à mesurer la qualité des interactions utilisateur/système, selon la perception du système. Dans le système de l'expérimentation, les récompenses sont distribuées de la manière suivante :

- -1 pour chaque rejet de la reconnaissance vocale (ASR).
- -1 pour chaque rejet de l'analyse sémantique (SLU).
- -1 pour chaque "time-out" (quand l'utilisateur ne répond pas au système dans un laps de temps prédéfini).
- -10 quand l'utilisateur raccroche.
- -10 après la réponse "non" à la question finale : "avez-vous réussi à installer votre Livebox ?"
- +5 après avoir fourni à l'utilisateur le message expliquant qu'il doit être chez lui pour installer sa Livebox.
- +5 après avoir fourni à l'utilisateur le message expliquant comment trouver sur internet la procédure d'installation de la Livebox.
- +5 après avoir transféré l'utilisateur vers le service de prise de rendez-vous avec un technicien.
- +10 après la réponse "oui" à la question finale : "avez-vous réussi à installer votre Livebox ?"

Ces récompenses sont légèrement réduite à chaque tour de dialogue (γ de la formule 4.1 vaut 0.975). Cette valeur très faible a été choisie parce que les dialogues peuvent être très longs (plus de 20 tours de dialogue) suivant la stratégie d'installation suivie. Après 20 tours de dialogue la récompense reste de 60% de sa valeur initiale.

Évolution de la performance globale

Les résultats expérimentaux montrent que les choix de l'algorithme d'apprentissage ont entraîné une amélioration de la performance globale du système dans le dialogue au court du temps, selon l'auto-évaluation du système définie en début de sous-section. Toutefois, la figure 6.1 doit être analysée avec un certain recul, à cause du faible volume de testeurs et de la variabilité inhérente à de telles interactions.

L'évolution initiale constante peut être expliquée par le fait que la politique est pratiquement entièrement exploratoire à ce stade de l'expérimentation. Ensuite, entre les abscisses [11,50] et [31,70], la performance augmente fortement, sans doute grâce à l'ap-

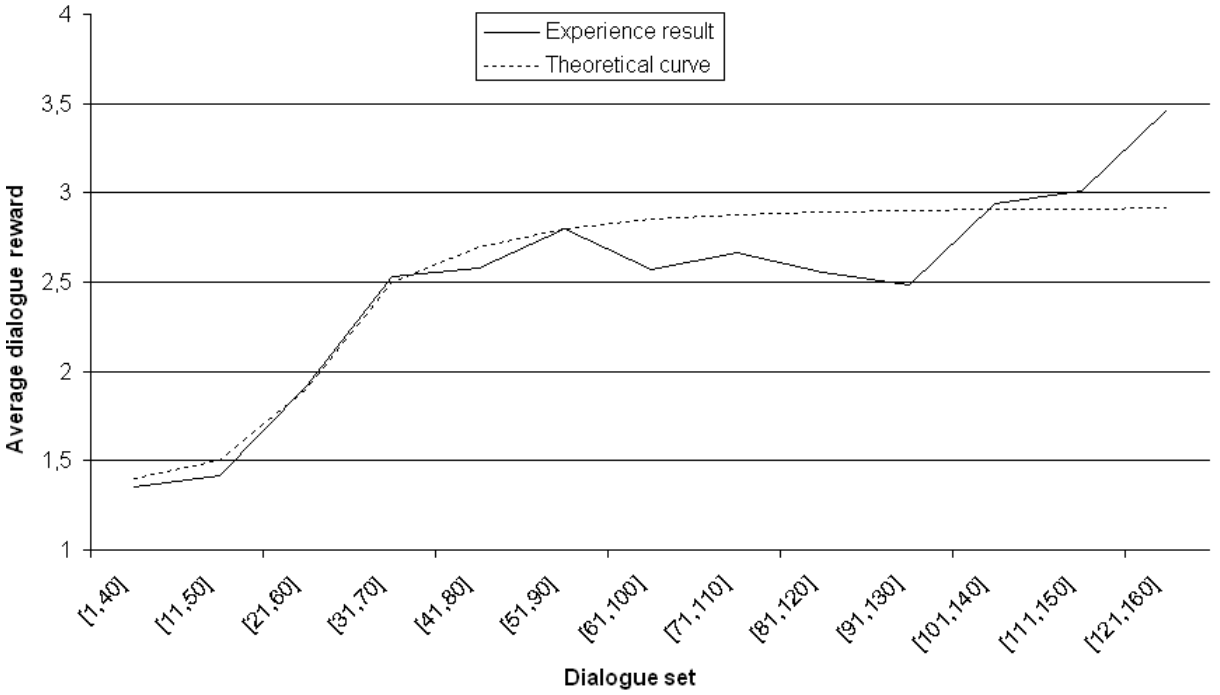


FIGURE 6.1 – Évolution de la performance globale au court du temps. Une lissage sur 40 dialogues a été réalisé de manière à avoir une courbe où la tendance est plus facile est à lire. La courbe en pointillés est une extrapolation de l'apprentissage par une sigmoïde, de manière à rendre compte que l'espérance réelle de récompense du système au cours du temps.

prentissage du système. La légère décroissance de la courbe autour de l'abscisse [81,120] est probablement le fruit du hasard, à cause de testeurs moins performants que la moyenne. De même, les récompenses très élevées obtenues à la fin de l'expérimentation doivent être relativisées par le fait qu'aucun apprentissage spécifique n'a été effectué à ce moment, et que cette variation ne peut être expliquée que par le hasard, avec des testeurs plus performants que la moyenne. La courbe en pointillés montre comment nous extrapolons l'espérance de performance du système au cours du temps.

L'amélioration globale numérique atteint approximativement 1.5. Ce nombre peut être comparé aux récompenses définies dans la section 6.3.1. Étant donné que nous n'avons pas eu de testeur qui ait raccroché, cette amélioration est équivalente à 1.5 erreur(s) d'ASR et SLU en moins par dialogue. Ce résultat montre que des systèmes très similaires peuvent varier énormément en terme de performance. Cela montre également que notre algorithme a permis d'améliorer significativement le système en un temps très restreint.

6.3.2 Convergence des points de décision

A cause de la limitation du nombre d'appels, les résultats ci-dessous doivent être confrontés à leur portée statistique. Plus précisément, nous comparons deux alternatives a_1 et a_2 sur la base de leurs moyennes et écart-types de récompenses à long terme respectifs pour déterminer la probabilité exacte que $Q_m(v, a_1) > Q_m(v, a_2)$ étant donné les observations que l'on vient de faire, ce qui constitue un indicateur sur la fiabilité du résultat. Pour des détails sur la méthodologie suivie, nous vous invitons à vous reporter vers l'annexe F.

Le message d'accueil

La table suivante donne les espérances de performances, les écart-types et le nombre de tentatives de chaque alternative pour le message d'accueil.

Alternatives	$\mathbb{E}(r_m(v,a))$	$\sigma(r_m(v,a))$	Effectif
neutre	2.2	5.0	11
expressive	0.9	7.1	33
calme	2.8	4.1	115

La table ci-dessous indique les probabilités de dominance deux à deux entre les différentes alternatives.

Alternative a_1	Alternative a_2	$P(\mathbb{E}(r_m(v,a_1)) \geq \mathbb{E}(r_m(v,a_2)))$
calme	expressive	0.93
neutre	expressive	0.73
calme	neutre	0.66

Nous pouvons donc conclure de manière assez fiable que l'alternative calme est meilleure que l'alternative expressive. En revanche la qualité de l'alternative neutre est plus difficile à classer.

Ordre des questions de court-circuitage

La table suivante donne les espérances de performances, les écart-types et le nombre de tentatives des deux ordres possibles pour poser les questions de court-circuitage.

Alternatives	$\mathbb{E}(r_m(v,a))$	$\sigma(r_m(v,a))$	Effectif
technicien $\rightarrow$ maison	-0.1	7.9	16
maison $\rightarrow$ technicien	2.7	4.6	143

La table ci-dessous indique les probabilités de dominance deux à deux entre les différentes alternatives.

Alternative a_1	Alternative a_2	$P\left(\mathbb{E}\left(r_m\left(v, a_1\right)\right) \geq \mathbb{E}\left(r_m\left(v, a_2\right)\right)\right)$
mais. $\rightarrow$ tech.	tech. $\rightarrow$ mais.	0.92

Ici, la différence de performance entre les deux alternatives est suffisamment importante pour justifier de n'avoir essayé que 16 fois de poser la question du technicien avant de poser la question "êtes-vous chez vous ?".

La question "êtes-vous chez vous ?"

La table suivante donne les espérances de performances, les écart-types et le nombre de tentatives de chaque alternative pour la question "êtes-vous chez vous ?".

Alternatives	$\mathbb{E}\left(r_m\left(v, a\right)\right)$	$\sigma\left(r_m\left(v, a\right)\right)$	Effectif
directive	-0.4	7.6	19
interrogative expressive	2.8	5.0	145
interrogative calme	-1.1	8.1	17
interrogative neutre	1.7	6.1	18

La table ci-dessous indique les probabilités de dominance deux à deux entre les différentes alternatives.

Alternative a_1	Alternative a_2	$P\left(\mathbb{E}\left(r_m\left(v, a_1\right)\right) \geq \mathbb{E}\left(r_m\left(v, a_2\right)\right)\right)$
inter. expressive	inter. calme	0.97
inter. expressive	directive	0.96
inter. neutre	inter. calme	0.87
inter. expressive	inter. neutre	0.76
inter. neutre	directive	0.71
directive	inter. calme	0.60

L'alternative interrogative expressive domine clairement toutes les autres. Cela peut sembler surprenant lorsqu'on le compare à l'alternative préférentielle du message d'accueil. Mais, la spécificité de cette synthèse est également que la partie interrogative est accentuée, ce qui peut avoir induit une meilleure compréhension de la qualité interrogative de la question et du moment quand l'utilisateur est supposé répondre. De plus, le message d'accueil a généralement le rôle d'apaiser l'utilisateur, alors que l'idée de la question "êtes-vous chez vous ?" est de faire comprendre à l'utilisateur qu'il doit interagir.

La stratégie d'installation

La table suivante donne les espérances de performances, les écart-types et le nombre de tentatives de chaque alternative de stratégies (elles sont décrites dans la sous-section 6.1.2).

Alternatives	$\mathbb{E}(r_m(v,a))$	$\sigma(r_m(v,a))$	Effectif
Stratégie 1	-0.1	7.4	31
Stratégie 2	-6.3	6.5	4
Stratégie 3	-6.7	5.4	2
Stratégie 4	-14	0	1
Stratégie 5	-5.9	11.1	2

Ce point de choix n'est atteint que lors du scénario 5, lors de l'installation complète de la Livebox. Le nombre de tentatives a donc été fortement réduit. Les résultats ne constituent pas de conclusions tangibles. La phase d'exploration a clairement été sous-estimée pour ces points de choix.

Conclusions générales des résultats des points de choix

On remarque que lors de l'expérience le système a toujours favorisé l'alternative ayant la meilleure espérance de récompense à long terme. Cependant, plusieurs alternatives concurrentes auraient mérité plus de tentatives pour confirmer statistiquement leur optimalité. Idéalement, il aurait fallu plus de dialogues et une meilleure stratégie d'exploration. C'est d'ailleurs cette expérience qui nous a mené à réfléchir aux compromis exploration/exploitation de la section 5.4.

6.3.3 Evaluation objective

Les indicateurs (KPI) du système sont nombreux. Dans cette expérience, nous avons mesuré seulement les KPI globaux au dialogue (nombre de tours de dialogue, durée moyenne du dialogue et les nombres d'erreurs d'ASR, de SLU ou d'inactivité). Nous évaluons ainsi la corrélation entre l'apprentissage sur l'auto-évaluation et l'évolution de ces KPI au cours du temps.

Implémentation

Nous classifions les KPI mesurés en deux catégories : les KPI visant à mesurer la durée d'un dialogue (la durée du dialogue en secondes et en tours de parole), et les KPI visant à calculer le nombre d'erreurs par dialogue. Les KPI de durée sont obtenus automatiquement alors que les KPI d'erreurs sont calculés à l'aide d'annotations manuelles de dialogues.

Les résultats expérimentaux des KPIs de durée

Les deux KPI que nous considérons dans cette partie sont la durée du dialogue en secondes et en tours de dialogue. La première se base sur l'unité de temps universelle

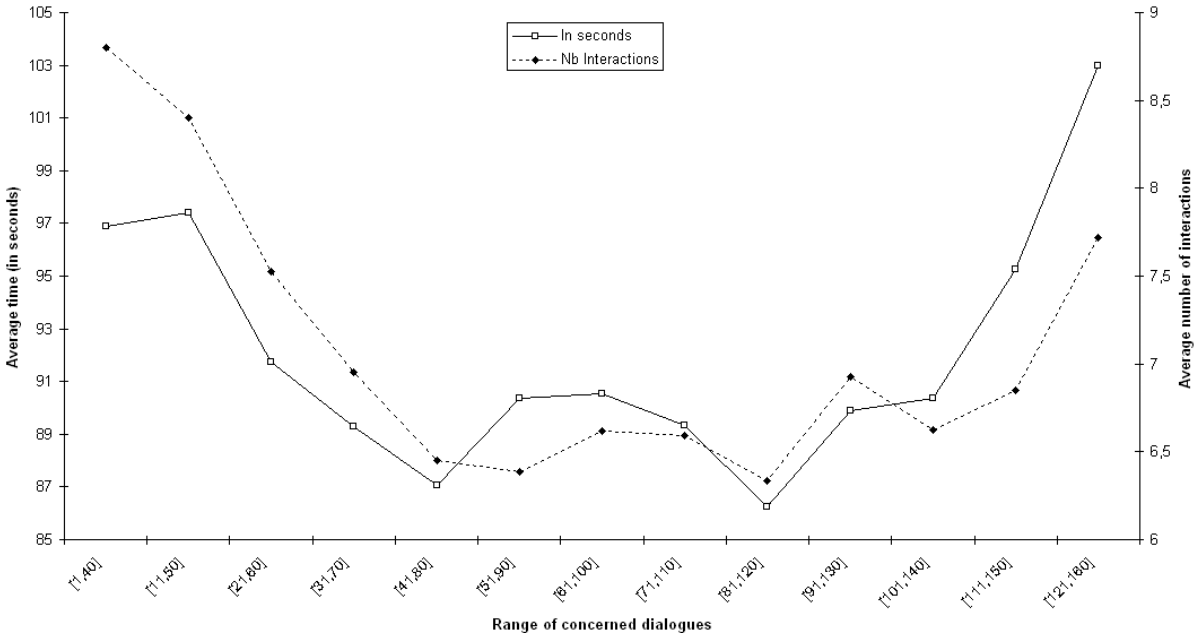


FIGURE 6.2 – Évolution des KPI de durée au cours de l'apprentissage du système

et peut naturellement être considéré comme un KPI objectif. Concernant la seconde en revanche, le système et l'utilisateur peuvent ne pas avoir eu la même vue des tours de dialogue. Dans le scénario de l'installation complète de la Livebox, cette inexactitude est renforcée par le fait que la reconnaissance vocale est très sensible au bruit occasionné par la manipulation de la Livebox, et que ce bruit génère un grand nombre d'interactions vides du point de vue du système.

