Table des matières

To correct
To explain more deeply

Chapitre 1

Introduction générale

- Ici on parlera des motivations qui ont aboutis à ce projet, des objectifs de ce dernier ainsi que ses perspectives

Chapitre 2

Assistants virtuels intelligents

2.1 Introduction

Depuis la commercialisation du premier ordinateur grand public (Xerox PARC Alto) en 1973, le monde découvrit pour la première fois ce qui allait devenir l'apparence basique de chaque ordinateur moderne. En effet, la compagnie Xerox fut la première à proposer une interface graphique dotée de fenêtres, d'icônes et d'une souris pour se déplacer et d'un clavier pour écrire du texte. Bien que basique, cette idée lança alors plusieurs autres grandes marques sur le même chemin (IBM, Apple, Compaq ...). Par la suite, beaucoup ont essayé d'améliorer la façon dont l'homme utilisait sa machine : souris plus précise, écran doté d'une plus grande résolution, clavier plus enrichi, voire même l'introduction des écrans tactiles dans certains systèmes embarqués.

Cependant, certains voyaient encore cette façon d'utiliser la machine comme trop primitive, et peu intuitive. En effet laissez un enfant devant un ordinateur et il prendrait un bon moment pour apprendre à éditer ne serait ce qu'un simple fichier. Pour citer Donald A. Norman :

> "We must design for the way people behave, not for how we would wish them to behave."[1]

que nous pouvons traduire par :

> "Nous devons concevoir selon le comportement des utilisateurs, et non pas selon la façon dont nous voudrions qu'ils se comportent."

L'humanité a fait beaucoup de chemin depuis les années 70, l'utilisation d'un ordinateur de nos jours avec les moyens classiques (souris, clavier, écran ...) est devenue une tâche triviale, voire même une seconde nature, cela reste cependant dû au fait que de plus en plus de jeunes enfants sont exposés depuis leur plus jeune âge au monde technologique qui les entoure, le processus d'apprentissage reste cependant présent, l'effort d'utiliser les outils communs reste lui aussi présent.

La plus naturelle et plus ancienne façon de communiquer pour l'homme a toujours été la parole. Le développement de langues toutes aussi riches et complexes les unes que les autres a permis à l'humanité de briser plusieurs barrières sociales. L'avancement le plus naturel pour cette façon de communiquer serait donc de l'étendre aux machines que l'homme a su construire et améliorer au fil des années.

Motivé par cette manière que l'on a de communiquer entre nous, et épaulé par les récentes technologies telles que l'apprentissage automatique, le traitement automatique du langage naturel et

l'intelligence artificielle, les plus brillants des chercheurs ont entamé leurs travaux dans cette toute nouvelle direction.

Les Assistants Virtuels Intelligents (Smart Personal Assistant, SPA [2]) sont donc le produit de plusieurs années de recherche, visant tout d'abord à faciliter certaines tâches pour l'utilisateur. Les premiers SPAs étaient conçus comme des agents de conversation ou Chatbots, limités dans leurs actions et dépendant toujours d'un moyen de communication textuel, ce n'était pas la forme désirée du SPA. Avec l'avancement des recherches sur la reconnaissance automatique de la parole (Automatic Speech Recognition, ASR) et l'émergence de l'apprentissage automatique, les tout premiers assistants virtuels utilisant l'ASR étaient spécialisés dans certains domaines comme des systèmes médicaux d'aide à la décision. Il a ensuite été plus aisé de briser la barrière et de réaliser ce qui était encore une esquisse d'un SPA personnalisé. Aujourd'hui, et ce depuis l'avènement de l'apprentissage profond et la popularisation des Smartphones, de nouveaux SPAs comme Apple Siri (voir 2.5.2) et Google assistant (voir 2.5.1) et Amazon Alexa (voir 2.5.3) ont fait leurs apparitions, offrant de plus en plus de services personnalisés et spécifiques à chaque utilisateurs.

Dans la suite de ce chapitre nous essayerons de mieux détailler ce qu'est un SPA, ce qui est demandé d'un tel système, ses domaines d'application, en enchaînant par une description d'une pseudo-architecture potentielle de ce système, pour enfin conclure sur les limitations actuelles et les motivations de ce projet.

2.2 L'importance du contexte pour un SPA

Informellement, un SPA est un type d'agent (voir 2.3.2) logiciel qui peut effectuer certaines tâches et proposer des services dédiés aux utilisateurs qui vont d'une simple tâche (Ouvrir une fenêtre, lancer une application ...) à la réalisation de requêtes un peu plus complexes comme réserver une table dans un restaurant en passant un appel vocal (voir 2.5.1.1). Pour répondre efficacement à toutes sortes de requêtes, un SPA se doit donc de garder trace du contexte courant de sa conversation avec l'utilisateur. Il doit disposer d'un système capable d'enregistrer les informations pertinentes et de savoir les réutiliser, mais aussi de pouvoir déduire lesquelles de ces informations sont manquantes. On parle ici de Context-Awarness ou Sensibilité au contexte, comme vu dans [2].

D'après [2] et [3], *Day* et *Abwod* définissent un contexte comme suit :

> "A context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"

qui peut être traduit par :

> "Un contexte est une information qui peut être utilisé pour caractériser l'état d'une entité. Une entité peut être une personne une place ou un objet, considérée comme pertinente à l'interraction entre l'utilisateur et l'application, ainsi qu'à ces deux derniers eux mêmes"

Il en découle que pour parvenir à développer un système qui puisse répondre aux besoins individuels et spécifiques de chaque personne, modéliser et prendre en compte le contexte semble être une solution prometteuse.

2.3 Caractéristiques principales d'un SPA

À partir de [2], nous pouvons dégager certaines caractéristiques principales qui peuvent être vues comme primordiales pour qualifier un assistant virtuel comme étant intelligent.

2.3.1 Sensible au contexte

Comme précédemment vu dans la définition du contexte (section 2.2), ce dernier peut être interprété comme tout aspect d'une entité (position d'un objet, couleur d'un objet, température d'une chambre, etc.). Un assistant dit intelligent doit donc être capable de capturer le concept du contexte, d'utiliser et de traiter toute information catégorisée comme contextuelle. Pour être plus précis, un SPA doit être sensible à l'évolution du contexte courant, par le biais de capteurs optiques, de microphones, ou tout ce qui pourrait amener l'utilisateur à faire évoluer la requête qu'il a émise. L'assistant devra donc proposer un système de mise à jour du contexte pour éliminer les informations inutiles et garder celles qui pourraient aider à répondre à la requête de l'utilisateur.

2.3.2 Évolutif

Comme vu dans la section 2.2, un SPA peut être vu comme un type d'agent. Pour rappel, d'après *Russel* et *Norvig* dans [4], un agent est une entité autonome pouvant interagir avec son environnement afin d'accomplir certaines tâches et peut être de plusieurs types :

- Agent à réflexes simples : agent exécutant ses actions à base de règles conditionnelles simples (c.à.d Si *Condition* alors *exécuter actions*), ils sont ainsi très simplistes et limités dans la portée de leurs actions.
- Agent basé modèle : semblable aux agents à réflexes simples, il est doté d'un modèle interne complexe censé représenter le monde extérieur auquel l'agent a accès. Cependant, il applique les actions de la même manière que le précédent type d'agents.
- Agent à but : ce type représente une amélioration des agents simples puisqu'il est doté d'un ensembles d'états buts à atteindre d'une façon ou d'une autre.
- Agent à utilité : il s'agit ici agents à buts qui tentent d'aboutir à leurs buts d'une manières optimisée (intelligente) utilisant une fonction de mesure adéquate pour le choix des différents états à atteindre.
- Agent apprenant : agent à utilité enrichi par un module d'apprentissage qui sert de juge pour répondre aux "critiques" des actions qu'il entreprend. Le terme agent évolutif est aussi employé.

Pour ce qui est des SPAs, les plus récents systèmes (ex : Amazon Alexa qui améliore son module de reconnaissance de la parole après chaque réponse non réfutée par l'utilisateur) peuvent être considérés comme des agents apprenants, répondant de ce fait à la contrainte évolutive imposée. Cependant, le domaine de l'auto-évolution des systèmes intelligents est encore un domaine nouveau qui se voit aidé par les récentes avancées dans l'apprentissage automatique [2].

2.3.3 Multimodal

Afin d'assurer une aisance d'utilisation, les SPAs sont fréquemment amenés à récupérer les requêtes (ou données) en entrée de la manière la plus naturelle possible (par exemple par le biais de la parole). Cependant, pour garantir une expérience d'utilisation adéquate, l'assistant sera souvent confronté à récupérer ces requêtes de différentes manières, que ce soit à travers une interface graphique (écran tactile) ou à travers un texte brut tapé au clavier, voire même à travers des expressions faciales ou des états cognitives/émotionnels [5], pour ensuite produire une réponse qui elle aussi pourrait éventuellement être de la forme textuelle, sonore ou les deux. Cette capacité à recevoir en entrée et/ou produire une sortie de plusieurs façons différentes est appelée la multi-modalité [6]. Cette caractéristique permet de masquer à l'utilisateur toute la complexité d'acquisition de ses requêtes.

2.3.4 Anthropomorphe

Plusieurs auteurs tendent à attribuer une grande importance à l'anthropomorphisme des SPAs [7], qui est

> "Un mécanisme qui pousse les êtres humains à induire qu'une entité non-humanoïde possède des caractéristiques et comportements propres à l'homme"[8]

Ce comportement humanoïde pousserait donc l'utilisateur à se sentir plus à l'aise avec l'assistant, le conduisant ainsi à adopter une façon de communiquer plus humaine et moins structurée qu'avec les autres machines. Ceci est une caractéristique majeure d'un SPA se disant personnalisé.

2.3.5 Multi-plateforme et Flexible

Malgré leurs récentes prouesses, certains SPAs sont encore restreints à un écosystème fortement dépendant du fabricant. Cowan et al. mentionnent dans [9] que Apple Siri est limité à l'environnement constitué des produits de la firme à la pomme, n'ouvrant par défaut que les applications de cette dernière quand une requête lui est transmise. C'est un comportement que les assistants devraient éviter, car une indépendance des plateformes utilisées est, certes, très complexe à instaurer, mais offre plus de possibilités aux utilisateurs et aux développeurs pouvant ainsi exploiter la puissance de certaines plateformes (Smartphones, TV connectées, etc). Avec l'émergence de l'IoT (Internet of Things) et des maisons intelligentes par exemple, c'est un tout nouveau terrain de jeu qui est présenté aux SPAs, offrant plus d'opportunités pour les utilisateurs.

2.4 Domaines d'applications des SPAs

Après avoir vu les différents aspects que les SPAS doivent traiter, nous nous intéresserons maintenant aux types de services et applications que ces derniers pourraient fournir pour démontrer qu'ils peuvent bel et bien faciliter certaines tâches à l'homme.

