

Chapitre 1

Composants de base d'un assistant personnel

1.1 Introduction

Dans 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 base. Nous ne traiterons que celles qui ont été les plus utilisées dans les travaux que nous avons examinés. La suite du chapitre sera organisée en sections qui décriront 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.

1.2 Schéma global d'un SPA

Durant nos lectures des différents travaux sur le domaine, nous avons pu dresser un schéma assez général qui englobe les principaux modules d'un SPA :

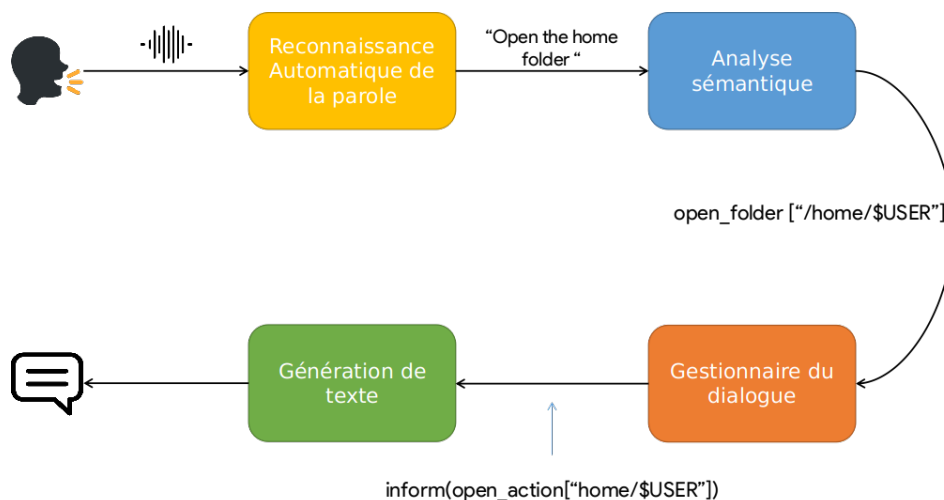


Figure 1.1 – Schéma abstrait d'un SPA Milhorat et al. (2014)

Le processus peut être résumé en des étapes cruciales (qui seront détaillées dans les sections suivantes) :

- L'utilisateur énonce une requête en utilisant sa voix. Le signal est ensuite transformé en sa version textuelle.
- La requête étant sous un format textuel brut ne peut pas être interprétée ou comprise par la machine. Une représentation sémantique est générée qui regroupera les principales informations contenues dans la requête, à savoir l'intention de l'utilisateur et ses arguments.
- Au fur et à mesure que l'utilisateur énonce des requêtes, le système doit pouvoir être capable d'utiliser le contexte du dialogue pour interagir avec ce dernier afin d'atteindre le but final du dialogue (exécuter une commande système, trouver des informations internet ou sur la machine locale ...). Le gestionnaire du dialogue communique avec son environnement en utilisant la même représentation sémantique produite par l'analyse précédente.
- Puisqu'il est préférable de cacher à l'utilisateur tout comportement interne au système, il serait préférable de transformer la trame sémantique (voir la figure 1.9) en texte naturel pour l'afficher en sortie.

Il est à noter que chacun des modules seront indépendants pour ce qu'il est de leur fonctionnement interne. Ainsi, aucun n'assumera un fonctionnement arbitraire des autres modules. Seul le format des informations qui circulent entre eux devra être établi au préalable pour un fonctionnement durable et robuste au changement.

Pour ce qui est de la suite du chapitre, nous allons présenter les différents modules ainsi que les techniques et architectures qui sont considérées comme étant l'état de l'art du domaine.

1.3 Notions et aspects théoriques

Dans cette section, nous présenterons différentes notions liées au domaine de l'apprentissage automatique, pour ensuite les citer dans les modules qui les utilise.

1.3.1 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 M Mitchell (2006), l'auteur définit 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?" M Mitchell (2006)

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 M Mitchell (2006), 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 diffèrent principalement par la manière dont le système apprend, du type de données sur lesquelles 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 externe. Les données doivent être séparées en deux parties (ensemble d’apprentissage et de test). Un *label* (ou classe) est associé à chaque instance pour permettre à l’algorithme de calculer un certain taux d’erreur qu’il essaiera de minimiser au fur et à mesure qu’une nouvelle instance lui est présentée Kotsiantis et al. (2006). Idéalement, le système pourra apprendre une certaine fonction $\hat{f} : X \rightarrow Y$ qui liera les entrées X aux sorties Y en minimisant l’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 Barlow (1989).
- **Apprentissage par renforcement** : Cette dernière catégorie regroupe des algorithmes qui apprennent par un système de *trial and error* (Essais et erreur). C’est à dire qu’en interagissant avec l’environnement, l’agent apprenant (l’algorithme) apprendra à accomplir une tâche précise en exécutant des actions qui modifieront son état et celui de son environnement (voir la section 1.6.3.3)

Dans les sections suivantes, nous nous intéresserons à quelques type d’algorithme d’apprentissage automatique.

1.3.2 Réseaux de neurones artificiels

Un réseau de neurones artificiels est une structure d’appariement non-linéaire entre un ensemble de caractéristiques en entrée et sortie 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 petites unités de calcul interconnectées appelées neurones, qui traitent l’information d’une manière parallèle dans le but de résoudre un problème bien spécifique Sonali (2014). Ils ont notamment connu un très grand succès lors des dernières années dans différents domaines comme la reconnaissance d’images Szegedy et al. (2015), la reconnaissance automatique de la parole Graves et al. (2013); Yu and Deng (2015) ou encore la classification de textes Velay and Daniel (2018); Liu et al. (2016).

Il existe une variété d’architectures de réseaux de neurones. Nous traiterons dans cette section trois d’entre elles :

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

1.3.2.1 Réseaux de neurones multicouches denses

La forme la plus basique que peut avoir un réseau de neurones est celle d'un réseau multicouches comme montré dans la figure 1.2. Elle se compose de trois parties :

- **Une couche d'entrée** : cette couche est composée d'un ensemble de neurones ou *perceptrons* Rosenblatt (1959). Elle représente l'information en entrée codifié en un vecteur numérique χ
- **Une ou plusieurs couches cachées** : le cur du réseau, c'est une succession de couche de neurones où 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 envoie 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 :

$$y = f_{\Phi}(\chi) \quad (1.1)$$

Le réseau calcule 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 parcours des données d'apprentissage Murtagh (1991)

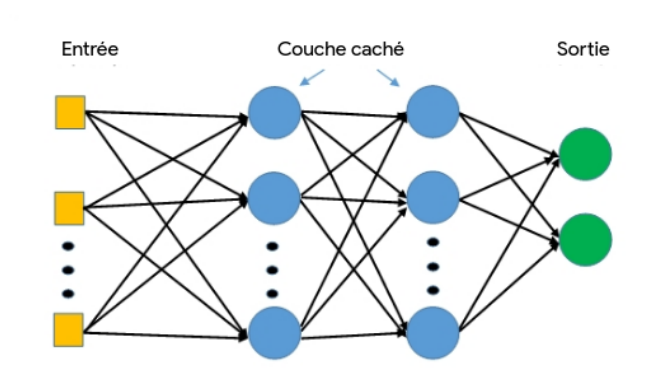


Figure 1.2 – Architecture basique d'un réseaux de neurones multicouches

1.3.2.2 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 computationnelles d'antan se sont vu vite brisées avec le développement de processeurs plus puissants. De plus, avec l'explosion des données sur internet, toutes les conditions nécessaires étaient

réunies pour la mise en place d'architectures plus complexes. 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 convolutif et récurrents LeCun et al. (2015)

1.3.2.3 Réseaux de neurones récurrents

Un aspect que les réseaux de neurones (profonds ou pas) ne peuvent capturer est la notion de séquentialité. En effet beaucoup de problèmes qui sont de nature séquentielle ne peuvent être modélisés 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 Lipton (2015).

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

$$h^{(t)} = \tanh(W^{hx} \times x^{(t)} + W^{hh} \times h^{(t-1)} + b_h) \quad (1.2)$$

$$\hat{y}^{(t)} = \text{softmax}(W^{hy} \times h^{(t)} + b_y) \quad (1.3)$$

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 Lipton (2015)

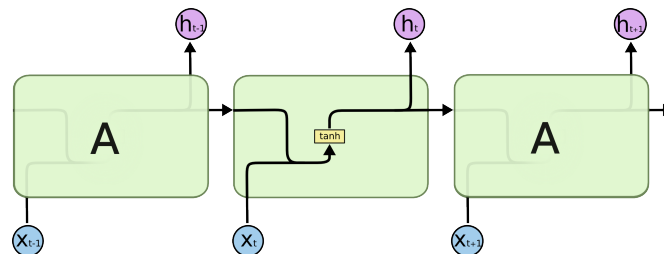


Figure 1.3 – Architecture interne d'un réseau de neurones récurrent à un instant t Olah (2015)

Un des principaux problèmes que rencontrent les RNNs est celui du *Vanishing gradient* traduit par le *Problème de disparition du gradient* Hochreiter (1998). Les relations à long terme entre 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 été introduites.