La figure 6.2 montre que les KPI de durée ne sont pas corrélées à l'auto-évaluation du dialogue de façon concluante. Les variations de 10% autour de leur moyenne sont probablement un résultat lié au bruit.

Les résultats expérimentaux des KPIs d'erreur

Quatre KPI objectifs sont considérés dans cette partie :

- Bruit de l'ASR : les erreurs de reconnaissance vocale occasionnées par le bruit ambiant (le testeur n'a pas parlé mais le module ASR s'est déclenché).
- Erreur de l'ASR : les erreurs de reconnaissance vocale de mécompréhension (le système n'a pas correctement compris l'utilisateur).
- Bruit+Erreur de l'ASR : la somme des deux premiers KPI.
- Erreur après le SLU : nombre d'erreurs après analyse sémantique. Certaines erreurs de reconnaissance vocale sont rattrapées par l'analyse sémantique. Ce KPI est le plus significatif dans la mesure où il recense les erreurs d'incompréhension totale du

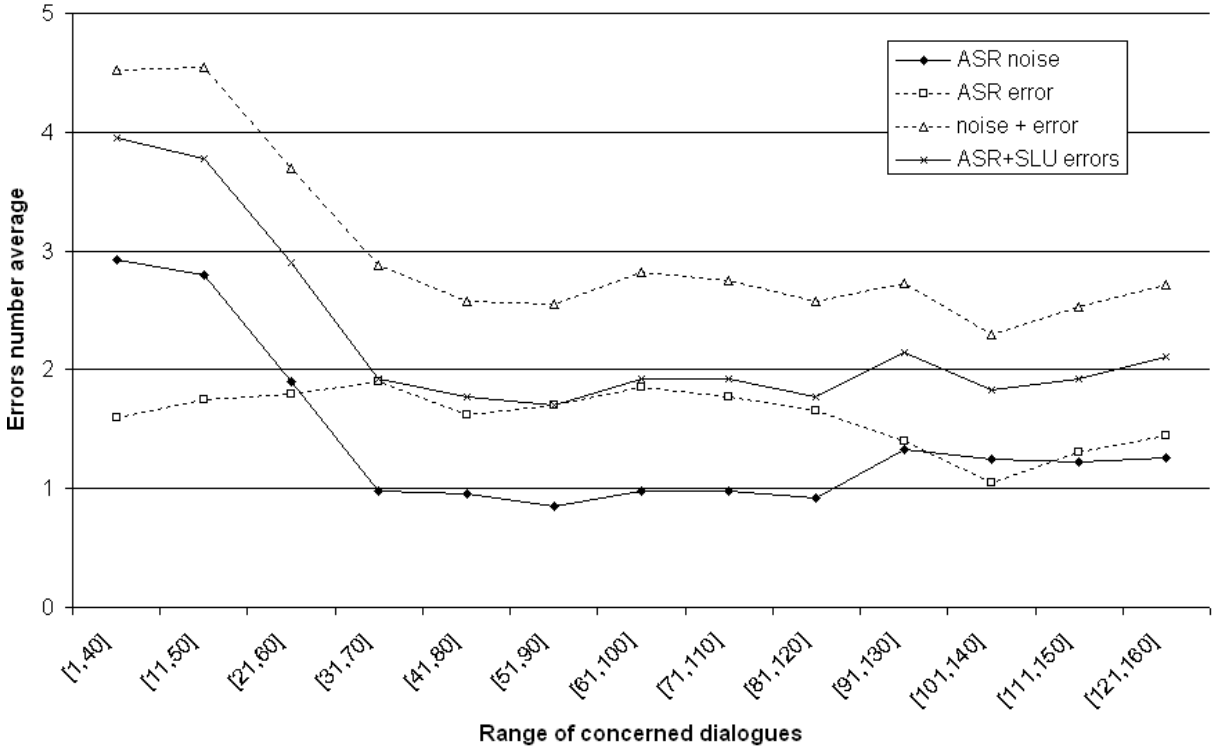


FIGURE 6.3 – Evolution des KPI d'erreur au cours de l'apprentissage du système

système.

La figure 6.3 révèle que le bruit de l'ASR est fortement corrélé aux récompenses à long terme obtenues par le système et donc que l'apprentissage réalisé par le système a contribué à baisser ce type d'erreur. Il semblerait donc que la façon de parler à l'utilisateur influence l'utilisateur sur sa manière d'interagir avec le système et de manipuler de la Livebox. Les erreurs de l'ASR semblent avoir moins été affectées par la modification de stratégie du système, sans doute parce que le vocabulaire était suffisamment simple pour que les utilisateur comprennent que le meilleur moyen d'interagir avec le système est de dire "oui" ou "non" lors des questions initiales et "suivant" lors de l'installation. L'erreur après le SLU est directement impactée par la réduction du déclenchement de l'ASR à cause du bruit. Au final, le nombre moyen d'erreurs après le SLU a été réduit par deux, ce qui est un résultat impressionnant après seulement 160 dialogues.

6.3.4 Évaluation subjective

La perception du système par l'utilisateur est mesurée à l'aide d'un questionnaire basé sur une échelle de Likert. L'évaluation a donc été réalisée sur une échelle à 5 niveaux, 1 étant la note minimale et 5 la note maximale. Ce questionnaire vise à évaluer la qualité des dialogues selon le testeur. La figure E.1 est une capture d'écran de ce questionnaire.

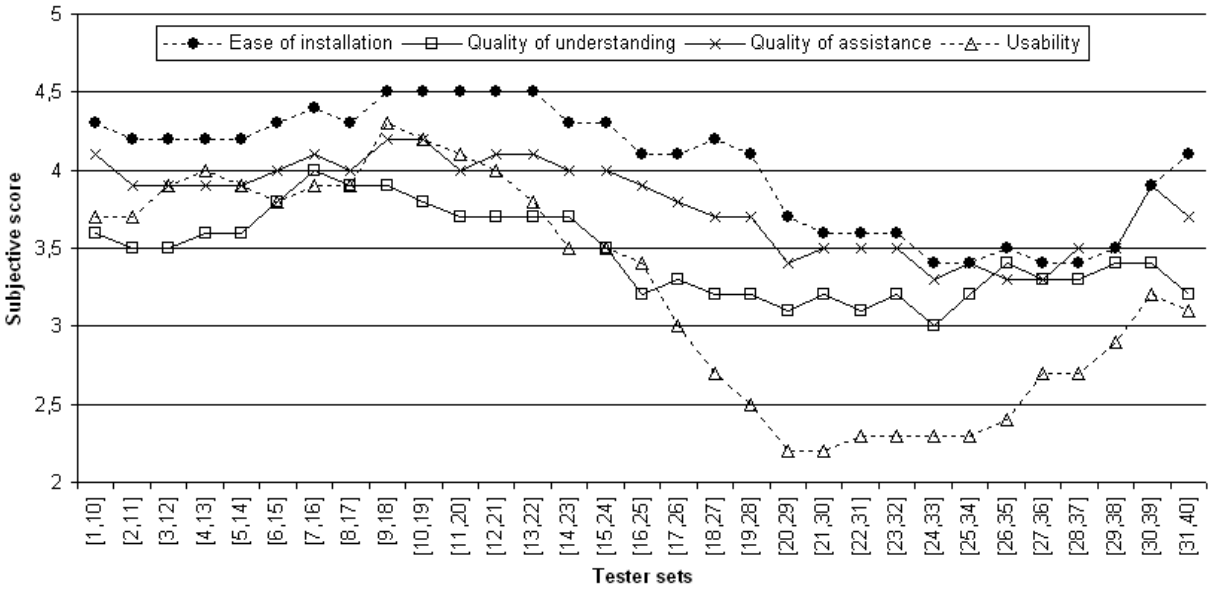


FIGURE 6.4 – Evaluation subjective.

Il comporte quatre questions : une première question sur la facilité de l'installation, la seconde sur la qualité de la compréhension, la troisième sur la pertinence de l'aide et enfin la dernière question sur l'utilisabilité du service.

Résultats expérimentaux

L'évaluation subjective ne reflète pas les résultats de l'auto-évaluation et de l'évaluation objective. Même pire, la performance moyenne semble décroître au cours du temps, en particulier en ce qui concerne la question de l'utilisabilité du service. Cette décroissance peut être expliquée par le changement de population des testeurs entre la première partie et la seconde partie de l'expérimentation. En effet, les quinze premiers testeurs sont des collègues d'Orange, alors que les vingt-cinq suivants sont des étudiants de l'École Nationale Supérieure des Télécommunications de Bretagne.

Si l'on excepte la chute observée à la mi-expérimentation, la variabilité est due au hasard et elle ne véhicule aucun enseignement, si ce n'est l'absence de corrélation entre l'auto-évaluation et l'évaluation subjective. La figure 6.5 montre d'une manière graphique, à l'aide d'un nuage de points, que les deux évaluations ne sont absolument pas corrélées. Le tableau ci-dessous confirme cette impression graphique. La dernière ligne du tableau affiche le coefficient de corrélation des évaluations de chaque question, par rapport à l'auto-évaluation. Ces nombres sont très proches de 0, ce qui signifie simplement qu'il n'y a aucune dépendance entre ces évaluations. Ce résultat démontre que les testeurs ne sont pas capables d'évaluer eux-même la qualité de l'interaction qu'ils ont eu avec le système. La non-corrélation entre les évaluations objective et subjective apparaît régulièrement dans

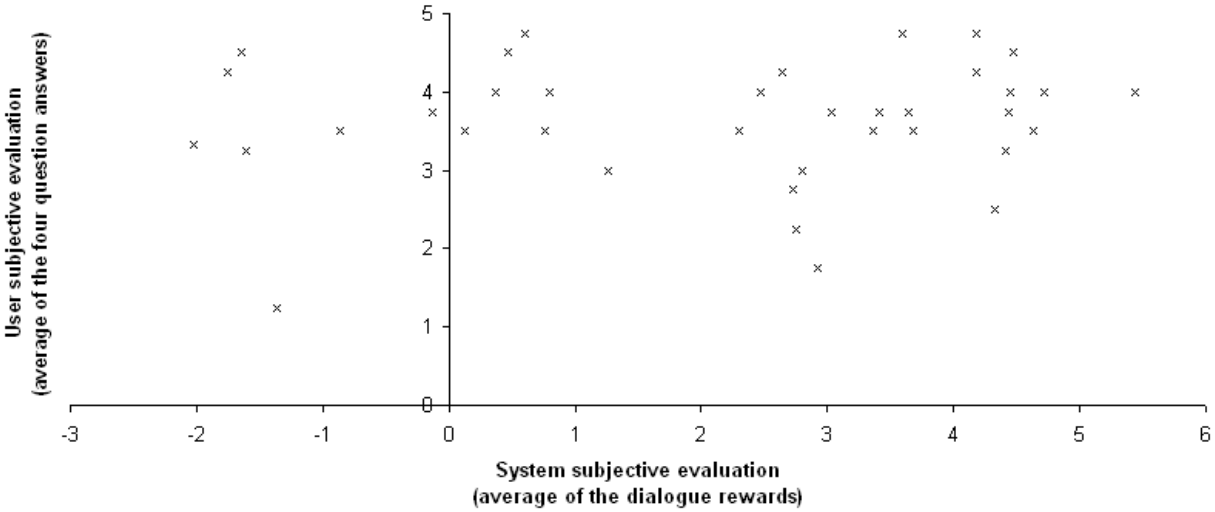


FIGURE 6.5 – System subjective evaluation vs. User subjective evaluation

les expérimentations sur les systèmes de dialogue (Lemon et al., 2006). Ceci s'explique par le fait qu'ils n'ont aucune référence sur la qualité présupposée d'un dialogue avec une machine. Ainsi, certains testeurs ont eu une très mauvaise expérience avec le système, mais ont tout de même rempli une bonne évaluation, parce qu'ils étaient enthousiastes à l'idée de parler à une machine. D'autres testeurs ont eu une bonne expérience, hormis quelques erreurs de reconnaissance vocale, mais ils sont restés très sévères dans leur jugement parce qu'ils ont trouvé inacceptable d'avoir ce genre d'erreurs dans un service industriel.

	Q1	Q2	Q3	Q4	Moyenne
Moyenne	4.1	3.4	3.8	3.3	3.7
Ecart-type	0,97	0,98	1,03	1,36	0,80
Coef. de Correl. avec l'autoeval.	0,16	0,05	0,14	-0,06	0,08

La ligne de la moyenne permet de voir comment les utilisateurs ont noté chaque point par rapport aux autres. Les utilisateurs ont de manière générale trouvé que l'installation était très simple et c'est la raison principale des bons scores obtenus à la question 1. Ils ont aussi ressenti les difficultés de la reconnaissance vocale avec le bruit ambiant et ils ont pénalisé en conséquence la réponse à la question 2. Les bons scores de la question 3 reflètent les résultats de la question 1. La manipulation est très simple, donc les conseils fournis le sont aussi. La mauvaise performance à la question 4 est probablement un corolaire de la bonne performance de la question 1 : les testeurs ont souvent pensé que le service est inutile étant donné la simplicité de la manœuvre.

