

Table des matières

1 Composants de base d'un assistant personnel	2
1.1 Introduction	2
1.2 Notions et aspects théoriques	2
1.2.1 Apprentissage automatique	2
1.2.2 Réseaux de neurones artificiels	3
1.3 Schéma global d'un SPA	9
1.4 Reconnaissance automatique de la parole (ASR)	10
1.4.1 Acquisition du signal et extraction d'attributs	11
1.4.2 Modélisation acoustique et modélisation du lexique	11
1.4.3 Modélisation de la langue	11
1.5 Compréhension du langage naturel (NLU)	11
1.5.1 Classification d'intentions	12
1.5.2 Extraction d'entités	13
1.5.3 Analyse sémantique	14
1.6 Gestion du dialogue	14
1.6.1 Modélisation	15
1.6.2 État du gestionnaire de dialogue	16
1.6.3 Politique de gestion de dialogue	18
1.7 Génération du langage naturel (NLG)	21
1.7.1 Détermination du contenu	21
1.7.2 Structuration de texte	21
1.7.3 Agrégation de phrases	22
1.7.4 Lexicalisation	22
1.7.5 Génération d'expressions référentielles (REG)	22
1.7.6 Réalisation linguistique	22
1.7.7 Systèmes basés encodeur-décodeur	24
1.8 Conclusion	26

To correct

To explain more deeply

Chapitre 1

Composants de base d'un assistant personnel

1.1 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 base ainsi que des techniques d'apprentissage automatique, nous traiterons 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 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 modules qui les utilisent.

1.2.1 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 [45], l'auteur a 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?"[45]

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 [45] 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 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 externe, 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 de minimiser au fur et à mesure qu'une nouvelle instance lui est présentée [37]. 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 [4]
- **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 1.6.3.3)

1.2.2 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 traitent l'information d'une manière parallèle dans le but de résoudre un problème bien spécifique [62]. Ils ont notamment connu un très grand succès lors des dernières années dans différents domaines comme la reconnaissance d'images [64], la reconnaissance automatique de la parole [29, 73] ou encore la classification de textes [66, 44].

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?

1.2.2.1 Réseaux de neurones multicouches denses

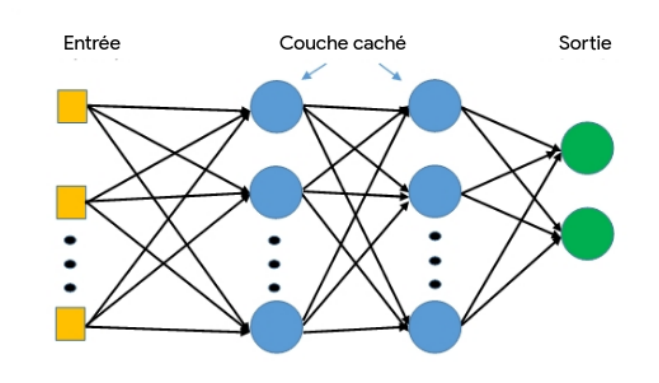


Figure 1.1 – Architecture basique d'un réseaux de neurones multicouches

La forme la plus basique que peut avoir un réseau de neurones est celle d'un réseau multicouches comme montré dans 1.1, elle se compose de trois parties :

- **Une couche d'entrée** : cette couche est composé d'un ensemble de neurones ou *perceptron* [58], 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 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 :

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

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 parcours des données d'apprentissage [49]

1.2.2.2 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 brisées 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 convolutif et récurrents [40]

1.2.2.3 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 [42].

Les réseaux de neurones récurrents (RNNs) sont des réseaux de neurones *feedforward* dont certaines connections 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[42]

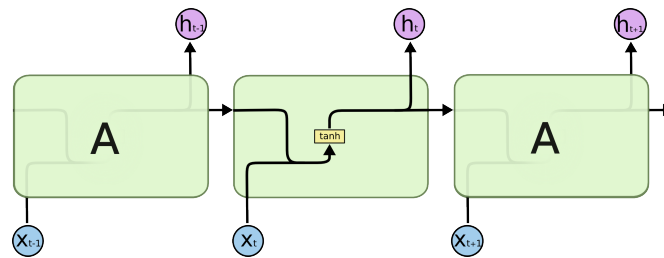


Figure 1.2 – Architecture interne d'un réseau de neurones récurrent à un instant t [51]

Un des principaux problèmes que rencontrent les RNNs est celui du *Vanishing gradient* traduit par la *Problème de disparition du gradient* [33], les relations à long termes 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.

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

Premièrement introduit en 1997 par *Sepp Hochreiter* et al. dans [34], 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, ainsi nous pouvons décortiquer les composantes d'une cellule LSTM (voir 1.3) comme suit :

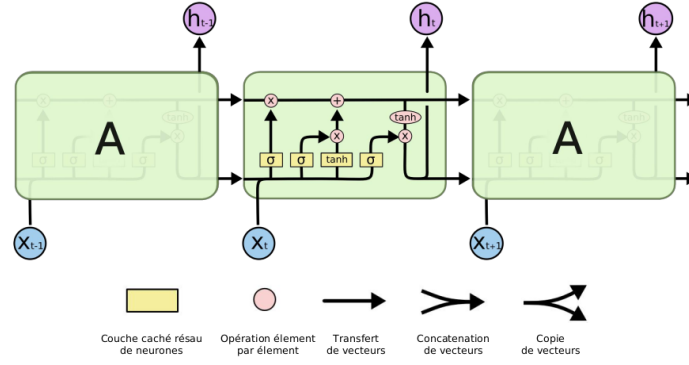


Figure 1.3 – Architecture interne d'une cellule mémoire dans un réseau LSTM [51]

- **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 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 [34].

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

- **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 [34, 42].