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

Introduite en 1997 par *Sepp Hochreiter* et al. dans Hochreiter and Schmidhuber (1997), cette architecture de réseaux de neurones récurrents est dotée 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, les composantes d'une cellule LSTM (voir 1.4) sont détaillées dans ce qui suit :

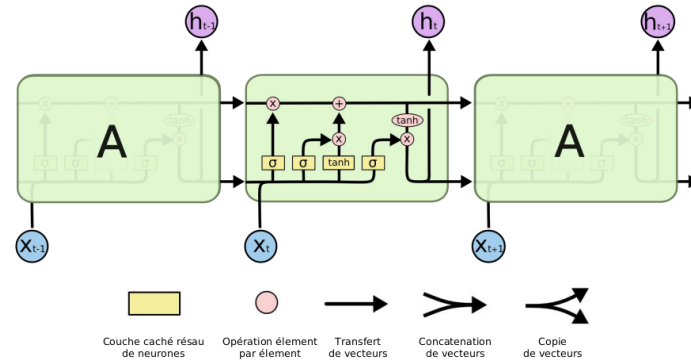


Figure 1.4 – Architecture interne d'une cellule mémoire dans un réseau LSTM Olah (2015)

- **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 pondérer les composants 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 Hochreiter and Schmidhuber (1997).

$$\begin{aligned} i_t &= \sigma(W_i \times [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \times [h_{t-1}, x_t] + b_C) \end{aligned} \quad (1.4)$$

- **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 Hochreiter and Schmidhuber (1997) ; Lipton (2015).

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \quad (1.5)$$

- **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 filtrées par les portes f_c et i_c Hochreiter and Schmidhuber (1997).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (1.6)$$

- **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 donnée en entrée et

l'état du réseau précédent pour ne laisser passer que certaines informations Olah (2015); Hochreiter and Schmidhuber (1997); Lipton (2015).

$$\begin{aligned} o_t &= \sigma(W_o \times [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \quad (1.7)$$

1.3.3 Modèle de Markov caché (Hidden Markov Models HMM)

Informellement, un modèle de Markov caché (HMM) est un outil 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 son état à l'instant $t - 1$. Il est alors dit que ce processus est Markovien ou qu'il satisfait la propriété de Markov Ghahramani (2002); Kemeny and Laurie Snell (1957).

D'un façon plus formelle, un HMM est un 5-tuples $\langle (S, V, \Pi, A, B) \rangle$ Rabiner and Juang (1986) où :

- $S = \{s_1 \dots s_N\}$ est l'ensemble fini des N états du processus sous-jacent.
- $V = \{v_1 \dots v_M\}$ est l'ensemble des M symboles qui constitue 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 évidemment $\sum_{i=1}^N \pi_i = 1$
- $A = \{a_{ij}\}$ est une matrice $N \times N$ 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 $i, j = 1 \dots N$.
- $B = \{b_i(v_k)\}$ 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 $i = 1 \dots N$.

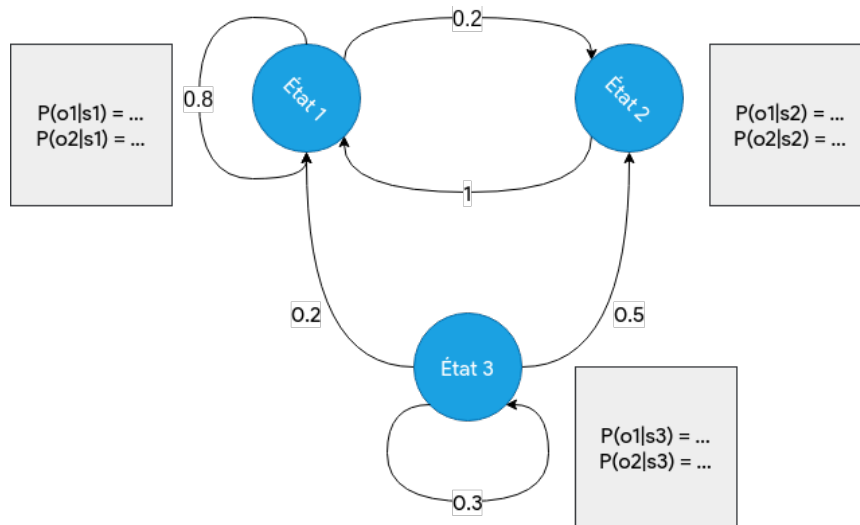


Figure 1.5 – Exemple d'une distribution de probabilité de transition ainsi qu'une distribution de probabilité d'observations pour un HMM à 3 états et 2 observations

Le but serait de trouver la séquence d'états $S_{1:K}$ qui maximise la probabilité Ghahramani (2002) :

$$P(Y_{1:K}|O_{1:K}) = P(O_1)P(O_1|S_1) \prod_{t=2}^K P(S_t|S_{t-1})P(O_t|S_t) \quad (1.8)$$

Pour y parvenir d'une manière efficace, un décodeur qui implémente l'algorithme de viterbi peut être utilisé Forney (1973); Bloit and Rodet (2008)

1.4 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 correspondant à la locution d'un utilisateur en un texte qui peut être interprété par la machine Al-Anzi and AbuZeina (2018). Différentes approches ont été développées au cours des années. Une architecture s'est ensuite dégagée où, les systèmes passent par deux phases : La phase d'apprentissage et la phase de reconnaissance. La première consiste à 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 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 Bloit and Rodet (2008), puis la séquence d'observations passe par le HMM et la meilleure séquence de mots est sélectionnée Yu and Deng (2015).

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

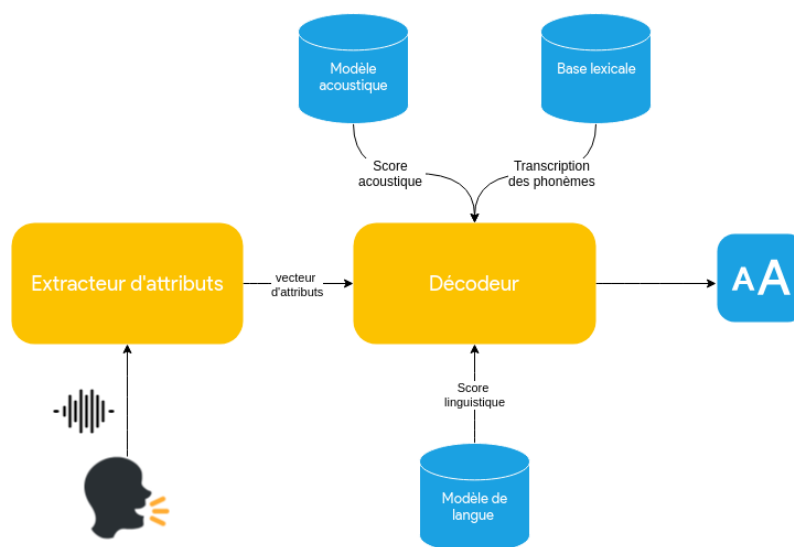


Figure 1.6 – Architecture des systèmes de reconnaissance de la parole Yu and Deng (2015)

1.4.1 Acquisition du signal 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 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 appliquée pour engendrer le Mel-Cpectrum qui est le vecteur d'attributs que nous recherchions Narang and Gupta (2015). Un tel processus peut utiliser plusieurs techniques pour la mise à l'échelle : MFCC (Mel-frequency cepstrum) Chauhan and Desai (2014), LPC (Linear Predictive coding) O'Shaughnessy (1988) et RASTA (RelAtive SpecTrAl) Rahali et al. (2014), chacune possédant ses avantages et ses inconvénients.

1.4.2 Modélisation acoustique et modélisation du lexique

C'est le cur 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 en utilisant un modèle HMM et en disposant d'un corpus de fichiers audio accompagnés de leurs transcriptions en phonèmes et en texte Ghai and Singh (2012); Rabiner and Juang (1986), ou bien un réseau de neurones profonds Yu and Deng (2015), ou encore une hybridation de ces derniers Deng et al. (2013). Le modèle acoustique procède après la reconnaissance de la séquence de phonèmes au décodage de ce dernier. Ceci est effectué en utilisant un dictionnaire linguistique qui transcrit chaque mots du vocabulaire en phonèmes qui le constituent. Ceci peut conduire 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.

1.4.3 Modélisation de la langue

Cette étape consiste à modéliser les aspects linguistiques de la langue que le système essaye de traiter. Souvent ces aspects sont spécifique au domaine d'application afin de restreindre l'espace de recherche des mots reconnaissables par le système afin de déterminer la séquence de mots en sortie la plus plausible. Les modèles utilisés sont basés soit sur des grammaires à contexte libre dans le cas où les séquences de mots reconnaissables sont peu variées et peuvent être modélisées par le biais de règles de la langue Glass and Kyle Mccandless (1994). Dans un cadre plus récent, des modèles probabilistes basés sur les N-grammes sont utilisés.