Le seul enseignement de la ligne sur l'écart-type est que la question 4 a reçu des évaluations plus disparates. Nous interprétons ce résultat par le fait que les testeurs ont pu avoir deux compréhensions de la question :

– "Utiliseriez-vous, personnellement, ce service ?" → faible évaluation
– "Pensez-vous que ce service serait utile pour certaines personnes ?" → forte évaluation

6.4 Conclusions de l'expérimentation

L'évaluation subjective a mené à des résultats mitigés, notamment en comparaison de l'évaluation automatique des récompenses. Il faut cependant garder à l'esprit que, dans un système industriel, l'objectif n'est pas de maximiser l'évaluation subjective, mais de maximiser les indicateurs objectifs, et principalement la complétion de la tâche. Comme l'expérience n'a mené que très rarement à des non-complétions de tâche, il était plus intéressant d'étudier dans cette expérimentation l'effet de l'apprentissage sur les autres indicateurs objectifs, notamment les erreurs de reconnaissance vocale. L'algorithme d'apprentissage a permis d'améliorer considérablement ces indicateurs.

Conclusion

1 Travaux réalisés

Cette thèse porte sur l'implantation de techniques statistiques avancées dans les systèmes de dialogue conventionnels, c'est-à-dire les systèmes de dialogue implantés manuellement. La plupart de ces systèmes de dialogue sont conçus à l'aide d'automates décisionnels décrivant la logique de dialogue. Ces automates sont réputés simplistes, difficiles à concevoir et sous-optimaux. Deux voies d'améliorations ont été explorées dans cette thèse : la gestion des incertitudes créées par les modules d'observation (la reconnaissance vocale et l'interprétation de langue parlée en ce qui concernes les systèmes vocaux) et l'optimisation de ces automates par apprentissage par renforcement en ligne.

Un travail préliminaire sur l'architecture des systèmes de dialogue a été réalisé de sorte à ce que les systèmes conventionnels soient capables de prendre en compte les diverses options créées par les listes N-best générées par les automates. Ce travail fait l'objet d'un livrable du projet européen CLASSiC (Putois et al., 2009; Laroche et al., 2010).

La gestion de l'incertitude a été traitée à l'aide d'un nouveau formalisme de logique probabiliste (le Logical Framework for Probabilistic Reasoning) qui a été publié dans une conférence internationale (Laroche et al., 2008) et dans un livrable du projet européen CLASSiC (Laroche et al., 2009c).

Le travail sur l'apprentissage intègre de multiples aspects de cette thèse allant de l'utilisation de nouvelles fonctionnalités de monotoring intégrées aux outils de conception jusqu'à des travaux d'optimisation de l'apprentissage en ligne, en passant par la définition un nouveau modèle de processus de décision (le Module-Variable Decision Process) et l'implantation de deux nouveaux algorithmes d'apprentissage par renforcement non-markoviens (le Compliance-Based Reinforcement Learning et le Module-Variable Temporal Difference Learning). Une expérimentation laboratoire a fait le "proof of concept" et a montré que l'optimisation sur des alternatives simples et a priori équivalentes permet d'améliorer de façon significative la qualité du service. Certains de ces aspects ont été publiés dans des conférences internationales (Laroche et al., 2009a), d'autres ont été l'objet de livrables du projet européen CLASSiC (Laroche et al., 2009c; Laroche et al., 2009b;

Putois et al., 2009; Laroche et al., 2010; Bretier et al., 2009), et enfin, les derniers aspects seront prochainement soumis dans d'autres conférences (notamment les travaux concernant le retour d'usage, l'exploration, la réduction de la complexité du CBRL et la comparaison avec le MVTDL).

Aujourd'hui, les algorithmes d'apprentissage sont utilisés dans une application de dialogue commerciale : la prise de rendez-vous avec le technicien de dépannage de la ligne fixe. Les données résultant de cette mise en production n'ont pas encore été collectées, si bien qu'elles ne figurent pas dans cette thèse, mais elles seront probablement des sources de publication future. C'est en effet le premier système de dialogue apprenant en ligne déployé commercialement, et c'est une service qui collecte plusieurs dizaines de milliers d'appels par an.

2 Perspectives

Du point de vue théorique, il reste encore beaucoup de facettes à explorer. En voici quelques unes.

Nous pensons principalement à l'intégration de la gestion de l'incertitude dans l'apprentissage. Cette tâche est particulièrement complexe dans la mesure où l'on ne dispose pas de modèle markovien ou même simplement probabiliste des connaissances. Elle pose un deuxième problème : celui de l'évaluation. L'exercice de la grille est difficilement adaptable à un modèle partiellement observable. Et même dans le cas où elle le serait, la politique optimale ne serait plus évidente, et en conséquence il serait difficile d'évaluer en absolu les politiques apprises.

Il reste également un travail théorique à réaliser au sujet de l'algorithme Compliance-Based Reinforcement Learning Monte Carlo pour mieux comprendre comment optimiser la fonction entre la compatibilité et le poids. Cette thèse a fait une étude d'optimisation de cette fonction sur l'exercice, sans toutefois réussir à justifier théoriquement cet optimum.

Enfin, un autre axe de recherche important concerne l'adaptation du modèle Module-Variable Decision Process aux espaces continus notamment en ce qui concerne l'ensemble des états locaux V_m. En fait, le MVDP ne présuppose jamais que les ensembles sont finis, ni même dénombrables, l'algorithme générique CBRL ne présuppose pas lui non plus que V_m est dénombrable, mais tous les algorithmes implantés dans cette thèse (CBRLMC et MVTDL) font l'hypothèse de domaines finis.

Du point de vue pratique, les futures étapes sont les suivantes.

Tout d'abord, lors du déploiement commercial, nous allons collecter massivement de données qui nous permettront de faire évoluer l'implantation du système en ligne et ainsi prouver toute la modularité de nos solutions d'apprentissage. Ce sera également une dé-

monstration de l'applicabilité et des bénéfices opérationnels de notre technique d'apprentissage en ligne. Cela renforcera la confiance accordée à ces techniques par l'industrie, qui les implantera progressivement dans des applications de plus en plus ambitieuses.

L'étape suivante concerne le passage de la méthode d'apprentissage à un échelle plus industrielle, en offrant aux développeurs d'applications tous les outils pour que l'apprentissage soit plus facile à aborder : rendre disponible dans l'outil de conception les objets graphiques de "point de choix" et de "récompense". Une fois la fonction entre la compatibilité et le poids comprise, l'algorithme CBRL ne nécessiterait aucun paramètre et un concepteur d'application sans connaissance en apprentissage par renforcement n'aurait pas besoin de fixer lui-même des paramètres obscurs. Toutefois, il resterait indispensable d'établir une méthodologie de bon usage : ne pas surcharger l'application en points de choix, ne pas surcharger les points de choix avec de trop grands espaces V_m, etc...

Ensuite, la gestion de l'incertitude doit être intégrée dans une application de dialogue de laboratoire dans le cadre du projet CLASSiC. En fonction des avancées architecturales et technologiques, l'architecture pourrait ensuite petit à petit migrer celle décrite dans le chapitre 1. Puis, en fonction des avancées théoriques, la gestion de l'incertitude et l'apprentissage en ligne s'intégreraient dans cette architecture statistique de bout en bout, pour créer un système de dialogue robuste et adaptatif.

Pour finir, nous pensons que les applications décrites dans cette thèse ne se limitent pas aux services vocaux interactifs, mais qu'elles peuvent s'étendre aux agents conversationnels (chatbots) et même dans tous les programmes experts interagissant avec l'utilisateur en temps réel (sans que l'on puisse vraiment parler de dialogue) ; nous pensons en particulier à l'intelligence artificielle dans les jeux vidéos.

Glossaire

Alternative Dans cette thèse, nous offrons la possibilité aux développeurs d'applications de dialogue de définir de plusieurs alternatives de conception, c'est-à-dire de proposer plusieurs options au système de dialogue lorsque celui-ci se trouve dans un état global de dialogue donné.

API Application Programming Interface ou interface de programmation. Ensemble de fonctions, procédures ou classes mises à disposition des développeurs par une bibliothèque.

Applicabilité Une démonstration est dite applicable lorsque le contexte permet l'application de la règle décrite par la démonstration, c'est-à-dire si la sémantique permet d'appliquer la démonstration. Ces considérations sémantiques incluent toutes les conditions qui ne font pas partie de la fiabilité. Voir la sous-section 2.3.1 pour plus de détails.

ASR Automatic Speech Recognition ou reconnaissance vocale. Module de dialogue servant à transformer un signal de parole en une suite de mots. Voir la sous-section 1.2.1 pour plus de détails.

ATMS Assumption-based Truth Maintainance System. Système de raisonnement créé par de Kleer (1986a); de Kleer (1986b); de Kleer (1986c). Voir la sous-section 2.5.2 pour plus de détails.

BDI Belief-Desire-Intention. Classe de logique modale dont l'objectif est de simuler les principales attitudes mentales des humains : croyance, désir et intention.

BO Back-Office. Classe de modules de dialogue. Nous réunissons sous cette appellation toutes les unités de calculs situées hors du système de dialogue. Voir la sous-section 1.2.1 pour plus de détails.

BOA Back-Office Act. Type d'acte correspondant aux appels à un BO. Voir la sous-section 1.2.2 pour plus de détails.

Bootstrapping Procédé permettant d'obtenir un résultat importants à l'aide de moyens réduits, aide "additionnelle". En ce qui concerne l'apprentissage par renforcement,

le bootstrapping est un procédé permettant une mise à jour rapide des fonctions d'état Q et V à l'aide d'une estimation des valeurs des états consécutifs.

CA Committing Act. Un acte est dit engageant ou committing lorsque la réalisation de celui-ci crée une obligation pour celui-ci qui l'exécute. Voir la sous-section 1.2.2 pour plus de détails.

CBRL Compliance-Based Reinforcement Learning. Nouvelle famille d'algorithme d'apprentissage par renforcement sans bootstrapping conçue pour les environnements non markoviens. La théorie est décrite dans la sous-section 4.5.1.

CBRLMC Compliance-Based Reinforcement Learning Monte-Carlo. Implémentation directe du CBRL en utilisant l'approche Monte Carlo. L'algorithme est décrit amplement dans la sous-section 4.5.1.

CLASSiC Computational Learning in Adaptive Systems for Spoken Conversation. Projet européen dans lequel s'inscrit la thèse. Le projet est introduit dans la sous-section 1.1.

CM Context Manager. Gestionnaire des connaissances du système de dialogue. Voir la sous-section 1.3.2 pour plus de détails.

Compatibilité Mesure de décision visant à définir la distance entre cette décision et une politique donnée. S'utilise également pour les épisodes, pour définir la distance entre ce qui a été décidé et ce que l'on déciderait actuellement. Voir la sous-section 4.5.1 pour plus de détails.

Contexte Lors d'un dialogue, le système observe des évènements et acquiert un ensemble de connaissances. Ces connaissances forment alors un contexte qui influence le comportement du système.

Corpus Ensemble de données. Les corpus sont souvent utilisés comme base d'apprentissage.

DA Dialogue Act ou acte de dialogue. Action élémentaire communicative. Voir Searle (1969) pour plus de détails.

Décision Un agent est amené à prendre une décision lorsqu'il se trouve sur un point de choix où plusieurs alternatives se proposent à lui. La décision consiste alors à choisir l'une des alternatives et à l'exécuter. Une décision est donc constitué d'un contexte et d'une action. Voir la sous-section 4.4.4 pour plus de détails.

Démonstration Item de la base de connaissance composé d'une proposition et de son support de démonstration. Voir les sous-sections 2.3.1 et **??** pour plus de détails.

DM Dialogue Management. Module ou ensemble de modules suivant les implémentations, servant à prendre les décisions concernant les stratégies de dialogue. Voir la sous-section 1.2.1 pour plus de détails.

DT Dialogue Turn ou tour de dialogue. Temps séparant deux énoncés système. Voir la sous-section 1.2.2 pour plus de détails.

Épisode Interaction entre un système et un environnement ayant un début et une fin. Elle est souvent assimilée à une suite de décisions. Pour un système de dialogue, un épisode est équivallent à un dialogue.

État global De manière à bien faire la distinction entre un état local lié à un noeud de l'automate et l'état global du système, nous avons défini ce terme contenant donc toutes les variables internes du système.

État local Un état local est l'ensemble des variables internes du système accessibles depuis un noeud de l'automate. Réduire cet état local a pour avantage de réduire les dépendances et donc de réduire la complexité de l'apprentissage.

Exploitation Une des particularités de l'apprentissage par renforcement est le dilemme exploitation/exploration. Le système est en phase d'exploitation lorsque celui-ci exploite les connaissances qu'il a acquises jusqu'ici de manière à optimiser ses futures récompenses.

Exploration Une des particularités de l'apprentissage par renforcement est le dilemme exploitation/exploration. Le système est en phase d'exploration lorsque celui-ci explore les actions qu'il connait mal de manière à enrichir sa connaissance.

Fiabilité Une démonstration est dite fiable dans un monde donné, si la démonstration est valide dans ce monde. Ceci implique nécessairement que la proposition démontrée est vraie dans ce monde. Voir la sous-section 2.3.1 pour plus de détails.

Hypothèse Markovienne Hypothèse généralement utilisée pour l'apprentissage par renforcement. Cette hypothèse stipule qu'un état donné à un instant t n'est dépendant que de l'état à l'instant précédent $t-1$ et de l'action exécuté à ce même instant. Cette dépendance peut être stochastique.

IA Internal Act ou acte interne. Acte non-observable de l'extérieur du système. Voir la sous-section 1.2.2 pour plus de détails.

IC Intervalle de Confiance. Méthode pour la gestion de l'équilibre entre l'exploration et l'exploitation.

Inférence logique Combinaison de deux connaissances permettant d'en générer une troisième. Par exemple, "Titi est un oiseau" et "les oiseaux volent" sont deux connaissances permettant d'inférer que "Titi vole".

Information Une information est un signal portant une connaissance, certaines ou incertaine. Une connaissance i véhicule une proposition p avec une certaine fiabilité numérique f. Voir la sous-section 2.3.1 pour plus de détails.