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

- **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 filtré par les portes f_c et i_c [34].

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

- **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 [51, 34, 42].

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

1.2.2.5 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' 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 [35].

$$P(w|t) = \frac{\text{Comptage}(t, w)}{\text{Comptage}(t)} \quad (1.8)$$

1.2.2.6 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 [23, 36].

D'un façon plus formelle, un HMM est un 5-tuple $\langle S, V, \Pi, A, B \rangle$ [53] 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 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 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 $i = 1 \dots N$.

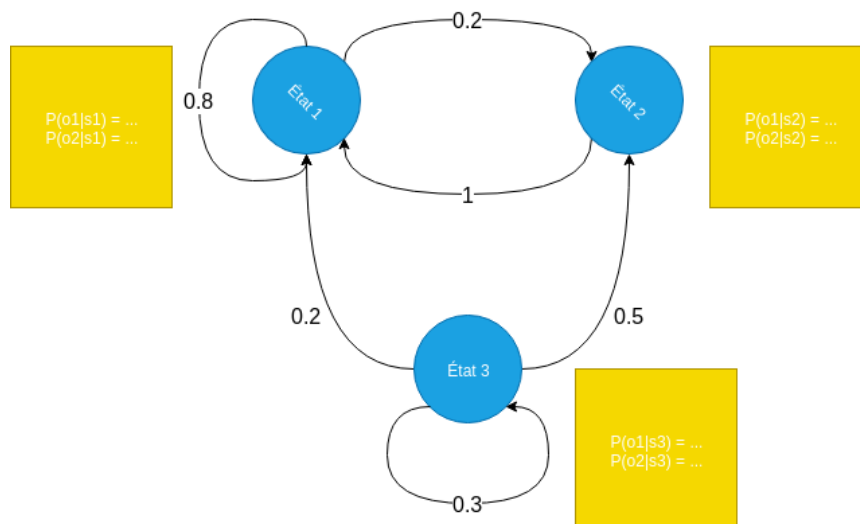


Figure 1.4 – Exemple d’une distribution de probabilité de transition ainsi qu’une distribution de probabilité d’observations pour un HMM à 3 états et 2 observations

Le but serait de trouver la séquence d'états $S_{1:K}$ qui maximiserait la probabilité [23] :

$$P(Y_{1:K}|O_{1:K}) = P(O_1)P(O_1|S_1) \prod_{t=2}^K P(S_t|S_{t-1})P(O_t|S_t) \quad (1.9)$$

Pour y parvenir d'une manière efficace, un décodeur qui implémente l'algorithme de viterbi peut être utiliser [20, 8]

1.3 Schéma global d'un SPA

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

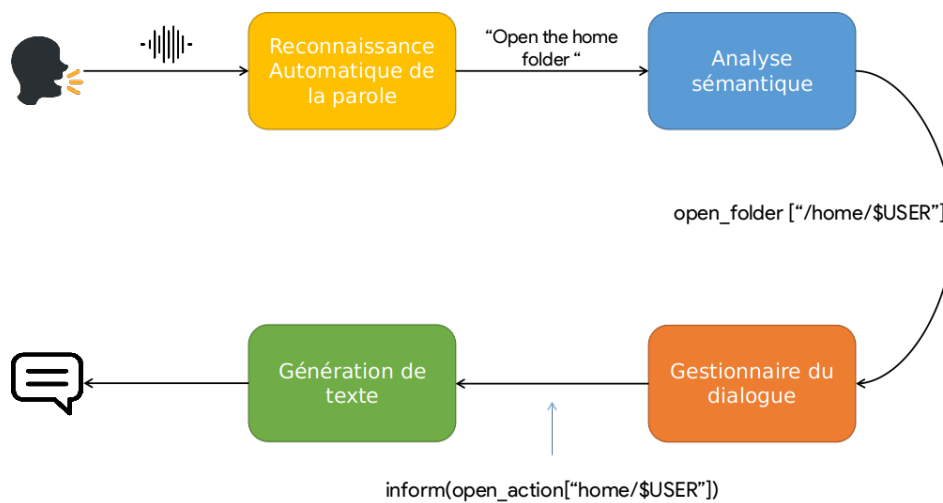


Figure 1.5 – Schéma abstrait d'un SPA [47]

Le processus peut être résumé en des étapes cruciales (qui seront détaillées dans la section suivante) :

- 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 compactera les principales informations contenues dans la requête, à savoir l'intentions de l'utilisateur et ses arguments.
- Au fur et à mesure que l'utilisateur énonce des requête, le système doit pouvoir être capable d'utiliser le contexte du dialogue pour interagir avec ce dernier pour pouvoir atteindre le but final du dialogue (exécuter uen commande système, trouver des informations sur le web 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 sera préférable de transformer la trame sémantique (voir 1.9) en texte naturelle pour l'afficher en sortie.

Il est à noter que chacun des modules seront indépendant 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 en est de la suite du chapitre, nous allons présenter les différents modules ainsi que les techniques et architectures qui sont considérés comme étant l'état de l'art du domaine.

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 la locution d'un utilisateur en un texte qui peut être interprété par la machine [1]. Différentes approches ont été développées 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 [8], la séquence d'observations passe par le HMM et la meilleure séquence de mots est sélectionnée [73].

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.