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

Dans un contexte textuel, un N-gramme est une suite de N mots consécutifs qui forment une sous-chaîne S' 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 Jurafsky and Martin (2008).

$$P(w|t) = \frac{\text{Comptage}(t, w)}{\text{Comptage}(t)} \quad (1.9)$$

Disposant d'un volume de données textuelles assez conséquent, l'utilisation de modèle basés sur les N-grammes ont prouvé leurs utilité Jurafsky and Martin (2008); Yu and Deng (2015); Roark et al. (2007) .

1.5 Compréhension du langage naturel (NLU)

Après avoir obtenu la requête de l'utilisateur, le module qui prend le relais est celui de la compréhension du langage naturel. Ce domaine fait partie du traitement automatique du langage naturel (NLP). Il se focalise principalement sur les tâches qui traitent le niveau sémantique voir même pragmatique de la langue tel que la classification de texte, l'analyse de sentiments ou encore le traitement automatiques des e-mails. Dans le contexte d'un SPA, le but final d'un module de compréhension du langage naturel est de construire une représentation sémantique de la requête de l'utilisateur. Étant donné une telle requête préalablement transformé en texte brut dans un langage naturel, le module essaiera d'en extraire une information primordiale qui est l'intention du locuteur.

1.5.1 L'intention

L'intention est un élément clé dans notre travail. Nous allons donc la définir et expliquer pourquoi elle a été choisie comme abstraction sémantique d'une requête d'un utilisateur.

Une intention (ou intent en anglais) est une représentation sémantique d'un but ou d'un sous-but de l'utilisateur, plusieurs requêtes de l'utilisateur peuvent avoir le même intent, ce qui rend l'utilisation d'une telle représentation essentielle pour que la machine puisse mieux comprendre l'utilisateur. La motivation vient du fait que la machine ne peut pas comprendre une requête formulé dans un langage non-formel. Il a fallu trouver un moyen de communication universel entre la machine et l'utilisateur de telle sorte que l'un puisse comprendre l'autre sans parler sa langue. L'intermédiaire entre ces deux parties est un traducteur bidirectionnel requête en langage naturel vers requête dans un langage formel et inversement.

Le choix de l'intention doit être minutieusement étudié au préalable pour ne pas trop subdiviser un type d'intention, tout en gardant un certain degré de spécificité. À titre d'exemple, si deux intentions *Ouvrir_fichier* et *Ouvrir_document* existent, on retrouve ici une répétition

de l'information *fichier* qui est aussi un *document*. Dans ce cas il suffit d'éliminer un synonyme pour ne garder qu'un seul cas.

Un autre cas est celui du sur-découpage d'intentions déjà assez semblable, *Ouvrir_fichier* et *Ouvrir_Repertoire* peuvent être regroupés en une seule intention *Ouvrir_Entit*. Bien-sûr ces choix sont purement conceptuel et dépendent fortement du problème traité par le système, choisir une représentation à la place d'une autre est lié à la nature de la tâche que le module doit réaliser Liu and Lane (2016); Goo et al. (2018)

1.5.2 Classification d'intentions

La classification d'intentions à partir d'un texte est une tâche réalisable avec des techniques récentes d'apprentissage automatique, plus précisément en utilisant des modèles basés sur les réseaux de neurones récurrents à mémoire court et long terme (LSTM) (voir la section 1.4). Dans Liu and Lane (2016) et Goo et al. (2018) deux architectures ont fait leurs preuves. Dans le premier travail, les auteurs ont utilisé une architecture BiLSTM (LSTM Bidirectionnel) pour capturer les contextes droit et gauche d'un mot donné Schuster and Paliwal (1997). Le dernier état caché retourné par la cellule LSTM est ensuite utilisé pour la classification. Une couche d'attention est ajoutée Chorowski et al. (2015) pour permettre au modèle de se focaliser sur certaines parties du texte en entrée. La deuxième architecture est un peu plus simpliste. En effet elle utilise des cellules LSTM basiques avec une couche de classification sur le dernier état. L'avantage par rapport à la première architecture est le temps d'apprentissage assez réduit au détriment d'une erreur de classification plus accrue mais toujours dans les normes.

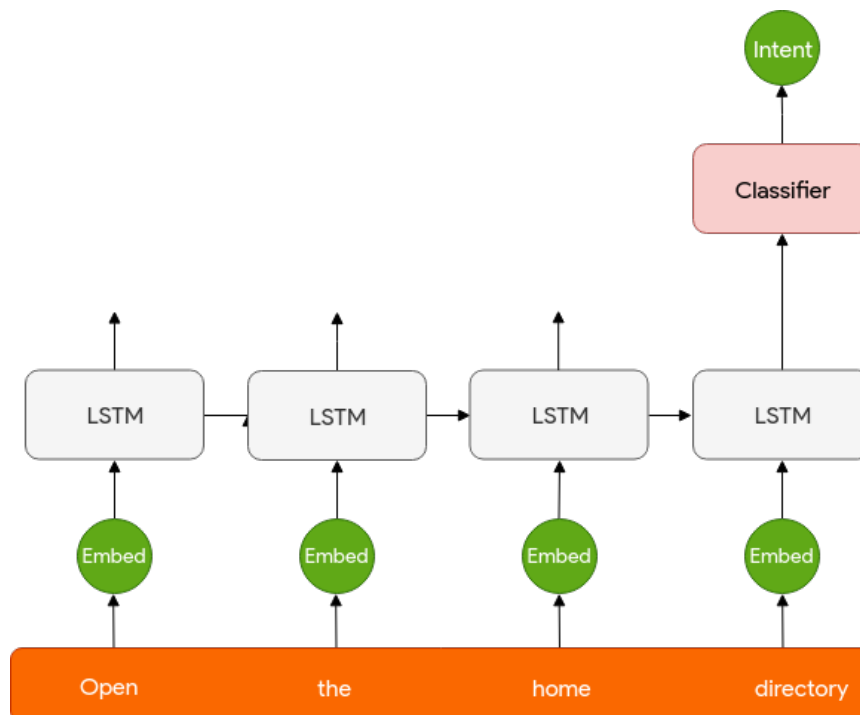


Figure 1.7 – Architecture de base d'un classificateur d'intents Liu and Lane (2016)

1.5.3 Extraction d'entités

Déterminer l'intention d'un utilisateur ne s'arrête pas qu'au stade de l'identification de l'*intent*. En effet, pour mieux représenter l'information récoltée depuis la requête, il faut en extraire des entités (nommées ou spécifiques du domaine) qui seront des arguments de l'intent identifié. Cette tâche consiste donc à, pour un intent I donné, extraire les arguments $Args$ qui lui sont appropriés depuis le texte de la requête. Plusieurs approches sont possibles, dans Goo et al. (2018) les auteurs ont traité le problème comme étant la traduction d'une séquence de mots en entrée en une séquence d'entités en sortie, le schéma suivant explicite le processus :

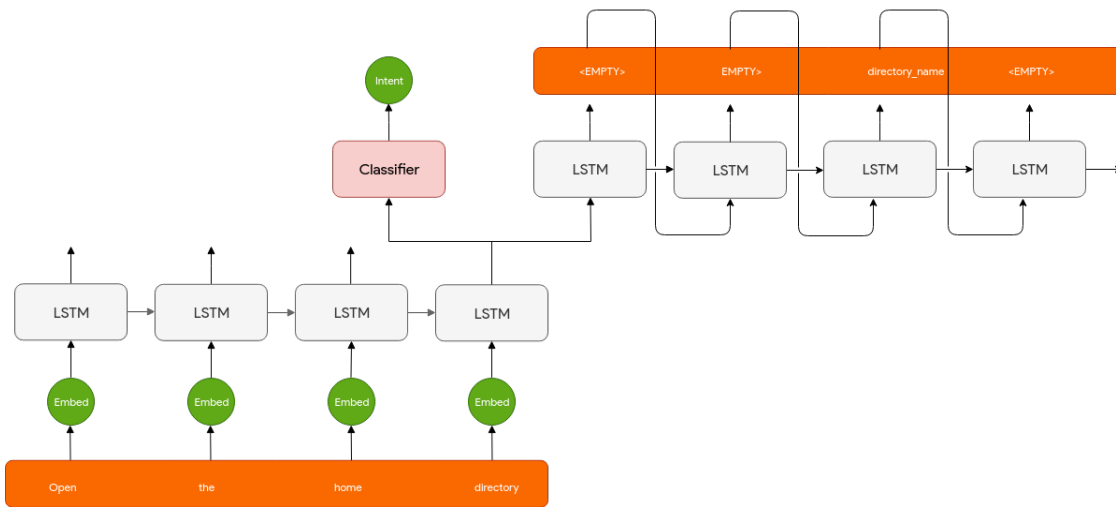


Figure 1.8 – Architecture de base d'un classificateur d'intents doublé d'un extracteur d'entité Goo et al. (2018)

1.5.4 Analyse sémantique

Disposant des deux modèles précédemment cités, le module NLU peut maintenant construire une représentation sémantique adéquate qui va être transmise au module suivant qui sera le gestionnaire du dialogue. L'avantage d'avoir en sortie une structure sémantique universelle est la non dépendance par rapport à la langue de l'utilisateur. En effet, le système pourra encore fonctionner si une correspondance entre la nouvelle langue et le format choisi pour la représentation sémantique peut être trouvée. Le module donnera donc en sortie une trame sémantique (Semantic frame) qui comprendra les informations préalablement extraites, à savoir l'intent et les arguments (slots) qui lui sont propres Liu and Lane (2016); Goo et al. (2018); Wang et al. (2018).