J2EE Java Enterprise Edition est une spécification pour la technique Java de Sun plus particulièrement destinée aux applications d'entreprise. Dans ce but, toute implémentation de cette spécification contient un ensemble d'extensions au framework Java standard (JSE, Java Standard Edition) afin de faciliter la création d'applications réparties.

JSP Le JavaServer Pages est une technique basée sur Java qui permet aux développeurs de générer dynamiquement du code HTML, XML ou tout autre type de page web. Cette technique permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique.

KPI Key Performance Indicator ou indicateurs clé de performance. Indicateurs mesurables d'aide décisionnelle dont le but est de représenter un aperçu d'évolution des facteurs clés de succès d'une application afin d'évaluer sa performance globale en fonction de la tâche qui lui est assignée.

LFPR Logic Framework for Probabilistic Reasoning. Formalisme logique permettant de raisonner sur des incertitudes en manipulant des probabilités. Voir la sous-section 2.3 pour plus de détails.

Likert scale L'échelle de Likert est une échelle répandue dans les questionnaires psychométriques. La personne interrogée exprime son degré d'accord ou de désaccord avec une affirmation.

Livebox La Livebox est un appareil électronique fourni par le Fournisseur d'accès à Internet Orange à ses abonnés ADSL.

LM Learning Manager. Composant servant non seulement à l'enregistrement du comportement du système et des récompenses obtenues, mais également à fournir une politique de comportement optimale étant donné l'apprentissage qu'il a pu avoir de ses enregistrements.

MDP Markov Decision Process ou processus de décision markovien. Modèle stochastique issu de la théorie de la décision et de la théorie des probabilités. Le modèle MDP peut être vu comme une chaîne de Markov à laquelle on ajoute une composante décisionnelle. Ce modèle repose sur l'hypothèse markovienne. Voir la sous-section 4.4.2 pour plus de détails.

Module Dans le modèle MVDP, un module est une entité logicielle capable de prendre des décision étant donné un contexte. Voir la sous-section 4.4.4 pour plus de détails.

Monde Dans le formalisme logique LFPR, un monde est un état des choses. Voir la sous-section 2.3.1 pour plus de détails.

Monde possible Monde consistant, c'est-à-dire vierge de toute contradiction. Voir la sous-section 2.3.1 pour plus de détails.

Monte Carlo On appelle méthode de Monte-Carlo toute méthode visant à calculer une valeur numérique, et utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. Le nom de ces méthodes, qui fait allusion aux jeux de hasard pratiqués à Monte-Carlo (Metropolis & Ulam, 1949).

MVDP Module-Variable Decision Process. Processus de décision permettant de s'affranchir de l'hypothèse markovienne. Voir la sous-section 4.4.4 pour plus de détails.

MVTDL Module-Variable Temporal Difference Learning. Adaptation du TD-Learning au processus de décision MVDP. Voir la sous-section 4.5.2 pour plus de détails.

NIA No Impact Act. Actes qui peuvent être exécutés sans être contrôlés par le système. Voir la sous-section 1.2.2 pour plus de détails.

NLG Natural Language Generation. Module servant à transformer une séquence d'actes de dialogue en une ou plusieurs phrases formant un énoncé. Voir la sous-section 1.2.1 pour plus de détails.

Off-policy Qualifie un algorithme d'apprentissage par renforcement pour renseigner qu'il est capable d'apprendre un comportement optimal même si le système ne suit pas sa politique.

Offline Qualifie un algorithme d'apprentissage par renforcement pour renseigner qu'il est conçu pour apprndre à partir d'un corpus.

Online Qualifie un algorithme d'apprentissage par renforcement pour renseigner qu'il est conçu pour apprendre en interaction avec l'environnement.

Point de choix Dans un studio de développement sous forme d'automate, un point de choix est un noeud de l'automate où plusieurs alternatives sont définies, de telle sorte que le système dispose de la liberté d'utiliser l'alternative qui lui semble la meilleure.

Politique Une politique se définit comme un ensemble d'actions coordonnées, mises en œuvre avec pour objectif d'obtenir une modification ou une évolution d'une situation donnée. Un algorithme d'apprentissage par renforcement définit une politique qui définit les actions du système.

POMDP Partially Observable Markov Decision Process ou processus de décision markovien partiellement observable. Modèle stochastique issu de la théorie de la décision (MDP) et de la théorie des probabilités permettant de l'apprentissage par renforcement en prenant en compte le caractère incertain d'évènements partiellement observables.

Proposition logique Une proposition donne une information sur un état de chose. Ainsi "2 + 2 = 4" ou "le livre est ouvert" sont deux propositions. En logique classique, une proposition peut prendre uniquement les valeurs vrai ou faux.

Q-Function Dans toute cette thèse, la Q-function (ou fonction Q) se rapporte à la fonction d'état-action utilisée dans les algorithmes d'apprentissage par renforcement. Celle-ci n'a rien à voir avec la Q-function utilisée en statistiques.

Q-Learning Algorithme d'apprentissage par renforcement (Watkins, 1989) reposant sur l'hypothèse markovienne et le bootstrapping. Diffère de SARSA par le fait qu'il est off-policy. Cas particulier du TD-Learning.

RBOA Read Back-Office Act. Sous-catégorie des BOA correspondant aux actes de lecture sans modification de la base de données. Voir la sous-section 1.2.2 pour plus de détails.

Récompense Processus d'évaluation utilisé dans les algorithmes d'apprentissage par renforcement dans le but d'encourager ou au contraire punir certains comportements. Ce principe est directement dérivé du système de récompense que l'on retrouve chez les mammifères (Olds & Milner, 1954).

Reinforcement Learning L'apprentissage par renforcement fait référence à une classe de problèmes d'apprentissage automatique, dont le but est d'apprendre, à partir d'expériences, ce qu'il convient de faire en différentes situations, de façon à optimiser une récompense numérique au cours du temps (Sutton & Barto, 1998).

Run S'utilise dans le domaine de l'apprentissage par renforcement. Séquence d'épisodes visant à reproduire l'apprentissage d'un système au cours du temps.

SARSA State-Action-Reward-State-Action (Rummery & Niranjan, 1994). Algorithme d'apprentissage par renforcement reposant sur l'hypothèse markovienne et le bootstrapping. Diffère du Q-Learning par le fait qu'il est on-policy. Cas particulier du TD-Learning.

Scheduler Composant servant à contrôler les ressources du système de dialogue et à choisir ses actions en fonction de ces ressources. Voir la sous-section 1.3.3 pour plus de détails.

SD Système de dialogue. Système permettant d'automatiser l'accomplissement d'une tâche à l'aide d'un dialogue avec l'utilisateur.

SLU Spoken Language Understanding ou module d'interprétation. Module servant à transformer une phrase en concepts système interprétable par le DM. Voir la sous-section 1.2.1 pour plus de détails.

ST System Turn ou tour système. Temps séparant deux actes externes du système. Voir la sous-section 1.2.2 pour plus de détails.

Subsomption La subsomption désigne une relation hiérarchique entre des concepts, dans les logiques de description. Cette notion est proche de la relation "est impliqué par" en logique classique, ou encore "contient" en logique ensembliste.

Support de démonstration Ensemble des mondes où la démonstration est à la fois fiable et applicable. Voir la sous-section 2.3.1 pour plus de détails.

Tautologie Phrase ou effet de style ainsi tourné que sa formulation ne puisse être que vraie. Peut aussi s'apparenter au truisme ou à une lapalissade. En logique, ce mot désigne une proposition toujours vraie.

TCA Time Consuming Act. Acte dont l'exécution implique une dépense de ressources temporelles. Voir la sous-section 1.2.2 pour plus de détails.

TD-Learning Temporal Difference Learning (Houk et al., 1995). Famille d'algorithmes d'apprentissage par renforcement reposant sur l'hypothèse markovienne et le bootstrapping . Les algorithmes SARSA et Q-Learning sont des implémentations du TD-Learning.

TTS Text-To-Speech ou synthèse vocale. Module servant à transformer une phrase en signal sonore. Voir la sous-section 1.2.1 pour plus de détails.

UCB Upper Confidence Bound. Méthode pour la gestion de l'équilibre entre l'exploration et l'exploitation reposant sur une estimation de l'espérance, la variance et le volume de données. Voir la sous-section 5.4.1 pour plus de détails.

UCB-tuned Upper Confidence Bound Tuned. Méthode pour la gestion de l'équilibre entre l'exploration et l'exploitation dérivée d'UCB reposant sur un calcul de l'espérance, la variance, la précision de la variance et le volume de données. Voir la sous-section 5.4.2 pour plus de détails.

VoiceXML Voice eXtensible Markup Language ou langage de balisage extensible vocal. Langage normalisé de programmation d'une application vocale.

VUI Voice User Interface. Interface pour les applications de dialogue. Permet de contrôler une machine en lui parlant.

VUI-Completeness Complétude VUI. Le comportement d'une application de dialogue se doit d'être complètement spécifié selon chaque situation qui puisse advenir durant l'interaction. Aucun énoncé de l'utilisateur, aussi imprédictible soit-il, ne doit jamais pousser le système dans un comportement inattendu. Voir la section 3.2 pour plus de détails.

WBOA Write Back-Office Act. Sous-catégorie des BOA correspondant aux actes d'écriture dans la base de données. Voir la sous-section 1.2.2 pour plus de détails.

XML Extensible Markup Language ou langage extensible de balisage. Langage informatique de balisage générique. Il sert essentiellement à stocker/transférer des données de type texte Unicode structurées en champs arborescents. Ce langage est qualifié

d'extensible car il permet à l'utilisateur de définir les balises des éléments. L'utilisateur peut multiplier les espaces de nommage des balises et emprunter les définitions d'autres utilisateurs.

Bibliographie

Abella, A., Wright, J., & Gorin, A. (2004). Dialog trajectory analysis. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 441–444).

Agrawal, R. (1995). Sample mean based index policies with o(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, *27*, 1054–1078.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*, 235–256.

Barto, A., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*, 341–379.

Baudoin, F. (2008). *Personnalisation des systèmes de dialogue en langage naturel : une méthode d'anticipation rationnelle d'actions communicatives.* Thèse de doctorat, LIP6.

Bellman, R. (1957a). *Dynamic programming.* Princetown University Press.

Bellman, R. (1957b). A markovian decision process. *Journal of Mathematics and Mechanics*, *6*, 679–684.

Bohlin, P., Cooper, R., Engdahl, E., & Larsson, S. (1999). Information states and dialogue move engines. *IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems.*

Boidin, C., Boeffard, O., Moudenc, T., & Damnati, G. (2009a). Towards Intonation Control in Unit Selection Speech Synthesis. *Proceedings of Interspeech.* Brighton (United Kingdom).

Boidin, C., Rieser, V., van der Plas, L., Lemon, O., & Chevelu, J. (2009b). Predicting how it sounds : Re-ranking dialogue prompts based on TTS quality for adaptive Spoken Dialogue Systems. *Proceedings of the Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue.* Brighton (United Kingdom).

Bos, J., Klein, E., Lemon, O., & Oka, T. (2003). Dipper : Description and formalisation of an information-state update dialogue system architecture.

Bouchon-Meunier (1995). *Logique floue et applications.* Addison Wesley.

Bouchon-Meunier, B. (2007). *La logique floue.* Que sais-je ? Presses universitaires de France. Quatrième édition.

Bouchon-Meunier, B., & Nguyen, H. (1996). *Les incertitudes dans les systèmes intelligents.* Que sais-je ? Presses universitaires de France.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. *In TARK* (pp. 195–210). Morgan Kaufmann.

Boutilier, C., & Brafman, R. (1997). Planning with concurrent interacting actions. *In Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97* (pp. 720–726).

Boutilier, C., Dearden, R., & Goldszmidt, M. (1995). Exploiting structure in policy construction. *in Proceedings of IJCAI-95* (pp. 1104–1111).

Bratman, M. (1987). *Intentions, plans, and practical reason.* Harvard University Press.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Monterey, CA : Wadsworth and Brooks.

Bretier, P. (1995). *La communication orale coopérative : contribution à la modélisation logique et à la mise en œuvre d'un agent rationnel dialoguant.* Thèse de doctorat, Université Paris Nord, France.

Bretier, P., Crook, P., Keizer, S., Laroche, R., Lemon, O., & G., P. (2009). *Initial evaluation of towninfo and self-help systems*Report D6.3). CLASSIC Project.

Cadic, D., Boidin, C., & d'Alessandro, C. (2009). Vocalic sandwich, a unit designed for unit selection TTS. *Proceedings of Interspeech.* Brighton (United Kingdom).

Cadic, D., & Segalen, L. (2008). Paralinguistic elements in speech synthesis. *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'08)* (pp. 1861–1864).

Choquet, G. (1953). Théorie des capacités. *Annales de l'institut Fourier*, *5*, 131–295.

Constantinides, P., Hansma, S., Tchou, C., A.I., R., & Rudnicky, E. (1998). A schema based approach to dialog control. *Proceedings of the International Conference on Spoken Language Processing* (pp. 409–412).

Crook, P., & Lemon, O. (2009). Accurate probability estimation of hypothesised user acts for POMDP approaches to dialogue management. *Proceedings of the 12th Annual Research Colloquium of the special-interest group for computational linguistics in the UK and Ireland (CLUKI 2009).*

Cuayáhuitl, H., Renals, S., Lemon, O., & Shimodaira, H. (2006). Reinforcement learning of dialogue strategies with hierarchical abstract machines. *Proceedings of IEEE/ACL Workshop on Spoken Language Technology (SLT).*

Cuayáhuitl, H., Renals, S., Lemon, O., & Shimodaira, H. (2007). Hierarchical dialogue optimization using semi-markov decision processes. *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*.

de Kleer, J. (1986a). An assumption-based TMS. *Artificial Intelligence*, *28*, 127–162.

de Kleer, J. (1986b). Extending the ATMS. *Artificial Intelligence*, *28*, 163–196.

de Kleer, J. (1986c). Problem solving with the ATMS. *Artificial Intelligence*, *28*, 197–224.