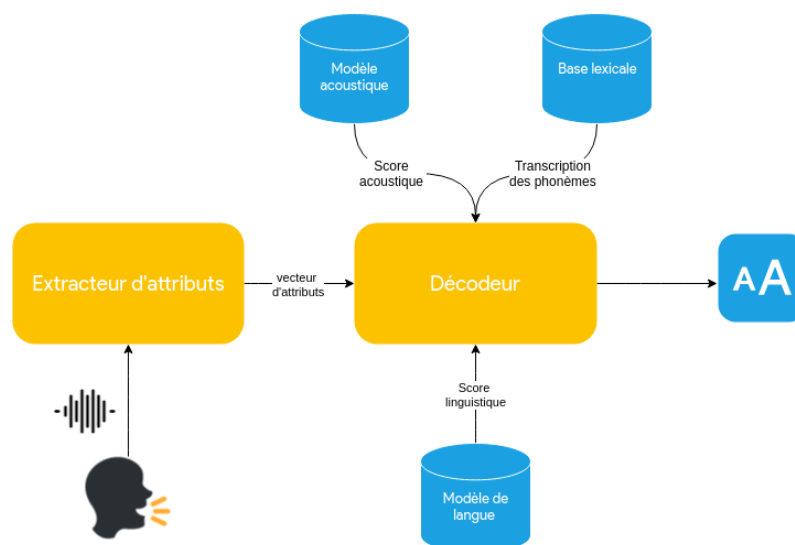


Figure 1.6 – Architecture des systèmes de reconnaissance de la parole [73]

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 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 [50]. Un tel processus peut utiliser plusieurs techniques pour la mise à l'échelle : MFCC (Mel-frequency cepstrum) [10], LPC (Linear Predictive coding) [52] et RASTA (RelAtive SpecTrAl) [54], 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 on utilisant un modèle HMM et en disposant d'un corpus de fichiers audio accompagnés de leurs transcriptions en phonèmes et en texte [24, 53], ou bien un réseau de neurones profonds [73], ou encore une hybridation de ces derniers [14]. 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.

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 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 [25]. 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 [35, 73, 57]

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 est principalement focalisé 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. Une intention (ou intent) est une abstraction 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.

1.5.1 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 basé sur les réseaux de neurones récurrents à mémoire court et long terme (LSTM) (voir 1.3). Dans [43] et [27] deux architectures ont fait leurs preuves, dans le premier travail les auteurs ont utilisé une architecture BLSTM (LSTM Bidirectionnel) pour capturer les contextes droit et gauche d'un mot donné [60], le dernier état caché retourné par la cellule LSTM est ensuite utilisé pour la classification, une couche d'attention est ajoutée [12] 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 est le temps d'apprentissage assez réduit au détriment d'une erreur de classification moins accrue mais toujours dans les normes.

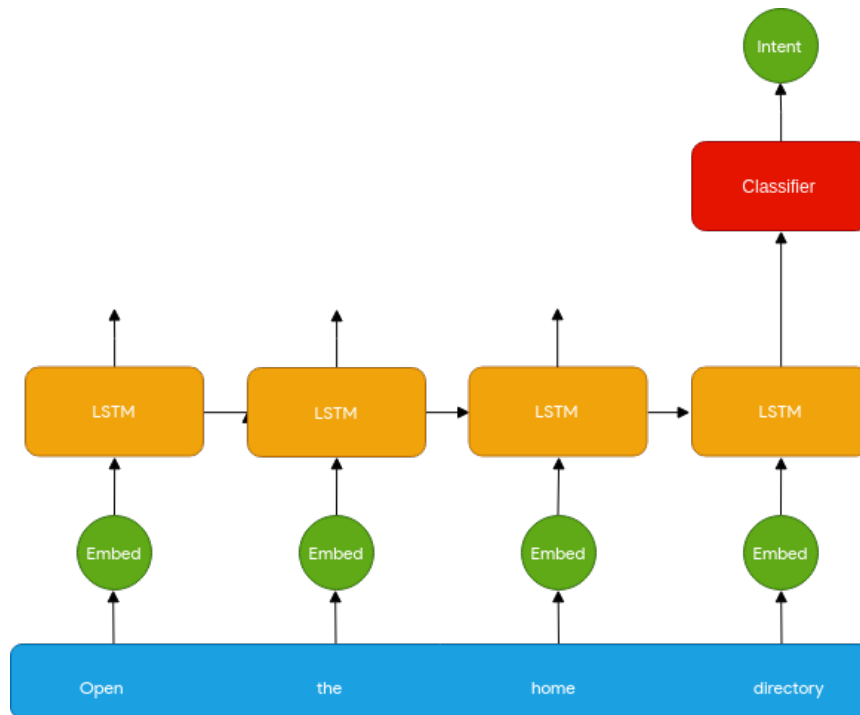


Figure 1.7 – Architecture de base d’un classificateur d’intents [43]

1.5.2 Extraction d’entités

Déterminer l’intention d’un utilisateur ne s’arrête pas qu’au stade de l’identification de ce dit *intent*, en effet pour mieux représenter l’information récoltée depuis la requête il faut en extraire des entités (nommés ou spécifiques au 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 [27] 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 :

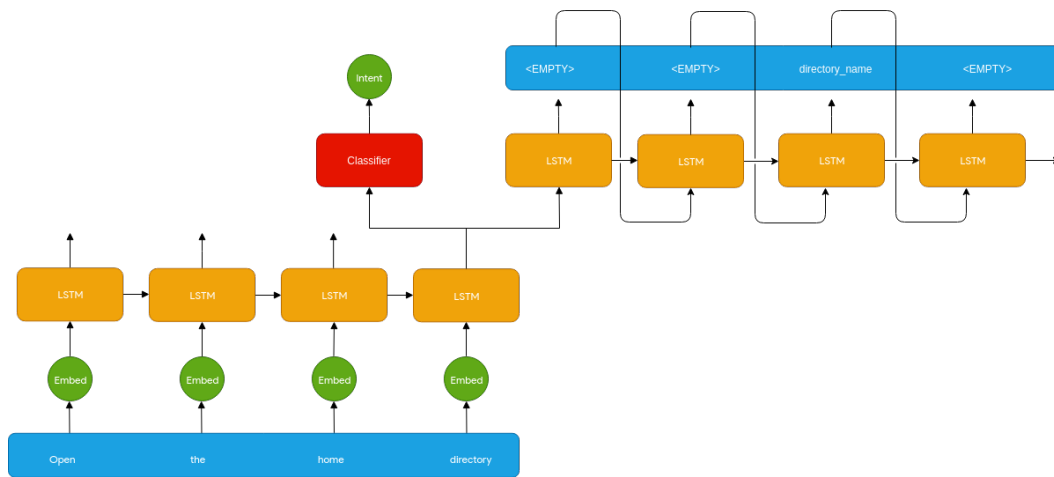


Figure 1.8 – Architecture de base d'un classificateur d'intents doublé d'un extracteur d'entité [27]

1.5.3 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 [43, 27, 67].