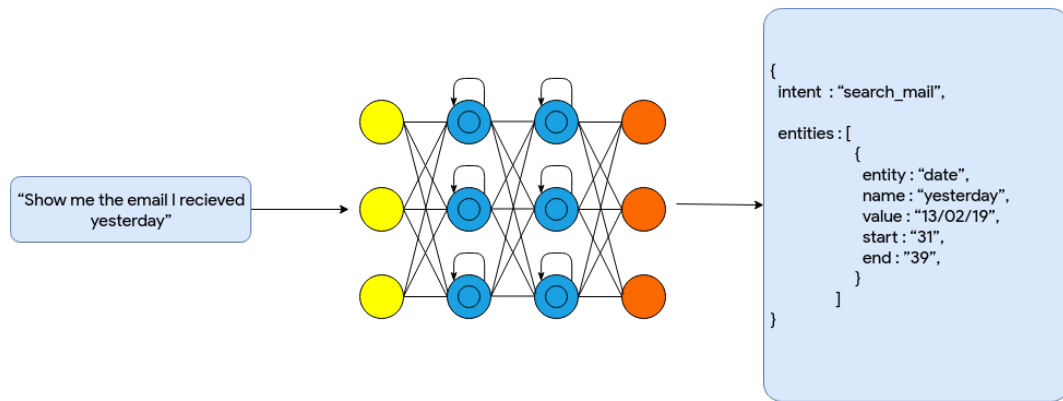


Figure 1.9 – Exemple d’une trame sémantique pour une requête donnée

1.6 Gestion du dialogue

La compréhension du langage naturel permet de transformer un texte en une représentation sémantique. Afin qu’un système puisse 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. Celle-ci est transmise au générateur du langage naturel (voir 1.7) pour afficher un résultat à l’utilisateur. Généralement, deux principaux modules sont présents dans les systèmes de gestion de dialogue :

- Un module qui met à jour l’état du gestionnaire de dialogue : le traqueur d’état.
- Un module qui détermine la politique d’action du gestionnaire de dialogue.

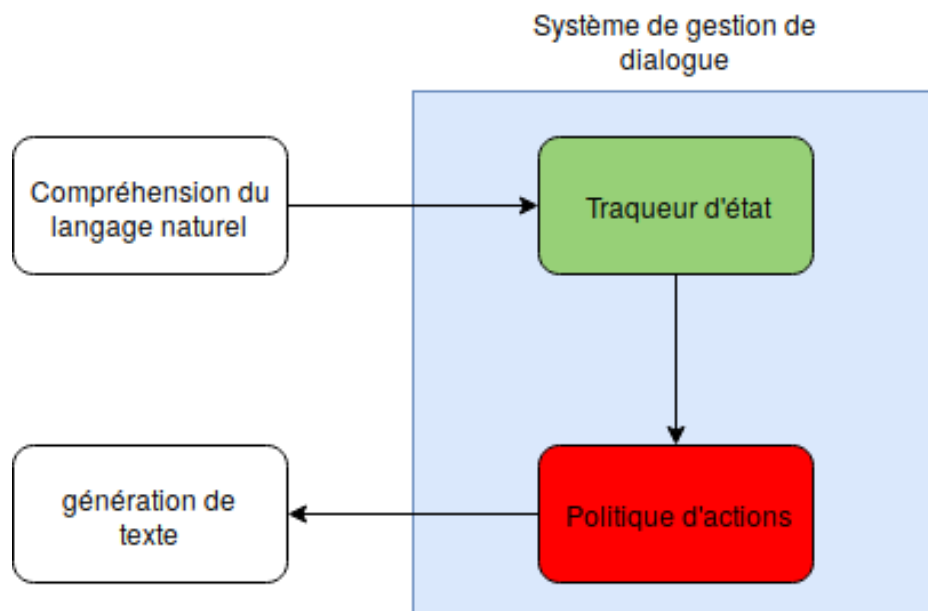


Figure 1.10 – Schéma général d’un gestionnaire de dialogue

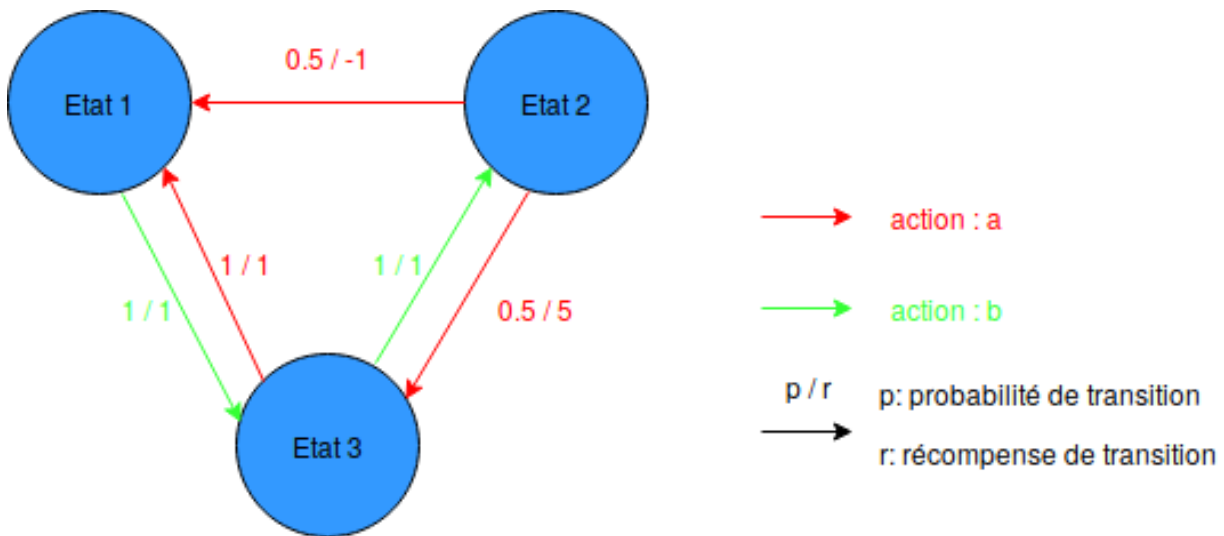
Ainsi, le gestionnaire de dialogue passe d’un état à un autre après chaque interaction avec l’utilisateur. Le problème à résoudre est donc de trouver la meilleure action à prendre étant dans un état donné.

1.6.1 Modélisation

Pour résoudre ce problème, un gestionnaire de dialogue peut être modélisé par un processus de décision Markovien (MDP) Richard Bellman" (1957). Ce dernier est modélisé par un 4-tuple (S, A, P, R) :

- S : ensemble des états du système.
- A : ensemble des actions du système.
- P : distribution de probabilités de transition 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 réalisé l'action a .
- R : est la récompense reçue immédiatement après avoir changé l'état avec une action donnée. $R(s'/s, a)$ est la récompense reçue après être passé à l'état s' sachant qu'on était à l'état s après avoir réalisé l'action a .

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' . De ce fait, il reçoit une récompense qui, dans notre cas, est une mesure sur les performances du système de dialogue. Le but est de maximiser les récompenses reçues.



1.6.2 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 Chen et al. (2017). Cette structure contient des emplacements à remplir sur un domaine donné. La figure 1.12 illustre un exemple de cadre sémantique.

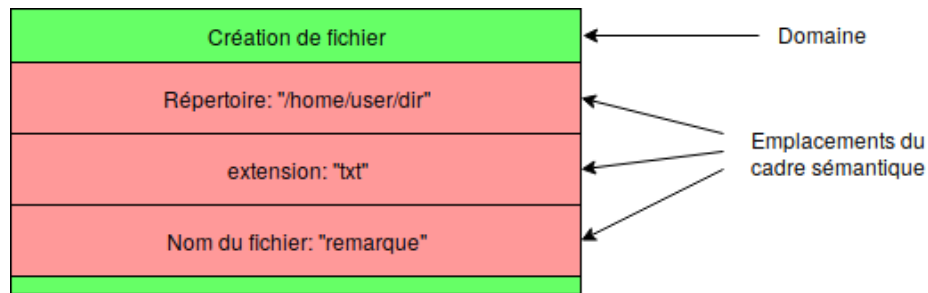


Figure 1.12 – Schéma représentant un cadre sémantique avec comme domaine : création de fichier

À l'arrivée d'une nouvelle information, un module dédié met à jour l'état du gestionnaire de dialogue. Comme l'action du système de dialogue est décidée à partir de son état, cette tâche est donc essentielle au bon fonctionnement du système. Plusieurs méthodes ont été proposées pour gérer le suivi de l'état du gestionnaire de dialogue Williams and Young (2007).