Dean, T., & Givan, R. (1997). Model minimization in markov decision processes. *In Proceedings of the Fourteenth National Conference on Artificial Intelligence* (pp. 106–111). AAAI.

Dempster, A. (1968). A generalization of bayesian inference. *Journal of the Royal Statistical Society*.

Dietterich, T. (1998). The MAXQ method for hierarchical reinforcement learning. *In Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 118–126). Morgan Kaufmann.

Dung, P. (1993). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming. *IJCAI* (pp. 852–859).

Eide, E., Aaron, A., Bakis, R., Hamza, W., Picheny, M., & Pitrelli, J. (2004). A corpus-based approach to "ahem" expressive speech synthesis. *in 5th ISCA Workshop on Speech Synthesis* (pp. 79–84).

Ferrans, J., Danielsen, P., Hunt, A., Porter, B., Lucas, B., Mcglashan, S., Tryphonas, S., Rehor, K., Burnett, D., & Carter, J. (2004). *Voice extensible markup language (voicexml) version 2.0* (Technical Report). W3C.

Fishman, G. (1996). *Monte Carlo : Concepts, algorithms, and applications*. Springer Series in Operations Research. New York : Springer-Verlag.

Furui, S. (1997). Recent advances in speaker recognition (invited paper). *AVBPA '97 : Proceedings of the First International Conference on Audio- and Video-Based Biometric Person Authentication* (pp. 237–252). London, UK : Springer-Verlag.

Gabsdil, M., & Lemon, O. (2004). Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. *ACL '04 : Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (p. 343). Morristown, NJ, USA : Association for Computational Linguistics.

Garcia, F. (1993). *Révision des croyances et révision du raisonnement pour la planification*. Thèse de doctorat, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse (France).

Garcia, F., & Serre, F. (2000). Average td(λ) : une version asymptotique de td(λ) pour une analyse de l'effet du paramètre λ. *Proceedings of RFIA*. Paris (France).

Gašić, M., Lefèvre, F., Jurčíček, F., Keizer, S., Mairesse, F., Thomson, B., Yu, K., & Young, S. (2009). Back-off action selection in summary space-based POMDP dialogue systems. *Proceedings of ASRU*. Merano (Italy).

Geist, M., Pietquin, O., & Fricout, G. (2009). Tracking in reinforcement learning. *Proceedings of the 16th International Conference on Neural Information Processing (ICONIP 2009)* (pp. 502–511). Bangkok (Thailand) : Springer LNCS. ENNS best student paper award.

Georgila, K., Henderson, J., & Lemon, O. (2005). Learning user simulations for information state update dialogue systems. *Proceedings of Eurospeech* (pp. 893–896). Lisbon (Portugal).

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning : Theory and practice*. Amsterdam : Morgan Kaufmann.

Governatori, G., & Maher, M. (2000). An argumentation-theoretic characterization of defeasible logic. *ECAI* (pp. 469–473). IOS Press.

Halpern, J. (2005). *Reasoning about uncertainty*. The MIT Press.

Henderson, J., & Lemon, O. (2008). Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management. *Proceedings of ACL*.

Henderson, J., Lemon, O., & Georgila, K. (2005). Hybrid reinforcement / supervised learning for dialogue policies from communicator data.

Hintikka, J. (1962). *Knowledge and belief : An introduction to the logic of the two notions*. Ithaca, N. Y. : Cornell University Press.

Hirose, K., Asano, Y., & Minematsu, N. (2006). Corpus-based generation of fundamental frequency contours using generation process model and considering emotional focuses. *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'06)*.

Houk, J. C., Adams, J. L., & Barto, A. G. (1995). A model of how the basal ganglia generate and use neural signals that predict reinforcement. In J. C. Houk, J. L. Davis and D. G. Beiser (Eds.), *Models of information processing in the basal ganglia*, 233–248. Cambridge, MA : MIT Press.

Huet, S., Gravier, G., & Sébillot, P. (2007). Morphosyntactic processing of n-best lists for improved recognition and confidence mesure computation. *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)* (pp. 2685–2688).

Janarthanam, S., & Lemon, O. (2009). A Wizard-of-Oz environment to study Referring Expression Generation in a Situated Spoken Dialogue Task. *Proceedings of ENLG*.

Jonson, R. (2006). Dialogue context-based re-ranking of asr hypotheses. *Spoken Language Technology Workshop, 2006. IEEE* (pp. 174–177).

Kambhampati, S. (1997). Refinement planning as a unifying framework for plan synthesis. *Artificial Intelligence*.

Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. (2009). Controlled experiments on the web : survey and practical guide. *Data Mining and Knowledge Discovery*.

Kohlas, J., & Monney, P. (1995). *A mathematical theory of hints. an approach to the dempster-shafer theory of evidence*, vol. 425 of *Lecture Notes in Economics and Mathematical Systems*. Springer.

Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, *6*, 4–22.

Laroche, R., Bouchon-Meunier, B., & Bretier, P. (2008). Uncertainty management in dialogue systems. *Proceedings of the European Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*.

Laroche, R., Putois, G., Bretier, P., & Bouchon-Meunier, B. (2009a). Hybridisation of expertise and reinforcement learning in dialogue systems. *Proceedings of Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue*. Brighton (United Knigdom).

Laroche, R., Putois, G., Bretier, P., & Young, S. (2009b). *Adaptive automata-based dm component, first prototype.*Prototype D1.1.2). CLASSIC Project.

Laroche, R., Putois, G., Young, S., Henderson, J., Lemon, O., Rieser, V., Liu, X., & Bretier, P. (2010). *Final communication architecture and module interface definitions*Report D5.1.2). CLASSIC Project.

Laroche, R., Young, S., Lemon, O., Putois, G., & Bretier, P. (2009c). *Requirements analysis and theory for statistical learning approaches in automaton-based dialogue management.*Report D1.1.1). CLASSIC Project.

Larsson, S., & Traum, D. (2000). Information state and dialogue management in the trindi dialogue move engine toolkit. *Nat. Lang. Eng.*, *6*, 323–340.

Lecœuche, R. (2001). Learning optimal dialogue management rules by using reinforcement learning and inductive logic programming. *NAACL*.

Lefèvre, F., Gašić, M., Jurčíček, F., Keizer, S., Mairesse, F., Thomson, B., Yu, K., & Young, S. (2009). k-nearest neighbor monte-carlo control algorithm for POMDP-based dialogue systems. *Proceedings of SIGDIAL*. London (United Kingdom).

Lemon, O., Georgila, K., & Henderson, K. (2006). Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users : the talk towninfo evaluation. *IEEE/ACL Spoken Language Technology*.

Lemon, O., & Pietquin, O. (2007). Machine learning for spoken dialogue systems. *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)* (pp. 2685–2688).

Levin, E., Pieraccini, R., & Eckert, W. (1998). Using markov decision process for learning dialogue strategies. *Proceedings of ICASSP1998*.

Levin, E., Pieraccini, R., & Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*.

Lewis, C. I., & Langford, C. H. (1932). *Symbolic logic*. New York and London : The Century Co.

Li, L., Williams, J., & Balakrishnan, S. (2009). Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. *Proceedings of Interspeech. Special Session : Machine Learning for Adaptivity in Spoken Dialogue*. Brighton (United Knigdom).

Li, X., & Huerta, J. (2007). How predicatable is asr confidence in dialogue applications ? *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)* (pp. 2685–2688).

Litman, D., Hirschberg, J., & Swerts, M. (2000). Predicting automatic speech recognition performance using prosodic cues. *In Proceedings of NAACL-00* (pp. 218–225).

Louis, V. (2002). *Conception et mise en œuvre de modèles formels de calcul de plans d'action complexes par un agent rationnel dialoguant*. Thèse de doctorat, Université de Caen, Caen (France).

Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed markov decision processes. *Oper. Res.*, *39*, 162–175.

Mallat Desmortiers, D. (2003). *Raisonnement automatique sur les croyances et les incertitudes d'un agent formalisé au sein de la théorie de l'interaction rationnelle*. Thèse de doctorat, LIPN.

McTear, M. F. (2004). *Spoken dialogue technology : Toward the conversational user interface*. Springer.

Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, *44*, 335–341.

Minematsu, N., Sekiguchi, M., & Hirose, K. (2002). Automatic estimation of one's age with his/her speech based upon acoustic modeling techniques of speakers. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on* (pp. I–137–I–140 vol.1).

Nilsson, N. (1986). Probabilistic logic. *Artificial Intelligence*, *28*, 71–87.

Olds, J., & Milner, P. (1954). Positive reinforcement produced by electrical stimulation of septal area and other areas of the brain. *Journal of Comparative & Physiological Psychology*, *47*, 419–427.

Paek, T. (2006). Reinforcement learning for spoken dialogue systems : Comparing strengths and weaknesses for practical deployment.

Paek, T. (2007). Toward evaluation that leads to best practices : Reconciling dialog evaluation in research and industry. *Proceedings of the Workshop on Bridging the Gap : Academic and Industrial Research in Dialog Technologies* (pp. 40–47). Rochester, NY : Association for Computational Linguistics.

Paek, T., & Pieraccini, R. (2008). Automating spoken dialogue management design using machine learning : An industry perspective. *Speech Communication*, *50*, 716–729.

Parr, R., & Russell, S. (1997). Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems*. The MIT Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems : Networks of plausible inference*. Morgan Kaufmann Publishers.

Pickering, M., & Garrod, S. (2004). Towards a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, *27*, 169–190.

Pieraccini, R., Caskey, S., Dayanidhi, K., Carpenter, B., & Phillips, M. (2001). Etude, a recursive dialog manager with embedded user interface patterns. *Automatic Speech Recognition and Understanding, 2001 IEEE Workshop on* (pp. 244–247).

Pieraccini, R., & Huerta, J. (2005). Where do we go from here ? research and commercial spoken dialog systems. *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue* (pp. 1–10).

Pieraccini, R., Levin, E., & Eckert, W. (1997). Amica : the att mixed initiative conversational architecture. *In Proceedings of Eurospeech* (pp. 1875–1878).

Pietquin, O. (2005). A probabilistic description of man-machine spoken communication. *Proceedings of the 5th IEEE International Conference on Multimedia and Expo (ICME 2005)* (pp. 410–413). Amsterdam (The Netherlands).

Pollock, J. (1987). Defeasible reasoning. *Cognitive Science*, *11*, 481–518.

Putois, G., Young, S., Henderson, J., Lemon, O., Rieser, V., Liu, X., Bretier, P., & Laroche, R. (2009). *Initial communication architecture and module interface definitions*Report D5.1.1). CLASSIC Project.

Rieser, V., & Lemon, O. (2006). Using machine learning to explore human multimodal clarification strategies. *Proceedings of the COLING/ACL on Main conference poster sessions* (pp. 659–666). Morristown, NJ, USA : Association for Computational Linguistics.

Rieser, V., & Lemon, O. (2008a). Automatic learning and evaluation of user-centered objective functions for dialogue system optimisation. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*. Marrakech (Morocco) : European Language Resources Association (ELRA). http ://www.lrec-conf.org/proceedings/lrec2008/.

Rieser, V., & Lemon, O. (2008b). Does this list contain what you were searching for ? Learning adaptive dialogue strategies for interactive question answering. *Natural Language Engineering (special issue on Interactive Question Answering)*, *15*, 55–72.

Rieser, V., & Lemon, O. (2008c). Learning effective multimodal dialogue strategies from wizard-of-oz data : Bootstrapping and evaluation. *Proceedings of ACL*. Columbus (Ohio, USA).

Rieser, V., & Lemon, O. (2008d). Simulation-based learning of optimal multimodal presentation strategies from Wizard-of-Oz data. *Proceedings of the AISB Symp. on Multimodal Output Generation (MOG)*. Aberdeen (Scotland.

Rieser, V., & Lemon, O. (2009a). Learning human multimodal dialogue strategies. *Natural Language Engineering*, *16*, 3–23.

Rieser, V., & Lemon, O. (2009b). Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. *Proceedings of EACL*.

Rohanimanesh, K., & Mahadevan, S. (2001). Decision-Theoretic planning with concurrent temporally extended actions. *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence* (pp. 472–479).

Rohanimanesh, K., & Mahadevan, S. (2003). Learning to take concurrent actions. *In Proceedings of the Sixteenth Annual Conference on Neural Information Processing Systems* (pp. 1619–1626). MIT Press.

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. *ACL '00 : Proceedings of the 38th Annual Meeting on Association for Computational Linguistics* (pp. 93–100). Morristown, NJ, USA : Association for Computational Linguistics.

Rummery, G., & Niranjan, M. (1994). *On-line q-learning using connectionist systems* (Technical Report). unknown.

Sadek, D. M. (1991). *Attitudes mentales et interaction rationnelle : vers une théorie formelle de la communication.* Thèse de doctorat, Université de Rennes I, France.

Sadek, M. D., Bretier, P., & Panaget, F. (1997). Artimis : Natural dialogue meets rational agency. *in Proceedings of IJCAI-97* (pp. 1030–1035). Morgan Kaufmann.

Searle, J. R. (1969). *Speech acts : An essay in the philosophy of language.* Cambridge, London : Cambridge University Press.

Shafer, G. (1976). *A mathematical theory of evidence.* Princetown University Press.

Shafran, I., Riley, M., & Mohri, M. (2003). Voice signatures. *Automatic Speech Recognition and Understanding, 2003 IEEE Workshop on.* http ://www.scientificcommons.org/42223516.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, *27*.

Shriver, S., Toth, A., Zhu, X., Rudnicky, A., & Rosenfeld, R. (2001). A unified design for human-machine voice interaction. *in Extended Abstracts of CHI* (pp. 247–248).

Si, J., Barto, A., Powell, W., & Wunsch, D. (2004). *Handbook of learning and approximate dynamic programming (ieee press series on computational intelligence)*, chapter 2, 47–64. Wiley-IEEE Press.

Sigmund, M., & Dostál, T. (2005). Automatic gender distinction by voice. *Artificial Intelligence and Applications* (pp. 633–636).

Singh, S., Kearns, M., Litman, D., & Walker, M. (1999). Reinforcement learning for spoken dialogue systems.

Singh, S., Litman, D., Kearns, M., & Walker, M. (2002). Optimizing Dialogue Management with Reinforcement Learning : Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, *16*, 105–133.

