Étude des modèles neuronaux profonds pour la compréhension automatique du langage naturel dans les habitats intelligents

Anastasiia Mishakova

Étude des modèles neuronaux profonds pour la compréhension automatique du langage naturel dans les habitats intelligents

MISHAKOVA

Anastasiia

Sous la direction de François PORTET et Michel VACHER

Laboratoire : GETALP-LIG

UFR LLASIC

Département Sciences du langage et FLE

Mémoire de master 2 Sciences du Langage orientation Recherche - 30 crédits

Parcours : Industries de la Langues

Année universitaire 2017-2018

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage, surtout mes encadrants, M. François Portet et M. Michel Vacher qui m'ont beaucoup aidée à toutes les étapes de la rédaction de ce mémoire et des expérimentations.

Je remercie aussi Thierry Desot, mon collègue avec qui j'ai travaillé sur ce projet, pour son collaboration et l'aide.

Enfin, j'adresse mes remerciements au CNRS qui a financé mon stage dans le cadre du projet VocADom.

UNIVERSITÉ Grenoble Alpes

DÉCLARATION

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

NOM : MISHAKOVA

PRENOM : Anastasiia

DATE : 21.06.2018 SIGNATURE : [redacted]

Résumé

Le but du projet VocADom est de créer un système de compréhension automatique des ordres vocaux qui pourrait être fonctionnel dans un habitat intelligent pour les personnes âgées. L'approche par apprentissage automatique nous semble plus flexible que l'approche par règles, ce dernier ne prenant pas en compte les énoncés qui ne se conforment pas à la grammaire ; et les études ont montré que les personnes âgées sont enclines à s'écarter de la grammaire imposée.

Pourtant il y a un manque de données d'apprentissage dans ce domaine. Pour contourner ce problème, une grammaire générant le corpus artificiel pour le domaine de l'habitat intelligent a été créé. Un des problèmes traités par ce mémoire est l'amélioration de la grammaire existante. Après amélioration, le corpus artificiel résultant compte 42195 phrases annotées. Il a été ensuite évalué en entraînant avec lui trois modèles états de l'art - Tri-CRF, att-RNN, RASA - ainsi que Tf-seq2seq, modèle séquence-vers-séquence. Ces modèles ont ensuite été testés sur un petit corpus d'ordres naturels enregistrés dans le cadre du projet VocADom. Cela nous a permis de voir les limites du corpus artificiel. Ces modèles ont ensuite été comparés aux modèles entraînés et testés sur le corpus de données réelles PORTMEDIA qui relève du domaine du tourisme. Cette comparaison nous a permis de savoir que la performance des modèles entraînés sur le corpus artificiel est due aux limitations du corpus artificiel plutôt qu'aux limitations des modèles eux-mêmes. Tous les modèles entraînés et testés sur PORTMEDIA avaient rendu de bons résultats mais les modèles entraînés sur le corpus artificiel et testés sur le corpus naturel VocADom ont été bien moins performants. Ceci montre la difficulté à prendre en charge la diversité lexicale et syntaxique d'un corpus réel.

Tous les modèles ont des performances comparables, mais Tf-seq2seq, contrairement à Tri-CRF, att-RNN et RASA, ne requiert pas de données alignées et est plus facile à entraîner.

Mots-clés : compréhension automatique de la langue, habitat intelligent, apprentissage automatique, modèles profonds, génération du corpus

Abstract

The goal of the VocADom project is to create a natural language understanding system for processing voice orders in a smart home for the elderly. Machine learning approach seems more flexible than the rule-based approach, which does not take into account orders that do not conform to grammar ; and studies have shown that older people are inclined to deviate from imposed grammar.

However there is a lack of learning data in this domain. To remedy the problem, a grammar generating the artificial corpus for the smart home domain was created. One of the goals of this work is to improve this grammar. After the improvement, the resulting artificial corpus has 42195 annotated sentences. It was then evaluated by training three state-of-the-art models on it - Tri-CRF, att-RNN, RASA, as well as Tf-seq2seq, a sequence-to-sequence model. The models were then tested on a small natural existing corpus recorded as part of the VocADom project. This allowed us to see the limitations of the artificial corpus. These models were then compared to the models that were trained and tested on the PORTMEDIA corpus of touristic domain. This comparison allowed us to know that the performance of the models trained on the artificial corpus is due to the limitations of the artificial corpus rather than the limitations of the models themselves. All the models trained and tested on PORTMEDIA gave good results but the models trained on the artificial corpus and tested on the VocADom corpus were much less efficient. This shows the difficulty of accounting for the lexical and syntactic diversity of the real data.

All models show comparable performances but Tf-seq2seq, unlike Tri-CRF, att-RNN et RASA, doesn't require aligned data and is easier to train.

Keywords : natural language understanding, smart home, machine learning, deep learning, corpus generation

Table des matières

Glossaire 11

1 Introduction **13**

2 État de l'art **15**

2.1 L'interaction vocale dans l'habitat intelligent 15

2.1.1 Études sur l'interaction dans l'habitat 15

2.1.2 Études sur les particularités langagières de l'interaction des personnes âgées 16

2.2 La compréhension de la langue naturelle 18

2.2.1 Définitions 18

2.2.2 Problèmes existants 19

2.2.3 Représentation sémantique par les slots 19

2.2.4 Modèles de compréhension automatique du langage naturel 20

Approche par règles 20

Approche statistique 21

Architecture : pipeline par opposition à de bout-à-bout 25

Détection d'intention et le remplissage des slots 27

Champ aléatoire conditionnel triangulaire 28

Logiciel commercial RASA 29

RNN du type encodeur-décodeur bidirectionnel basé sur l'attention . . 29

Tf-seq2seq 31

2.2.5 Développement d'un modèle 32

2.2.6 Évaluation du modèle 33

2.3 Les corpus 35

2.3.1 Généralités 35

Annotation des corpus 35

Découpage des corpus 36

2.3.2 Corpus existants 36

ATIS 36

MEDIA et PORTMEDIA 37

SWEET-HOME 38

VocADom 39

2.4 Travaux préliminaires dans le cadre du projet VocADom 41

3 Problématique et méthode **45**

4 Génération et évaluation du corpus artificiel **47**
4.1 La modification de la grammaire du corpus artificiel . . . 47
4.1.1 Redéfinition des intentions . . . 47
4.1.2 Enrichissement sémantique . . . 48
4.1.3 Enrichissement syntaxique . . . 48
4.1.4 Homogénéisation de la langue de la grammaire . . . 48
4.1.5 Ajout de KEYWORD . . . 49
4.1.6 Génération . . . 49
4.2 Annotation des corpus VocADom et SWEET-HOME . . . 50
4.3 Comparaison du corpus artificiel, VocADom, SWEET-HOME et PORTMEDIA 52
4.4 L'évaluation du corpus artificiel . . . 53
4.4.1 L'entraînement de RASA, Tri-CRF, att-RNN . . . 53
4.4.2 Entraînement du Tf-seq2seq . . . 55
Modification et pré-traitement des corpus . . . 55
Répartition des données . . . 57
Comparaison du corpus de l'entraînement (Artificiel + SWEET-HOME) avec le corpus de test (VocADom) . . . 57
Paramètres des modèles . . . 58
Pré-traitement des prédictions et des cibles et l'évaluation . . . 58
Résultats . . . 59

5 L'étude de Tf-seq2seq : différents modes et tâches **61**
5.1 Mots vs caractères . . . 61
5.2 Prédiction des intentions et slots vs slots seulement . . . 62

6 Récapitulatif des résultats de tous les modèles **63**

7 Discussion **65**

8 Perspectives de la recherche **67**

9 Conclusion **71**

Références **73**

Glossaire

Intention - l'objectif du locuteur que l'on vise à annoter dans la tâche de compréhension automatique. Par exemple, dans la phrase *Trouve un restaurant qui sert de la cuisine italienne*, l'intention pourrait être annotée comme *find_restaurant*.

Slots - les morceaux de l'énoncé pertinents du point de vue de l'intention, que l'on vise à annoter dans la tâche de compréhension automatique. Par exemple, dans la phrase *Trouve un restaurant qui sert de la cuisine italienne*, on pourrait annoter le slot *cuisine=italienne*. Ici, *cuisine* est un slot-label et *italienne* est un slot-value.

Chapitre 1

Introduction

Les travaux rapportés dans ce mémoire se sont déroulés dans le cadre du projet ANR VocADom[1], système vocal d'interaction avec la domotique pour aider les habitants âgées au quotidien. Ce mémoire porte plus particulièrement sur la compréhension automatique de la langue dans l'habitat intelligent pour personnes âgées.

Le développement des technologies doit permettre de faciliter la vie des personnes âgées et des personnes à mobilité réduite en augmentant leur habitat avec des systèmes intelligents domotiques (Chan *et al.*, 2008). L'habitat intelligent peut alors assister les personnes âgées dans l'exécution de tâches quotidiennes sans perturber leur confort et leur vie privée.

Une expérimentation a été menée dans un appartement intelligent dans le cadre du projet SWEET-HOME (Vacher *et al.*, 2015) impliquant des utilisateurs potentiels (personnes âgées et malvoyantes) commandant le système domotique à l'aide d'ordres vocaux avec une grammaire rigide. Cette grammaire implique que chaque ordre se conforme à la structure *mot-clé action objet* comme par exemple *Nestor allume la lumière*. L'étude a permis de repérer que les façons de parler des personnes âgées sont assez hétérogènes et tandis que les uns respectent la grammaire des ordres, les autres s'en écartent. Par ailleurs, ces utilisateurs sont souvent enclins à personnifier le système en utilisant des formules propres à la communication humaine (formules de politesse notamment), à oublier le mot clé ou à le mettre dans une position autre que celle prévue au début de l'ordre, et à utiliser des formes grammaticales inattendues (par exemple, usage de l'infinitif au lieu de l'impératif). Tout cela doit être pris en compte par les concepteurs des systèmes domotiques. Les systèmes à base de règles ne sont pas suffisamment flexibles pour cela, c'est pourquoi les systèmes basés sur un apprentissage semblent plus adaptés. Malheureusement, nous ne possédons que deux petits corpus : le corpus VocADom (Vacher, 2018) (1646 phrases) et SWEET-HOME (Vacher *et al.*, 2014) (727 phrases), tous les deux enregistrés dans des conditions réalistes d'un habitat intelligent. Le premier est un corpus d'ordres spontanés et lus ; le deuxième est un corpus d'ordres lus uniquement. Nous appelons ces deux corpus « réalistes » car ils ont été enregistrés dans des conditions réalistes. Les deux corpus ne sont pas annotés. Ainsi, il n'existe pas de corpus réaliste suffisamment grand pour apprendre un tel système.