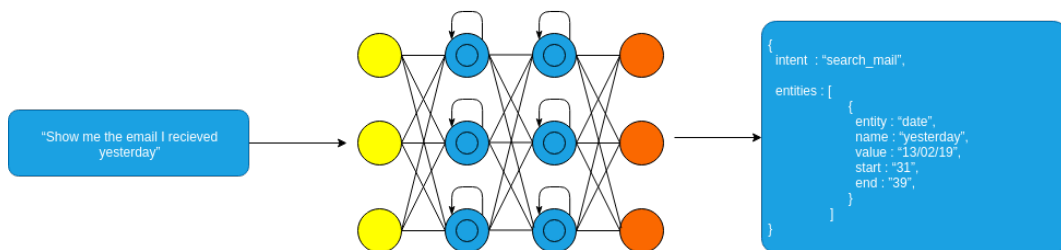


Figure 1.9 – Exemple d'une trame sémantique pour une requête donnée

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.

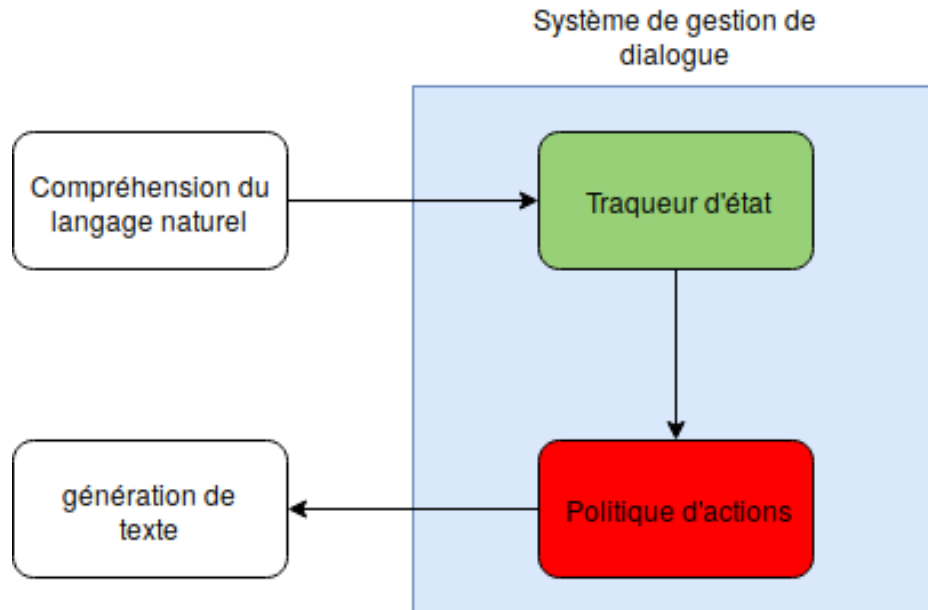


Figure 1.10 – Schéma général d'un gestionnaire de dialogue

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 en 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)[56]. 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.

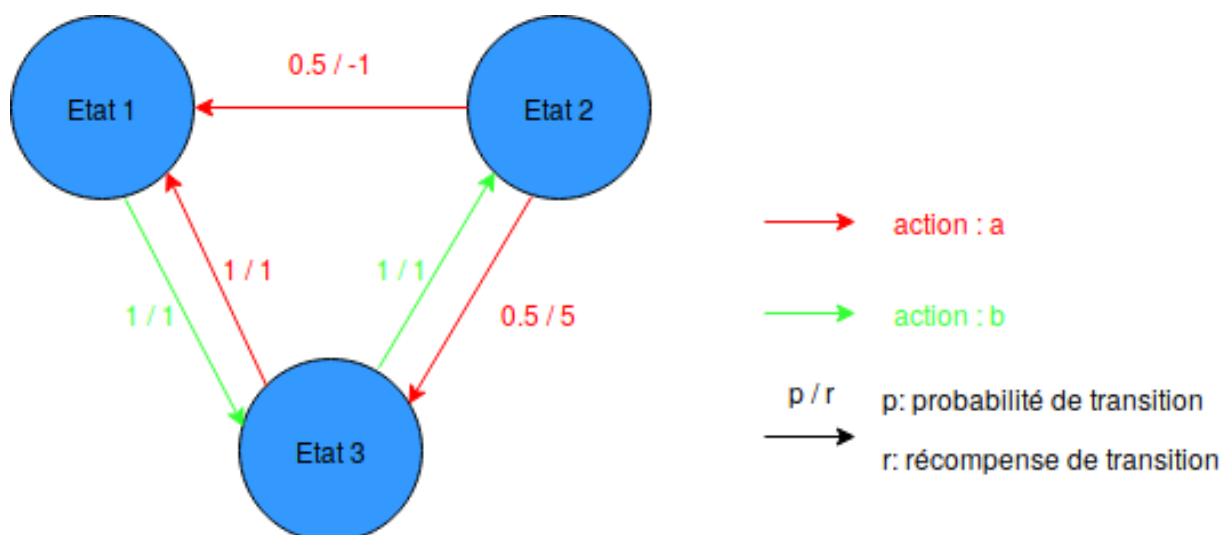


Figure 1.11 – Schéma représentant les transitions entre états dans un MDP

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 [11]. Cette structure contient des emplacements à remplir sur un domaine donné. La figure 1.12 illustre un exemple de cadre sémantique.