Smets, P. (1990). The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 447–458.

Smets, P. (1992). Resolving misunderstandings about belief functions. *Int. J. Approx. Reasoning*, *6*, 321–344.

Sondik, E. J. (1971). *The optimal control of partially observable markov decision processes.* Thèse de doctorat, Stanford, California.

Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning* (pp. 9–44).

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning : An introduction (adaptive computation and machine learning).* The MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps : a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Thomson, B., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Yu, K., & Young, S. (2008a). User study of the Bayesian Update of Dialogue State approach to dialogue management. *Proceedings of Interspeech*. Brisbane (Australia).

Thomson, B., Yu, K., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., & Young, S. (2008b). Evaluating semantic-level confidence scores with multiple hypotheses. *Proceedings of Interspeech*. Brisbane (Australia).

Tran, D., & Sharma, D. (2003). Automatic gender recognition. *ICECS'03 : Proceedings of the 2nd WSEAS International Conference on Electronics, Control and Signal Processing* (pp. 1–5). Stevens Point, Wisconsin, USA : World Scientific and Engineering Academy and Society (WSEAS).

von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton University Press.

Walker, M., Kamm, C., & Litman, D. (2000). Towards developing general models of usability with paradise.

Walker, M., Litman, D., Kamm, C., & Abella, A. (1997). PARADISE : a framework for evaluating spoken dialogue agents. *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics* (pp. 271–280). Morristown, NJ, USA : Association for Computational Linguistics.

Watkins, C. (1989). *Learning from delayed rewards*. Thèse de doctorat, Cambridge University, Cambridge (England).

Williams, J., & Balakrishnan, S. (2009). Estimating probability of correctness for asr n-best lists. *Proceedings of the 10th SIGdial Conference on Discourse and Dialogue*.

Williams, J. D. (2008). The best of both worlds : Unifying conventional dialog systems and POMDPs. *International Conference on Speech and Language Processing*.

Williams, J. D., & Young, S. (2005). Scaling up POMDPs for dialog management : The summary POMDP method. *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on* (pp. 177–182).

Williams, J. D., & Young, S. (2007). Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, *21*, 393–422.

Wright, J., Kapilow, D., & Abella, A. (2005). Interactive visualization of human-machine dialogs. *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)* (pp. 2517–2520). Lisbon (Portugal).

Young, S. (2006). Using POMDPs for dialog management. *Spoken Language Technology Workshop, IEEE*.

Young, S., Schatzmann, J., Weilhammer, K., & Ye, H. (2005). The hidden information state approach to dialogue management. *In Proceedings of ICASSP*.

Young, S. J. (2000). Probabilistic methods in spoken-dialogue systems. *Philosophical Transactions of the Royal Society* (pp. 1389–1402).

Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 3–28.

Zurif, E. (1995). *Brain regions of relevance to syntactic processing*, chapter 13. The MIT Press.

Annexe A

Relation d'indépendance entre des générations de plan pour le dialogue

A.1 Le Problème

Supposons que la partie de consommation (voir section 1.2.2) ait défini deux interprétations exclusives de ce que l'utilisateur vient de dire :

– "italian restaurant" est la requête avec une probabilité p

– "indian restaurant" est la requête avec une probabilité $1 - p$

Par souci de simplicité, nous considérons qu'il n'y a que trois actes différents possibles :

1. feedback sur italian restaurant + question sur la localisation
2. feedback sur indian restaurant + question sur la localisation
3. question de désambiguïsation ("did you mean italian or indian ?")

Il a été démontré que l'utilisation de question de désambiguïsation est utile lorsqu'il y a beaucoup de bruits, pour restreindre les possibilités offertes à l'utilisateur. Cette étude montre que la question de désambiguïsation ne peut être planifiée si les plans de dialogue sont exclusifs, c'est-à-dire si l'on considère qu'il y a exactement un plan de dialogue pertinent dans un état donné.

A.2 Plans exclusifs

Considérer que les plans sont exclusifs revient à chercher le meilleur plan dans chacun des contextes possibles. Dans le cas que nous traitons, la définition de meilleur plan est une tâche simple :

– "italian restaurant" est la requête utilisateur $\Rightarrow$ le plan (1) est le meilleur

– "indian restaurant" est la requête utilisateur $\Rightarrow$ le plan (2) est le meilleur

En conclusion, le plan de désambiguïsation (3) ne pourra jamais être choisi.

A.3 Plans indépendents

Considérer que les plans sont indépendants revient à chercher les plans qui sont pertinents dans chacun des contextes possibles. Il se peut donc que plusieurs plans π_1 et π_2 soient pertinents dans un même état de dialogue s. Dans le cas que nous traitons, la définition des plans pertinents peut être réalisé de la manière suivante :

- "italian restaurant" est la requête utilisateur $\Rightarrow$ le plan (1) est pertinent (règle certaine)
- "italian restaurant" est la requête utilisateur $\Rightarrow$ le plan (3) est pertinent (règle incertaine de probabilité q)
- "indian restaurant" est la requête utilisateur $\Rightarrow$ le plan (2) est pertinent (règle certaine)
- "indian restaurant" est la requête utilisateur $\Rightarrow$ le plan (3) est pertinent (règle incertaine de probabilité q)

En conclusion, si $p > 0.5$ et $p > q$, le plan (1) sera choisi, si $p < 0.5$ et $1 - p > q$, le plan (2) sera choisi et finalement si $q > p$ et $q > 1 - p$, alors le plan (3) sera choisi.

A.4 Conclusion

Ce besoin de conserver des plans indépendants est encore renforcé par le NLG et le TTS. Par exemple, supposons que si l'utilisateur cherche un restaurant indien, le NLG ne propose qu'une paraphrase : "You are looking for an indian restaurant. Where do you want to eat ?" mais que s'il cherche un restaurant italien, le NLG en propose deux : "You are looking for an italian restaurant. Where do you want to eat ?" et "You are looking for a pizzeria. Where do you want to eat ?". Le fait de considérer les deux plans pour le restaurant italien comme exclusifs, fait que le chemin du restaurant italien est pénalisé par le fait d'avoir deux alternatives.

Annexe B

API des contrôleurs

B.1 API du Log Manager

L'API du Log Manager peut être complexe parce que ce module requiert de trouver un formalisme universel pour décrire les états internes des modules. Ceci ne peut clairement pas être atteint. A minima, il doit être capable de rassembler tous les échanges entre les modules et la manière dont ils ont été initialement paramétrés. Comme les échanges inter-modules sont normalisés, cet objectif minimal devrait pouvoir être atteint. Nous considérons donc chacun des modules comme une boîte noire qui déclare sa paramétrisation en début de dialogue. Puis le système se charge de mémoriser dans le Log Manager tous les messages entre ces modules. De cette manière, les dialogues peuvent être entièrement reconstitués. Il y a donc deux méthodes à l'API du Log Manager :

- `public void declare(moduleID, parametersValues)`
- `public void log(string)`

B.2 API du Context Manager

L'API du Context Manager tel qu'implémenté dans le cadre formel Logical Framework for Probabilistic Reasoning est relativement simple.

- `public int add_hypothesis(proposition, rating, dependency)` : ajoute une nouvelle hypothèse probabilisée à la base de connaissances incertaines et retourne l'identifiant de la nouvelle connaissance.
- `public int add_knowledge(proposition, dependency)` : ajoute à la base de connaissances incertaines une nouvelle déduction dépendant d'autres connaissances incertaines et retourne l'identifiant de la nouvelle connaissance.
- `public double confidence(proposition)` : calcule le degré de confiance concer-

nant la proposition en argument.
- `public void delete_knowledge(Id)` : retire une connaissance de la base de connaissances incertaines, ainsi que les déductions qui ont suivi.

Pour faciliter l'utilisation des fonctionnalités décrites dans l'exemple d'utilisation du Context Manager dans l'annexe C, nous y avons ajouté une API avancée :

- `public double confidence(proposition, dependency)` : calcule le degré de confiance concernant la proposition en argument dans le contexte décrit par la dépendance en argument. Il s'agit du calcul de $p(a|b) = \frac{p(ab)}{p(b)}$ où a est la proposition et b le contexte exprimée par l'argument dependency.
- `public Vector<Vector<Object» distribution(proposition, dependency)` : la proposition en argument contient au moins une variable universelle et le Context Manager fournit la liste des propositions de la base de connaissances s'unifiant avec celle-ci ainsi que leur probabilité.
- `public Vector<Integer> add_independant_hypothese_set(proposition/ probability list, dependency)` : ajoute n nouvelles hypothèses qui sont toutes indépendantes (c'est la fonction utilisée quand le système souhaite ajouter plusieurs connaissances dont les dépendances sont inconnues). La méthode retourne la liste des identifiants des nouvelles connaissances.
- `public Vector<Integer> add_exclusive_hypothese_set(proposition/ probability list, dependency)` : ajoute n nouvelles hypothèses qui sont toutes mutuellement exclusives (c'est la fonction utilisée quand le système souhaite ajouter plusieurs connaissances qu'il sait être incompatibles). La méthode retourne la liste des identifiants des nouvelles connaissances.
- `public Vector<Integer> add_knowledge_set(proposition/dependency list)` : ajoute n nouvelles hypothèses dont les dépendances sont décrites dans la dépendency list. La méthode retourne la liste des identifiants des nouvelles connaissances.
- `public void knowledge_to_hypothese(identifiant)` : le système retire tous les ancêtres de la connaissance de l'identifiant en argument et il remplace cette connaissance par une hypothèse de telle manière que le degré de croyance reste inchangé.
- `public void clean_knowledge_base()` : le système retire toutes les connaissance obsolètes de la base de connaissance. C'est une fonction a priori dépendante de l'application.
- `public void reset_knowledge_base()` : retire toutes les connaissances de la base.

B.3 API du Scheduler

En ce qui concerne l'API du Scheduler, elle n'a pas besoin d'être définie puisque le Scheduler est le processus principal du système.

Annexe C

Exemple d'utilisation d'un Contexte Manager basé sur la connaissance

Nous utilisons dans cette annexe un Context Manager dont les scores de confiance sont des probabilités.

La section B fournit la définition de l'API utilisée lors de cet exemple. La figure C.1 permet de suivre plus facilement les échanges entre les divers modules. De manière à simplifier l'exemple, les échanges avec le Scheduler sont ignorés. On considère que tous les calculs dans chacun des modules sont réalisés, sans prendre en compte les contraintes temps réel. Celles-ci sont traitées dans l'annexe D.

Un appel est reçu. Le Context Manager (CM) est remis à zéro à l'aide de la méthode `reset_knowledge_base()`. Le gestionnaire de dialogue (DM) choisit l'acte dialogue `welcome_user_action` que le générateur (NLG) va transformer en phrase, phrase que la synthèse (TTS) transformera en signal sonore. Le Scheduler informe le CM quelle action vient d'être effectuée à l'aide de la méthode `add_hypothesis(reset_knowledge_base(), 1, null)`.

L'utilisateur répond au système. Le signal est reçu par le Scheduler, qui en informe le CM à l'aide la méthode `add_hypothesis(userInput(1, wavePath), 1, null)`. Le CM répond à cette méthode avec l'identifiant de cette nouvelle connaissance : `idInput1`. Ensuite, le Scheduler transmet le signal à la reconnaissance vocale (ASR) pour le début du traitement de la parole. L'ASR renvoie une N-best list probabilisée. La plupart des ASR ne garantissent pas que la somme des probabilités des alternatives vaille 1. L'annexe 2.6 décrit la façon dont la normalisation est effectuée. Au final, l'ASR regroupe plusieurs traitements du signal sonores dont les probabilités de fiabilité sont exclusives (et donc dont la somme des probabilité est inférieure à 1) :

- `asrSentenceProcessing1, proba1` ;
- `asrSentenceProcessing2, proba2` ;

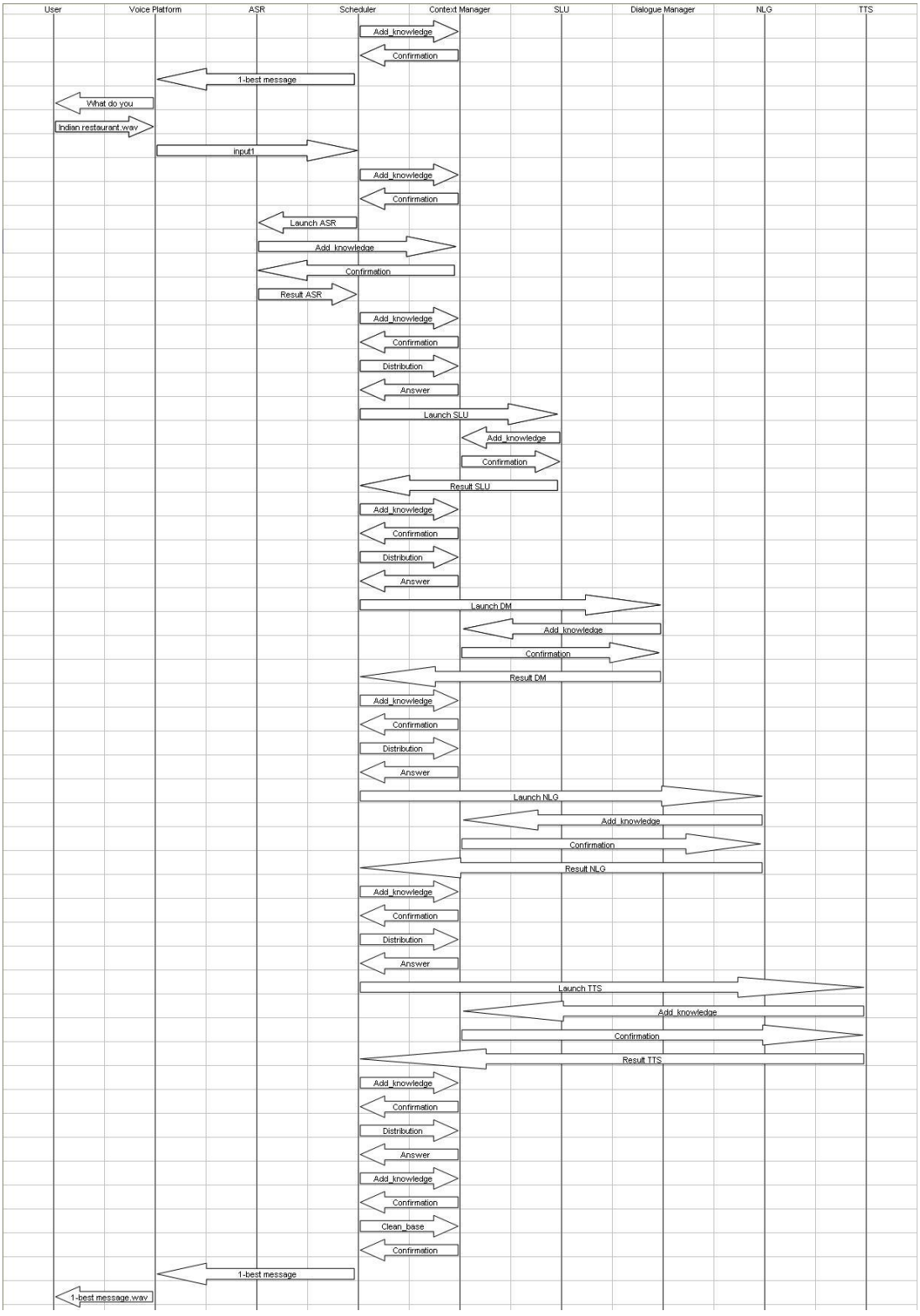


FIGURE C.1 – Les divers appels entre modules pendant la durée de l'exemple traité dans l'annexe C.