Pour résoudre entre autre ce problème, le projet ANR VocADom a été lancé en 2017. Ce projet vise à développer un système domotique adapté aux personnes âgées et malvoyantes, commandé par la voix. Les problèmes traités par le projet sont le contrôle par la voix à distance, l'adaptation du système à la grammaire propre au locuteur, la prise de décision à partir du contexte, l'extraction de la parole dans le bruit et l'adaptation aux attentes des utilisateurs.

1. Site du projet ANR VocADom https ://vocadom.imag.fr/ Date de dernière consultation : 16.01.2018

En effet, le but du système est qu'il soit capable de fonctionner dans les conditions réelles du domicile de la manière la plus naturelle que possible pour l'utilisateur.

Les études menées dans le cadre de ce projet nécessitent de disposer de lieux d'expérimentation spécifiques et adaptés, c'est le cas de l'appartement intelligent DOMUS du LIG (Laboratoire d'Informatique de Grenoble). Cet appartement a été conçu pour observer l'interaction des utilisateurs avec l'environnement intelligent (Vacher *et al.*, 2015). DOMUS est un appartement de 30 mètres carrés, il consiste en une salle de bain, une cuisine, une chambre et un salon. L'appartement est équipé de 150 capteurs (pour la consommation de l'eau et de l'énergie, la température, etc.) et actionneurs (pour la lumière, les rideaux, etc.) et de microphones disposés dans le plafond (16 disposés en antennes de 4 microphones et 7 microphones HF indépendants).

Ce mémoire se focalise uniquement sur le module de compréhension automatique du langage naturel. Dans le cadre de ce projet, nous avons deux objectifs.

Premièrement nous souhaitons générer un corpus artificiel d'ordres annotés pour apprendre un système de compréhension automatique, faute d'un grand corpus réaliste. Dans ce but, nous allons améliorer la grammaire existante de génération créée dans le cadre du projet VocADom afin de prendre en compte les particularités du langage des utilisateurs de l'habitat intelligent. Cette grammaire ne générant que 62 phrases, il nous faudrait donc l'enrichir lexicalement et syntaxiquement. Ensuite il faut déterminer la qualité du corpus artificiel résultant ; c'est-à-dire, d'évaluer comment un système entraîné sur notre corpus artificiel fonctionnerait dans des conditions réelles. Dans ce but, nous allons entraîner quatre modèles neuronaux (Tri-CRF, RASA, att-RNN et Tf-seq2seq) sur ce corpus et puis nous allons les tester sur le corpus réaliste VocADom. On va également comparer ces résultats avec les résultats des 4 modèles appris et testés sur le corpus PORTMEDIA, relevant du domaine du tourisme, pour être sûr que nos conclusions sont tirées sur la base des limitations des corpus plutôt que les limitations des modèles eux-mêmes.

Avant d'expliquer le deuxième objectif, il faut dire que même si Tri-CRF, RASA et att-RNN sont l'état de l'art dans la compréhension automatique, Tf-seq2seq a un avantage sur eux en ce qu'il ne requiert que l'entraînement d'un seul modèle pour la prédiction des intentions, slot-labels et slot-values, alors que Tri-CRF, RASA et att-RNN en requiert deux ou trois. En plus, Tf-seq2seq ne requiert pas d'alignement des données, qui est coûteux. Un autre aspect dans lequel il est différent est qu'il n'a pas de séparation explicite entre les intentions et les slots. Il est intéressant de savoir si malgré ces particularités, Tf-seq2seq peut être compétitif par rapport à trois autres modèles. Nous allons donc étudier la performance de Tf-seq2seq en variant la mode : mot et caractère, et la tâche : la prédiction des intentions et slots, et la prédictions des slots seulement.

Ce mémoire s'organise comme suit : une première partie sera constituée d'un état de l'art qui présent les études existantes sur l'interaction vocale dans l'habitat intelligent, et en particulier les études menées avec les personnes âgées. Nous introduirons ensuite les concepts liés à la compréhension automatique de la langue : ses notions de base, les problèmes existants, les modèles existants ainsi que d'autres aspects. Suivront des informations générales sur les corpus et la description des corpus existants. Puis nous décrirons les travaux préliminaires dans le cadre du projet ANR VocADom, qui ont abouti à la création de la grammaire. Enfin, nous présenterons la problématique de ce travail avant de passer à la partie expérimentation où nous décrivons l'implémentation des trois objectifs mentionnés ci-dessus.

Chapitre 2

État de l'art

2.1 L'interaction vocale dans l'habitat intelligent

Comme notre recherche porte sur l'interaction vocale des personnes âgées avec un système domotique dans l'habitat intelligent, il faut tout d'abord étudier les recherches existantes sur l'interaction vocale avec la maison connectée, et plus précisément, celles qui ont été menées avec les personnes âgées et préciser les particularités de leur langage dans ce contexte. En effet, même si quelques études ont impliqué des personnes âgées, la plupart des études portant sur l'interaction avec l'habitat intelligent ont été faites avec des adultes d'âge moyen.

2.1.1 Études sur l'interaction dans l'habitat

L'étude de Callejas et López-Cózar (Callejas et López-Cózar, 2009) a été effectuée à partir des opinions exprimées par les personnes âgés et elle porte sur des fonctionnalités à prendre en compte par les systèmes contrôlant des habitats intelligents. Pour cela, un sondage a été mené sur 200 participants âgés de 50 à 80 ans. Les questions posées ont porté sur l'utilité de fonctionnalités telles que le contrôle de la lumière, de la température, des stores, de la TV, de la musique, des appareils électroménagers, mais aussi sur la capacité du système à gérer un agenda. Les gens ont jugé toutes les fonctionnalités utiles mais ils n'utiliseraient pas souvent certaines fonctionnalités. L'agenda a été jugé la fonctionnalité la plus utile. Par contre, la plupart des gens ont dit qu'ils n'utiliseraient jamais le contrôle des appareils électroménagers. Mais cela peut être dû au fait que, selon le questionnaire, la plupart des participants, n'effectue pas de tâches ménagères toute seule. Comme dans d'autres études similaires, les gens ont exprimé la peur que le fait d'utiliser le système puisse provoquer la perte de certaines capacités, telles que la mémoire, quand on utilise la fonction agenda. Il n'y avait pas de remarques sur les particularités d'interaction langagière.

Une des études menées dans le cadre du projet SWEET-HOME (Portet *et al.*, 2013) porte sur l'évaluation subjective des participants âgés du système domotique SWEET-HOME dans le cadre de l'habitat intelligent DOMUS, déjà mentionné dans le chapitre 1. L'étude conclut que les gens apprécient la sécurité que le système peut fournir (portes ou fenêtre restés ouvertes). En revanche, certains participants ont jugé que le système encourage un mode de vie relâché, peu actif ce qui peut avoir des effets néfastes sur la santé.

Quant aux particularités du langage des ordres vocaux, il n'y a pas beaucoup d'observations sur cela car ce n'était pas le but de l'étude. Pourtant on remarque que la majorité des participants préfèrent contrôler la maison par mots-clés (et non par des ordres spontanés) car ils considèrent

que c'est plus efficace.

L'étude sur l'utilisation de l'habitat intelligent de Tiiu Koskela et Kaisa Väänänen-Vainio-Mattila (Koskela et Väänänen-Vainio-Mattila, 2004) ne porte pas sur les personnes âgées et l'habitat utilisé est commandé par des applications (et non par la voix), mais elle nous semble intéressante quand même car c'est une des rares études dans laquelle les participants ont habité une maison connectée pendant longtemps (6 mois), ce qui leur a permis de donner leur opinion « sur le long terme » et non pas seulement leurs « premières impressions ».

L'étude a été faite en trois étapes. D'abord, pour définir les besoins des utilisateurs, vingt-deux jeunes adultes ont été questionnés sur les fonctionnalités qu'ils voudraient avoir dans un habitat intelligent. Puis, trois interfaces de contrôle de l'habitat intelligent ont été conçues : sur le portable, le TV et l'ordinateur. Enfin, ces trois interfaces ont été évaluées pendant une expérimentation avec 2 personnes qui habitaient 6 mois dans cet habitat. Il a été constaté qu'il est nécessaire d'avoir deux types du contrôle : un « pattern control » qui automatise le comportement du système lors de certains actes routiniers (par exemple, allumer la lumière quand le réveil sonne le matin) et un « instant control, c'est-à-dire, le contrôle direct par l'utilisateur. Les participants ont exprimé la peur que si le système avait été commandé par la voix, il n'aurait pas compris ce qu'ils auraient dit.

2.1.2 Études sur les particularités langagières de l'interaction des personnes âgées

Nous avons pu mettre en évidence que les études décrites ci-dessus sont plutôt concentrées sur l'évaluation des systèmes de l'habitat intelligent, et non sur les spécificités linguistiques de l'interaction. Il y a peu d'études sur ces dernières. Nous allons malgré tout en décrire deux.

L'étude menée dans le cadre du projet MeMo (Usability Workbench for Rapid Product Development) (Möller *et al.*, 2008) compare la façon de parler des personnes âgés et jeunes au système de l'habitat intelligent. Elle a trouvé deux patterns chez les personnes âgées. Premièrement, ils utilisent plus d'unités lexicales relevant de l'interaction sociale (*s'il te plaît, merci*), alors que les personnes jeunes ont tendance à utiliser le « command style ». Deuxièmement, les personnes âgés utilisent plus de mots que les jeunes pour formuler un ordre. Portant, on ne peut pas projeter ces généralisations sur l'ensemble des personnes âgées car les écarts-types sont beaucoup plus important chez les personnes âgées que chez les jeunes en ce qui concerne ces deux particularités langagières. Nous en tirons donc la conclusion que les gens âgées utilisent une façon de parler proche de celle utilisée avec les humains, mais ce n'est pas généralisable sur tous les gens âgées.