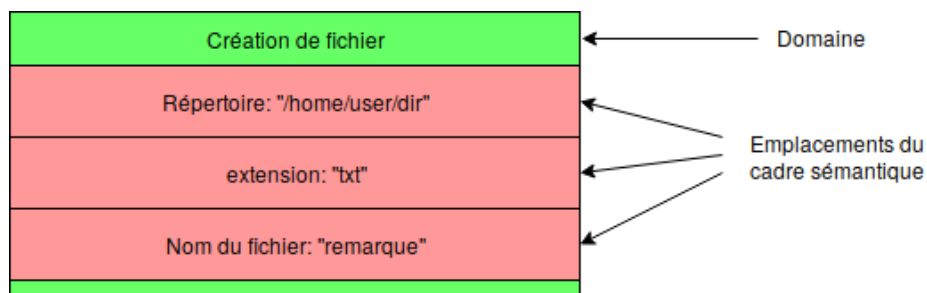


Figure 1.12 – Schéma représentant un cadre sémantique avec comme domaine : création de fichier

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 [71].

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 [26]. Cependant, les bases de

règles sont très susceptibles à faire des erreurs[11] car elles sont peu robustes face aux incertitudes.

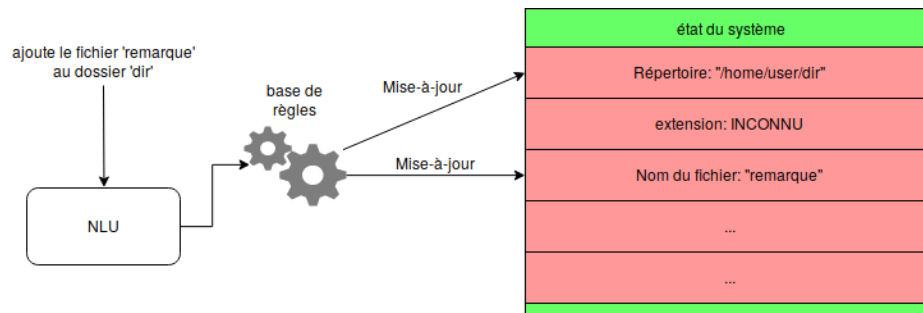


Figure 1.13 – Schéma représentant la mise-à-jour de l'état par un système basé règles

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)[72] 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.

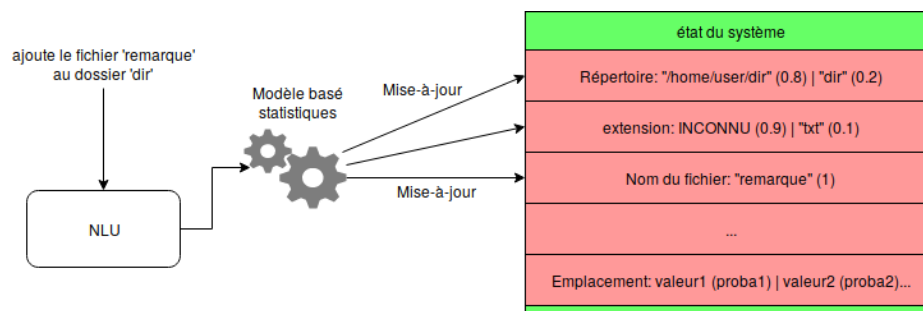


Figure 1.14 – Schéma représentant la mise-à-jour de l'état par un système basé statistiques

Processus de décision markovien partiellement observable (POMDP) Comme dans les processus de décision markovien, un POMDP[3] 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.

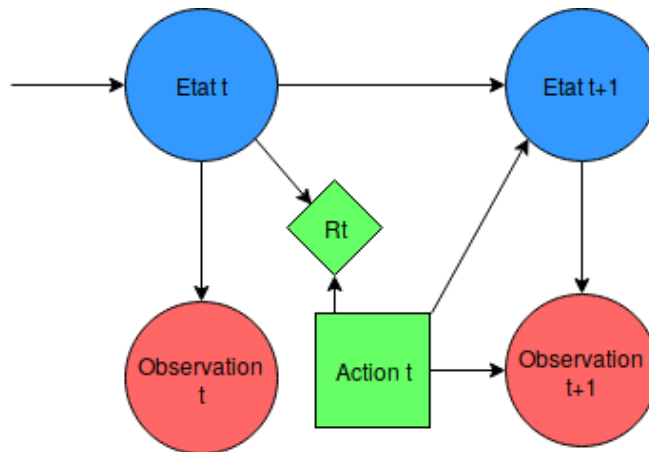


Figure 1.15 – *Diagramme d'influence dans un POMDP*

1.6.2.3 Suivi de l'état avec réseaux de neurones profonds

Récemment, des approches utilisant les réseaux de neurones profonds (voir 1.2.2.2) 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[32]. 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, naboutit pas à des résultats flexibles qui peuvent suivre le flux du dialogue convenablement[41].

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[31], 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 Seq2Seq² afin de produire directement une sortie à partir des informations reçues de l'utilisateur[70, 61].