– asrSentenceProcessing3, proba3 ;

– asrSentenceProcessing4, proba4 ;

– ...

L'ASR peut ensuite informer le CM du résultat de la reconnaissance vocale à l'aide de la méthode suivante :

– add_exclusive_hypothese_set([asrSentenceProcessing#, proba#], idInput1)

L'ASR peut également ajouter des connaissances non verbales telles que le sexe du locuteur :

– asrGenderProcessing5, proba5 ;

– asrGenderProcessing6, proba6 ;

L'ajout de ces connaissances dans le CM se feront de la manière suivante :

– add_exclusive_hypothese_set([asrGenderProcessing#, proba'#], idInput1)

Où proba# et proba'# réfèrent aux probabilités des reconnaissances asrSentenceProcessing# et asrGenderProcessing#, et où # prend les valeurs de 1 à n. Le fait que les hypothèses soit exclusives interdit à deux hypothèses de la même liste N-best d'être vraie simultanément.

Ensuite l'interpréteur (SLU) a accès aux résultats de reco :

– reco(''I want an indian restaurant''), asrSentenceProcessing1 & idInput1 ;

– reco(''I'd want an indian restaurant''), asrSentenceProcessing2 & idInput1 ;

– reco(''I want an italian restaurant''), asrSentenceProcessing3 & idInput1 ;

– reco(''I'd want an italian restaurant''), asrSentenceProcessing4 & idInput1 ;

– ...

– gender(user, female), asrGenderProcessing5 & idInput1 ;

– gender(user, male), asrGenderProcessing6 & idInput1 ;

Ensuite les éléments du N-best sont ajoutées grâce à la méthode suivante :

– add_knowledge_set([proposition#, dependency#])

Le CM associe donc les informations idReco# et idGender# aux connaissances qui ont permis de l'inférer, c'est-à-dire le fait d'avoir reçu le signal idInput1 et le traitement de l'ASR ayant mené à la conclusion correspondante : asrSentenceProcessing# ou asrGenderProcessing#. Le Scheduler a ensuite besoin de connaître la distribution des reconnaissances vocales pour savoir lesquels faire traiter au SLU :

– distribution(reco(?), null)

Le CM renvoit la liste suivante :

– reco(''I want an indian restaurant''), 0.27, idReco1 ;

– reco(''I'd want an indian restaurant''), 0.25, idReco2 ;

– reco(''I want an italian restaurant''), 0.23, idReco3 ;

– reco(''I'd want an italian restaurant''), 0.22, idReco4 ;

– ...

Les modules Scheduler et SLU considèrent que seuls les quatre premiers résultats ont une probabilité suffisamment grande pour qu'ils soient traités :

– `sluDAProcessing1, 0.96` ;
– `sluDAProcessing2, 0.02` ;
– `sluDAProcessing3, 0.97` ;
– `sluDAProcessing4, 0.99` ;
– `sluDAProcessing5, 0.98` ;
– `sluLanguageLevelProcessing1, 0.65` ;
– `sluLanguageLevelProcessing2, 0.25` ;
– `sluLanguageLevelProcessing3, 0.95` ;
– `sluLanguageLevelProcessing4, 0.65` ;
– `sluLanguageLevelProcessing5, 0.25` ;
– `sluLanguageLevelProcessing6, 0.95` ;

Où `sluDAProcessing1` et `sluDAProcessing2` sont les traitements exclusifs de l'interprétation de `idReco1`, où `sluLanguageLevelProcessing1` et `sluLanguageLevelProcessing2` sont les traitements exclusifs du niveau de langage de `idReco1`, où `sluDAProcessing3` est le traitement de l'interprétation de `idReco2`, où `sluLanguageLevelProcessing3` est le traitement du niveau de langage de `idReco2`, où `sluDAProcessing4` est le traitement de l'interprétation de `idReco3`, où `sluLanguageLevelProcessing4` et `sluLanguageLevelProcessing5` sont les traitements exclusifs du niveau de langage de `idReco3`, où `sluDAProcessing5` est le traitement de l'interprétation de `idReco4` et où `sluLanguageLevelProcessing6` est le traitement du niveau de langage de `idReco4`. Ces hypothèses sont ajoutées au CM. Et la distribution de connaissance dans le CM concernant les résultats du SLU est la suivante :

– inputDA([goal="restaurant", speciality="indian"]), sluDAProcessing1 & idReco1 ;
– inputDA([goal="restaurant", speciality="asian"]), sluDAProcessing2 & idReco1 ;
– inputDA([goal="restaurant", speciality="indian"]), sluDAProcessing3 & idReco2 ;
– inputDA([goal="restaurant", speciality="italian"]), sluDAProcessing4 & idReco3 ;
– inputDA([goal="restaurant", speciality="italian"]), sluDAProcessing5 & idReco4 ;
– languageLevel(user, low), sluLanguageLevelProcessing1 & idReco1 ;
– languageLevel(user, high), sluLanguageLevelProcessing2 & idReco1 ;
– languageLevel(user, high), sluLanguageLevelProcessing3 & idReco2 ;
– languageLevel(user, low), sluLanguageLevelProcessing4 & idReco3 ;
– languageLevel(user, high), sluLanguageLevelProcessing5 & idReco3 ;
– languageLevel(user, high), sluLanguageLevelProcessing6 & idReco4 ;

Lorsque le Scheduler demande au CM une distribution des actes de dialogue que l'utilisateur vient de formuler, le CM renvoie :

– inputDA(topic=‘‘restaurant’’ & foodtype=‘‘Indian’’), 0.5, idInputDA1 | idInputDA2;
– inputDA(topic=‘‘restaurant’’ & foodtype=‘‘Italian’’), 0.44, idInputDA3 | idInputDA4
– inputDA(topic=‘‘restaurant’’ & foodtype=‘‘Asian’’), 0.01, idInputDA5

Où | signifie le “ou” logique. Cette liste d'interprétations est envoyée au DM qui fait la consommation des actes reçus et qui acquiert ainsi la connaissance sur la requête de l'utilisateur et la transmet au CM :

– add_knowledge(topic=‘‘restaurant’’ & foodtype=‘‘Indian’’, idInputDA1 | idInputDA2)
– add_knowledge(topic=‘‘restaurant’’ & foodtype=‘‘Italian’’, idInputDA3 | idInputDA4)
– add_knowledge(topic=‘‘restaurant’’ & foodtype=‘‘Indian’’, idInputDA5);

Le DM génère cinq actes différents, chacun étant le fruit de stratégies indépendantes (et non exclusives, voir annexe A pour une explication plus complète) :

– dmProcessing1, 0.95;
– dmProcessing2, 0.95;
– dmProcessing3, 0.75;
– dmProcessing4, 0.60;
– dmProcessing5, 0.60;

Et la sortie du DM au Scheduler est la suivante :

– outputDA(explicit_feedback(foodtype=‘‘indian’’)), dmProcessing1 & inputDA1;
– outputDA(explicit_feedback(foodtype=‘‘italian’’)), dmProcessing2 & inputDA2;
– outputDA(ambiguity_question(foodtype=‘‘indian’’|‘‘italian’’)), dmProcessing3 & (inputDA1 | inputDA2);
– outputDA(implicit_feedback(foodtype=‘‘indian’’)+openQ), dmProcessing4 & inputDA1;
– outputDA(implicit_feedback(foodtype=‘‘italian’’)+openQ), dmProcessing5 & inputDA2;

La chose importante à comprendre est que les différents plans de dialogue sont en compétition les uns avec les autres, mais il se peut qu'il y ait certains contextes où plusieurs d'entre eux sont pertinents. Ce explique le fait que la somme des probabilités puisse dépasser 1.

Une fois passé le DM, la consommation est terminée et la production a déjà commencé (voir la section 1.2.2). Pendant l'étape de production, il n'y a plus de renforcement possible entre plusieurs alternatives (telles que idInputDA1 et idInputDA2 se sont renforcées en idOutputDA3), parce qu'il n'y a aucune possibilité que deux actes de dialogue différents se génèrent de la même façon. La suite fonctionne de la même manière que ce que l'on a vu jusqu'à maintenant. Le NLG génèrera possiblement plusieurs formulations pour chaque acte de dialogue et il notera ces différentes formulations selon la confiance qu'il porte à chacune. Le même genre de mécanique est valable pour le TTS.

A la fin du tour de dialogue, le Scheduler choisira le meilleur mouvement de dialogue en se basant principalement sur les probabilités de pertinence associées à chaque alternative qu'ils représentent. Ensuite, comme beaucoup de connaissances très spécifiques à ce tour de dialogue ont été ajoutées, il est indispensable de procéder à un nettoyage du CM, qui consiste à ne garder que les informations qui resteront pertinentes pour les tours de dialogue prochains. Les informations du type `reco(?)`, `inputDA(?)`, `outputDA(?)`, `sentence(?)`, `wav(?)` ne sont plus pertinentes, seules les connaissances obtenues par inférence sur leur base ont un sens pour le reste du dialogue. De même, il est nécessaire d'ajouter au CM la connaissance du mouvement de dialogue qui vient d'être réalisé.

Annexe D

Exemple de fonctionnement du Scheduler

Par souci de simplicité, les divers échanges entre le Scheduler et le Context Manager ne sont pas décrits dans cette annexe. L'annexe C traite le même exemple sous l'angle du Context Manager. Il est d'ailleurs recommandé de lire l'annexe C avant celle-ci. Pour faciliter la lecture, le système est considéré mono-tâche dans cette annexe, c'est-à-dire qu'il ne peut exécuter deux tâches simultanément. En effet, il est plus simple de suivre les exécutions lorsqu'elles sont toutes exécutées en série.

Nous faisons démarrer cet exemple au moment où l'utilisateur dit son premier énoncé. Le Scheduler demande à la reconnaissance vocale (ASR) de traîter le signal avec la commande suivante : `execute(ASR, input.wav)` puis l'ASR répond en insérant dans l'Output Set l'action consistant à faire l'analyse sémantique (SLU) du N-best suivant :

- `reco="I want an indian restaurant", 0.27;`
- `reco="I'd want an indian restaurant", 0.25;`
- `reco="I want an italian restaurant", 0.23;`
- `reco="I'd want an italian restaurant", 0.22;`
- `reco="I want an egyptian restaurant", 0.005;`
- `reco="I'd want an egyptian restaurant", 0.003;`
- ...

L'ajout du nouvel élément à l'Output Set a créé un évènement, de même que la fin de la tâche de l'ASR. Le Scheduler récupère donc une nouvelle tâche à exécuter et le système dispose de toutes ses ressources pour traîter la tâche. Le Scheduler choisit de ne traîter que les six premiers résultats de reconnaissance. Il laisse en attente les autres résultats dans l'Output Set. Ce choix de ne garder que les six meilleures alternatives est motivé par les contraintes temps réel du dialogue, qui font que le système de dialogue veut économiser sa puissance de calcul.

De même que le Scheduler, le SLU peut juger que les 5ème et 6ème résultats de reconnaissance sont trop improbables pour qu'il soit nécessaire de perdre du temps à les traîter. Le SLU est donc capable de se mettre en pause sur un traitement en indiquant au Scheduler quelle est la criticalité de ce traitement et le temps prévu nécessaire à ce traitement.

Ensuite le SLU termine sa tâche en ajoutant son N-best d'hypothèses d'interprétation à proposer au gestionnaire de dialogue (DM) à l'Output Set.

- `inter=(topic=‘‘restaurant’’ & foodtype=‘‘Indian’’), 0.97, 1st-ASR-Id`;
- `inter=(topic=‘‘restaurant’’ & foodtype=‘‘Indian’’), 0.96, 2nd-ASR-Id`;
- `inter=(topic=‘‘restaurant’’ & foodtype=‘‘Italian’’), 0.99, 3rd-ASR-Id`;
- `inter=(topic=‘‘restaurant’’ & foodtype=‘‘Italian’’), 0.98, 4th-ASR-Id`;
- interprétations peu probables

Le Scheduler met en pause les interprétations peu probables, puis il décide de faire traîter en priorité par le DM les deux meilleures hypothèses d'interprétation.