1.6.2.1 Suivi de l'état avec une base de règles

La méthode traditionnelle utilisée consiste à écrire manuellement les règles à suivre lors de l'arrivée d'une nouvelle information pour mettre à jour l'état Goddeau et al. (1996). Cependant, les bases de règles sont très susceptibles à faire des erreurs Chen et al. (2017) car elles sont peu robustes face aux incertitudes.

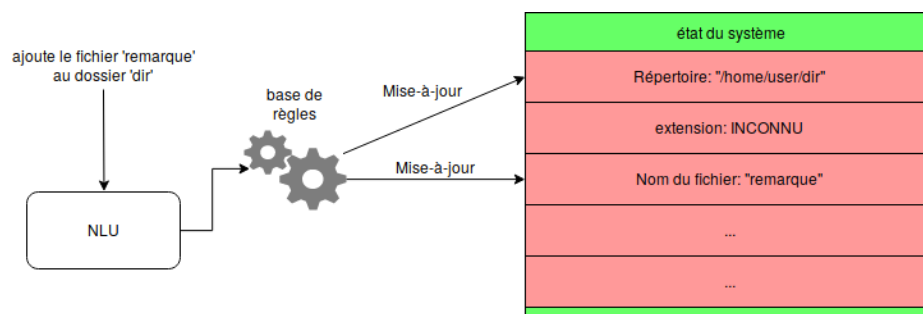


Figure 1.13 – Schéma représentant la mise-à-jour de l'état par un système basé règles

1.6.2.2 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. Ce qui nécessite une nouvelle modélisation non déterministe du problème. Les processus de décision markovien partiellement observable (POMDP) Young et al. (2010) sont des variants des MDP capables de répondre à ce besoin 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.

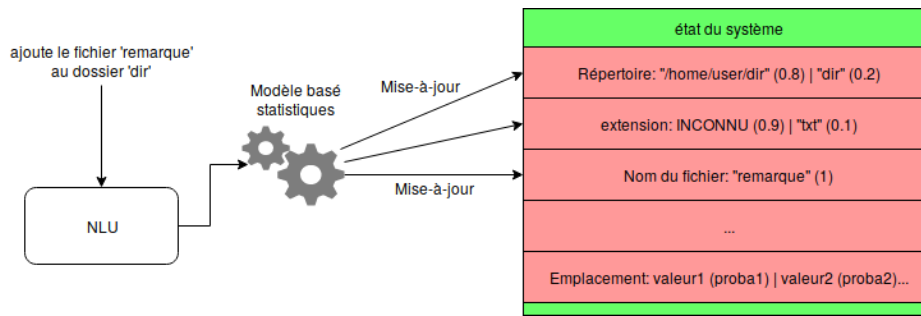


Figure 1.14 – Schéma représentant la mise-à-jour de l'état par un système basé statistiques

Processus de décision markovien partiellement observable (POMDP) Comme dans les processus de décision markovien, un POMDP Åström (1965) passe d'un état à un autre en prenant une des actions possibles. Cependant, ce dernier ne connaît 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 mêmes que ceux d'un MDP (voir 1.6.1).
- M : l'ensemble des observations.
- O : distribution de probabilités sur les observations o en connaissant l'état s et l'action a 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.

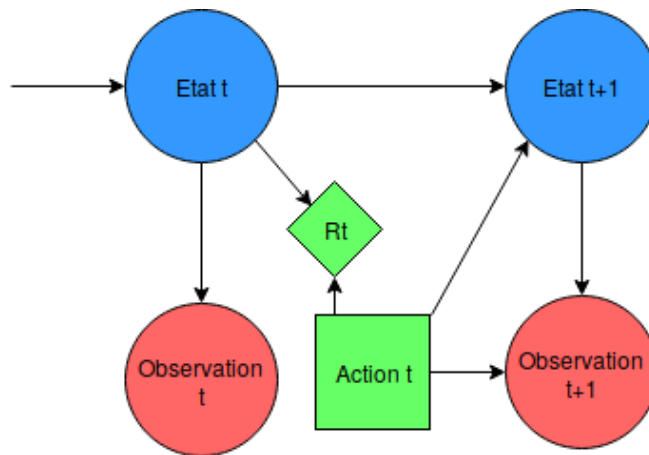


Figure 1.15 – Diagramme d'influence dans un POMDP

1.6.2.3 Suivi de l'état avec réseaux de neurones profonds

Récemment, des approches utilisant les réseaux de neurones profonds (voir 1.3.2.2) ont fait leur apparition. 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 Henderson et al. (2013). En conséquence, il peut être utilisé comme modèle de suivi d'état pour un processus partiellement observable.

1.6.3 Politique de gestion de dialogue

La première partie est 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 obtenues.

1.6.3.1 Gestion de dialogue avec une base de règles

Les premières approches utilisaient des systèmes de règles destinés à un domaine bien spécifique. Elles étaient déployées dans plusieurs domaines d'application pour leur simplicité. Cependant, le travail manuel nécessaire reste difficile à faire et, généralement, aboutit pas à des résultats flexibles qui peuvent suivre le flux du dialogue convenablement Lee et al. (2010).

1.6.3.2 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 dans le 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 Henderson et al. (2008), encore moins des corpus annotés avec les récompenses à chaque transition. Néanmoins, il existe des approches de bout en bout qui exploitent des architectures avec réseaux de neurones profonds et traitent le problème comme Seq2Seq¹ afin de produire directement une sortie à partir des informations reçues de l'utilisateur Wen et al. (2017); Serban et al. (2016).

1. Les modèles Seq2Seq sont des architectures de réseaux de neurones profonds qui prennent en entrée une séquence et produisent en sortie une autre séquence en utilisant les réseaux de neurones récurrents (RNN).

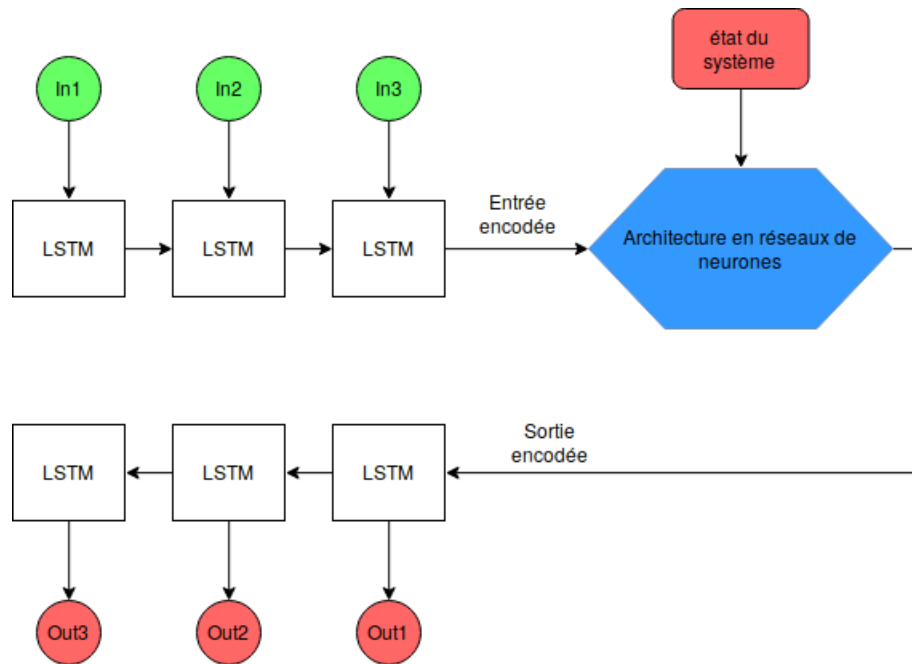


Figure 1.16 – Schéma de gestion de dialogue de bout en bout avec architecture Seq2Seq

1.6.3.3 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. Celle-ci doit maximiser une mesure d'évaluation qui est sous forme de récompenses obtenues après chaque action (Weisz et al. (2018)). L'environnement est souvent modélisé 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 dans des états donnés. Il peut ainsi estimer la fonction de récompense pour pouvoir faire le choix d'actions optimales.