2.4.1 Vie quotidienne

À la base, les SPAs étaient destinés à un usage très personnel comme la gestion des achats dans les supermarchés, ou des guides touristiques de plusieurs destinations de voyage. Cette spécificité a commencé à s'estomper petit à petit avec l'émergence de nouveaux systèmes dédiés à des applications plus générales, comme les maisons intelligentes ou les assistants de planification de tâches. Ceci a permit de mettre encore plus l'accent sur cet aspect de convivialité que les tout premiers SPAs ont tenté de perfectionner. Ainsi, ces assistants spécialisés dans des domaines restreints (Tourisme, shopping, détente, etc) ont été regroupés dans un seul système plus polyvalent, capable de répondre à des besoins quotidiens divers et variés, allant même à fournir une assistance aux personnes âgées pour leur faciliter les tâches rudimentaires devenues trop fatigantes.

2.4.2 Assistance professionnelle

Les SPAs ont aussi une place dans le monde professionnel. Dans [10] il est cité que dans les situations où la marge d'erreur est très petite (par exemple dans les système de manufacturing [1]) l'assistance d'un SPA est nécessaire, servant d'un aide à l'humain pour la prise de décision.

Par exemple, dans un environnement de travail hétérogène (Nouveaux/anciens employés, Hiérarchies des postes ...) les SPAs pourraient décharger les employés les plus expérimentés de la tâche

1. Manufacturing ici dans le sens chaîne de montage industrielle, par exemple dans des usines.

d'assister les nouveaux arrivants, pour ainsi aider ces derniers dans leurs tâches et permettre aux autres de se focaliser sur les leurs.

2.4.3 E-Apprentissage

Les SPAs peuvent aussi être utiles dans l'enseignement, aussi bien dans un milieu académique que professionnel. D'une part ils pourraient occuper plusieurs rôles dans les établissements scolaires (correcteur automatique de copies, enseignant interactif ...) [11] et, d'autre part, accompagner les employés durant leurs formations professionnelles.

Ainsi, en considérant les caractéristiques d'un SPA, la sensibilité au contexte est reliée aux expériences antérieures de l'apprenant, permettant au SPA d'adapter son processus d'enseignement en conséquence.

En ce qui concerne l'aspect évolutif du SPA, il lui permet de préférer une approche d'enseignement à une autre selon les résultats de ses apprenants.

2.5 Exemples de SPAs

Pour illustrer la puissance des SPAs les plus récents, nous présentons dans cette section les quatre produits qui dominent le marché courant :

2.5.1 Google assistant

Lancé en 2016 sous forme d'un chatbot intégré dans l'application Google Allo, Google Assistant s'est vu ensuite être directement intégré sur les système d'exploitation Android (que ce soit sur smartphones ou tablettes, et plus récemment sur Google Home [2]). Google Assistant est un assistant à tout faire qui a été développé par les ingénieurs de Google dans le but de faciliter la recherche sur internet, la planification des tâches, l'ajustement des réglages de l'appareil, etc. Son point fort est sa capacité à engager une conversation bi-directionnelle avec l'utilisateur, assurant ainsi une interaction personnalisée variant d'un utilisateur à un autre. Cette capacité lui permet par exemple de proposer certains résultats de recherche selon les précédentes interactions avec l'utilisateur ou de lui proposer une activité si ce dernier lui mentionne qu'il s'ennuie (voir figure 2.1).

2. Appareil servant à contrôler les composants d'une smart-house ainsi que l'utilisation des différents services de Google

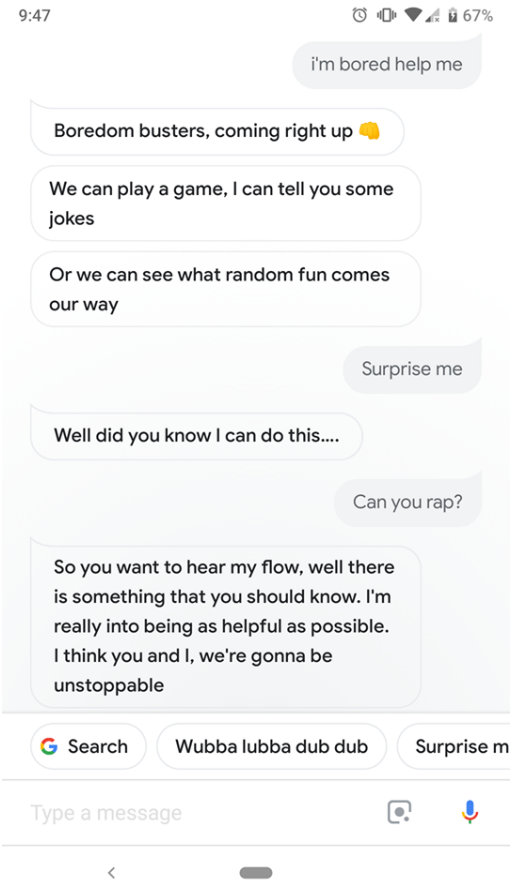


FIGURE 2.1 – *Conversation aléatoire avec Google Assistant*

FIGURE 2.2 – *Requête simple formulée à Google Assitant*

2.5.1.1 Google duplex

Une des nouveautés impressionnante de Google Assistant est la fonctionnalité Google Duplex. Toujours en phase de développement, ce module est capable de passer des appels a de vraies personnes et d'avoir une conversation avec elles afin de réaliser une tâche demandée par l'utilisateur comme par exemple réserver une chambre d'hôtel, une table au restaurant, etc.

FIGURE 2.3 – *Google duplex réservant une place dans un salon de coiffure*

2.5.2 Apple Siri

Siri est l'assistant virtuel développé par Apple. Contrairement aux SPAs durant sa sortie, Siri proposait une nouvelle façon de communiquer avec l'utilisateur, à travers une interface de requêtes vocales, et une façon de converser très humanoïde (satisfaisant ainsi le critère d'anthropomorphisme 2.3.4). Siri est capable de répondre à des questions précises (voir figure 2.6), de proposer des recommandations, déléguer la requête à des services web ou d'autres applications (voir figures 2.4 et 2.5). Il a l'avantage (et l'inconvénient) d'être uniquement disponible que sur les appareils qui composent l'écosystème d'Apple (MacBook, iPhone, iWatch, etc).

FIGURE 2.4 – *Intégration aux applications [12]*

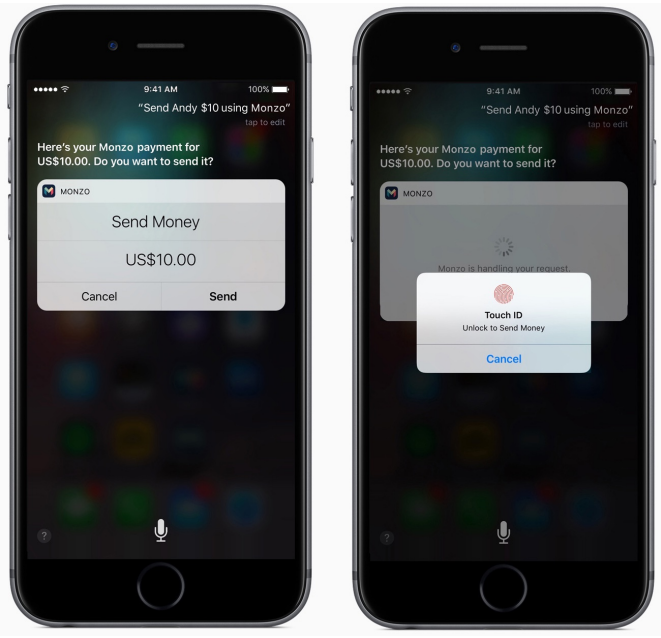


FIGURE 2.5 – *Service paiement 1 [12]*

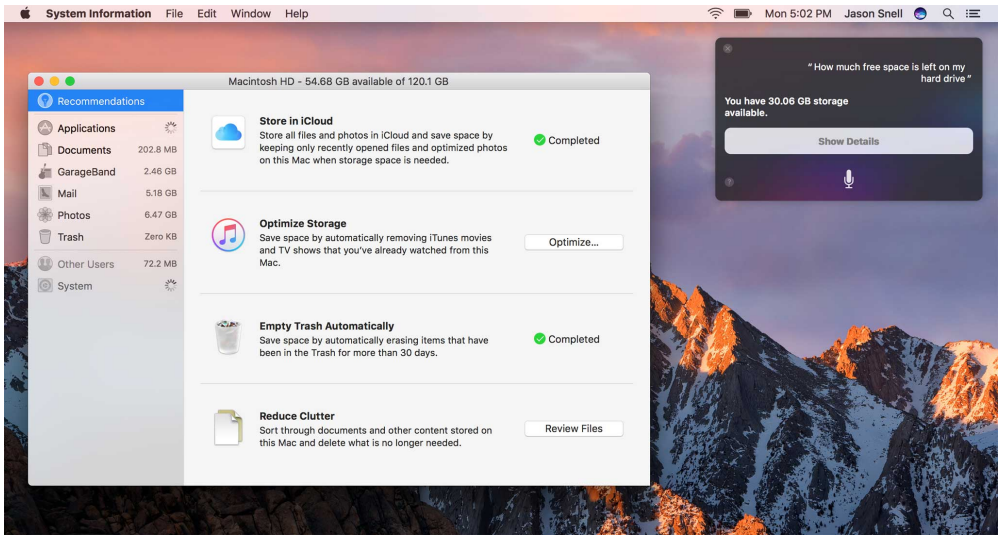


FIGURE 2.6 – *Siri sur un laptop [13]*

2.5.3 Amazon Alexa

Amazon Alexa est un assistant exclusivement intégré au dispositif Amazon Echo (un haut-parleur portatif). À l'instar de Siri, il est aussi capable de communiquer avec l'utilisateur par le biais de la parole, pouvant ainsi exécuter diverse commandes comme jouer de la musique, réciter des livres audios, annoncer des news en temps réel (Résultats sportifs, tendances politiques, etc). Son atout majeur est sa capacité à s'intégrer à plusieurs appareils-connectés (Contrôleur de thermostat ou de lumières ambiantes dans une Smart-House) ainsi que la possibilité d'ajout des Skills (ou compétences) de la part des développeurs tiers pour enrichir la panoplie de services que peut offrir Alexa.

2.5.4 Microsoft Cortana

Cortana est la tentative de la part de Microsoft d'intégrer un assistant dans son système d'exploitation Windows 10 et WindowsPhone. Il propose divers services de base tel que planifier des tâches, exécuter des commandes via la parole, et analyser des résultats de recherche sur le moteur de recherche de Microsoft, Bing, pour répondre à des questions.

2.6 Conclusion

À travers les sections précédentes, nous avons essayé de présenter les différents aspects d'un assistant virtuel intelligent (caractéristiques, exemples, architectures possibles, etc). Nous avons donc pu apprécier la potentielle puissance d'un tel système s'il venait à être perfectionner d'avantage.

En effet, en examinant les domaines d'applications, il est facile de déduire que le recours à un SPA peut grandement faciliter certaines tâches, que ce soit celles qui sont les plus triviales pouvant retarder d'autres tâches plus importantes, ou bien celles qui doivent faire appel à la précision

et à la grande capacité de calcul des machines, assurant ainsi des résultats précis et rapidement délivrés.