Même si l'étude précédente a été menée en allemand, on peut y observer les tendances similaires que l'on remarque dans l'étude de Michel Vacher et al. (Vacher *et al.*, 2015), menée en français dans l'habitat intelligent DOMUS avec le système domotique SWEET-HOME. Cette expérimentation a été menée avec 11 participants qui étaient âgés entre 50 et 91 ans, et dont certains présentaient des troubles de la vision (sans être aveugles) ; ils habitaient seuls dans leur logement, étaient sur le point de perdre leur autonomie mais restaient encore autonomes. Chaque participant a reçu une liste des ordres à lire, qui se conformaient à la grammaire suivante :

basicCmd = key initiateCommand object | key emergencyCommand
key = "Nestor"
initiateCommand = "ouvre" | "ferme" | "baisse" | "éteins" | "monte" | "allume" | "descend" | "appelle" | "donne"

emergencyCommand = "au secours" | "à l'aide"
object = [determiner] (device | person | organisation)
determiner = "mon" | "ma" | "l'" | "le" | "la" | "les" | "un" | "des" | "du"
device = "lumière" | "store" | "rideau" | "télé" | "télévision" | "radio" | "heure" | "température"
person = "fille" | "fils" | "femme" | "mari" | "infirmière" | "médecin" | "docteur"
organisation = "samu" | "secours" | "pompiers" | "supérette" | "supermarché"

Les participants n'avaient pas d'accès à cette grammaire.

Ensuite, les participants ont été encouragés à donner leurs propres ordres au système. On peut relever 3 points intéressants pour notre travail parmi les résultats de cette dernière étape de l'expérimentation, quand les participants ont prononcé des ordres spontanés :

* tous les participants ne se conforment pas à la grammaire des ordres, deux groupes se dégagent : ceux qui respectent la grammaire (dans la plupart de cas, les gens avec des troubles de la vision car ils sont plus familiarisés avec les technologies d'assistance) et ceux qui ne l'ont pas fait. L'analyse des ordres a montré que 21 mots sur 90 étaient conformes à la grammaire.

  Les déviations de la syntaxe étaient les suivantes :

  1. l'absence du mot clé *Nestor* qui devait commencer la commande ;
  2. la présence d'une pause trop importante entre le mot clé *Nestor* et le reste de la phrase ;
  3. l'utilisation de l'infinitif au lieu de l'impératif : *Nestor, éteindre la radio* au lieu de *Nestor, éteins la radio*. On peut supposer que le premier énoncé est vu comme un langage de communication avec la machine ;
  4. utilisation des formes de politesse *Pouvez-vous éteindre la lumière, s'il vous/te plaît*. Ces formes reflètent l'attitude des participants envers le système – proche de l'attitude à tenir envers un humain ;
  5. le transfert du mot clé *Nestor* à la fin de l'ordre.

  La moitié des participants ont remarqué qu'il serait préférable de pouvoir modifier le nom Nestor et deux participants auraient préféré ne pas avoir de nom pour le système. Quatre participants sur six n'ont pas utilisé le nom du système lorsque ils n'étaient pas guidés.

* En ce qui concerne les ordres, ils peuvent contenir des informations implicites, par exemple, la localisation de la lampe dans la phrase *Allume la lampe*. Cette information a été obtenue à partir du contexte (la localisation de l'utilisateur). On ne peut pas savoir si le système a correctement interprété les ordres ambigus ou si les utilisateurs n'ont tout simplement pas pensé à préciser leur ordre. Il y avait risque de confusion, s'agissait-il réellement de la lampe dans la chambre ? La question de la décision en contexte reste donc ouverte.

* En ce qui concerne la perception de la grammaire, les avis des participants ont été recueillis à l'aide d'un entretien. Les participants disent ne pas avoir rencontré de difficultés en utilisant la grammaire. La moitié des participants a trouvé les ordres intuitifs. Les autres ont considéré que l'impératif ne convient pas pour s'adresser au système. Une participante a dit qu'elle préfère parler au système de la même façon qu'elle parle aux gens.

Les auteurs de l'étude constatent que le système est utilisable mais que la grammaire doit être adaptée aux utilisateurs en fonction des problèmes relevés.

En conclusion, il semble qu'un système de compréhension automatique de la parole dans un habitat intelligent doive prendre en compte la variation naturelle de la parole pour être fonctionnel. Dans la section suivante, nous allons décrire comment fonctionne la compréhension de la langue naturelle.

2.2 La compréhension de la langue naturelle

2.2.1 Définitions

En termes simples, la compréhension de la langue vise à extraire le sens de la langue naturelle. La langue est le moyen de communication le plus naturel pour les humains ; beaucoup de recherches aujourd'hui portent sur l'implémentation de systèmes basés sur la communication entre la machine et l'humain : dialogue homme-machine.

Les applications les plus courantes des systèmes dialogiques sont orientées tâche, c'est-à-dire que les systèmes visent à comprendre les besoins de l'utilisateur et à les satisfaire avec un nombre limité de tours de dialogues (*dialogue turns*) (Bordes et Weston, 2016).

Il est d'abord nécessaire de définir ce qu'est un tour de dialogue. Selon Traum et Heemann, le tour de dialogue est la parole émise par un locuteur sans interruptions de la part de son interlocuteur (Traum et Heeman, 1996).

Il est également nécessaire de définir ce qui est l'*intention*, une notion cruciale dans notre travail car la compréhension automatique du langage a deux objectifs : déterminer les intentions des énoncés de l'utilisateur, ce que l'on appelle la prédiction de l'intention, et, identifier des morceaux pertinents de l'énoncé ou *slots*, opération appelée remplissage des *slots* (*slot-filling*) (Tur et De Mori, 2011). L'intention peut être défini comme l'objectif du locuteur que l'on vise à annoter dans la tâche de compréhension automatique. Les slots peuvent être définis comme les morceaux de l'énoncé pertinents du point de vue de l'intention.

La notion de l'intention est très proche à celle de l'*acte de langage* (*speech act* en anglais), qui est l'activité communicative définie par l'intention d'un locuteur et par l'effet qu'il a sur son interlocuteur (Crystal, 2011). Dans ce travail, nous sous-entendons que l'acte de langage est le synonyme de l'intention car nous ne nous intéressons pas à l'effet sur l'interlocuteur mais à l'intention.

L'intérêt des intentions et des slots dans la compréhension de la langue naturelle peut être montré sur un exemple : la phrase *Trouve un restaurant qui sert de la cuisine italienne* pourrait être annoté avec l'intention *find_restaurant*, et avec le slot *cuisine=italienne*. Ici, *cuisine* est un slot-label et *italienne* est un slot-value. Une fois que les intentions sont déterminées et que les slots remplis, ils sont transmis au gestionnaire de dialogue qui forme une requête pour la base des connaissances et détermine la réponse donnée à l'utilisateur.

Certains systèmes déterminent la réponse en se basant non seulement sur la phrase que l'utilisateur vient de prononcer, mais également sur l'historique du dialogue. L'état du dialogue est une structure de données qui contient l'historique du dialogue à un moment donné (Williams *et al.*, 2016). En pratique, l'état du dialogue code généralement l'objectif de l'utilisateur dans la conversation ; par exemple, dans le domaine des horaires d'autobus, il peut coder quel arrêt de bus l'utilisateur veut quitter, où il va, et si le système a déjà proposé un bus suivant cet itinéraire. Ce processus est appelé le *dialogue state tracking*. L'architecture du système standard de la compréhension automatique de la langue sera décrite plus en détails dans la section 2.2.4.

Il faut remarquer que dans notre travail, nous utilisons le terme *énoncé (utterance)* comme l'équivalent du terme*phrase*.

2.2.2 Problèmes existants

Avant de présenter les approches et les modèles utilisés pour la compréhension de la langue naturelle, nous pouvons dégager les problèmes existants en rapport avec cette tâche, sans tenir compte du domaine spécifique d'application.

Le module de compréhension de la langue dépend de son entrée qui est le résultat de la reconnaissance de la parole. Il existe plusieurs problèmes dans cette phase de traitement. Premièrement, la parole spontanée contient des pauses, hésitations, auto-corrections et faux départs (*false starts*), aussi bien que des fragments de mots. Il y a des tentatives fructueuses de traiter ces particularités de la parole. Par exemple, la performance des systèmes s'est améliorée grâce à l'intégration de modèles acoustiques des hésitations (Butzberger *et al.*, 1992; Ward, 1989).

Il reste aussi le problème des mots inconnus. On ne peut pas l'éviter car il est impossible de connaître l'ensemble des mots de vocabulaire des utilisateurs ; de plus, il y a toujours des changements dans le domaine (par exemple, dans la réservation des restaurants, il y a toujours de nouveaux restaurants qui s'ouvrent). Des modèles ad hoc ont été conçus pour détecter de nouveaux mots (Hetherington et Zue, 1993), aussi bien que les manières de les faire apprendre au système (Asadi et Makhoul, 1991). Mais cette question reste loin d'être résolue.

La qualité du signal de la parole a aussi une grande importance. Le traitement de la parole de basse qualité, plus précisément, la parole téléphonique, a été l'objet d'une attention particulière dans les recherches qui s'explique par le fait que les premiers systèmes conversationnels ont été accessibles par téléphone (Billi *et al.*, 1998).

En ce qui concerne la phase de compréhension de la langue, les systèmes doivent prendre en compte les caractéristiques de la parole spontanée déjà mentionnées ci-dessus : les hésitations, faux débuts, absence de la grammaire stricte etc. Si le système fait l'analyse syntaxique sur une entrée textuelle saturée par ces particularités de la parole spontanée, il va échouer. C'est pourquoi les approches sémantiques qui extraient les mots clés ont été utilisées pour dériver le sens. (Ward, 1990).