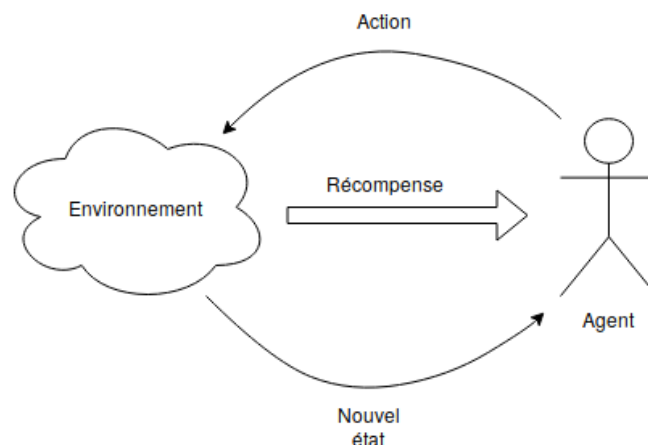


Figure 1.17 – Schéma d'interaction agent-environnement dans l'apprentissage par renforcement

Il existe plusieurs méthodes que l'agent peut utiliser pour estimer la fonction de récompense

Bertsekas (2012), notamment Deep Q Learning (DQN) Mnih et al. (2015) qui utilise les réseaux de neurones profonds pour trouver une approximation à cette fonction. Dans cette approche, à chaque fois que l'agent prend 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. Donc, la nécessité d'utiliser un **simulateur utilisateur** qui est un programme capable de se comporter 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 une fonction de récompense. Il existe plusieurs méthodes pour créer un simulateur utilisateur :

- Les simulateurs utilisateur basés règles : dans ce cas une liste de règles est écrite manuellement que le simulateur doit suivre pour communiquer avec le système Schatzmann et al. (2007).
- Les simulateurs utilisateur basés 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 Georgila et al. (2005).
- Les simulateurs utilisateur basés 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 de l'utilisateur sachant l'état du système ou l'action du système directement dans le modèle HMM Cuayáhuatl et al. (2005).
- Les simulateurs 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 Chandramohan et al. (2011).

1.7 Génération du langage naturel (NLG)

Le domaine de la génération automatique du langage naturel est l'un des domaines dont les limites 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 entrées, c'est à dire, sur quoi se basera le système pour générer le texte. D'après Reiter and Dale (1997) 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 and Portet (2012), utilisent les

mêmes techniques pour des entrées linguistiques. Enfin, la génération du langage naturel peut être très proche de la gestion de dialogue Dethlefs (2014). 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 Reiter and Dale (1997).

1.7.1 Détermination du contenu

Cette partie consiste à sélectionner les informations de l'entrée dont le système veut transmettre leur 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 Yu et al. (2007). De plus, le choix de l'information peut aussi dépendre de l'utilisateur et de ses connaissances Dethlefs (2014). Ce qui requiert de mettre au point un système qui détecte les informations pertinentes à l'utilisateur.

1.7.2 Structuration de texte

Après la détermination du contenu, le système doit ordonner les informations 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.

1.7.3 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 répétitions. Un exemple de cela peut être la description de la météo à Alger au cours de la matinée :

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

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

1.7.4 Lexicalisation

Le système choisit les mots et les expressions à utiliser pour communiquer le contenu des phrases sélectionnées. La difficulté de cette tâche revient à l'existence de plusieurs manières 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 Gatt and Krahmer (2018).

1.7.5 Génération d'expressions référentielles (REG)

Cette partie du système se focalise sur 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 son 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 Reiter and Dale (1997). 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. Le choix entre eux dépend de comment l'entité a été mentionnée auparavant, si elle l'a été, et des détails qui l'ont accompagnés.

1.7.6 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) et des approches statistiques Gatt and Krahmer (2018).

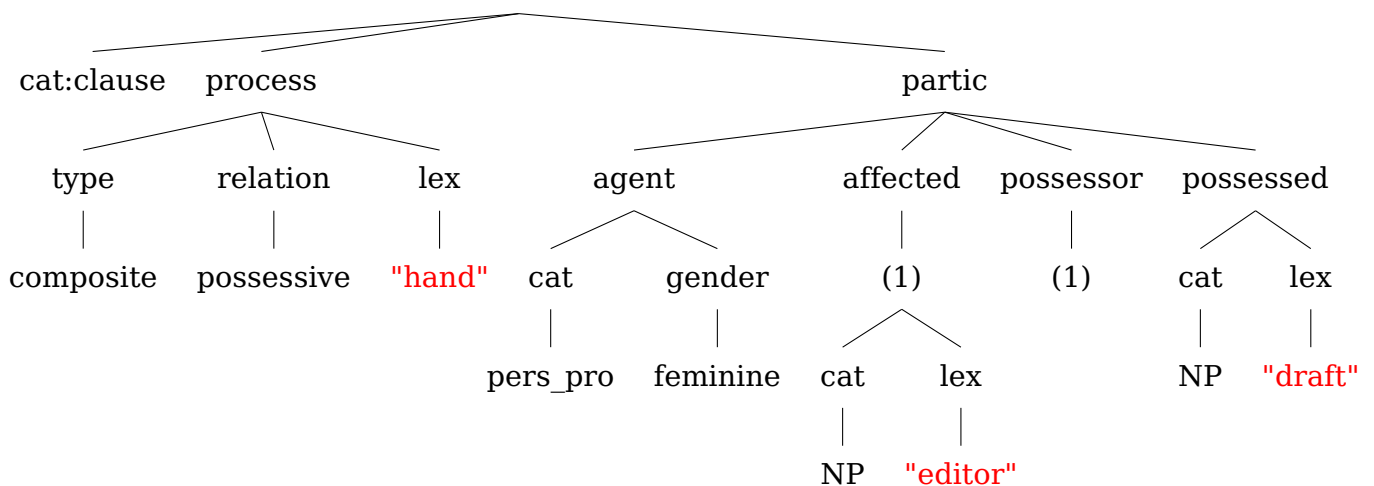


Figure 1.18 – Un exemple d'entrée de SURGEElhadad and Robin (1996)

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 côté, il est facile de contrôler la qualité des sorties syntaxiquement et sémantiquement tout en utilisant des règles de remplissage complexe Theune et al. (2001). 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 essayé de faire un apprentissage de modèles de phrases à partir d'un corpus Angeli et al. (2012), cette méthode reste inefficace lorsqu'il s'agit d'applications qui nécessitent un grand nombre de variations linguistiques.

Systèmes basés grammaire : La réalisation peut se faire en suivant une grammaire du langage. Celle-ci contient les règles morphologiques et de structures de la langue, notamment la grammaire systémique fonctionnelle (SFG) Halliday and Matthiessen (2004) qui a été largement utilisée comme dans NIGEL Mann and Matthiessen (1983) ou KPML Bateman (1997). 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 Elhadad and Robin (1996) : Qui génère la phrase : She hands the draft to the editor.

Comme les modèles de phrases, les systèmes basés grammaire nécessitent 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 Gatt and Krahmer (2018).

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 utilisant Gatt and Krahmer (2018) :

- La première se base sur une petite grammaire qui génère plusieurs alternatives qui sont ensuite ordonnées selon un modèle statistique basé sur un corpus pour sélectionner la phrase la plus probable (par exemple Langkilde-Geary (2000)Langkilde-Geary (2000)).
- La deuxième méthode utilise les informations statistiques directement au niveau de la génération pour produire la solution optimale (exemple : Belz (2008)Belz (2008)).

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. Elle peut être aussi extraite à partir des données, comme utilisation des Treebanks² pour déduire les règles de grammaire Espinosa et al. (2008).

D'autres approches statistiques n'utilisent pas des grammaires mais se basent sur des classificateurs. Ces derniers peuvent être cascades de telle sorte à décider quel constituant utiliser dans quelle position ainsi que les modifications nécessaires pour générer un texte correct. À noter qu'une telle approche, ne nécessitant pas l'utilisation de grammaire, utilise des entrées plus abstraites et moins détaillées linguistiquement. À voir même la possibilité qu'elle s'étende 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 des systèmes basés encodeur-décodeur qui sont récemment plus utilisés.

1.7.7 Systèmes basés encodeur-décodeur

Une architecture souvent utilisée dans le traitement du langage naturel est l'encodeur-décodeur. En particulier, son utilisation dans les tâches seq2seq 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 adaptés pour convertir une représentation abstraite de l'information en langage naturel Ferreira et al. (2017).

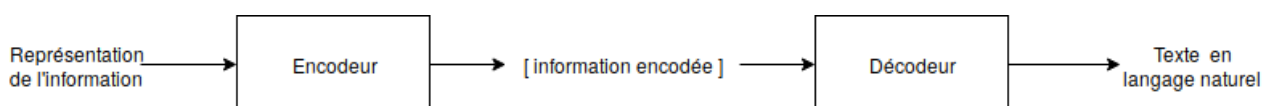


Figure 1.19 – Schéma d'une architecture encodeur-décodeur pour NLG

Beaucoup d'approches de génération de langage naturel en gestion de dialogue utilisent des encodeur-décodeur. Wen et al. (2015)Wen et al. (2015) utilise par exemple des LSTMs

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

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 pour encoder l'état du gestionnaire de dialogue et l'entrée reçue suivie par un décodeur pour générer le texte de la réponse Sordani et al. (2015) ; Serban et al. (2016) ; Goyal et al. (2016).

1.8 Conclusion

Au terme de ce chapitre et après avoir étudié l'état de l'art sur les systèmes existants, nous avons pu construire un bagage théorique assez complet dans le domaine des SPAs. En assimilant les différentes approches, il est maintenant temps de passer à la conception de notre propre système. Le chapitre suivant introduira nos propositions pour le développement de notre propre SPA qui sera destiné à la manipulation d'un ordinateur.