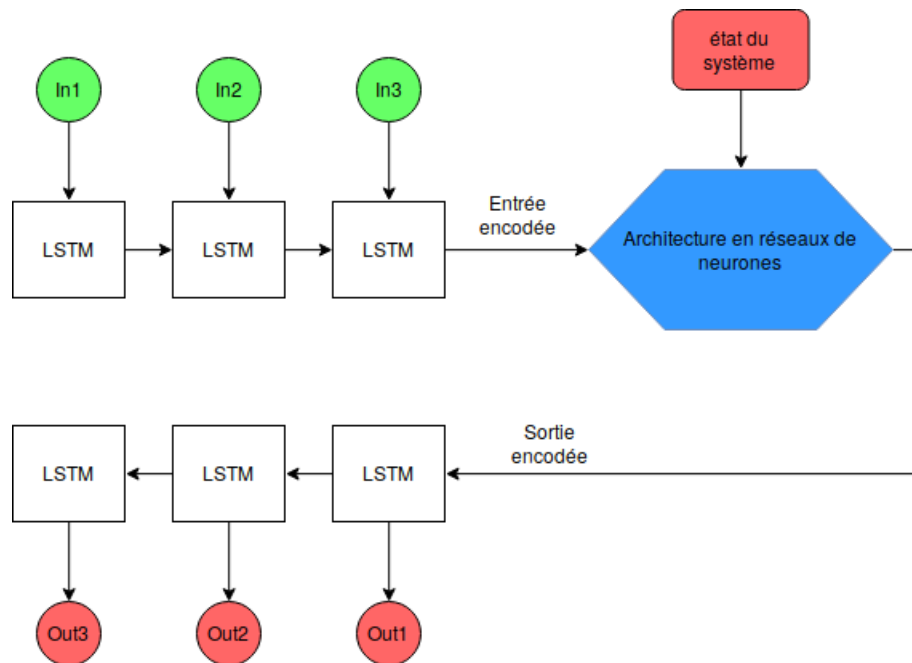


Figure 1.16 – Schéma de gestion de dialogue de bout en bout avec architecture Seq2Seq

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[68]. 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.

2. 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).

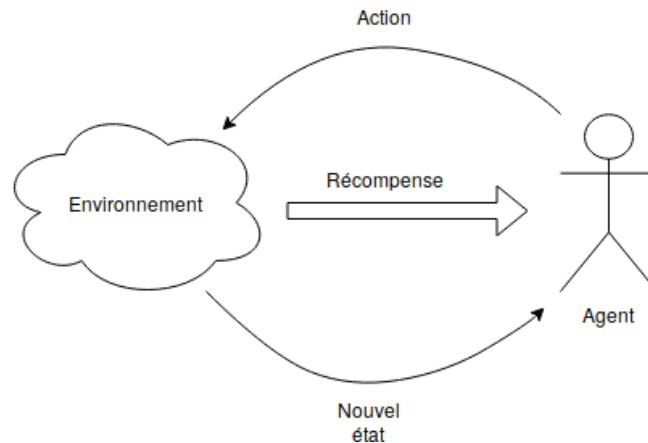


Figure 1.17 – Schéma d'interaction agent-environnement dans l'apprentissage par renforcement

Il existe plusieurs méthodes que l'agent peut utiliser pour estimer la fonction de récompense [7], notamment Deep Q Learning (DQN) [48] 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. Doù, la nécessité d'utiliser un **simulateurs 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 basé règles : dans ce cas une liste de règles est écrite manuellement que le simulateur doit suivre pour communiquer avec le système[59].
- Les simulateurs 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[22].
- Les simulateurs 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 de l'utilisateur sachant l'état du système ou l'action du système directement dans le modèle HMM[13].
- 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[9].

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[18]. 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 [55] 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 [38], utilisent les mêmes techniques pour des entrées linguistiques. Enfin, la génération du langage naturel peut être très proche de la gestion de dialogue[15]. 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 [55].

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[74]. De plus, le choix de l'information peut aussi dépendre de l'utilisateur et de ses connaissances[15]. 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[21].

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[55]. 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 [21].

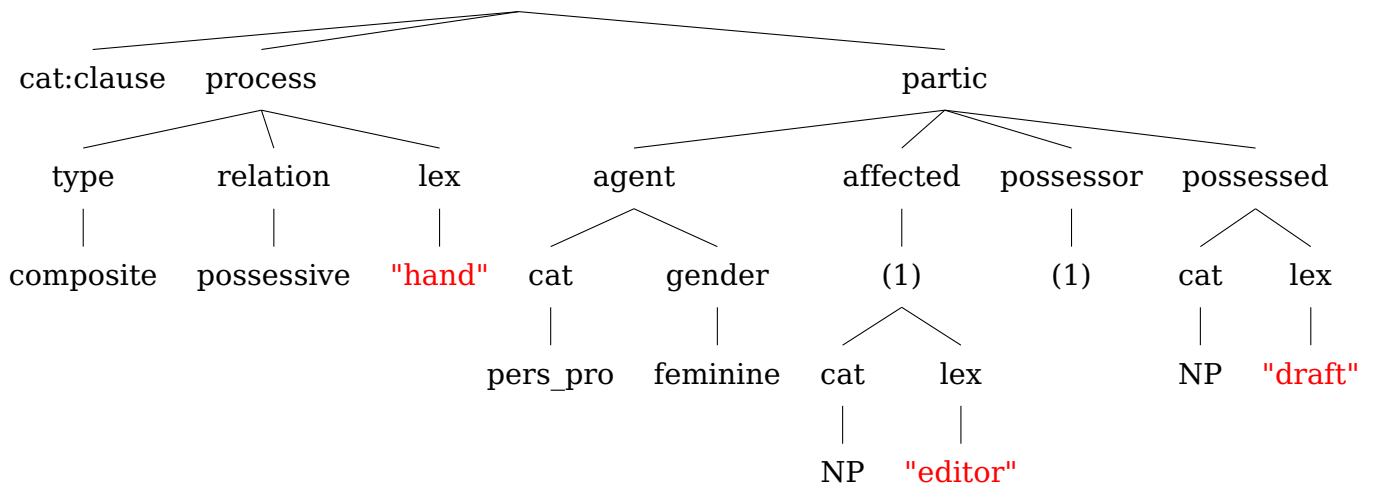


Figure 1.18 – Un exemple d'entrée de SURGE[16]

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 [65]. 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[2], 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)[30] qui a été largement utilisée comme dans NIGEL[46] ou KPML[5]. 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[16] : 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[21].