En outre, le système doit traiter les anaphores (par exemple, *what's their phone number*) et les ellipses (par exemple, si l'utilisateur dit *I want to go from Boston to Denver* et puis ajoute *show me only United flights* », il faudrait comprendre qu'il ne veut pas connaître tous les vols de United, mais seulement les vols de Boston à Denver). Pour résoudre cela, il faut prendre en compte le contexte, c'est-à-dire, les autres phrases. Cela est fait à l'aide du dialog state tracking.

En plus de ces problèmes génériques, nous rencontrons aussi un défi lié au domaine de l'application : les particularités de la parole des personnes âgées présentées dans la section 2.1. Vu ce grand nombre de variabilité de données en entrée, la robustesse d'un système de compréhension de la langue naturelle est un critère très important.

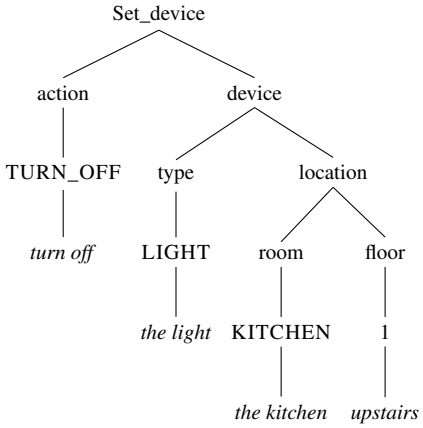
2.2.3 Représentation sémantique par les slots

Dans la section 2.2.1, nous avons déjà mentionné le principe de la compréhension automatique de la langue qui combine la détection de l'intention et le remplissage des slots. Par exemple, le système de réservation des restaurants pourrait avoir des slots tels que la localisation, la fourchette de prix et la cuisine (Yu *et al.*, 2017). Pour la cuisine, le slot-label pourrait être *cuisine* et le slot-value pourrait être *italienne* ou *grecque*. De tels systèmes à slots sont utilisés dans plusieurs domaines comme la réservation de bus (Raux *et al.*, 2005), de billets d'avions (Zue *et al.*, 1994), l'orientation spatiale (direction giving en anglais) (Yu *et al.*, 2015).

Les slots peuvent être représentés de façon hiérarchique ou plate (Tur et De Mori, 2011). La figure 2.1a montre une représentation hiérarchique des slots pour la commande *turn off the*

light in the kitchen upstairs (Raimondo, 2017). Les slots peuvent être composés de subslots. Le remplissage des slots dans le contexte de cette approche est une étiquetage de certains tokens avec les slots correspondants (par exemple, le mot *kitchen* est étiqueté comme *room*).

La figure 2.1b montre la représentation plate des slots pour la même commande. Le slot le plus haut correspond à l'intention dans cette approche (textitset_device). Les autres slots décrivent les informations nécessaires pour exprimer cette intention. Cette approche réduit la compréhension de la sémantique à la tâche de l'étiquetage de chaque segment de la phrase. Les slot-labels sont indiqués en police standard, les slot-values sont en majuscules, et les segments du texte associés sont en italique.



(a) Représentation hiérarchique des slots

set_device(action = TURN_OFF = turn off,
device = LIGHT = *the light*,
location-room = KITCHEN = "in the kitchen",
location-floor = 1 = "upstairs")

(b) Représentation plate des slots

FIGURE 2.1 – Représentations hiérarchiques et plates des slots (Raimondo, 2017)

Le représentation hiérarchique présente comme avantage de séparer l'action que le système doit effectuer du reste des informations.

Le problème est que la représentation plate comme sur 2.1b n'est pas toujours utilisable. Par exemple, même si des informations sur la localisation sont présentes, il n'est pas possible de préciser à quoi/qui elles se réfèrent : à la cuisine (kitchen) ou à la lumière (light) ou à tous les deux. En conséquence, il faut explicitement lier les informations sur la localisation de l'objet avec le slot *device : light*. Cela impose d'utiliser une représentation structurée (imbriquée), c'est à dire, que les slots peuvent contenir des slots. Par exemple : *device :{name :light,location :room1}*. Dans le cas où des objets identiques sont dans des pièces différentes (par exemple, les lampes dans la chambre et dans le salon), il est préférable d'utiliser la représentation hiérarchique pour résoudre l'ambiguïté des ordres (par exemple, *Allume la lumière*). Par contre, il est plus difficile d'établir un modèle qui pourrait non seulement extraire des intentions/slots mais aussi les structurer dans une hiérarchie.

2.2.4 Modèles de compréhension automatique du langage naturel

Approche par règles

Les modèles existants peuvent être répartis dans les deux groupes : soit basés sur les règles, soit sur les statistiques (Tur et De Mori, 2011).

En ce qui concerne l'approche par règles, elle est utilisée dans un des premiers systèmes de dialogue homme-machine, l'agent conversationnel ELIZA, développé par Joseph Weizenbaum

entre 1964 et 1966, qui simule un psychothérapeute rogérien en reformulant la plupart des affirmations du « patient » sous forme de questions (Weizenbaum, 1966). Cet agent conversationnel était basé sur la reconnaissance des motifs (pattern recognition en anglais) et la substitution simple des mots. Un autre agent conversationnel bien connu est ALICE (Wallace, 2009) développé en 2000 à l'aide d'Artificial Intelligence Markup Language (AIML), qui est un langage basé également sur la reconnaissance des motifs. Les deux agents conversationnels suivent la technique du dialogue qui s'appelle stimilus-réponse, où le stimulus est la question de l'utilisateur et la réponse du machine se base uniquement sur cette dernière question. Dans ALICE, la technologie AIML a été responsable pour l'appariement des motifs et la mise en relation de l'entrée de l'utilisateur avec la base des connaissances de l'agent conversationnel (Serban *et al.*, 2015). La figure 2.2 représente un exemple des règles écrites avec AIML prise de (Serban *et al.*, 2015) où le « pattern » est l'entrée de l'utilisateur et le « template » est la réponse du machine.

Une autre approche par règles est une grammaire hors contexte comme dans le système TINA (Seneff, 1992) qui vise à créer une interface entre la syntaxe et la sémantique (dépendante du domaine). Sur la figure 2.3, on peut voir un exemple du traitement de la phrase *What street is the Hyatt hotel on* par TINA. En utilisant la grammaire hors contexte, d'abord le système fait l'analyse syntaxique (2.3a), et puis il remplace les éléments non-terminaux syntaxiques (par exemple, PROPER_NOUN) par éléments non-terminaux (par exemple, hotel_name) (2.3b).

Un autre exemple de l'approche par règles est le système PHOENIX (Ward, 1991). Dans ce système, les énoncés de l'utilisateur sont donnés à un réseau de transition récursive (Recursive Transition Network en anglais) qui est un automate à états finis où les états sont des slots et les transitions peuvent être des mots ou des sous-réseaux. Comme l'exemple précédent, cette représentation est aussi une forme de grammaire hors contexte. Un fragment du réseau PHOENIX est donné sur la figure 2.4; il s'agit du fragment *PriceRange* qui vise à déterminer la fourchette de prix désirée par l'utilisateur. Ce fragment comporte 4 transitions qui sont elles-mêmes des sous-réseaux : *PriceExact* (qui reconnaît les séquences du type *100 dollars*), *PriceApproximate* (qui utilise le sous-réseau *PriceExact* pour reconnaître les séquences du type *around 100 dollars*), aussi bien que *PriceLowerBound* (frontière inférieure du prix) et *PriceHigherBound* (frontière supérieure du prix) qui utilisent également le sous-réseau *PriceExact* pour reconnaître les séquences du type *at least 100 dollars*.

L'avantage de l'approche par règles est qu'elle ne demande pas de données étiquetées ; mais elle présente un inconvénient important : il est très difficile d'écrire une grammaire qui couvrirait la diversité linguistique (ce qui est important dans notre travail vu que le langage des personnes âgées a des particularités propres). C'est pourquoi l'utilisation de systèmes statistiques est une alternative.

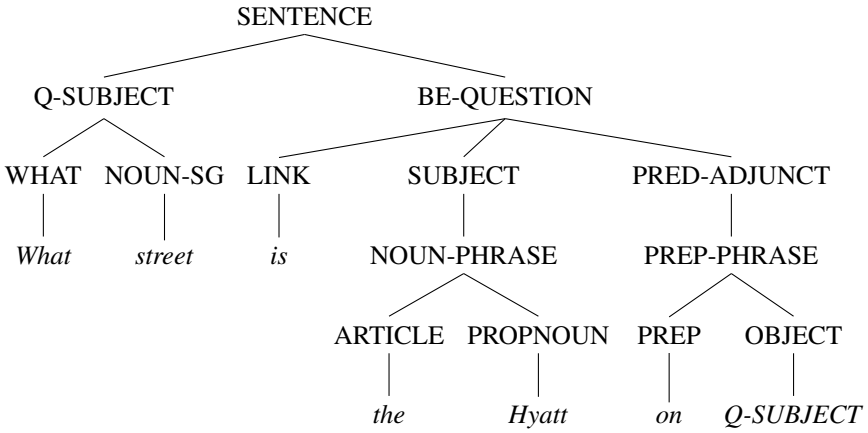
Approche statistique

Il y a deux types de modèles statistiques : génératifs et conditionnels. On présente à tous les deux une suite de mots W, et on cherche à trouver son sens M. En d'autres termes, on pourrait dire que l'on veut étiqueter les mots avec leur sens.