À la fin de ce chapitre nous pouvons donc mettre en valeur la place primordiale que pourraient avoir les SPAs s'ils arrivaient à maturité, c.à.d briser la barrière qui sépare les humains de la machine, parvenant ainsi à faire partie de la vie quotidienne des utilisateurs.

Dans le prochain chapitre nous allons principalement aborder les aspects techniques des différents composants du SPA que nous désirons réaliser.

Chapitre 3

Composants de base d'un assistant personnel

3.1 Introduction

Durant ce chapitre, nous allons détailler un peu plus l'aspect technique d'une architecture typique pour un SPA, nous commencerons d'abord par définir des notions de bases ainsi que des techniques d'apprentissage automatique, nous traiterons celles qui ont été les plus utilisées pour les travaux que nous avons trouvé. La suite du chapitre sera organisé en sections qui décrirons chacune le fonctionnement d'une partie du SPA, en citant les travaux et références qui relatent de cette dernière. Nous terminerons sur une conclusion qui introduira le chapitre suivant.

3.2 Notions et aspects théoriques

Dans cette section nous essaierons de présenter différentes notions liées au domaine de l'apprentissage automatique, pour ensuite les citer dans les module qui leur font appel.

3.2.1 Apprentissage automatique

Le plus souvent, un domaine scientifique peut être défini à travers le, ou les types de problèmes dont il essaye de trouver une solution, d'après [14], l'auteur à défini ce problème sous forme d'une question :

> "How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes ?"[14]

que nous pouvons traduire par :

> "Comment pourrions nous développer un système informatique qui pourrait s'améliorer à travers l'expérience, et quelles seraient les lois qui régiraient le processus d'apprentissage ?"

D'après [14] l'apprentissage automatique est un paradigme qui stipule qu'un système informatique peut apprendre à effectuer un ensemble de tâches T, sachant qu'il dispose d'un ensemble de données E, tout en améliorant sa performance P.

Il existe plusieurs sous catégories d'apprentissage automatique, elles différent principalement par la manière dont le système apprend, du type de données sur les quelles il apprend, ainsi que du but de son apprentissage (classification, régression,...). Nous pouvons citer les catégories suivantes :

- **Apprentissage supervisé** : Les algorithmes de cette catégories ont besoin d'une assistance externes, les données doivent être séparées en deux parties (ensemble d'apprentissage et de test), un *label* ou classe doit être associée à chaque instance pour permettre à l'algorithme de calculer un certain taux d'erreur qu'il essayera d'améliorer au fur et à mesure qu'une nouvelle instance est présentée [15]. Idéalement le système pourra apprendre une certaines fonction $\hat{f} : X \to Y$ qui liera les entrées X aux sorties Y en minimisant une erreur E_Y
- **Apprentissage non supervisé** : Ici, les algorithmes ne disposent pas d'un étiquetage des données, ils essayeront donc d'apprendre des *pattern* ou motifs fréquents pour grouper les données similaires. De tels algorithmes ne se préoccupent pas de la classe, mais de la similarité entre un groupe de données [16]
- **Apprentissage par renforcement** : Cette dernière catégories d'algorithmes apprennent par un système de *trial and error*[1] en interagissant avec l'environnement pour accomplir une tâche (voir 3.6.5)

3.2.2 Réseaux de neurones artificiels

Un réseau de neurones artificiels est une structure d'appariement non-linéaire inspiré de la façon dont les systèmes nerveux biologiques fonctionnent. Ils permettent de modéliser les relations sous-jacentes des données. ils sont composés d'un nombre arbitrairement large de plus petites unités de calcul interconnectées appelées neurones, qui traient l'information d'une manière parallèle dans le but de résoudre un problème bien spécifique [17]. Ils ont notamment connu un très grand succès lors des dernières années dans différents domaines comme la reconnaissance d'images [18], la reconnaissance automatique de la parole [19] [20] ou encore la classification de textes [21] [22].

Il existe une variété d'architectures de réseaux de neurones, nous traiterons dans cette section de trois d'entre elles :

- Réseaux de neurones multicouches denses
- Réseaux de neurones profonds
- Réseaux de neurones récurrents et leurs variantes

1. Essais et erreur ?

3.2.2.1 Réseaux de neurones multicouches denses

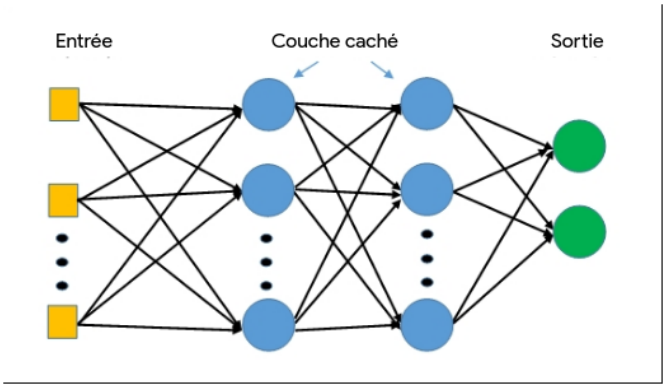


FIGURE 3.1 – *Architecture basique d'un réseaux de neurones multicouches*

La forme la plus basique que puisse avoir un réseaux de neurones est celle d'un réseaux multi-couches comme montré dans 3.1, elle se compose de trois parties :

- **Une couche d'entrée :** cette couche est composé d'un ensemble de neurones ou *perceptron* [23], elle représente l'information en entrée codifié en un vecteur numérique χ
- **Une ou plusieurs couches cachées :** le coeur du réseau, c'est une succession de couche de neurones dont chaque couche ω_i reçoit un signal sous forme d'une ou plusieurs valeurs numériques depuis une couche antérieure ω_{i-1} ou bien la couche d'entrée χ, puis envois en sortie un autre signal de même nature qui est une combinaison non-linéaire du signal en entrée vers une couche suivante ω_{i+1} ou bien la couche de sortie
- **Une couche de sortie :** elle permet de calculer une valeur y qui peut être vu comme la prédiction du modèle Φ par rapport à son entrée χ :

$$y = f_\Phi(\chi) \tag{3.1}$$

Le réseau calculera une erreur $e(y, \hat{y}$ en fonction de sa valeur en sortie et de la valeur exacte puis corrigera cette erreur au fur et à mesure du parcoure des données d'apprentissage [24]

3.2.2.2 Réseaux de neurones profonds

Les réseaux de neurones à une seule couche sont dit *shallow*, ils présentaient l'avantage d'être assez rapide durant la phase d'apprentissage, cependant les limites computationnelle d'antan se sont vu vite brisé avec le développement de processeurs plus puissants, de plus avec l'explosion des données sur le web, toutes les conditions nécessaires pour la mise en place d'architectures plus complexes étaient réunies. Les réseaux de neurones profonds sont une adaptation des réseaux multi-couches classiques avec généralement plus de 3 couches cachées.

même si le principe reste le même, l'apprentissage profond est puissant car les architectures les plus complexes permettent d'extraire automatiquement les caractéristiques qui sont importantes mais non visibles, c'est le cas des réseaux de neurones constitutionnels et récurrents [25]

3.2.2.3 Réseaux de neurones récurrents

Un aspect que les réseaux de neurones (profonds ou pas) ne peuvent capturé est la notion de séquentialité, en effet beaucoup de problèmes qui sont de nature séquentielle ne peuvent être modélisé par les architectures dites classiques, comme l'analyse d'un texte, l'introduction d'une notion de séquence permet donc de capturer des dépendances entre certains états et leurs voisins, on parle ici de contexte [26].

Les réseaux de neurones récurrents (RNNs) sont des réseaux de neurones *feedforward* dont certaines connections en sortie sont réintroduite comme entrées dans le réseau durant une étape ultérieur du processus d'apprentissage, ceci introduit la notion de temps dans l'architecture, ainsi à un instant t, un neurones récurrent recevra en entré la données $x^{(t)}$ ainsi que la valeur de l'état caché $h^{(t-1)}$ résultante de l'étape précédente du réseau, la valeur en sorite $\hat{y}^{(t)}$ est calculée en fonction de l'état caché $h^{(t)}$, les équations suivantes montrent les calculs effectués :

$$h^{(t)} = \tanh(W^{hx} \times x^{(t1)} + W^{hh} \times h^{(t-1)} + b_h) \tag{3.2}$$

$$\hat{y}^{(t)} = softmax(W^{hy}\ timesh^{(t1)} + b_y) \tag{3.3}$$

Où W^{hx} est la matrice de poids entre la couche d'entrée et l'état caché et W^{hh} est la matrice de poids de récurrence (entre l'état t et $t-1$) les deux vecteurs b_h et b_y sont les vecteurs de biais et $\times$ est l'opération du produit matriciel[26]

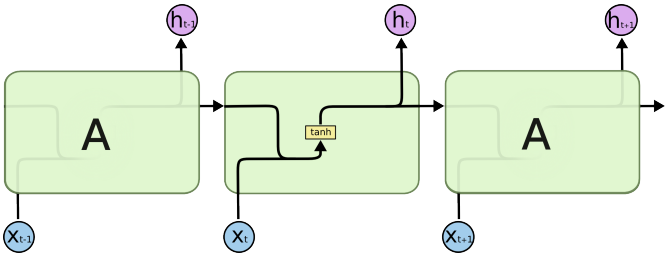


FIGURE 3.2 – *Architecture interne d'un réseau de neurones récurrent à un instant t [27]*

Un des principaux problèmes que rencontrent les RNNs est celui du *Vanishing gradient* traduit par la *Problème de disparition du gradient* [28], les relations à long termes entre dans les séquences ne sont donc pas capturées. Ainsi, pour remédier à ce problème, des architectures de réseaux de neurones dotées d'un module de mémoire ont étés introduites.

3.2.2.4 Réseaux de neurones récurrents à mémoire court et long terme (LSTM)

Premièrement introduit en 1997 par *Sepp Hochreiter* et al. dans [29], cette architecture de réseaux de neurones récurrents est doté d'un système de *portes* qui filtrent l'information qui y passe ainsi que d'un état interne de la cellule mémoire d'un LSTM , ainsi nous pouvons décortiquer les composantes d'une cellule LSTM (voir 3.3) comme suit :

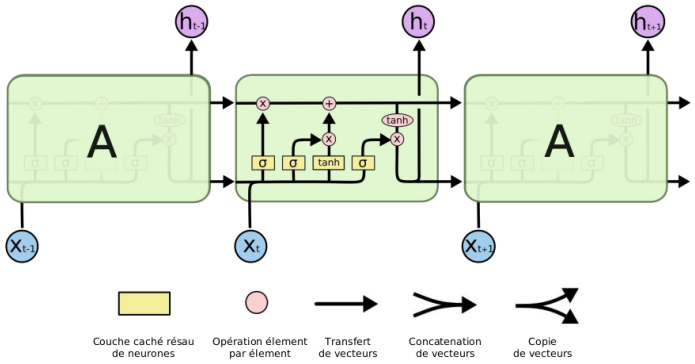


FIGURE 3.3 – *Architecture interne d'une cellule mémoire dans un réseau LSTM [27]*

- **Porte d'entrée (Input gate) :** C'est une unité de calcul qui laisse passer certaines informations en entrée en utilisant une fonction d'activation sigmoïde pour pondéré les composantes du vecteur d'entrée à une étape t et celles du vecteur d'état interne à l'étape $t-1$ (1 : laisser passer, 0 : ne pas laisser passer) et ainsi générer un vecteur candidat C_t [29].

$$\begin{aligned} i_t &= \sigma(W_i \times [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \times [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.4}$$

- **Porte d'oubli (Forget gate) :** De manière similaire, cette porte permet de spécifier au fur et à mesure de l'apprentissage les informations à oublier, qui sont donc peu importantes [29][26].