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 [21] :

- 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)[39]).
- La deuxième méthode utilise les informations statistiques directement au niveau de la génération pour produire la solution optimale (exemple : Belz (2008)[6]).

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 l'utilisation des Treebanks³ pour déduire les règles de grammaire [17].

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 [19].

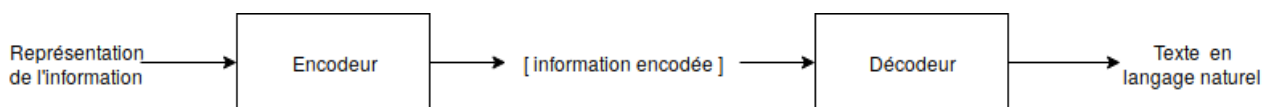


Figure 1.19 – Schéma d'une architecture encodeur-décodeur pour NLG

Beaucoup d'approches de génération de langage naturel en gestion de dialogue utilisent des encodeur-décodeur. Wen et al. (2015)[69] utilise par exemple des LSTMs 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 récurrents pour encoder

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

l'état du gestionnaire de dialogue et l'entrée reçue suivie par un décodeur pour générer le texte de la réponse[63, 61, 28].

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.

Table des figures

1.1	Architecture basique d'un réseaux de neurones multicouches	4
1.2	Architecture interne d'un réseau de neurones récurrent à un instant t [51]	5
1.3	Architecture interne d'une cellule mémoire dans un réseau LSTM [51]	6
1.4	Exemple d'une distribution de probabilité de transition ainsi qu'une distribu- tion de probabilité d'observations pour un HMM à 3 états et 2 observations	8
1.5	Schéma abstraitif d'un SPA [47]	9
1.6	Architecture des systèmes de reconnaissance de la parole [73]	10
1.7	Architecture de base d'un classificateur d'intents [43]	13
1.8	Architecture de base d'un classificateur d'intents doublé d'un extracteur d'en- tité [27]	14
1.9	Exemple d'une trame sémantique pour une requête donnée	14
1.10	Schéma général d'un gestionnaire de dialogue	15
1.11	Schéma représentant les transitions entre états dans un MDP	16
1.12	Schéma représentant un cadre sémantique avec comme domaine : création de fichier	16
1.13	Schéma représentant la mise-à-jour de l'état par un système basé règles	17
1.14	Schéma représentant la mise-à-jour de l'état par un système basé statistiques .	17
1.15	Diagramme d'influence dans un POMDP	18
1.16	Schéma de gestion de dialogue de bout en bout avec architecture Seq2Seq . .	19
1.17	Schéma d'interaction agent-environnement dans l'apprentissage par renforce- ment	20
1.18	Un exemple d'entrée de SURGE[16]	23
1.19	Schéma d'une architecture encodeur-décodeur pour NLG	24

Bibliographie

- [1] Fawaz Al-Anzi and Dia AbuZeina. Literature survey of arabic speech recognition. pages 1–6, 03 2018.
- [2] Gabor Angeli, Christopher D. Manning, and Daniel 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, 2012. Association for Computational Linguistics.
- [3] K. J. Åström. Optimal control of Markov Processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, January 1965.
- [4] H.B. Barlow. Unsupervised learning. *Neural Computation*, 1(3):295–311, 1989.
- [5] John A. Bateman. Enabling technology for multilingual natural language generation : The kpml development environment. *Nat. Lang. Eng.*, 3(1), March 1997.
- [6] Anja Belz. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.*, 14(4), October 2008.
- [7] Dimitri Bertsekas. *Dynamic Programming & Optimal Control, Vol II : Approximate Dynamic Programming*. Athena Scientific, 01 2012.
- [8] Julien Bloit and Xavier Rodet. Short-time viterbi for online hmm decoding : Evaluation on a real-time phone recognition task. pages 2121 – 2124, 05 2008.
- [9] Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. User simulation in dialogue systems using inverse reinforcement learning. In *INTER-SPEECH*, page 10251028, 2011.
- [10] 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 (ICGC-CEE)*, pages 1–5, March 2014.
- [11] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems : Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35, November 2017.
- [12] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015.
- [13] Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. Human-computer dialogue simulation using hidden markov models. *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 290–295, 2005.