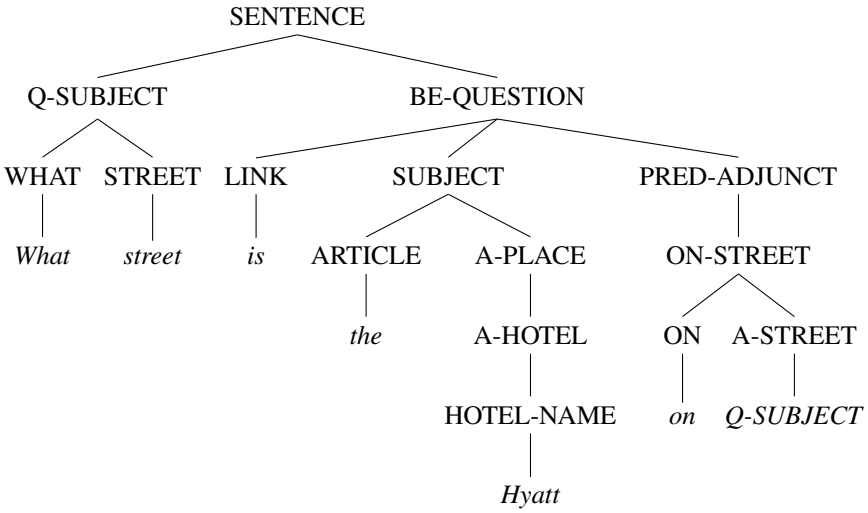
Avant d'aborder les modèles statistiques, il faut mentionner que la langue naturelle doit être convertie par une représentation qui la rende utilisable par un modèle statistique : par exemple, les sacs de mots (un vecteur utilisant les fréquences des mots dans un document) ou la repré-

<pattern>I like *</pattern> <template>I like </star> too</template>

FIGURE 2.2 – Un exemple de règles écrites avec AIML (Serban *et al.*, 2015)



(a) Analyse syntaxique



(b) Ajout d'éléments sémantiques

FIGURE 2.3 – Exemple de l'analyse de la phrase *What street is the Hyatt hotel on* par TINA (Seneff, 1992)

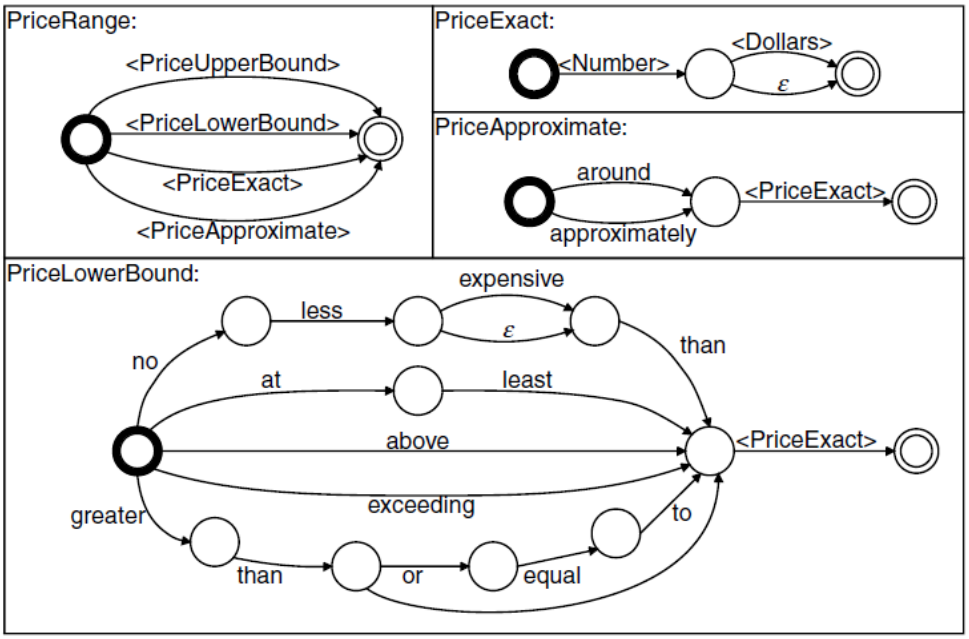


FIGURE 2.4 – Un fragment du réseau de transition récursive utilisé dans PHOENIX (Ward, 1991)

sentation vectorielle des mots (*word-embeddings*). Les word-embeddings (également connus comme modèles distributionnels sémantiques) sont des vecteurs qui sont dérivés de la table de coocurrence des mots. (Mikolov *et al.*, 2013). Pour chaque mot, son embedding est composé de nombres de cooccurrence (c'est-à-dire, l'occurrence des deux mots dans la même fenêtre fixe) avec tous les autres mots. Naturellement, les vecteurs sont creux (sparse vectors en anglais), c'est-à-dire qu'ils contiennent beaucoup de valeurs nulles parce que la plupart des cooccurrences ne se réalisent jamais. Pour convertir un vecteur en forme plus condensée, on utilise des méthodes mathématiques qui génèrent des vecteurs denses (dense vectors en anglais). Le principe de la représentation vectorielle des mots est que le mot est caractérisé par son contexte. Ces données sont reçues en entrée par les modèles statistiques.

D'abord, nous aborderons les modèles génératifs. Étant donné la suite de mots W, les modèles génératifs trouvent la représentation sémantique de sa signification M qui maximise la probabilité jointe $P(M, W)$ qui pourrait être réécrite de la façon suivante :

$$P(W, M) = P(W|M)P(M)$$

Ce modèle génératif a deux modèles distincts pour les deux parties de ce produit. Le modèle de lexicalisation $P(W|M)$ assigne la probabilité à la phrase (la séquence des mots) W étant donné la structure sémantique M. Le modèle sémantique à priori $P(M)$ attribue la probabilité à priori à cette structure sémantique (le sens M). Il y a beaucoup de types différents de modèles sémantiques à priori aussi bien que de lexicalisation. On utilise souvent les chaînes de Markov cachées (HMM). La figure 2.5 illustre ce principe présenté dans (Tur et De Mori, 2011).

Ici, les slots sont les états 0, 1, 2 et 3. On attribue un sens à chaque slot : par exemple, on peut supposer que le slot 2 a le sens *FromCity* (de quelle ville veut-on partir).

a_{00}, a_{01} ... etc. sont les probabilités de transition d'un slot à l'autre (ou de rester sur le même slot). C'est un modèle sémantique à priori.

b_0, b_1 ... etc. sont les probabilités d'avoir un mot (par exemple, *Seattle*) dans un slot (par

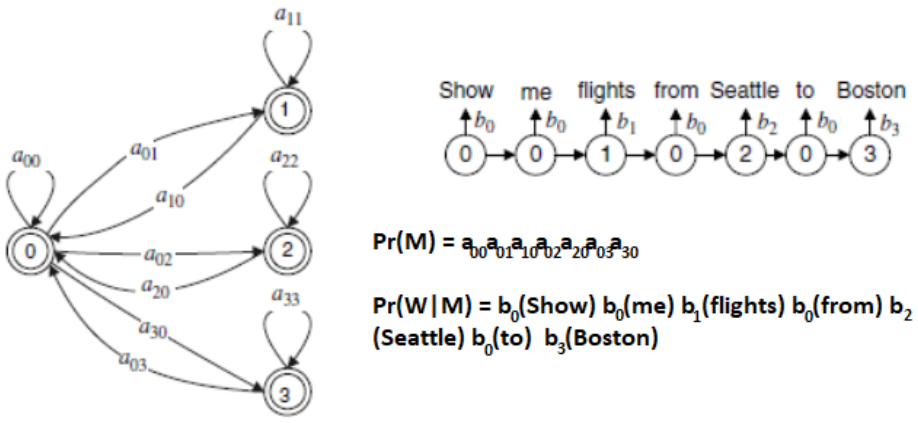


FIGURE 2.5 – Le principe du fonctionnement du modèle génératif HMM

exemple, dans le slot 2 qui a le sens *FromCity*). On les appellent les probabilités d'émission car il s'agit de l'émission des états observables (les mots) à partir des états cachés (qui représentent le sens). C'est un modèle de lexicalisation. Sur la figure, pour le slot 2, on ne montre que la probabilité b_2 de la présence du mot *Seattle*, mais en pratique, chaque slot est associé à plusieurs probabilités d'émission car on peut avoir plusieurs mots différents dans un même slot.

Les deux ensembles de probabilités sont obtenus à partir de l'apprentissage sur les données.

En ce qui concerne les modèles conditionnels, ils visent à maximiser la probabilité conditionnelle $P(M|W)$. Le modèle conditionnel assigne une étiquette M à chaque W. Il y en a de différents types. Pour la compréhension de la langue naturelle, les champs aléatoires conditionnels (Conditional Random Fields, ou CRF en anglais) sont couramment utilisés (cf. figure 2.6 prise de (Jeong et Lee, 2008)).

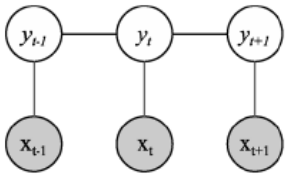


FIGURE 2.6 – Le principe du fonctionnement du CRF à chaîne linéaire (Jeong et Lee, 2008)

C'est un CRF à chaîne linéaire pour l'intervalle du temps de $t-1$ à $t+1$. Chaque nœud blanc désigne une variable aléatoire, (c'est-à-dire, un état caché, ou dans notre cas, le sens y) et le nœud gris correspondant représente sa valeur observée (dans notre cas, le mot x).

Puisque le nombre de toutes les séquences d'étiquettes possibles augmente de façon combinatoire et qu'il serait trop long de calculer la probabilité pour chaque séquence, il est courant de restreindre l'historique des états précédents qu'on prend en compte ce qui s'appelle une contrainte de Markov.

L'avantage des modèles conditionnels sur les modèles génératifs est le fait que ces derniers prennent du temps pour modéliser la génération d'observations qui sont en pratique toujours connues.

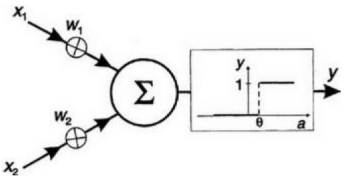


FIGURE 2.7 – Le principe du fonctionnement d'un neurone (Gurney, 1997)

D'autres méthodes statistiques génératives couramment utilisées sont les réseaux neuronaux (Gurney, 1997). Ils peuvent être utilisés en apprentissage supervisé, c'est-à-dire, entraînés sur les données étiquetées, ou non-supervisé, sur les données non-étiquetées.

Le fonctionnement des réseaux neuronaux se base sur celui de son unité, le neurone. Les neurones interconnectés forment un réseau ou chaque neurone a une seule sortie mais peut avoir plusieurs entrées.

Un neurone reçoit les entrées $x_1, \ldots x_n$. Puis leur somme est pondérée, et la sortie du neurone est calculée à l'aide d'une fonction d'activation (par exemple, une sigmoïde) qui prend cette somme pondérée en entrée. Chaque poids dans le réseau est ajusté lors de l'entraînement en calculant sa contribution à la sortie finale. Ce processus s'appelle la rétro-propagation du gradient (backpropagation).