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{3.5}$$

- **État interne de la cellule (Internal cell's state) :** C'est une sorte de convoyeur qui fait circuler l'information à travers la cellule, cet état est mis à jour à travers la combinaison des deux valeurs précédemment filtré par les portes f_c et i_c[29].

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.6}$$

- **Porte de sortie (Output gate) :** Pour renvoyer un résultat comme état interne du réseau, la cellule filtre son vecteur d'état et le combine avec la donné en entrée et l'état du réseau précédent pour ne laisser passer que certaines informations [27][29][26].

$$\begin{aligned} o_t &= \sigma(W_o \times [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{3.7}$$

3.2.2.5 Modèle basé N-grammes

Très souvent quand il est nécessaire de traiter un ensemble de mots, phrases ou bien tout entité atomique qui constitue une séquence, il serait intéressant de pouvoir assigner une probabilité de vraisemblance à une séquence (de mots par exemple) en particulier.

Dans un contexte textuel, un N-gramme est une suite de N mots consécutifs qui forment une sous-chaîne $S\prime$ d'une chaîne de caractères S. Un exemple d'un ensemble de bi-grammes (2-grammes) pour la phrase "Ouvre le fichier *home*" serait donc : (Ouvre,le), (le,fichier), (fichier,*home*).

Ainsi, en disposant d'un corpus de texte assez large et diversifié et d'une méthode de comptage efficace, il serait possible de calculer la vraisemblance d'apparition d'un mot w après une certaine séquence de mots t sous forme d'une probabilité P [30].

$$P(w|t) = \frac{Comptage(t,w)}{Comptage(t)} \tag{3.8}$$

3.2.2.6 Modèle de Markov caché (Hidden Markov Models HMM)

Informellement, un modèle de Markov caché (HMM) est un outil de de représentation pour modéliser la distribution de probabilité d'une séquence d'observations. Il est dit *caché* pour deux raisons. Premièrement, il est assumé qu'une observation O à un instant t (dénotée O_t) est le résultat d'un certain processus (souvent stochastique) dont l'état S_t est caché à l'observateur. Deuxièmement, l'état S_t du processus caché ne dépend uniquement que de sont état à l'instant $t-1$, il est alors dit que ce processus est Markovien ou qu'il satisfait la propriété de Markov [31][32].

D'un façon plus formelle, un HMM est un 5-tuple $< (S, V, \Pi, A, B >$ [33] où :

- $S = \{s_1 ... s_N\}$ est l'ensemble fini des N états du processus sous-jacent.
- $V = \{v_1 ... v_M\}$ est l'ensemble des M symboles qui constituent un certain vocabulaire.
- $\Pi = \{\pi_i\}$ est une distribution initiale des probabilité d'états tel que π_i est la probabilité de se trouver à l'état i à l'instant $t = 0$, bien évidement $\sum_{i=1}^{N} \pi_i = 1$
- $A = \{a_{ij}\}$ est une matrice NxN dont chaque entrée a_{ij} est la probabilité de transition d'un état i à un état j avec $\sum_{j=1}^{N} a_{ij} = 1$ pour tout $i, j = 1...N$.
- $B = \{b_i(v_k)\}$ est l'ensemble des distribution de probabilités d'émission (ou d'observation), donc $b_i(v_k)$ est la probabilité de générer un symbole v_k du vocabulaire étant donné un certain état i avec $\sum_{k=1}^{M} b_i(v_k) = 1$ pour tout $i = 1...N$.

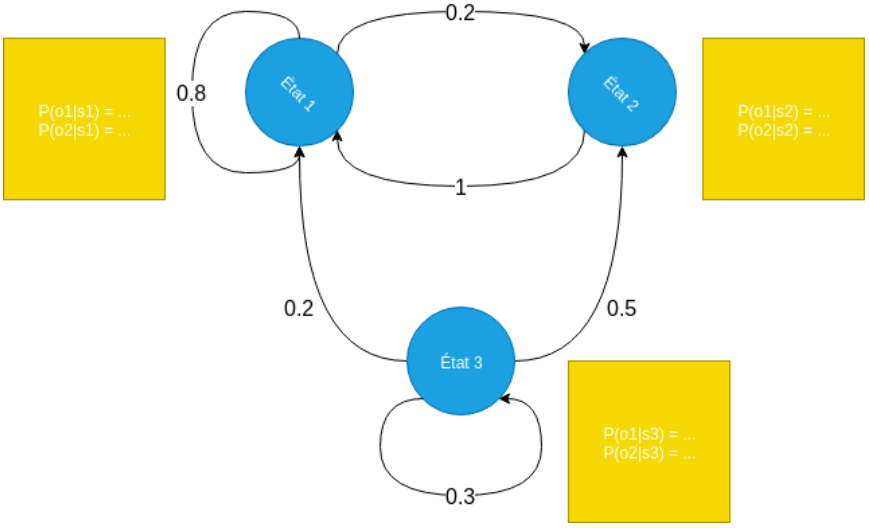


FIGURE 3.4 – *Exemple d'une distribution de probabilité de transition ainsi qu'une distribution de probabilité d'observations pour un HMM à 3 états et 2 observations*

Le but serait de trouver la séquence d'états $S_{1:K}$ qui maximiserait la probabilité [31] :

$$P(Y_{1:K}|O_{1:K}) = P(O_1)P(O_1|S_1)\prod_{t=2}^{K} P(S_t|S_{t-1})P(O_t|S_t) \tag{3.9}$$

Pour y parvenir d'une manière efficace, un décodeur qui implémente l'algorithme de viterbi peut être utiliser [34][35]

3.3 Architecture d'un SPA

3.4 Reconnaissance automatique de la parole (ASR)

Le premier module qui compose le système est celui de la reconnaissance automatique de la parole (ASR), le but d'un tel système (ou sous-système dans notre cas) est de convertir un signal audio décrivant la locution d'un utilisateur en un texte qui peut être interprété par la machine [36]. Différentes approches ont été développé au court des années, une architecture s'est ensuite dégagée, les systèmes passent par deux phases : La phase d'apprentissage et la phase de reconnaissance. La première consiste a collecter les données qui constituent le corpus d'apprentissage, un ensemble de fichiers audios avec leurs transcriptions en texte et en phonèmes, le signal est ensuite traité pour en extraire des vecteurs de caractéristiques (ou attributs), la suite de l'apprentissage consiste à initialiser le HMM, lui passer les vecteurs précédemment extraits puis de l'enregistrer pour la phase suivante qui est la reconnaissance. Dans la deuxième phase le signal audio passe par le même procédé d'extraction des attributs, en utilisant un algorithme de décodage approprié [35], la séquence d'observation passe par le HMM et la meilleure séquence de mots est sélectionnée [20].

Une architecture assez générale s'est dégagée, quatre modules sont impliqué dans le processus de la reconnaissance, nous les citerons dans les sections qui suivent en citant les modèles utilisés dans chacun.

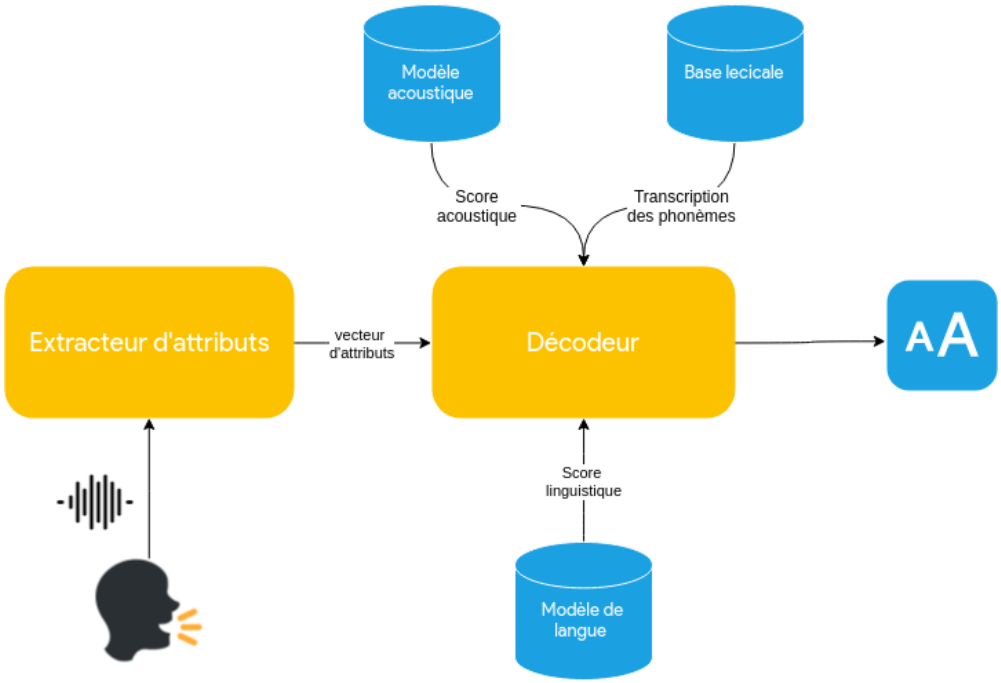


FIGURE 3.5 – *Architecture des systèmes de reconnaissance de la parole [20]*

3.4.1 Acquisition du signale et extraction d'attributs

l'étape consiste à extraire une séquence de vecteurs caractéristiques à partir du signal audio, fournissant une représentation compacte de ce dernier. Le processus démarre par la segmentation du signal en *trames*, une transformation de fourrier est ensuite appliquée à ces dernières pour engendrer un spectre de magnitude qui sera ensuite passé à un module de transformation en spectre de Mel (Mel-Spectrum) qui est une sorte de changement d'échelle, finalement, une transformation inverse de fourrier est appliqué pour engendrer le Mel-Cpestrum qui est le vecteur d'attributs que nous recherchions [37]. Un tel processus peut utiliser plusieurs techniques pour la mise à l'échelle : MFCC (Mel-frequency cepstrum) [38], LPC (Linear Predictive coding) [39] et RASTA (RelAtive SpecTrAl) [40], chacune possédant ses avantages et ses inconvénients.

3.4.2 Modélisation acoustique et du lexique

C'est le cœur du système de reconnaissance, le but est de construire un modèle permettant d'identifier une séquence de phonèmes à partir d'une séquence d'observations de vecteurs d'attributs, ceci peut être fait on utilisant un modèle HMM et en disposant d'un corpus de fichiers audio accompagnés de leurs transcriptions en phonèmes et en texte [41] [33], ou bien un réseau de neurones profonds [20], ou encore une hybridation de ces derniers [42]. Le modèle acoustique procède après la reconnaissance de la séquence de phonèmes au décodage de ce derniers, ceci est fait en utilisant un dictionnaire linguistique qui transcrit chaque mots du vocabulaire en les phonèmes qui le constituent, ceci peut induire à un cas d'ambiguïté où plusieurs mots peuvent avoir la même transcription phonétique (ou bien aucun mot ne correspond à cette séquence). Pour lever cette ambiguïté un modèle de langue est nécessaire pour décider quelle est la séquence de mots la plus probable qui coïncide avec la séquence de phonèmes observée.