- [14] li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications : An overview. pages 8599–8603, 10 2013.
- [15] Nina Dethlefs. Context-sensitive natural language generation : From knowledge-driven to data-driven techniques. *Language and Linguistics Compass*, 8, 03 2014.
- [16] Michael Elhadad and Jacques Robin. An overview of surge : a reusable comprehensive syntactic realization component. In *International Natural Language Generation Workshop*, 1996.
- [17] Dominic Espinosa, Michael White, and Dennis Mehay. Hypertagging : Supertagging for surface realization with ccg. In *ACL*, 2008.
- [18] Roger Evans, P. Piwek, and Lynne Cahill. What is nlg ? In *Proceedings of the Second International Conference on Natural Language Generation*, 2002.
- [19] Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. Linguistic realisation as machine translation : Comparing different mt models for amr-to-text generation. In *INLG*, 2017.
- [20] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.
- [21] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation : Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1), January 2018.
- [22] Kallirroi Georgila, James Henderson, and Oliver Lemon. Learning user simulations for information state update dialogue systems. In *INTERSPEECH*, pages 893–896, 2005.
- [23] Zoubin Ghahramani. 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, 2002.
- [24] Wiqas Ghai and Navdeep Singh. Literature review on automatic speech recognition. *International Journal of Computer Applications*, 41(8):42–50, March 2012. Full text available.
- [25] R Glass and Michael Kyle Mccandless. Automatic acquisition of language models for speech recognition. 10 1994.
- [26] D Goddeau, Helen Meng, Joseph Polifroni, Stephanie Seneff, and S Busayapongchai. A form-based dialogue manager for spoken language applications. volume 2, pages 701 – 704, 11 1996.
- [27] Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 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, June 2018. Association for Computational Linguistics.
- [28] Raghav Goyal, Marc Dymetman, and Eric Gaussier. Natural language generation through character-based rnns with finite-state prior knowledge. In *COLING*, pages 1083–1092, 2016.

- [29] 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*, pages 6645–6649, May 2013.
- [30] M. A. K. Halliday and Christian M. I. M. Matthiessen. *Introduction to Functional Grammar*. Hodder Arnold, London, 3 edition, 2004.
- [31] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Comput. Linguist.*, 34(4):487–511, December 2008.
- [32] Matthew Henderson, Blaise Thomson, and Steve J. Young. Deep neural network approach for the dialog state tracking challenge. In *SIGDIAL Conference*, pages 467–471, 2013.
- [33] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998.
- [34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
- [35] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, 2nd Edition*. Prentice Hall, 2008.
- [36] John G. Kemeny and J. Laurie Snell. Markov processes in learning theory. *Psychometrika*, 22(3):221–230, Sep 1957.
- [37] Sotiris Kotsiantis, I Zaharakis, and P Pintelas. Machine learning : A review of classification and combining techniques. *Artificial Intelligence Review*, 26:159–190, 11 2006.
- [38] Cyril Labbé and François Portet. Towards an abstractive opinion summarisation of multiple reviews in the tourism domain. *CEUR Workshop Proceedings*, 917, 01 2012.
- [39] Irene Langkilde-Geary. Forest-based statistical sentence generation. In *ANLP*, 2000.
- [40] Yann LeCun, Y Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.
- [41] Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, Donghyeon Lee, and Gary Geunbae Lee. Recent approaches to dialog management for spoken dialog systems. *JCSE*, 4:1–22, 2010.
- [42] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.
- [43] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *CoRR*, abs/1609.01454, 2016.
- [44] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *CoRR*, abs/1605.05101, 2016.
- [45] Tom M Mitchell. The discipline of machine learning. 01 2006.
- [46] William C. Mann and Christian M. I. M. Matthiessen. *Nigel : A systemic grammar for text generation*. 1983.
- [47] Pierrick Milhorat, Stephan Schlögl, Gerard Chollet, Boudy , Anna Esposito, and G Pelosi. Building the next generation of personal digital assistants. 03 2014.

- [48] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharmashan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.
- [49] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, jul 1991.
- [50] Shreya Narang and Ms. Divya Gupta. Speech feature extraction techniques : A review. 2015.
- [51] Christopher Olah. Understanding lstm networks – colah’s blog. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>, Août 2015. (Accessed on 02/19/2019).
- [52] Douglas D. O’Shaughnessy. Linear predictive coding. *IEEE Potentials*, 7:29–32, 1988.
- [53] R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3:4–16, 1986.
- [54] Hajer Rahali, Zied Hajaiej, and Nouredine Ellouze. 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, 10 2014.
- [55] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1), March 1997.
- [56] title = "A Markovian Decision Process Richard Bellman". *Indiana Univ. Math. J.*, *fjournal* = "Indiana University Mathematics Journal", 6:679–684, 1957.
- [57] Brian Roark, Murat Saraclar, and Michael Collins. Discriminative n-gram language modeling. *Comput. Speech Lang.*, 21(2), April 2007.
- [58] F. Rosenblatt. *Perceptron simulation experiments (Project Para)*. 1959.
- [59] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *HLT-NAACL*, 2007.
- [60] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11), November 1997.
- [61] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 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, 2016.
- [62] B.Maind. Sonali. Research paper on basic of artificial neural network. page 1, 2014.
- [63] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*, pages 196–205, 2015.
- [64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

- [65] M. Theune, E. Klabbers, J. R. De Pijper, E. Krahmer, and J. Odijk. From data to speech : A general approach. *Natural Language Engineering*, 7(1), March 2001.
- [66] Marc Velay and Fabrice Daniel. Seq2seq and multi-task learning for joint intent and content extraction for domain specific interpreters. *CoRR*, abs/1808.00423:3–4, 2018.
- [67] Yu Wang, Yilin Shen, and Hongxia Jin. 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, June 2018. Association for Computational Linguistics.
- [68] Gellért Weisz, Paweł Budzianowski, Pei hao Su, and Milica Gax0161ix0107. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26:2083–2097, 2018.
- [69] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei hao Su, David Vandyke, and Steve J. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, 2015.
- [70] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449, 2017.
- [71] J. D. Williams and S. Young. Scaling pomdps for spoken dialog management. *Trans. Audio, Speech and Lang. Proc.*, 15(7):2116–2129, September 2007.
- [72] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model : A practical framework for pomdp-based spoken dialogue management. *Comput. Speech Lang.*, 24(2):150–174, April 2010.
- [73] Dong Yu and Li Deng. *Automatic Speech Recognition*. Springer London, 2015.
- [74] Jin Yu, Ehud Reiter, Jim Hunter, and Chris Mellish. Choosing the content of textual summaries of large time-series data sets. *Natural Language Engineering*, 13(1), March 2007.