La figure 2.7 présente l'illustration d'un neurone à deux entrées x_1 et x_2 qu'il pondère avec les poids w_1 et w_2, et utilise ensuite une fonction d'activation pour fournir la sortie y. Deux modèles neuronaux récurrents (RNN) seront décrits plus en détails dans la section 2.2.4.

L'avantage de l'approche basée sur les statistiques est qu'elle s'adapte bien aux nouvelles données. Elle comporte cependant les inconvénients suivants : elle demande un grand volume des données d'entraînement, et la qualité du système dépend beaucoup de la qualité de ces données. Par exemple, si les données d'entraînement sont produites en laboratoire, le système ne sera peut-être pas capable de montrer une bonne performance dans la vie réelle. C'est pourquoi il est préférable de recueillir les données dans des conditions proches des conditions réelles d'utilisation.

Une autre avantage bien évidente des méthodes statistiques est qu'elles ne demandent pas des règles écrites manuellement. Par exemple, dans le cas de l'approche slot-filling utilisée dans les modèles de compréhension du langage, il est impossible de coder manuellement tous les slots et leur valeurs possibles. C'est pourquoi les réseaux neuronaux sont souvent utilisés au lieu des systèmes par règles : ces systèmes sont appris sur les vrais dialogues annotés du domaine, ce qui rend inutile de modéliser le domaine manuellement. Ils peuvent même être appris pour faire des requêtes sur des bases de données ce qui est souvent une partie intégrante des systèmes orientés tâche. C'est pourquoi dans (Bordes et Weston, 2016), même si les systèmes par règles ont obtenu des résultats meilleurs que les réseaux neuronaux dans les situations de simulation de réservation du restaurant, les réseaux neuronaux ont conduit aux meilleurs résultats dans les situations réelles.

Architecture : pipeline par opposition à de bout-à-bout

Les systèmes de dialogue traditionnels sont des pipelines de modules connectés tels que Natural Language Understanding, Dialog State Tracking, Dialog Management, and Natural Language Generation (Raux *et al.*, 2005; Rudnicky *et al.*, 1999; Liu et Lane, 2017).

L'architecture classique de ce genre des modèles est montrée sur la figure 2.8 (Young *et al.*, 2013) : pour chaque phrase t, le module de la compréhension du langage naturel (Natural Lan-

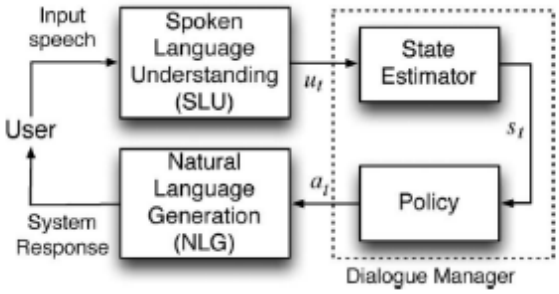


FIGURE 2.8 – Les composants d'un système de dialogue basé sur un pipeline (Young *et al.*, 2013)

guage Understanding) convertit l'entrée en représentation sémantique prédéfinie u_t, en l'annotant avec par exemple les intentions et les slots. Cette sortie est passée au Dialogue Manager qui est constitué de Dialogue State Tracking (appelé State Estimator sur la figure 2.8) et de Policy. Le module Dialogue State Tracking met à jour l'état du dialogue s_t. Comme il était précisé dans la section 2.2.1, l'état du dialogue s_t contient l'historique, c'est-à-dire, les variables dont on a besoin pour suivre le progrès du dialogue. Ce sont les intentions et les slots qui déterminent les vœux de l'utilisateur. Puis l'action suivante du système a_t est déterminée à l'aide des règles (Policy).

Cette architecture contient aussi un générateur du texte qui prend l'action en entrée et génère un énoncé en langue naturelle.

L'étude (Liu et Lane, 2017) constate que ces modèles ont 2 désavantages. Premièrement, ils sont difficilement adaptables au nouveaux domaines car chaque module doit être entraîné séparément. Deuxièmement, les erreurs issues des modules différents se propagent dans le système et rendent difficile l'identification de la source de l'erreur. Comme alternative, ils proposent un système de dialogue orienté tâche de bout-en-bout (end-to-end en anglais). Le système de bout-en-bout peut être défini comme un système constitué d'un seul module effectuant le traitement plutôt que plusieurs modules agissant comme une pipeline.

Le système cité ici comme exemple de l'approche de bout-en-bout est prise de l'étude (Liu et Lane, 2017) et basé sur un réseau neuronal avec belief tracking (cf. figure 2.9, avec nos modifications). Belief tracking est un type de dialogue state tracking ; la différence est que belief tracking sous-entend le suivi possible de plusieurs « versions » de l'état du dialogue, chacune contenant des informations sur les intentions du locuteur et les slots correspondants. Chaque « version » de l'état du dialogue est associé à une probabilité. Après chaque énoncé de l'utilisateur, le modèle met à jour l'état de dialogue, c'est-à-dire ; la distribution des probabilités pour les valeurs candidates pour remplir les slots. Par exemple, dans le domaine de la réservation de restaurants, le modèle met à jour la distribution des probabilités sur les exigences de l'utilisateur en ce qui concerne le type de cuisine, la fourchette de prix etc.

Dans l'exemple de la figure 2.9, l'utilisateur demande *Is there another restaurant that serves Italian food*. Les tours de dialogue sont encodés en utilisant le LSTM RNN. Le LSTM est expliqué en détails dans la section 2.2.4.

A chaque pas de temps t, le vecteur de l'état du LSTM sert à calculer la distribution des probabilités des valeurs de slots (slot-values) sur les souhaits de l'utilisateur telles que le lieu (*area_slot*), la cuisine (*food_slot*) et le prix (*price_slot*). Dans ce cas, l'utilisateur veut que le restaurant fasse de la cuisine italienne (*slot food="Italian"*), et se trouve dans la partie ouest de la ville (*slot area="west"*), mais il ne précise pas le prix, donc le slot *prix* est associé à la valeur

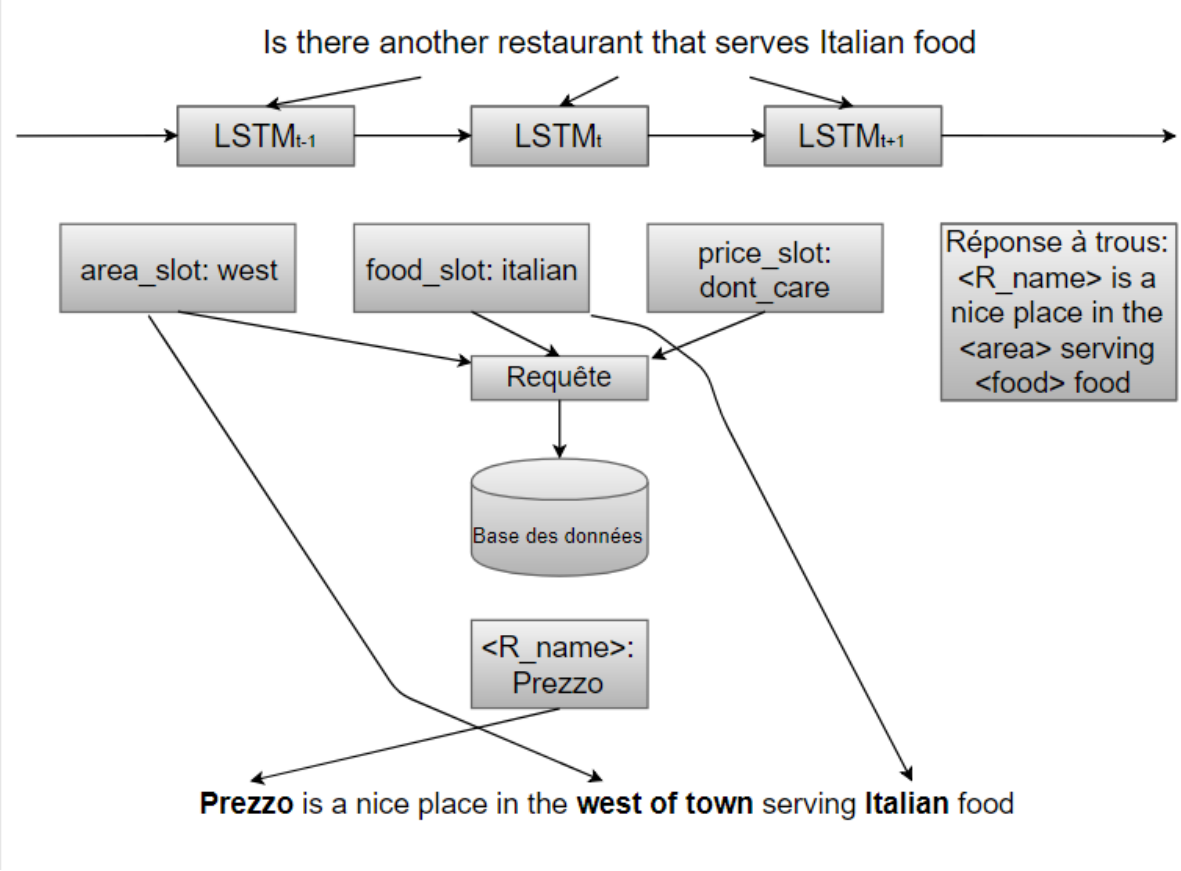


FIGURE 2.9 – Système de dialogue orienté tâche de bout-à-bout basé sur un réseau neuronal avec belief tracking (Liu et Lane, 2017) (avec nos modifications)

par défaut *dont_care*. Il n'est pas expliqué d'où provient le vœux de l'utilisateur concernant la localisation mais on peut supposer que ce slot a été rempli pendant la requête précédente.

Le vecteur de l'état du LSTM sert aussi à choisir une réponse à trous parmi les phrases dans la base, par exemple *R-name is a nice place in the <area> serving <food> food*.