3.4.3 Modélisation de la langue

Cette étape consiste à modéliser les aspects linguistiques de la langue que le système essaye de traiter, souvent spécifique au domaine d'application pour restreindre l'espace de recherche des mots reconnaissables par ce dernier et déterminer la séquence de mots en sortie la plus plausible. les modèles utilisés sont basé soit sur des grammaires à contexte libre dans le cas où les séquences de mots reconnaissables sont peut variées et peuvent être modélisées par le biais de règles de la langue [43]. Dans un cadre plus récent, les modèles probabilistes basé sur les N-grammes sont utilisés. Disposant d'un volume de données textuelles assez conséquent, l'utilisation de modèle basés sur les N-grammes ont prouvé leurs utilité [30][20][44]

3.5 Compréhension du langage naturel (NLU)

3.5.1 Définition

3.5.2 Classification d'Intent

3.5.3 Extraction d'entités

3.5.4 Analyse sémantique

3.6 Gestion du dialogue

La compréhension du langage naturel permet de transformer un texte en une représentation sémantique. Afin qu'un système puissent réaliser un dialogue aussi anthropomorphe que possible, il doit décider, à partir des représentations sémantiques reçues au cours du dialogue, quelle action à prendre à chaque étape de la conversation afin de la transmettre au générateur du langage naturel 3.7 pour afficher le résultat à l'utilisateur. On distingue deux principaux modules généralement présent dans les systèmes de gestion de dialogue :

- Un module qui met à jour l'état du gestionnaire de dialogue.
- Un module qui détermine la politique d'action du gestionnaire de dialogue.

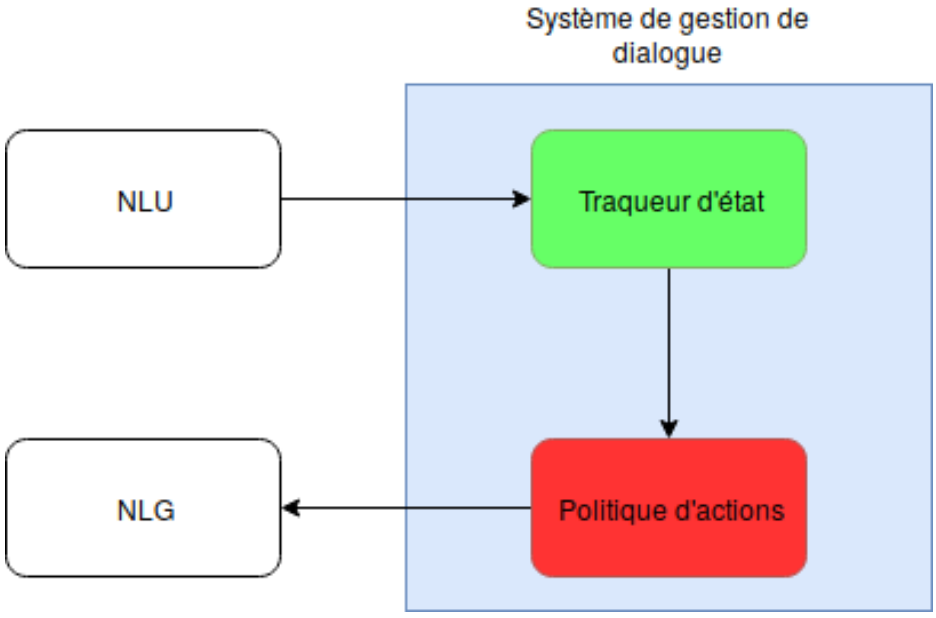


FIGURE 3.6 – *Schéma général d'un gestionnaire de dialogue*

3.6.1 Processus de décision Markovien (MDP)

Un gestionnaire de dialogue peut être modélisé par un processus de décision Markovien[45]. Ce dernier est modélisé par un 4-tuple (S,A,P,R)(*) :

- S : ensemble d'états du système.
- A : ensemble d'actions du système.
- P : distribution de probabilités de transitions entre états sachant l'action prise. P(s'/s,a) est la probabilité de passer à l'état s' sachant qu'on était à l'état s après avoir pris l'action a.
- R : est la récompense reçue immédiatement après avoir changer d'état avec une action donnée. R(s'/s,a) est la récompense reçu après avoir passer à l'état s' sachant qu'on était à l'état s après avoir pris l'action a.

D'après (*) un MDP, à tout instant t, est dans un état s, dans notre cas c'est l'état du gestionnaire de dialogue. Il peut prendre une action a afin de passer à un nouvel état s', et sur ce fait, il reçoit une récompense, qui ,dans notre cas, est une mesure sur les performance du système de dialogue. Le but est de maximiser les récompense reçu.

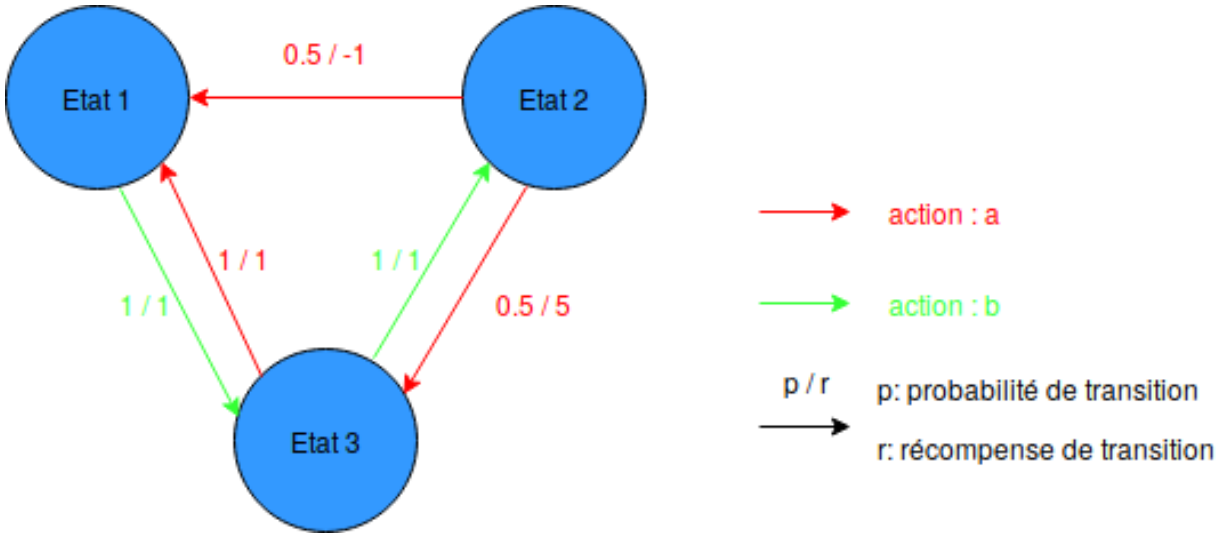


FIGURE 3.7 – *Schéma représentant les transitions entre états dans un MDP*

3.6.2 État du gestionnaire de dialogue

L'état d'un système de dialogue est une représentation sémantique qui contient des informations sur le but final de l'utilisateur ainsi que l'historique de la conversation. La représentation souvent utilisée dans les systèmes de dialogue est celle du cadre sémantique [46]. Cette structure contient des emplacements à remplir sur un domaine donné, la figure 3.8 illustre un exemple de cadre sémantique.

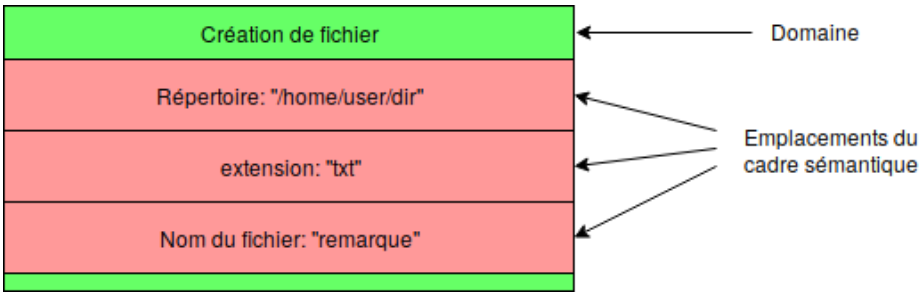


FIGURE 3.8 – *Schéma représentant un cadre sémantique avec comme domaine : création de fichier*

À l'arriver d'une nouvelle information, un module dédié met à jour l'état du gestionnaire du dialogue. Comme l'action du système de dialogue est décidée à partir de son état, cette tâche est donc essentiel au bon fonctionnement du système. Plusieurs méthodes ont été donc proposé pour gérer le suivi de l'état du gestionnaire de dialogue.

3.6.2.1 Suivi de l'état avec une base de règles

La méthode traditionnelle utilisée est d'écrire manuellement les règles à suivre lors de l'arrivé d'une nouvelle information pour mettre à jour l'état[47]. Cependant, les bases de règles sont très susceptibles à faire des erreurs[46] comme ils sont moins robuste face aux incertitudes.

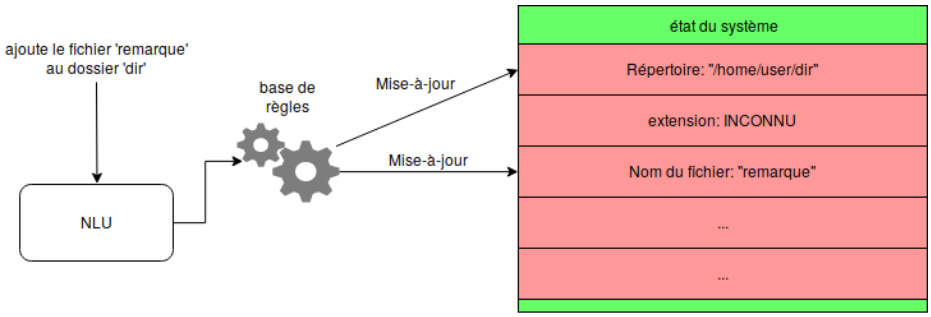


FIGURE 3.9 – *Schéma représentant la mise-à-jour de l'état par un système basé règles*

3.6.2.2 Suivi de l'état avec des méthodes statistiques

Le suivi dans ce cas se fait en gardant une distribution de probabilités sur l'état du système. D'où, l'utilisation des processus de décision markovien partiellement observé (POMDP)[48] qu'on introduira par la suite. Dans ce cas, le système garde une distribution de probabilités sur les valeurs possibles des différents emplacements du cadre sémantique.