Conclusion générale

Nous sommes sans doute actuellement entrain de vivre une époque importante de l'intelligence artificielle. Le temps ou nous rêvions encore de converser avec une machine semble à la fois proche en terme de temps, mais loin en terme de progrès. Les utilisateurs réguliers de produits technologiques sont confrontés à une technologie nouvelle mais prometteuse. Ce secteur d'activités peut donc s'avérer très prolifique si les efforts fournis sont suffisamment conséquents.

La création de ce nouveau besoin a été la principale source de motivation pour ce projet. Cependant, et par faute de temps ainsi que de moyens techniques (surtout en ce qui concerne le coté matériel pour l'apprentissage automatique), certains modules n'ont pas été exploités à leur maximum. Cela ne nous a pas empêché de réaliser un travail dont nous sommes particulièrement fières. Mais nous gardons toutefois un esprit critique, ainsi qu'une objectivité envers le travail fourni.

Tout au long de la réalisation de ce mémoire, nous avons étudié l'état actuel des assistants personnels intelligents. Nous avons dû passer une majeure partie de cette étape à comprendre les fondements théoriques et conceptuelles de chaque composants de ces systèmes. Principalement a cause de la grande densité de techniques, concepts et théories qui sont nouvelles pour nous.

Après avoir assimilé la totalité des concepts, et qui font office d'état de l'art du domaine. Nous sommes arrivé à certaines conclusions. Tout d'abord, développer un système en partant de rien était un travail assez massif. Dépassant de loin le cadre d'un projet de fin d'études de Master. Nous avons donc mit l'accent sur certains modules. Ces modules, à savoir le module de compréhension du langage naturel et le module de gestion du dialogue, dépendaient énormément de notre problématique. Le module de reconnaissance automatique de parole a été sujet à une amélioration spécifique à nos besoins tout en exploitant un système de base déjà existant (à savoir DeepSpeech).

Avec une idée claire du travail à réaliser, nous avons pu entamer la conception de chaque module en y incorporant nos ajouts et modifications. Beaucoup de ces modifications sont le fruit de longues séances de débat et de discussions.

L'étape de réalisation est celle qui a pris le plus de temps. Entre la découverte de nouvelles technologies à utiliser, la collecte des données et l'apprentissage des différents modèles.

La partie de construction du corpus pour le modèle de langue du module de reconnaissance automatique de la parole a été soumise à beaucoup d'optimisations incrémentales, en tombant à chaque fois sur un nouveau problème, ou bien un obstacle matériel (manque de puissance de calcul). L'ajout de ce modèle de langue a amélioré les résultats. Cela s'accordait avec nos prédictions théoriques. Cependant, les résultats n'étaient pas assez encourageants. Surtout si le but est de détrôner les systèmes propriétaires comme celui de Google qui réalise un score quasi-parfait sans apprentissage supplémentaire. L'avenir de systèmes Open source pour la reconnaissance automatique de la parole reste cependant prometteur. Ces derniers offrent un moyen libre de mener des études et contribuer au développement à grande échelle de cette discipline. Une perspective future pour ce module serait de lancer notre propre plateforme de collecte de données. Cette idée a déjà été discutée dans la partie de l'étude de l'état de l'art. Malheureusement le temps a cruellement manqué. L'investissement de la communauté dans de telles initiatives n'en reste pas moins indéniable. Comme l'a démontré le projet CommonVoice de Mozilla.

La construction de l'ensemble d'apprentissage à partir de zéro était l'étape la plus longue de la réalisation du module de compréhension du langage naturel. Plus l'ensemble grandissait, plus il était difficile de maintenir sa validité sans l'intervention d'un soutien externe. Les ajouts faits au modèle d'apprentissage pour ce module ont été expérimentalement validés dans le chapitre Réalisations et résultats. L'ajout de l'information morphosyntaxique a permis de donner plus de valeur sémantique à chaque mots de la requête. L'introduction d'erreurs aléatoires a quad à elle permit la gestion d'éventuelles erreurs que pourrait engendré le module de reconnaissance automatique de la parole. Cependant, vu que la tâche à accomplir était relativement simple et limitée, le réel impacte de cet ajout ne peut pas être certifié et validé dans un cadre plus général. Une autre problématique est celle du manque de données d'apprentissage consacrées au domaine de la manipulation d'ordinateurs. Ces données sont généralement construites manuellement par les développeurs du système. Faire appel à une collecte massive de données est une des perspectives envisagées. Le développement d'un outil d'aide à l'annotation d'un corpus était elle aussi le sujet d'un long débat. La mise à l'échelle de cette plateforme pourrait grandement faire avancer la tâche fastidieuse qu'est la collecte de données. Un autre point à soulever est celui du manque de diversité dans les tâches réalisable par l'assistant. Cet ensemble de tâche reste facilement extensible. Il suffit d'ajouter des exemples assez exhaustifs à l'ensemble de données. Cela reste néanmoins une tâche lourde et manuelle à plus grande échelle. L'automatisation de cette étape n'a pas été discutée pendant la réalisation de ce travail.

En ce qui concerne le gestionnaire de dialogue, nous l'avons conçu pour qu'il soit facilement ajustable et mis à l'échelle. L'utilisation d'une architecture hiérarchisée d'agents de dialogue permet de grandement réduire la complexité de développement et d'ajout d'un nouveau gestionnaire de tâches dans le système. L'utilisation d'un système de ges-

tion d'état du dialogue plus riche que ce qui est proposé en temps normal a permis à l'agent apprenant de mieux comprendre les échanges effectués avec l'utilisateur. L'utilisation d'un graphe de connaissances au lieu de trames sémantiques est justifié par le fait que ces derniers offrent une plus grande flexibilité dans la représentation de l'état du dialogue. Pour ce qui est des agents de dialogue, ils sont entraînés en utilisant des techniques d'apprentissage par renforcement. Ils interagissent avec le simulateur d'utilisateur, qui est un programme permettant de simuler le comportement d'un utilisateur avec la machine. Ce simulateur est capable d'estimer à quel point l'agent apprenant est efficace dans la tâche qui lui est confiée. Le but de l'agent est d'estimer la satisfaction de l'utilisateur en utilisant un réseau de neurones comme fonction d'approximation. Son but final est d'atteindre le but de l'utilisateur à travers l'optimisation d'une politique d'actions basée sur un système de récompenses. Cependant, l'utilisation du graphe de connaissances et de l'apprentissage profond a un prix. Pour essayer de connecter ces deux composants, nous avons dû condenser le graphe tout entier en un vecteur de taille fixe. Il peut être ensuite injecté au réseau de neurones pour effectuer la prédiction de l'action à entreprendre. Nous avons exploré deux approches pour cela en les comparant expérimentalement. En effet, la codification de la totalité du graphe en un seul vecteur était une tâche assez difficile. La taille de ce dernier augmente exponentiellement avec l'ajout de nouvelles connaissances. Ce qui rend le processus d'encodage en un vecteur de taille fixe très compliqué. De plus, les nœuds du graphe ont été codifiés avec des entiers distincts comme identificateurs. Ce qui a pour effet de faire perdre de l'information sémantique au graphe encodé. Le décodage du graphe s'en trouve grandement affecté dans la phase d'apprentissage de l'estimateur d'actions. Comme perspectives futures, il est envisageable d'utiliser une méthode d'apprentissage semi-supervisée pour l'encodage des nœuds du graphes. Cette ajout pourrait permettre d'enrichir la valeur sémantique de ces nœuds, et facilitera la tâche au réseau de neurones de l'agent apprenant pour le décodage du graphe. Principalement grâce au fait que les nœuds dont les sens sont arbitrairement proches auront des codifications similaires.

Pour le dernier module, à savoir le module de génération du langage naturel. Il fût assez simple à réaliser. Son fonctionnement est très rudimentaires et intuitif. Il s'agissait d'utiliser un ensemble de phrases modèles et de remplacer les valeurs manquantes avec des valeurs réelles. Une méthode plus sophistiquée comme l'utilisation de modèles d'apprentissage automatique basés encodeur-décodeur, ou bien basés convertisseur de graphes de connaissances en texte pourraient être utilisées. Cependant, ces architectures requièrent un très grand volume de données d'apprentissage annotés et spécifique à notre problématique.

Pour ce qui est de l'application, nous avons fait le choix d'utiliser une architecture trois tiers basée web. Ce choix fût motivé par le fait que l'utilisation d'un serveur offre une puissance de calcul considérablement plus élevée que celle d'une machine personnel en local. L'interface reste assez simple et épurée. Le but est de minimiser l'interaction avec l'utilisateur avec tout autre moyen de communication que la parole. Cependant cette interface à aussi pour but de montrer les fonctionnalités du système. Elle est donc plus orientée vers le développement. Une amélioration possible serait le déploiement du serveur dans un service de Cloud Hosting. Cela permettrait de minimiser les temps d'inférence et de

post-traitement. Le développement d'une interface plus légère et plus orientée vers les cas pratiques est une amélioration possible. Garder les deux cas de figures (utilisation et développement) est aussi possible.