Pour faire une requête sur la base des données, on utilise un patron de la commande et on complète les trous par les meilleures hypothèses pour chaque slot issues de Belief tracker (chercher un restaurant qui a la cuisine italienne dans la partie ouest de la ville). Cette requête génère un pointeur sur une entité qui correspond à la question de l'utilisateur. Dans notre cas, c'est le nom du restaurant *Prezzo*.

En utilisant les slot-values et le pointeur vers l'entité, le système remplir la réponse à trous : *Prezzo is a nice place in the west of town serving Italian food*.

Détection d'intention et le remplissage des slots

Le but des modèles de compréhension de la langue est de détecter l'intention et de remplir les slots. On peut traiter ces deux tâches séparément.

La détection d'intentions peut être considérée comme soit une tâche de l'étiquetage des séquences (sequence labelling tasks en anglais), soit une tâche de la classification des séquences (sequence classification task en anglais) (Jeong et Lee, 2008). Beaucoup de méthodes différentes sont utilisés pour la détection d'intentions. Les méthodes les plus répandues pour le moment sont basées sur les réseaux de neurones (Core et Allen, 1997; Ji *et al.*, 2016; Kalchbrenner et

Blunsom, 2013; Ravuri et Stolcke, 2015) alors que les autres utilisent les HMM (Tran *et al.*, 2017).

En ce qui concerne le remplissage des slots (slots filling), il existe aussi différentes approches : modèles génératifs tels que HMM/CFG composite models (Pieraccini *et al.*, 1992; Wang *et al.*, 2005), hidden vector state model (HVS) (He et Young, 2003), et modèles conditionnels tels que les champs aléatoires conditionnels (Conditional Random Field, ou CRF) (Lafferty *et al.*, 2001; yi Wang et Acero, 2006) qui seront décrit en détails dans la section 2.2.4. Malgré des années de recherches, le slot-filling demeure une tâche difficile en TAL (Mesnil *et al.*, 2015) ce qui a motivé des recherches fructueuses en approche par réseaux de neurones (Huang *et al.*, 2017), y compris les RNN (Mesnil *et al.*, 2015; Zhang et Wang, 2016; Liu et Lane, 2016).

La plupart du temps, les deux tâches (la détection de l'intention et le remplissage des slots) sont effectuées simultanément car elles sont corrélées (Zhang et Wang, 2016). Les études (Zhang et Wang, 2016) et (Liu et Lane, 2017) montrent que cette méthode est plus efficace que les systèmes où ces deux tâches sont séparées parce que chacune bénéficie des prédictions de l'autre. En plus, cela simplifie le système car un seul modèle devra être entraîné et utilisé.

Les modèles neuronaux encoder-decoder ont été récemment appliqués dans les différents problèmes d'apprentissage des séquence tels que la traduction automatique (Sutskever *et al.*, 2014) et la reconnaissance de parole (Chan *et al.*, 2015). Le principe est d'encoder la séquence d'entrée en vecteur dense, et ensuite de l'utiliser pour générer la séquence correspondante de sortie (Liu et Lane, 2016). Dans (Liu et Lane, 2016), un tel modèle montre de bons résultats sur la compréhension de la langue. Nous allons décrire son architecture en détails dans la section 2.2.4.

Champ aléatoire conditionnel triangulaire

Le modèle Tri-CRF (Triangular Conditional Random Field, ou Champ aléatoire conditionnel triangulaire en français) de (Jeong et Lee, 2008), est une extension d'un CRF à chaîne linéaire (CRF) décrit dans la section 2.2.4 appliquée à l'étiquetage des intentions et des slots. Ce modèle est constitué d'une séquence observée des mots X, d'une séquence cachée représentant les slots correspondants Y (la figure ne montre que l'intervalle du temps de $t-1$ à $t+1$) auxquelles s'ajoute une troisième variable Z (d'où le nom) qui est l'intention associée à l'énoncé. x_0 est une séquence observée correspondant à l'ensemble de la phrase car l'intention est prédite à partir de toute la phrase. On ajoute aussi les liens de dépendance entre cette variable et la séquence cachée.

Il s'agit d'un modèle discriminatif qui suppose que les intentions génèrent des slots avec une certaine probabilité, puis que ces slots couplés avec l'intention, à leur tour, génèrent les mots dans un énoncé. Donc simultanément Tri-CRF prédit l'intention et la séquence des slots. Comme il était déjà dit, les modèles remplissant simultanément les tâches de détection de l'intention et le remplissage des slots se sont montrées plus efficaces (Zhang et Wang, 2016; Liu et Lane, 2017). Le modèle de base Tri-CRF est illustré à la figure 2.10.

Dans l'étude (Jeong et Lee, 2008), cette méthode montre une meilleure performance sur un corpus artificiel (généré) et sur un corpus réel que les systèmes de base de référence (baseline) pour la tâche de la compréhension automatique. Les CRFs sont également parfois utilisés comme un composant du système de la compréhension automatique, comme dans RASA, qui va être décrit dans le paragraphe suivant.

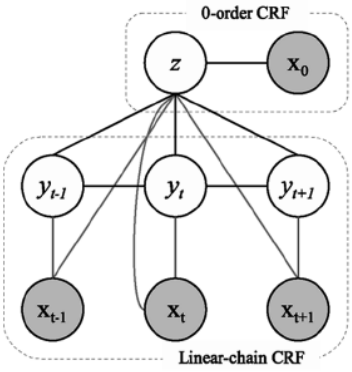


FIGURE 2.10 – Le principe du fonctionnement du tri-CRF(Jeong et Lee, 2008)

Logiciel commercial RASA

Rasa est un logiciel open-source destiné à faciliter le développement d'agents conversationnels commerciaux où la prédiction de l'intention et le remplissage des slots sont effectués séparément.

Selon une étude récente comparant des systèmes de compréhension automatique, la performance de RASA a été jugée parmi les meilleures, c'était en outre le seul système qui pouvait être parametré (Braun *et al.*, 2017). Le modèle Rasa utilisé s'appelle « spacy sklearn » et il est basé sur des word embeddings pré-appris sur les données de Wikipedia, OpenSubtitles et Wikinews. Le vocabulaire final contient 1 184 651 mots. RASA utilise un CRF pour remplir les slots. Pour identifier les intentions, le modèle utilise les Séparateurs à Vaste Marge (Support Vector Machine, ou SVM) linéaire. Le SVM linéaire est un classifieur qui permet de séparer des données en 2 classes (Cortes et Vapnik, 1995). Pour cela, les données sont représentées comme vecteurs dans un espace que l'on vise à séparer avec une frontière, en maximisant la marge entre cette frontière et les échantillons les plus proches.

RNN du type encodeur-décodeur bidirectionnel basé sur l'attention

Cette section parle du RNN du type encodeur-décodeur bidirectionnel basé sur l'attention remplissant simultanément les tâches de détection de l'intention et de remplissage des slots. La figure 2.11 illustre le système présenté dans (Liu et Lane, 2016). La gauche de la ligne pointillé constitue la partie encodeur du système, la droite la partie décodeur.

L'encodeur est un LSTM (Long Short Term Memory) RNN, c'est-à-dire, un RNN composés d'unités LSTM. Il faut d'abord expliquer ce qu'est le LSTM RNN.

Les RNN classiques sont des réseaux à connexions récurrentes qui prennent en compte à un instant t un certain nombre d'états passés ; mais ils sont confrontés au problème de l'erreur rétro-propagée qui diminue trop vite à chaque pas de propagation (le problème du vanishing gradient en anglais) (Chung *et al.*, 2014). Grâce à une fonction d'activation complexe apprise, les LSTM RNN résolvent ce problème. C'est pourquoi les LTSM RNN modélisent mieux des dépendances longues (long-terms dependencies en anglais) que les RNN simples.

L'unité LSTM peut mémoriser des valeurs (dont le nom) grâce aux cellules mémoire. L'unité LSTM de base est montrée sur la figure 2.12 : i, f et o sont respectivement la porte d'entrée, la porte d'oubli et la porte de sortie. c et $\tilde{c}$ sont la cellule mémoire et la cellule mémoire mise à jour respectivement. Chaque porte agit comme un neurone dans le sens qu'elle utilise une fonction d'activation de la somme pondérée des entrées. De cette façon, elles contrôlent le flux

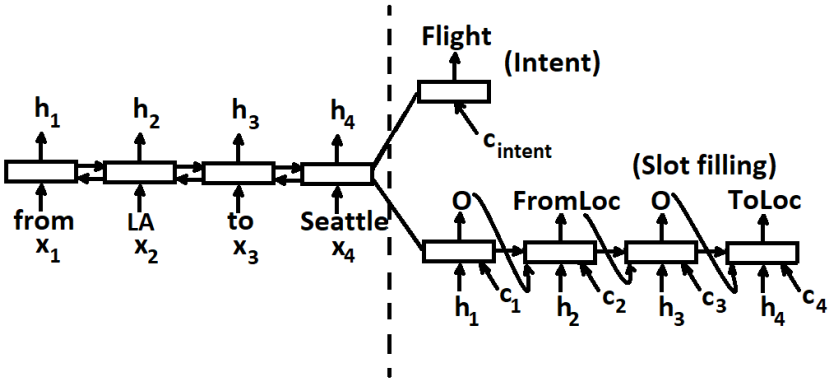


FIGURE 2.11 – Modèle RNN du type encodeur-décodeur bidirectionnel basé sur attention (Liu et Lane, 2016)

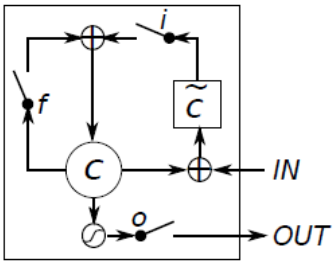


FIGURE 2.12 – L'unité LSTM de base (Chung *et al.*, 2014)

de données qui passe par cette unité.