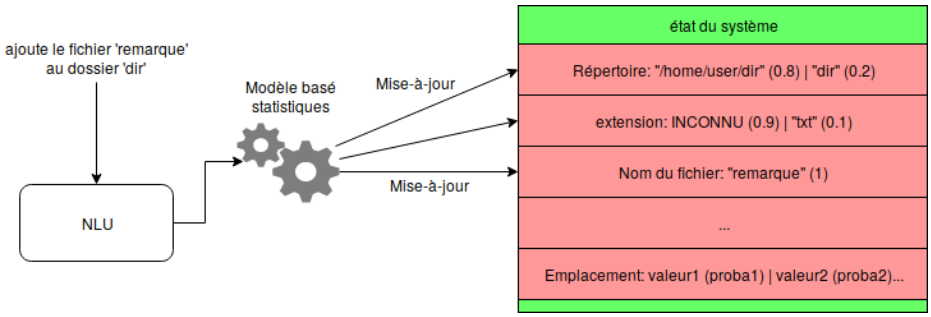


FIGURE 3.10 – *Schéma représentant la mise-à-jour de l'état par un système basé statistiques*

3.6.2.3 Processus de décision markovien partiellement observable (POMDP)

Comme dans les processus de décision markovien, un POMDP[49] passe d'un état à un autre en prenant une des actions possibles. Cependant, ce dernier ne connait pas l'état exacte dans lequel il se trouve à un instant t. Il reçoit par contre une observation, dans notre cas c'est l'action de l'utilisateur, à partir de laquelle il peut estimer une distribution de probabilités sur l'état actuel. Pour résumer cela, un POMDP est un 6-tuple (S,A,P,R,M,O) :

- Les 4 premiers composants sont les même que celui d'un MDP 3.6.1.
- M : l'ensemble des observation.
- O : distribution de probabilités sur les observations o sachant en connaissant l'état et l'action prise pour y arriver. O(o|s,a) est la probabilité d'observer o sachant qu'on se trouve à l'état s et qu'on a pris l'action a pour y arriver.

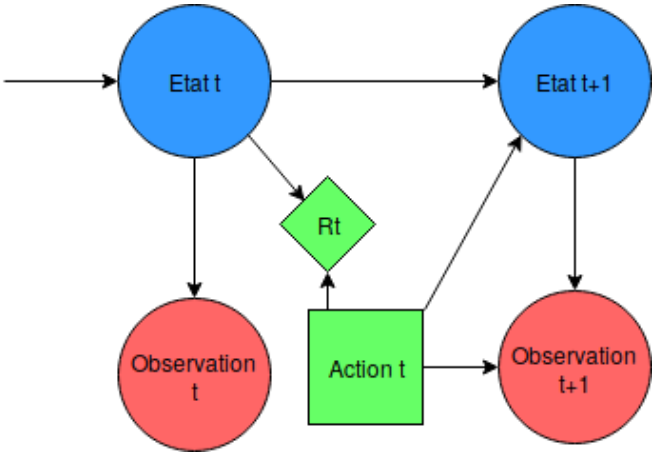


FIGURE 3.11 – *Diagramme d'influence dans un POMDP*

3.6.2.4 Suivi de l'état avec réseaux de neurones profonds

Récemment, des approches utilisant les réseaux de neurones profonds(refpart2) ont fait leurs apparitions. En effet, l'utilisation des architectures profondes permet de capter des relations complexes entre les caractéristiques d'un dialogue et ainsi mieux estimer l'état du système. Le réseau de neurones estime les probabilités de toutes les valeurs possibles d'un emplacement du cadre sémantique[50]. En conséquence, il peut être utilisé comme modèle de suivi d'état pour un processus partiellement observable.

3.6.3 Politique de gestion de dialogue

La première partie était dédiée au module qui suit l'état du système de dialogue. Dans cette partie, Nous allons présenter des approches proposées afin d'arriver au but du MDP, c'est à dire quelles actions prendre pour maximiser la somme des récompenses obtenus.

3.6.3.1 Gestion de dialogue avec une base de règles

Les premières approches utilisaient des systèmes de règles destiné à un domaine bien spécifique. Elles étaient déployés dans plusieurs domaines d'application pour sa simplicité. Cependant, le travail manuel nécessaire reste difficile à faire, et, généralement, n'aboutit pas à des résultats flexibles qui peuvent suivre le flux du dialogue convenablement[51].

3.6.4 Gestion de dialogue par apprentissage

La résolution d'un MDP revient à trouver une estimation de la fonction de récompense afin de pouvoir choisir la meilleure action. La majorité des approches récentes utilise l'apprentissage par renforcement pour but d'estimer la récompense obtenue par une action et un état donnés. Cette préférence par rapport aux approches supervisées revient à la difficulté de produire des corpus de dialogues[52], encore moins des corpus annotés avec les récompenses à chaque transition. Néanmoins, il existe des approches de bout en bout qui exploite des architectures avec réseaux de neurones profonds et traite le problème comme Seq2Seq(part2) afin de produire directement une sortie à partir des informations reçues par l'utilisateur[53][54].

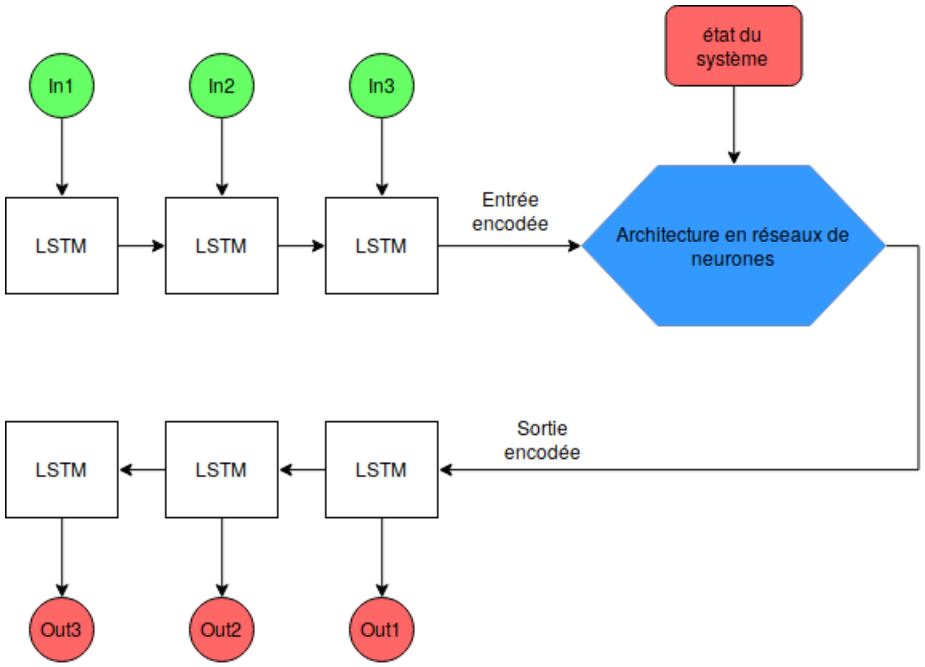


FIGURE 3.12 – *Schéma de gestion de dialogue de bout en bout avec architecture Seq2Seq*

3.6.5 Apprentissage par renforcement

L'apprentissage par renforcement est une approche qui a pour but d'estimer une politique d'actions à prendre dans un environnement pour maximiser une mesure d'évaluation qui est sous forme de récompenses obtenues après chaque action[55]. L'environnement est souvent modéliser comme un MDP, ou éventuellement POMDP. L'agent d'apprentissage passe donc d'un état à un autre en prenant des actions dans cet environnement. L'apprentissage se fait dans ce cas en apprenant par l'expérience de l'agent, à savoir les récompenses obtenues par les actions prises et de quels états elles étaient prises. Il peut ainsi estimer la fonction de récompense pour pouvoir faire le choix d'actions optimales.

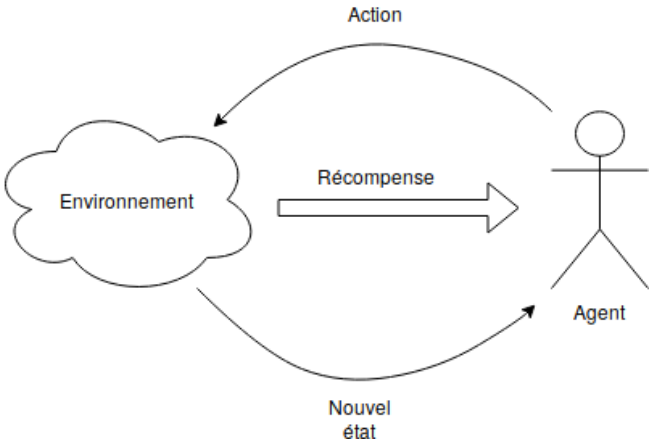


FIGURE 3.13 – *Schéma d'interaction agent-environnement dans l'apprentissage par renforcement*

Il existe plusieurs méthodes que l'agent peut prendre afin qu'il estime la fonction de récompense [56], notamment Deep Q Learning (DQN) [57] qui utilise les réseaux de neurones profonds pour trouver une approximation à cette fonction. Dans cette approche, à chaque fois que l'agent prenne une action, il compare la récompense reçue avec celle estimée par le réseau de neurones, il applique ensuite l'algorithme de rétro-propagation pour corriger les erreurs du réseau.

Le système de dialogue est modélisé par un MDP. Seulement, ce dernier inclut l'utilisateur comme partie de l'environnement. En conséquence, pour appliquer l'apprentissage par renforcement, il est nécessaire qu'un utilisateur communique avec le système pour qu'il apprenne. D'où, la nécessité d'utiliser les simulateurs d'utilisateur.

3.6.6 Simulateur d'utilisateur

Un simulateur d'utilisateur est un programme qui se comporte comme un humain interagissant avec un système de dialogue. Celui-là doit pouvoir estimer sa satisfaction après une interaction. en d'autres termes, il doit comporter d'une fonction de récompense. Il existe plusieurs méthodes pour créer un simulateur d'utilisateur :

- Les simulateurs d'utilisateur basé règles : dans ce cas une liste de règles est écrite manuellement que le simulateur doit suivre pour communiquer avec le système[58].
- Les simulateurs d'utilisateur basé n-grammes : ils traitent un dialogue comme une séquence d'actions. Ils prennent les n-1 actions pour estimer l'action la plus probable que peut prendre le simulateur à partir des statistiques tirées d'un corpus de dialogues[59].
- Les simulateurs d'utilisateur basé HMM : les états du modèle sont les états du système, les observations sont ses actions. Ainsi un HMM peut estimer l'état le plus probable du système pour prendre une action. Il existe d'autres variantes, IHMM et IOHMM, qui incluent les probabilités conditionnelles des actions utilisateur avec l'état système et l'action système directement dans le modèle HMM[60].
- Les simulateurs d'utilisateur avec apprentissage par renforcement : Comme le gestionnaire de dialogue, le simulateur apprend par renforcement au même temps. Dans ce cas la fonction de récompense pour les deux agents peut être apprise à partir des dialogues humain-humain[61].