Pour conclure, nous estimons que la totalité du projet était une énorme occasion d'approfondir nos connaissances. Que ce soit celles qui nous ont été enseignées durant notre cursus, comme l'apprentissage automatique, le traitement automatique du langage naturel, le web-sémantique et la représentation de connaissances. Ou bien celles que nous avons apprises au cours de notre étude de la littérature, comme l'apprentissage par renforcement, le traitement automatique de la parole, le nettoyage des données, etc. Ce projet nous a aussi initié au travail en équipe pour la réalisation d'un projet assez conséquent. Tout en étant encadrés par nos supérieurs. Finalement, nous pensons que la plus grande satisfaction vient du fait que nous avons réalisé, dans un certain délai restreint, un travail qui traite d'un sujet récent et ambitieux. Et ainsi, de poser la première pierre à l'édifice pour, idéalement, encourager les chercheurs en Algérie à s'investir dans ce domaine.

Bibliographie

- Al-Anzi, F. and AbuZeina, D. (2018). Literature survey of arabic speech recognition. pages 1–6.
- Angeli, G., Manning, C. D., and Jurafsky, D. (2012). 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. Association for Computational Linguistics.
- Åström, K. J. (1965). Optimal control of Markov Processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205.
- Barlow, H. (1989). Unsupervised learning. *Neural Computation*, 1(3):295–311.
- Bateman, J. A. (1997). Enabling technology for multilingual natural language generation : The kpml development environment. *Nat. Lang. Eng.*, 3(1).
- Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.*, 14(4).
- Bertsekas, D. (2012). *Dynamic Programming & Optimal Control, Vol II : Approximate Dynamic Programming*. Athena Scientific.
- Bloit, J. and Rodet, X. (2008). Short-time viterbi for online hmm decoding : Evaluation on a real-time phone recognition task. pages 2121 – 2124.
- Chandramohan, S., Geist, M., Lefèvre, F., and Pietquin, O. (2011). User simulation in dialogue systems using inverse reinforcement learning. In *INTERSPEECH*, page 10251028.
- Chauhan, P. M. and Desai, N. P. (2014). 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)*, pages 1–5.
- Chen, H., Liu, X., Yin, D., and Tang, J. (2017). A survey on dialogue systems : Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35.
- Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. *CoRR*, abs/1506.07503.
- Cuayáhuítl, H., Renals, S., Lemon, O., and Shimodaira, H. (2005). Human-computer dialogue simulation using hidden markov models. *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 290–295.

- Deng, L., Hinton, G., and Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications : An overview. pages 8599–8603.
- Dethlefs, N. (2014). Context-sensitive natural language generation : From knowledge-driven to data-driven techniques. *Language and Linguistics Compass*, 8.
- Elhadad, M. and Robin, J. (1996). An overview of surge : a reusable comprehensive syntactic realization component. In *International Natural Language Generation Workshop*.
- Espinosa, D., White, M., and Mehay, D. (2008). Hypertagging : Supertagging for surface realization with ccg. In *ACL*.
- Evans, R., Piwek, P., and Cahill, L. (2002). What is nlg? In *Proceedings of the Second International Conference on Natural Language Generation*.
- Ferreira, T. C., Calixto, I., Wubben, S., and Krahmer, E. (2017). Linguistic realisation as machine translation : Comparing different mt models for amr-to-text generation. In *INLG*.
- Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.
- Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation : Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1).
- Georgila, K., Henderson, J., and Lemon, O. (2005). Learning user simulations for information state update dialogue systems. In *INTERSPEECH*, pages 893–896.
- Ghahramani, Z. (2002). Hidden markov models. chapter An Introduction to Hidden Markov Models and Bayesian Networks, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA.
- Ghai, W. and Singh, N. (2012). Literature review on automatic speech recognition. *International Journal of Computer Applications*, 41(8):42–50. Full text available.
- Glass, R. and Kyle Mccandless, M. (1994). Automatic acquisition of language models for speech recognition.
- Goddeau, D., Meng, H., Polifroni, J., Seneff, S., and Busayapongchai, S. (1996). A form-based dialogue manager for spoken language applications. volume 2, pages 701 – 704.
- Goo, C.-W., Gao, G., Hsu, Y.-K., Huo, C.-L., Chen, T.-C., Hsu, K.-W., and Chen, Y.-N. (2018). Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.
- Goyal, R., Dymetman, M., and Gaussier, E. (2016). Natural language generation through character-based rnns with finite-state prior knowledge. In *COLING*, pages 1083–1092.
- Graves, A., Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.
- Halliday, M. A. K. and Matthiessen, C. M. I. M. (2004). *Introduction to Functional Grammar*. Hodder Arnold, London, 3 edition.

- Henderson, J., Lemon, O., and Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Comput. Linguist.*, 34(4):487–511.
- Henderson, M., Thomson, B., and Young, S. J. (2013). Deep neural network approach for the dialog state tracking challenge. In *SIGDIAL Conference*, pages 467–471.
- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.
- Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing, 2nd Edition*. Prentice Hall.
- Kemeny, J. G. and Laurie Snell, J. (1957). Markov processes in learning theory. *Psychometrika*, 22(3):221–230.
- Kotsiantis, S., Zaharakis, I., and Pintelas, P. (2006). Machine learning : A review of classification and combining techniques. *Artificial Intelligence Review*, 26:159–190.
- Labbé, C. and Portet, F. (2012). Towards an abstractive opinion summarisation of multiple reviews in the tourism domain. *CEUR Workshop Proceedings*, 917.
- Langkilde-Geary, I. (2000). Forest-based statistical sentence generation. In *ANLP*.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.
- Lee, C., Jung, S., Kim, K., Lee, D., and Lee, G. G. (2010). Recent approaches to dialog management for spoken dialog systems. *JCSE*, 4:1–22.
- Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019.
- Liu, B. and Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *CoRR*, abs/1609.01454.
- Liu, P., Qiu, X., and Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. *CoRR*, abs/1605.05101.
- Mitchell, T. (2006). The discipline of machine learning.
- Mann, W. C. and Matthiessen, C. M. I. M. (1983). *Nigel : A systemic grammar for text generation*.
- Milhorat, P., Schlögl, S., Chollet, G., , B., Esposito, A., and Pelosi, G. (2014). Building the next generation of personal digital assistants.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518.

- Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197.
- Narang, S. and Gupta, M. D. (2015). Speech feature extraction techniques : A review.
- Olah, C. (2015). Understanding lstm networks – colah’s blog. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. (Accessed on 02/19/2019).
- O’Shaughnessy, D. D. (1988). Linear predictive coding. *IEEE Potentials*, 7:29–32.
- Rabiner, R. and Juang, B. H. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, 3:4–16.
- Rahali, H., Hajaiej, Z., and Ellouze, N. (2014). Robust features for speech recognition using temporal filtering technique in the presence of impulsive noise. *International Journal of Image, Graphics and Signal Processing*, 6:17–24.
- Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1).
- Richard Bellman", t. . A. M. D. P. (1957). *Indiana Univ. Math. J.*", *fjournal* = "Indiana University Mathematics Journal", 6:679–684.
- Roark, B., Saraclar, M., and Collins, M. (2007). Discriminative n-gram language modeling. *Comput. Speech Lang.*, 21(2).
- Rosenblatt, F. (1959). *Perceptron simulation experiments (Project Para)*.
- Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. J. (2007). Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *HLT-NAACL*.
- Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11).
- Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, pages 3776–3783. AAAI Press.
- Sonali, B. (2014). Research paper on basic of artificial neural network. page 1.
- Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, W. B. (2015). A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*, pages 196–205.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.
- Theune, M., Klabbers, E., De Pijper, J. R., Krahmer, E., and Odijk, J. (2001). From data to speech : A general approach. *Natural Language Engineering*, 7(1).
- Velay, M. and Daniel, F. (2018). Seq2seq and multi-task learning for joint intent and content extraction for domain specific interpreters. *CoRR*, abs/1808.00423:3–4.

- Wang, Y., Shen, Y., and Jin, H. (2018). A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.
- Weisz, G., Budzianowski, P., hao Su, P., and Gax0161ix0107, M. (2018). Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26:2083–2097.
- Wen, T.-H., Gasic, M., Mrksic, N., hao Su, P., Vandyke, D., and Young, S. J. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.
- Wen, T.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., hao Su, P., Ultes, S., Vandyke, D., and Young, S. J. (2017). A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449.
- Williams, J. D. and Young, S. (2007). Scaling pomdps for spoken dialog management. *Trans. Audio, Speech and Lang. Proc.*, 15(7):2116–2129.
- Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model : A practical framework for pomdp-based spoken dialogue management. *Comput. Speech Lang.*, 24(2):150–174.
- Yu, D. and Deng, L. (2015). *Automatic Speech Recognition*. Springer London.
- Yu, J., Reiter, E., Hunter, J., and Mellish, C. (2007). Choosing the content of textual summaries of large time-series data sets. *Natural Language Engineering*, 13(1).