- `execute(DM, 2-best-inter)`

Où la liste des meilleures hypothèses est la suivante :

- `inter=(topic=‘‘restaurant’’ & foodtype=‘‘Indian’’), 0.5`;
- `inter=(topic=‘‘restaurant’’ & foodtype=‘‘Italian’’), 0.44`

Le DM génère cinq alternatives de stratégie différentes : la première propose de résoudre l'ambiguïté entre indien et italien, la seconde propose de faire le requête au back office (BO) sur les restaurants indiens, la troisième la même requête sur les restaurants italiens, la quatrième propose de demander une confirmation explicite que l'utilisateur a dit indien, et enfin la cinquième stratégie est la confirmation explicite sur italien. Les deuxième et troisième alternatives sont les ajouts suivants à l'Output Set :

- `add_output(no_commitment, current_status, productive, time_to_execute, 0.5)`;
- `add_output(no_commitment, current_status, productive, time_to_execute, 0.44)`

Les trois autres alternatives sont des actes de dialogues que la génération (NLG) et la synthèse (TTS) devront verbaliser :

- `DA=ambiguity_question(foodtype=‘‘indian’’|‘‘italian’’), 0.71`
- `DA=YesNo_question(foodtype=‘‘indian’’), 0.6`
- `DA=YesNo_question(foodtype=‘‘italian’’), 0.55`

A partir du moment où l'on passe en phase de génération, il y a plus vraiment la nécessité de gérer plusieurs alternatives en un seul flux. Il suffit alors de prendre la meilleure et de la générer du mieux possible. Ainsi, la stratégie de désambiguïsation, qui est la meilleure jusqu'ici sera traitée et au final sa probabilité d'être “bonne” sera probablement

supérieure aux alternatives (c'est-à-dire supérieure à 0.6), quoi qu'il arrive. Mais si celle-ci venait à descendre en dessous d'autres alternatives, il serait intéressant d'essayer ensuite de générer les alternatives suivantes.

De même, selon les contraintes de temps réel, le Scheduler peut souhaiter compléter les interprétations dans le SLU et ses prises en compte d'hypothèses dans le DM, qui sont restées en attente dans l'Output Set pendant tout le traitement.

Au final, le Scheduler décide d'exécuter l'acte de dialogue de demande de désambiguïsation. C'est un Committing Act (voir la section 1.2.2) qui entraîne les actions suivantes :

- Il stoppe toutes les tâches en cours d'exécution : dans notre exemple, cette partie peut être ignorée puisque le système est mono-tâche.
- Il vide son "Output Set" : les plans d'actions concurrents, les tâches d'interprétation, etc . . .sont toutes supprimées. Elles ne sont plus nécessaire puisque le système a déjà pris une décision qui l'engage pour le reste du dialogue.
- Il informe les divers modules de la chaîne de dialogue des différents Committing Acts qui sont en train d'être exécutés : chaque module reçoit donc le message avec ces informations. La plupart d'entre eux ne sont pas des machines à états et ils ignoreront ce message. Mais le DM est généralement une machine à état et il a besoin de savoir quel acte il a effectué pour savoir dans quel état de dialogue le système se trouve.
- Il nettoie le Context Manager, en supprimant les informations obsolètes et en synthétisant les informations à conserver, de manière à ne garder que le strict nécessaire. Voir l'annexe C, pour plus de précisions sur ce qui doit être conservé ou supprimé dans la base de connaissances incertaines.

Annexe E

Directives pour l'expérimentation laboratoire

Cette annexe décrit les directives adressées aux superviseurs pour l'expérimentation laboratoire de l'installation de la Livebox, de manière à réduire au maximum les variations entre les expériences.

E.1 Protocole de passage d'une expérimentation :

- Préparer la boîte de Livebox fermée dans la même disposition qu'à l'achat.
- Le superviseur demande au testeur de s'installer en face du téléphone, isolé des autres testeurs, et des sources de distractions (télévision, musique, etc...), avec à sa disposition la boîte de la Livebox.
- Le superviseur distribue la feuille avec les instructions au testeur.
- Le superviseur lit oralement le texte avec le testeur.
- Si le testeur dit qu'il a déjà entendu parler de ce système en particulier, le remercier d'avoir été franc et lui expliquer qu'on ne peut pas lui faire tester le système. Dans le cas contraire, lui demander s'il a des questions au sujet de l'expérimentation. Ne répondre à aucune question concernant le système, mais répondre si besoin est aux questions concernant la façon don't interagir avec le système (que signifie s'exprimer naturellement → parler comme vous parleriez avec un opérateur humain, que faire si on ne comprend pas ce que dit le système → ne pas paniquer, on évalue notre travail, pas le testeur, etc...). Ne pas répondre non plus aux questions portant sur l'expérimentation. Il ne faut pas que le testeur connaisse l'objectif du test.
- Distribuer le scénario n°1. Le superviseur lit le scénario à haute voix. Demander si le testeur a des questions concernant le scénario (ne pas répondre aux questions à

propos du systèmes ou de l'expérimentation).
- Faire passer le scénario n°1.
- Noter le numéro de session et le scénario emprunté.
- Le scénario n°1 se termine en l'un des 4 autres scénarii. Le scénario que l'on a joué doit alors être défaussé et on doit expliquer au testeur de ne plus se mettre dans la même situation (par exemple s'il a demandé l'aide d'un technicien, lui dire que dorénavant, il ne souhaitera plus faire appel à un technicien). Si aucun des scénarii n'a été terminé (si la personne raccroche d'elle-même, ou si le système bugge), alors ne défausser aucun des 4 scénarii.
- Faire exécuter les scénarii restant dans l'ordre de leurs numéros. A chaque fois, lire le scénario à haute voix et demander au testeur s'il a des questions. Noter les numéros de session et de scénario sur le questionnaire d'évaluation. Ne pas répéter les scénarii 2, 3, 4 et 5 en cas d'échec ou de bug. En revanche, si l'un des scénario 2, 3 ou 4 se termine en un autre scénario, noter dans le questionnaire que le scénario effectué a bien été celui qui a été fait lors du dialogue. Par exemple, si on dit au testeur d'appliquer le scénario n°1 mais que la question du technicien apparait avant et qu'il y a répond favorablement, alors cela correspond au scénario n°4 et l'on notera dans le questionnaire que le scénario qui vient d'être effectué est le n°4. On défausse alors le scénario n°4 et on demande au testeur de répéter le scénario n°2 en refusant l'option technicien.
- A la fin des 4 dialogues (ou 5 si le premier dialogue a abouti à un échec ou un bug), remercier le testeur et lui demander de remplir le questionnaire. Vous pouvez ensuite discuter librement du système et de l'expérimentation.
- Bien lui répéter de ne pas parler de cette expérience autour de lui pendant la durée de l'expérimentation, de manière à ne pas se priver de testeurs potentiels par la suite.

E.2 Choses à faire ou ne pas faire :

- Ne jamais intervenir lors d'un dialogue avec le système.
- Ne pas influencer le testeur avec notre ressenti de la qualité du système.
- Mettre court à tout type de communication durant un dialogue (ne pas donner de signal au testeur).
- Empêcher le testeur de raccrocher quand celui-ci a atteint le succès dans le dialogue.
- Ne pas empêcher le testeur de raccrocher quand, au contraire, il raccroche parce qu'il est excédé par le système.
- Noter le numéro de session.

– En aucun cas, il ne faut répéter un scénario une deuxième fois.
– N'exprimez aucune émotion. Soyez décontracté (mais sérieux, évitez de faire des blagues), sans montrer de lassitude ou d'excitation.

E.3 Feuilles distribuées aux testeurs :

– Instructions destinées au testeur : Vous allez participer à une expérimentation portant sur un service de dialogue automatique pour l'aide à la première installation de la Livebox Mini. Vous venez de recevoir votre Livebox et vous entreprenez de l'installer. Pour ce faire, vous avez le réflex d'appeler le service gratuit d'Orange automatique. Vous pouvez vous exprimer librement et naturellement. Question préliminaire : avez-vous déjà entendu parler de cette expérimentation ? A ce sujet, nous vous demanderons de ne pas non plus parler de l'expérience que vous allez avoir avec vos collègues ou vos amis, avant le 20 mai. Nous allons vous demander de vous placer dans des scénarii avant d'effectuer chaque appel. Avez-vous des questions ?
– Scénario n°1 : Placez-vous dans la situation où vous appelez un service pour demander de l'aide pour l'installation de votre Livebox. Essayez de fixer au maximum le contexte dans lequel vous vous trouveriez.
– Scénario n°2 : Vous avez un peu de temps libre à votre travail. Vous en profitez pour appeler le service d'installation à votre Livebox.
– Scénario n°3 : Vous êtes chez vous. Vous n'avez pas encore installé votre Livebox, mais vous avec tout de même une connexion internet via votre Iphone.
– Scénario n°4 : Vous êtes chez vous mais vous n'avez pas accès à internet pour l'instant. Vous ne voulez pas perdre de temps à installer vous-même votre Livebox et vous êtes favorable à la venue d'un technicien à votre domicile.
– Scénario n°5 : Vous êtes chez vous mais vous n'avez pas accès à internet pour l'instant. Vous ne souhaitez pas non plus d'aide de la part d'un technicien.
– Questionnaire : voir figure E.1.

- Vous avez réussi à installer facilement votre Livebox.

Pas du tout d'accord	Pas d'accord	Ni en désaccord ni d'accord	D'accord	Tout à fait d'accord

- Le système a bien compris ce que vous lui disiez.

Pas du tout d'accord	Pas d'accord	Ni en désaccord ni d'accord	D'accord	Tout à fait d'accord

- Le système vous a bien renseigné.

Pas du tout d'accord	Pas d'accord	Ni en désaccord ni d'accord	D'accord	Tout à fait d'accord

- Vous utiliseriez un système tel que celui-ci pour installer votre Livebox.

Pas du tout d'accord	Pas d'accord	Ni en désaccord ni d'accord	D'accord	Tout à fait d'accord

FIGURE E.1 – Questionnaire distribué à tous les testeurs pour leur faire évaluaer le système.

Annexe F

Comparaison des moyennes de deux distributions de tirages

Évaluer la position relative des espérances de deux lois probabilistes à travers un minimum de tirages est une tâche mathématique complexe. L'évaluation directe des tirages peut être faussée par la variance inhérente à ces lois. Cette annexe présente le cadre mathématique formel permettant de mesurer la précision des conclusions et d'associer une probabilité de dominance des espérances des deux lois inconnues grâce aux observations de tirage de variables aléatoires suivant ces lois.

F.1 Théorème central limite

Lorsque l'on observe les tirages de variables aléatoires suivant deux lois inconnues, il faut prendre en compte la variance de ces variables aléatoires. Dans le cas où l'on mesure des phénomènes complexes, tels que ceux basés sur des actions humaines, il y a tellement de facteurs incontrôlables, qu'il est impossible d'obtenir un modèle fiable des distributions des lois correspondantes. Hormis la variance et l'espérance, il est impossible de caractériser plus précisément ces distributions.

Cependant, lorsque l'on s'intéresse à la comparaison de suites de variables aléatoires définies sur un même espace de probabilité et suivant une même loi de probabilité D, il est possible d'appliquer le théorème central limite. Le théorème central limite établit que lorsque le nombre de tirages n augmente, la distribution S^n de la somme de ces variables aléatoires tend, quand n est suffisamment grand, vers une gaussienne d'espérance $n\mu$ et de variance $n\sigma^2$ où μ et σ^2 sont respectivement l'espérance et la variance de la loi de probabilité D.

Dans la suite de cette étude, nous ferons la supposition que les échantillons n_1 et n_2

des lois respectives D_1 et D_2 sont suffisamment grands pour considérer que S_1^n et S_2^n sont des gaussiennes.

F.2 La comparaison des moyennes de deux gaussiennes

F.2.1 Comparaison d'une gaussienne et d'un point

A partir du tirage de n variables aléatoires X_n suivant une même distribution inconnue D, d'espérance inconnue μ et de variance inconnue σ^2, nous obtenons une variable aléatoire $S^n = \sum_i X_i$ suivant une loi gaussienne, d'espérance $n\mu$ et de variance $n\sigma^2$. La somme observée $n\mu^*$ est alors un tirage suivant cette loi S^n. Par inversion de la loi gaussienne, la position de l'espérance $n\mu$ de S^n suit également une loi gaussienne ayant les mêmes caractéristiques mais centrée sur la somme observée et ayant une variance égale à la variance observée $N\left(n\mu^*, n\sigma^{*2}\right)$.

Ainsi, étant donné l'observation d'une espérance μ^* et d'une variance σ^{*2}, l'estimation de la probabilité que l'espérance μ soit supérieure à une valeur donnée μ_0 est alors calculée de cette façon :

$$P\left(\mu \geq \mu_0\right) = \frac{1}{\sqrt{2\pi}} \int_{\mu_0}^{\infty} e^{-\frac{1}{2}\frac{\left(x-\mu^*\right)^2}{\frac{\sigma^{*2}}{n}}} dx \tag{F.1}$$

$$= \frac{1}{2} erfc\left(\frac{\mu_0 - \mu^*}{\sqrt{\frac{2}{n}}\sigma^*}\right) \tag{F.2}$$

F.2.2 Comparaison de deux gaussiennes

Maintenant, nous prenons en compte deux sommes S_1^n et S_2^n de lois aléatoires D_1 et D_2. Les notations précédentes sont conservées en y ajoutant les indices 1 et 2 pour distinguer les éléments de chaque loi. Le problème revient à faire la somme de la formule F.2 pour les valeurs μ_0 pondérées par la gaussienne D_2. Au final, nous obtenons la formule suivante :

$$P\left(\mu_1 \geq \mu_2\right) = \int \frac{1}{2} erfc\left(\frac{x-\mu_1^*}{\sqrt{2}\frac{\sigma_1^*}{\sqrt{n_1}}}\right) \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\frac{\left(x-\mu_2^*\right)^2}{\frac{\sigma_2^{*2}}{n_2}}} dx \tag{F.3}$$

F.3 Méthodes de calcul de ces probabilités

Malheureusement la formule F.3 n'est pas calculable exactement, mais par échantillonnage, il est relativement aisé d'en obtenir une approximation. Ceci est utile pour les

significances calculées par exemple dans la section 6.3. Mais pour une utilisation en ligne, la complexité est trop grande et nous utilisons un algorithme calculant une approximation qui s'est révélée suffisante : on considère que la gaussienne ayant la variance empirique $\frac{\sigma^{*2}}{n}$ la plus petite est un point et on applique la formule F.2 qui est elle directe à calculer. C'est ce l'approximation qui a servi à faire les tests de l'algorithme d'exploration "intervalle de confiance" dans la sous-section 5.4.5.