3.7 Génération du langage naturel (NLG)

Le domaine de la génération automatique du langage naturel est l'un des domaines dont les bordures sont difficiles à définir (Evans et al., 2002). il est vrai que la sortie d'un tel système est clairement du texte. Cependant, l'ambiguïté se trouve dans ses l'entrées, c'est à dire, sur quoi se basera le système pour générer le texte. D'après (Reiter & Dale, 1997)[62] la génération du langage naturel est décrite comme étant le sous domaine de l'intelligence artificielle qui traite la construction des systèmes de génération de texte à partir d'une représentation non-linguistique de l'information, celle-ci peut être une représentation sémantique, des données numériques, une base de connaissances ou même des données visuelles (images ou vidéos). Ceci dit, d'autres travaux ,comme Labbé & Portet (2012)[63], utilisent les même techniques pour des entrées linguistique. Enfin la génération du langage naturel peut être très proche de la gestion de dialogue[64], en effet, le texte généré doit prendre en compte l'historique de la conversation et le contexte de l'utilisateur. Il existe six tâches trouvées fréquemment dans les systèmes de génération de texte [62].

3.7.1 Détermination du contenu

Cette partie consiste à sélectionner les informations de l'entrée dont le système veut transmettre le contenu sous forme de texte naturel à l'utilisateur. En effet, les données en entrée peuvent contenir plus d'informations que ce que l'on désire communiquer[65], de plus, cette information peut aussi dépendre de l'utilisateur et de ses connaissances[64]. Ce qui requiert de mettre au point un système qui détecte les informations pertinentes à l'utilisateur.

3.7.2 Structuration de texte

Après la détermination du contenu, le système doit ordonner les information à transmettre. Ceci dépend grandement du domaine d'application qui peut exiger des contraintes d'ordre temporelle ou de préférence par importance des idées. Les informations à transmettre en elles-mêmes sont souvent reliées par sens ce qui implique une certaine structuration de texte à respecter.

3.7.3 Agrégation de phrases

Certaines informations peuvent être transmises dans une même phrase. Cette partie introduit des notions de la linguistique afin que le texte généré soit plus lisible et éviter les répétition. Un exemple de cela peut être la description de la météo à Alger au cours de la matinée :

- Il va faire 16° à Alger à 7h.
- Il va faire 17° à Alger à 8h.
- Il va faire 18° à Alger à 9h.
- Il va faire 18° à Alger à 10h.

Ceci peut être agrégé en un texte plus compacte : "La température moyenne à Alger sera de 17° entre 7h et 10h."

3.7.4 Lexicalisation

Le système choisit les mots et les expressions à utiliser pour communiquer le contenu des phrases sélectionner. La difficulté de cette tâche revient à l'existence de plusieurs manière d'exprimer la même idée. Cependant, certains mots ou expressions sont plus appropriés en certaines situations que d'autres. En effet, "inscrire un but" est une façon inadéquate d'exprimer un but contre son camp[66].

3.7.5 Génération d'expressions référentielles (REG)

Cette partie du système se focalise dans la génération d'expressions référentielles qui peuvent être entre autres : noms propres, groupes nominaux ou pronoms et ceci a pour but d'identifier les entités du domaine. Cette tâche semble être très proche de sa prédécesseur ; elle s'avère néanmoins plus délicate dû à la difficulté de confier suffisamment d'information sur l'entité afin de la différencier des autres[62]. Le système doit faire un choix de l'expression référentielle en se basant sur plusieurs facteurs, par exemple "Mohammed", "Le professeur" ou "Il" font référence à la même personne. Cependant, le choix entre eux dépendrait de si l'entité a été mentionnée auparavant et des détails l'accompagnant par exemple.

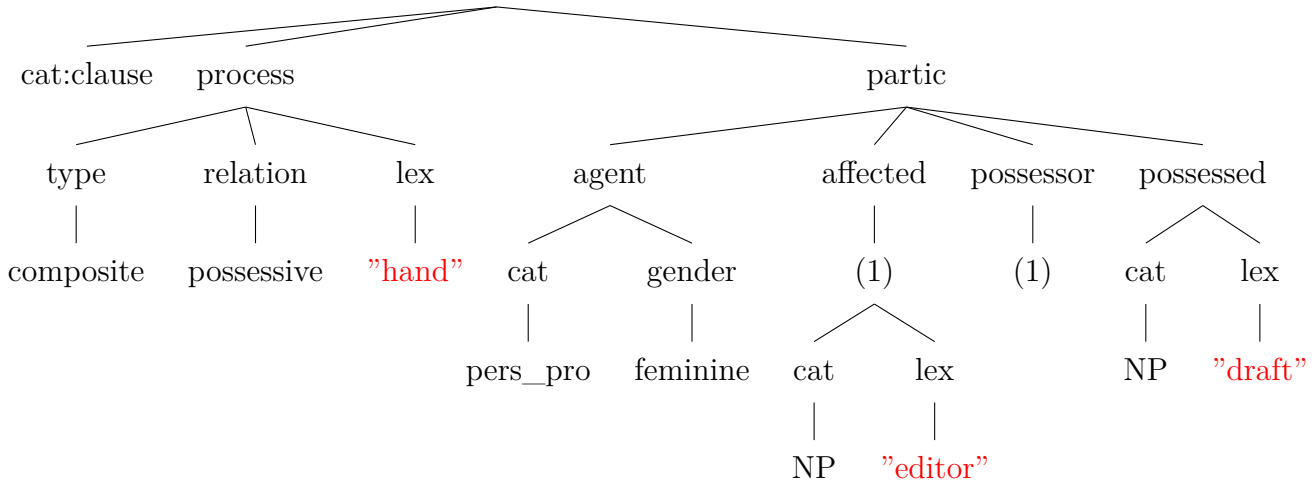
3.7.6 Réalisation linguistique

La dernière tâche consiste à combiner les mots et expressions sélectionnés pour construire une phrase linguistiquement correcte. Ceci requiert l'utilisation des bonnes formes morphologiques des mots, les ordonner et éventuellement l'addition de certains mots du langage afin de réaliser une structure de phrase grammaticalement et sémantiquement correcte. Plusieurs méthodes ont été proposées, principalement les méthodes basées sur des règles manuellement construites (modèles de phrases, systèmes basés grammaires) ou des approches statistiques [66].

Modèles de phrases : La réalisation se fait en utilisant des modèles de phrases prédéfinis. Il suffit de remplacer des espaces réservés par certaines entrées du système. Par exemple, une application dans un contexte météorologique pourrait utiliser le modèle suivant : "la température à

[ville] atteint [température]° le [date]".
Cette méthode est utilisée lorsque les variations des sorties de l'application sont minimales. Son utilisation a l'avantage et l'inconvénient d'être rigide. D'un coté il est facile de contrôler la qualité des sorties syntaxiquement et sémantiquement tout en utilisant des règles de remplissage complexe [67]. Cependant, lorsque le domaine d'application présente beaucoup d'incertitude, cette méthode exige un travail manuel énorme, voire impossible à faire, pour réaliser une tâche pareille. Bien que certains travaux ont essayer de faire un apprentissage de modèles de phrases à partir d'un corpus[68] cette méthode reste inefficace lorsqu'il s'agit d'application qui nécessite un grand nombre de variations linguistiques.

Systèmes basés grammaire : La réalisation peut se faire en suivant une grammaire du langage. Celle-si contient les règles morphologiques et de structures de la langues, notamment la grammaire systémique fonctionnelle (SFG)[69] a été largement utilisé comme dans NIGEL[70] ou KPML[71]. L'exploitation des grammaires dans la génération du texte nécessite généralement des entrées détaillées. En plus des composantes du lexique sélectionnées, des descriptions de leurs rôles ainsi que leurs fonctions grammaticales sont souvent exigées. Un exemple d'entrée d'un système basé grammaire est celui de SURGE[72] :



Qui génère la phrase : "She hands the draft to the editor".

Comme les modèles de phrases, les systèmes basés grammaire nécessite un énorme travail manuel. En particulier, il est difficile de prendre en compte le contexte en définissant les règles de choix entre les variantes possibles du texte résultat à partir des entrées[66].

Approches statistiques : Il existe plusieurs méthodes basées sur des statistiques pour la tâche de réalisation. Certains se basent sur des grammaires probabilistes, cette dernière a l'avantage de minimiser le travail manuel tout en couvrant plus de cas de réalisation. Il existe principalement deux approches l'utilisant[66] :

- La première se base sur une petite grammaire qui génère plusieurs alternatives qui sont ensuite ordonnés selon un modèle statistique basé sur un corpus pour sélectionner la phrase la plus probable (par exemple Langkilde-Geary (2000)[73]).

- La deuxième méthode utilise les informations statistiques directement au niveau de la génération pour produire la solution optimale (exemple : Belz (2008)[74]).

Dans les deux méthodes sus-citées la grammaire de base peut être manuellement faite, dans ce cas, les informations statistiques aideront à la détermination de la solution optimale, ou elle peut être extraite à partir des données, comme l'utilisation des Treebanks [2] pour déduire les règles de grammaire[75].

D'autres approches statistiques n'utilisent pas des grammaires mais se basent sur des classificateurs. Ces derniers peuvent être cascadés de telle sorte à décider quel constituant utiliser dans quelle position ainsi que les modifications nécessaires pour générer un texte correcte. À noter qu'une telle approche, ne nécessitant pas l'utilisation de grammaire, utilise des entrées plus abstraites et moins détaillées linguistiquement. À voire même la possibilité de s'étendre aux autres tâches de NLG, c'est à dire un système qui accomplit plusieurs tâches de NLG en parallèle en utilisant les entrées initiales. Dans la suite de ce travail nous allons présenter certains de ces systèmes qui sont plus utilisés récemment.

3.7.7 Systèmes basés encodeur-décodeur

Une architecture souvent utilisée dans le traitement du langage naturel est l'encodeur-décodeur (part2). En particulier, son utilisation dans les tâches seq2seq (part2) ce qui permet de mettre en correspondance une séquence de taille variable en entrée avec une autre séquence en sortie. Les modèles seq2seq peuvent être adapter pour convertir une représentation abstraite de l'information en langage naturel[76].

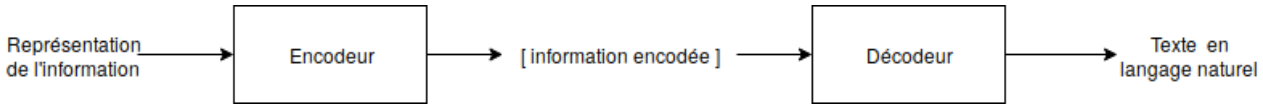


FIGURE 3.14 – *Schéma d'une architecture encodeur-décodeur pour NLG*

Beaucoup d'approches de génération de langage naturel en gestion de dialogue utilise des encodeur-décodeur. Wen et al. (2015)[77] utilise par exemple des LSTMs(part2) sémantiquement conditionnés ; il ajoute aux LSTMs classiques une couche contenant des informations sur l'action prise par le gestionnaire de dialogue pour assurer que la génération représente le sens désiré. D'autres travaux utilisent des réseaux de neurones récurrent (part2) pour encoder l'état du gestionnaire de dialogue et l'entrée reçu suivis par un décodeur pour générer le texte de la réponse[78][54][79].

2. un Treebank est un texte analysé qui contient des informations syntaxiques ou sémantiques sur les structures de phrases

Table des figures

Bibliographie

[1] D. A. Norman, *The design of everyday things*. New York : Basic Books, 2002.

[2] R. Knote, A. Janson, L. Eigenbrod, and M. Söllner, "The what and how of smart personal assistants : Principles and application domains for is research," in *Multikonferenz Wirtschaftsinformatik (MKWI)*, 2018.

[3] H.Gellersen, *Handheld and Ubiquitous Computing : First International Symposium, HUC'99, Karlsruhe, Germany, September 27-29, 1999, Proceedings (Lecture Notes in Computer Science)*. Springer, 1999.

[4] S. J. Russell and P. Norvig, *Artificial Intelligence : A Modern Approach*. Pearson Education, 2 ed., 2003.

[5] T. Dingler, "Cognition-aware systems as mobile personal assistants," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct - UbiComp '16*, ACM Press, 2016.

[6] E. Luger and A. Sellen, ""like having a really bad PA"," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, ACM Press, 2016.

[7] R. Trappl, ed., *Your Virtual Butler*. Springer Berlin Heidelberg, 2013.

[8] A. Purington, J. G. Taft, S. Sannon, N. N. Bazarova, and S. H. Taylor, ""alexa is my new BFF"," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '17*, ACM Press, 2017.

[9] B. R. Cowan, N. Pantidi, D. Coyle, K. Morrissey, P. Clarke, S. Al-Shehri, D. Earley, and N. Bandeira, ""what can i help you with ?"," in *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '17*, ACM Press, 2017.

[10] J. Imtiaz, N. Koch, H. Flatt, J. Jasperneite, M. Voit, and F. van de Camp, "A flexible context-aware assistance system for industrial applications using camera based localization," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, IEEE, sep 2014.

[11] A. Janson and M. T. de Gafenco, "Engaging the appropriation of technology-mediated learning services - a theory-driven design approach," in *ECIS*, 2015.

[12] "Apple shares examples of siri's third-party app integration on ios 10." `https://www.idownloadblog.com/2016/09/01/apple-siri-ios-10-app-integration/`. (Accessed on 10/29/2018).

[13] "macos sierra review : Hey siri, where did my files go ? - six colors." `https://sixcolors.com/post/2016/09/sierra-review/`, 2016. (Accessed on 10/29/2018).

[14] T. M Mitchell, "The discipline of machine learning," 01 2006.

[15] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Machine learning : A review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, 11 2006.

[16] H. Barlow, "Unsupervised learning," *Neural Computation*, vol. 1, no. 3, pp. 295–311, 1989.

[17] B. Sonali, "Research paper on basic of artificial neural network," p. 1, 2014.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[19] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, May 2013.

[20] D. Yu and L. Deng, *Automatic Speech Recognition*. Springer London, 2015.

[21] M. Velay and F. Daniel, "Seq2seq and multi-task learning for joint intent and content extraction for domain specific interpreters," *CoRR*, vol. abs/1808.00423, pp. 3–4, 2018.

[22] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *CoRR*, vol. abs/1605.05101, 2016.

[23] F. Rosenblatt, *Perceptron simulation experiments (Project Para)*. 1959.

[24] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, pp. 183–197, jul 1991.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[26] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.

[27] C. Olah, "Understanding lstm networks – colah's blog." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, Août 2015. (Accessed on 02/19/2019).

[28] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, pp. 107–116, Apr. 1998.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[30] D. Jurafsky and J. H. Martin, *Speech and Language Processing, 2nd Edition*. Prentice Hall, 2008.

[31] Z. Ghahramani, "Hidden markov models," ch. An Introduction to Hidden Markov Models and Bayesian Networks, pp. 9–42, River Edge, NJ, USA : World Scientific Publishing Co., Inc., 2002.

[32] J. G. Kemeny and J. Laurie Snell, "Markov processes in learning theory," *Psychometrika*, vol. 22, pp. 221–230, Sep 1957.

[33] R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, pp. 4–16, 1986.

[34] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[35] J. Bloit and X. Rodet, "Short-time viterbi for online hmm decoding : Evaluation on a real-time phone recognition task," pp. 2121 – 2124, 05 2008.

[36] F. Al-Anzi and D. AbuZeina, "Literature survey of arabic speech recognition," pp. 1–6, 03 2018.

[37] S. Narang and M. D. Gupta, "Speech feature extraction techniques : A review," 2015.

[38] P. M. Chauhan and N. P. Desai, "Mel frequency cepstral coefficients (mfcc) based speaker identification in noisy environment using wiener filter," in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, pp. 1–5, March 2014.

[39] D. D. O'Shaughnessy, "Linear predictive coding," *IEEE Potentials*, vol. 7, pp. 29–32, 1988.

[40] H. Rahali, Z. Hajaiej, and N. Ellouze, "Robust features for speech recognition using temporal filtering technique in the presence of impulsive noise," *International Journal of Image, Graphics and Signal Processing*, vol. 6, pp. 17–24, 10 2014.

[41] W. Ghai and N. Singh, "Literature review on automatic speech recognition," *International Journal of Computer Applications*, vol. 41, pp. 42–50, March 2012. Full text available.

[42] l. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications : An overview," pp. 8599–8603, 10 2013.

[43] R. Glass and M. Kyle Mccandless, "Automatic acquisition of language models for speech recognition," 10 1994.

[44] B. Roark, M. Saraclar, and M. Collins, "Discriminative n-gram language modeling," *Comput. Speech Lang.*, vol. 21, pp. 373–392, Apr. 2007.

[45] t. . A. M. D. P. Richard Bellman" *Indiana Univ. Math. J.", fjournal = "Indiana University Mathematics Journal*, vol. 6, pp. 679–684, 1957.

[46] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems : Recent advances and new frontiers," *SIGKDD Explor. Newsl.*, vol. 19, pp. 25–35, Nov. 2017.

[47] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, "A form-based dialogue manager for spoken language applications," vol. 2, pp. 701 – 704, 11 1996.

[48] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The hidden information state model : A practical framework for pomdp-based spoken dialogue management," *Comput. Speech Lang.*, vol. 24, pp. 150–174, Apr. 2010.

[49] K. J. Åström, "Optimal control of Markov Processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, pp. 174–205, January 1965.

[50] M. Henderson, B. Thomson, and S. J. Young, "Deep neural network approach for the dialog state tracking challenge," in *SIGDIAL Conference*, pp. 467–471, 2013.

[51] C. Lee, S. Jung, K. Kim, D. Lee, and G. G. Lee, "Recent approaches to dialog management for spoken dialog systems," *JCSE*, vol. 4, pp. 1–22, 2010.

[52] J. Henderson, O. Lemon, and K. Georgila, "Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets," *Comput. Linguist.*, vol. 34, pp. 487–511, Dec. 2008.

[53] T.-H. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. hao Su, S. Ultes, D. Vandyke, and S. J. Young, "A network-based end-to-end trainable task-oriented dialogue system," in *EACL*, pp. 438–449, 2017.

[54] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, pp. 3776–3783, AAAI Press, 2016.

[55] G. Weisz, P. Budzianowski, P. hao Su, and M. Gax0161ix0107, "Sample efficient deep reinforcement learning for dialogue systems with large action spaces," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, pp. 2083–2097, 2018.

[56] D. Bertsekas, *Dynamic Programming & Optimal Control, Vol II : Approximate Dynamic Programming*. Athena Scientific, 01 2012.

[57] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[58] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. J. Young, "Agenda-based user simulation for bootstrapping a pomdp dialogue system," in *HLT-NAACL*, pp. 149–152, 2007.

[59] K. Georgila, J. Henderson, and O. Lemon, "Learning user simulations for information state update dialogue systems," in *INTERSPEECH*, pp. 893–896, 2005.

[60] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Human-computer dialogue simulation using hidden markov models," *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pp. 290–295, 2005.

[61] S. Chandramohan, M. Geist, F. Lefèvre, and O. Pietquin, "User simulation in dialogue systems using inverse reinforcement learning," in *INTERSPEECH*, p. 10251028, 2011.

[62] E. Reiter and R. Dale, "Building applied natural language generation systems," *Natural Language Engineering*, vol. 3, pp. 57–87, Mar. 1997.

[63] C. Labbé and F. Portet, "Towards an abstractive opinion summarisation of multiple reviews in the tourism domain," *CEUR Workshop Proceedings*, vol. 917, pp. 87–94, 01 2012.

[64] N. Dethlefs, "Context-sensitive natural language generation : From knowledge-driven to data-driven techniques," *Language and Linguistics Compass*, vol. 8, pp. 99–115, 03 2014.

[65] J. Yu, E. Reiter, J. Hunter, and C. Mellish, "Choosing the content of textual summaries of large time-series data sets," *Natural Language Engineering*, vol. 13, pp. 25–49, Mar. 2007.

[66] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation : Core tasks, applications and evaluation," *J. Artif. Int. Res.*, vol. 61, pp. 65–170, Jan. 2018.

[67] M. Theune, E. Klabbers, J. R. De Pijper, E. Krahmer, and J. Odijk, "From data to speech : A general approach," *Natural Language Engineering*, vol. 7, pp. 47–86, Mar. 2001.

[68] G. Angeli, C. D. Manning, and D. Jurafsky, "Parsing time : Learning to interpret time expressions," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, NAACL HLT '12, (Stroudsburg, PA, USA), pp. 446–455, Association for Computational Linguistics, 2012.

[69] M. A. K. Halliday and C. M. I. M. Matthiessen, *Introduction to Functional Grammar*. London : Hodder Arnold, 3 ed., 2004.

[70] W. C. Mann and C. M. I. M. Matthiessen, "Nigel : A systemic grammar for text generation.," 1983.

[71] J. A. Bateman, "Enabling technology for multilingual natural language generation : The kpml development environment," *Nat. Lang. Eng.*, vol. 3, pp. 15–55, Mar. 1997.

[72] M. Elhadad and J. Robin, "An overview of surge : a reusable comprehensive syntactic realization component," in *International Natural Language Generation Workshop*, pp. 1–4, 1996.

[73] I. Langkilde-Geary, "Forest-based statistical sentence generation," in *ANLP*, pp. 170–177, 2000.

[74] A. Belz, "Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models," *Nat. Lang. Eng.*, vol. 14, pp. 431–455, Oct. 2008.

[75] D. Espinosa, M. White, and D. Mehay, "Hypertagging : Supertagging for surface realization with ccg," in *ACL*, pp. 183–191, 2008.

[76] T. C. Ferreira, I. Calixto, S. Wubben, and E. Krahmer, "Linguistic realisation as machine translation : Comparing different mt models for amr-to-text generation," in *INLG*, pp. 1–10, 2017.

[77] T.-H. Wen, M. Gasic, N. Mrksic, P. hao Su, D. Vandyke, and S. J. Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," in *EMNLP*, pp. 1711–1721, 2015.

[78] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and W. B. Dolan, "A neural network approach to context-sensitive generation of conversational responses," in *HLT-NAACL*, pp. 196–205, 2015.

[79] R. Goyal, M. Dymetman, and E. Gaussier, "Natural language generation through character-based rnns with finite-state prior knowledge," in *COLING*, pp. 1083–1092, 2016.