Pour ce qui concerne le modèle RNN bidirectionnel du type encodeur-décodeur, l'encodeur est fait de deux parties : forward et backward. L'encodeur forward lit la séquence des tokens de gauche à droite et génère un état caché fh_i à chaque itération i (le temps i va de 1 à T). L'encodeur backward lit la même séquence de tokens de droite à gauche et génère un état caché bh_i à chaque itération i. Donc pour chaque pas du temps i, l'encodeur lit un mot x_i et émet un état caché h_i qui est une concaténation de fh_i et bh_i.

$$h_i = [fh_i, bh_i]$$

La partie décodeur est constituée de deux sous-parties : le décodage des slots et le décodage de l'intention. Le décodage de l'intention est fait par un LTSM RNN. L'état initial du décodeur des slots est calculé à partir du dernier état (qui contient l'information sur toute la séquence) de l'encodeur backward. Á chaque instant i, l'état du décodeur s_i est calculé comme résultat d'une fonction qui reçoit en entrée l'état précédent du décodeur (s_{i-1}), l'état caché précédent du décodeur (h_i), la sortie précédente du décodeur (y_{i-1}, c'est-à-dire le slot précédent) et un vecteur d'attention c_i. La sortie de s_i est le slot prédit.

$$s_i = f(s_{i-1}, y_{i-1}, h_i, c_i)$$

Sur la figure, l'absence de l'attribution du slot à un mot est marqué comme O.

Le vecteur d'attention c_i est un vecteur somme de tous les états cachés du l'encodeur à l'instant i, pondérés avec des poids appris ce qui assure la prise en compte de l'attention (focus) pour les différents parties de la séquence. Par exemple, dans la phrase *Ce restaurant sert de la cuisine italienne*, il faut faire plus attention au mot *italienne* pour la prédiction du slot *food*.

L'avantage de ce modèle d'un RNN simple pour l'annotation des séquences est due à la capacité du le RNN de type encodeur-décodeur de reporter une séquence vers une autre séquence de longueur différente.

Le modèle doit pas seulement prédire les slots mais aussi les aligner avec les segments correspondants de la phrase. On n'aligne pas les intentions car on suppose qu'une phrase n'a qu'une seule intention. Le modèle décrit utilise le soft alignment, c'est-à-dire qu'il attribue des probabilités pour chaque mot d'être associé à des slots différents (contrairement au hard alignment quand chaque mot est associé à un seul slot). C'est fait à l'aide du mécanisme d'attention : au lieu d'utiliser seulement un état caché h_i à chaque pas, on utilise aussi le vecteur d'attention c_i qui peut fournir une information additionnelle sur le contexte, surtout des dépendances longues qui ne sont pas bien capturées par l'état caché (les états cachés du RNN contiennent l'information de toute la séquence mais l'information peut se perdre lors de la propagation).

Tf-seq2seq

Tf-seq2seq est lui aussi un réseau neuronal récurrent du type encodeur-décodeur (Britz *et al.*, 2017), il est développé pour Tensorflow qui est un outil open source d'apprentissage automatique (Abadi *et al.*, 2015). Sur le site officiel de Tf-seq2seq[1], les développeurs proposent de l'utiliser pour la traduction automatique, le résumé automatique de textes, la modélisation des assistants virtuels (chatbots), la génération des légendes pour des images et d'autres applications.

La configuration par défaut de Tf-seq2seq utilise un encodeur du type RNN Bidirectionnel et un décodeur basé sur l'attention ; il a donc la même architecture que le modèle att-RNN utilisé dans la section 2.2.4. Par contre, chaque unité de l'encodeur et du décodeur est un Gated Reccurent Unit (GPU) et non un LSTM comme dans att-RNN. Les GPU ont été proposés pour la première fois par (Cho *et al.*, 2014). Ils ont une structure similaire aux LSTM décrit dans la section2.2.4, mais ils n'ont pas de porte de sortie.

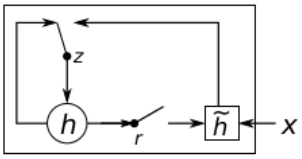


FIGURE 2.13 – L'architecture d'une unité GRU selon (Britz *et al.*, 2017)

La figure 2.13 montre une représentation d'une unité GPU proposée par (Cho *et al.*, 2014). La porte de mise à jour z choisit si l'état caché h devrait être mis à jour avec le nouveau état caché $\tilde{h}$. La porte de mise à jour r décide si l'état caché précédent doit être ignoré ou non.

La figure 2.14 montre l'architecture de Tf-seq2seq. La partie encodeur se situe à gauche de la ligne en pointillés, la partie décodeur est sur la droite, chaque partie est une séquence d'états cachés. Comme dans att-RNN, les vecteurs d'attention assurent que lors la prédiction d'un slot donné, le décodeur va faire plus attention à certaines parties de la phrase d'entrée qu'aux autres.

Contrairement à Tri-CRF, RASA et att-RNN, Tf-seq2seq ne demande pas de données alignées (alignement des mots de la source avec les slots de la cible) et ne prédit qu'une séquence.

1. https ://github.com/google/seq2seq

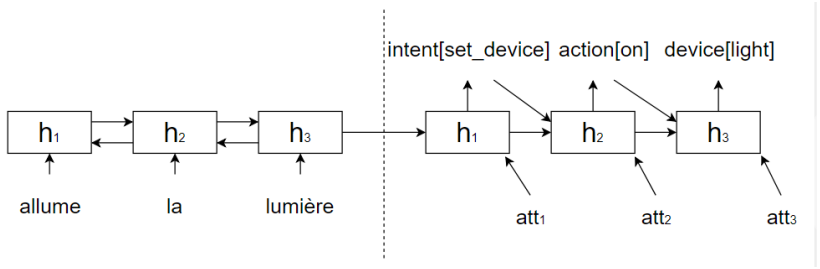


FIGURE 2.14 – L'architecture de Tf-seq2seq (Abadi *et al.*, 2015). Seules les couches cachées sont montrées

2.2.5 Développement d'un modèle

Après le choix d'un type de modèle d'apprentissage et d'un corpus, intervient le processus de développement du modèle sur un cycle à 6 étapes (Stubbs et Pustejovsky, 2012) (cf. figure 2.15).

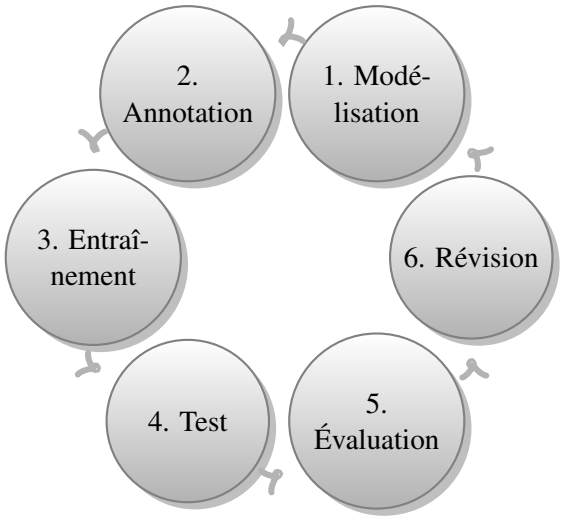


FIGURE 2.15 – Cycle de développement d'un modèle selon (Stubbs et Pustejovsky, 2012)

Les étapes sont les suivantes :

1. Modélisation. Il faut d'abord développer un schéma d'annotation qui est un ensemble de fonctions codant les descriptions structurelles et les propriétés spécifiques des données d'entrée. Ceci est décrit plus en détails dans la section 2.3.1.
2. Annotation du corpus. Le corpus est annoté selon le schéma d'annotation.
3. Entraînement. L'algorithme est entraîné sur la partie entraînement du corpus annoté.
4. Test. L'algorithme est testé sur la partie développement. Les résultats sont utilisés pour l'analyse des erreurs. Une fois les sources des erreurs trouvées, l'algorithme peut être ajusté et entraîné à nouveau. Cette procédure peut être répétée jusqu'à l'obtention de résultats satisfaisants.
5. Évaluation. La méthode la plus courante pour évaluer la performance de l'algorithme consiste à calculer avec quelle précision il annote l'ensemble de données. Dans la section 2.2.6, nous donnons un aperçu des métriques d'évaluations.

6. Révision. Cela consiste à retoucher le modèle afin de le rendre plus robuste et fiable. Par exemple, dans le cas du système de détection d'entités nommées, il peut s'avérer que l'annotation manque une catégorie importante. Par exemple, il peut s'avérer que les annotateurs ne savent pas comment traiter les noms des événements, tels que Pâques, ou Thanksgiving. Ils les annotent comme Temps mais on pourrait ajouter une nouvelle catégorie au modèle, c'est-à-dire, Événement. Puis ce cycle recommence, et les révisions apportent généralement une meilleure performance. Finalement, une fois que le développement du modèle est terminé, l'algorithme est exécuté sur le corpus de test. Cela montre à quel point l'algorithme fonctionne sur de nouvelles données.

2.2.6 Évaluation du modèle

Pour évaluer un système de compréhension automatique de la langue naturelle, on utilise 2 types d'évaluation standard : la comparaison entre la sortie du modèle en forme de représentation sémantique et la référence, et l'évaluation de bout-à-bout (end-to-end en anglais) (Tur et De Mori, 2011).

1. La comparaison entre la sortie du modèle en forme de représentation sémantique et la référence sous-entend l'utilisation des métriques suivantes :

   Le taux de bonne reconnaissance, ou l'accuracy au niveau de la phrase/énoncé (Sentence/utterance Level Semantic Accuracy en anglais) : le nombre d'énoncés qui ont reçu la représentation sémantique correcte divisé par le nombre total d'énoncés.

   Le Slot Error Rate (SER) : le nombre de slots insérés/supprimés/substitués divisé par le nombre de slots de la représentation sémantique de référence. Le SER mesure la performance du système au niveau des slots.

   La précision est le nombre de slots correctement détectés par le système divisé par le nombre total de slots détectés par le système.

   Le rappel est le nombre de slots correctement détectés par le système divisé par le nombre total de slots de la référence.

   Le F-score se calcule de la façon suivante :

$$F1 = 2\frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

   Il faut remarquer que si le système utilise aussi la détection d'intentions, les mesures sur les intentions sont faites de la même manière que sur les slots.

   Une autre remarque importante est que l'accuracy (accuracy en anglais) et la précision (precision en anglais) sont deux métriques différentes :

$$\text{Précision} = \frac{TP}{TP + FP},$$

   alors que

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

   où TP sont les résultats vrais positifs, TF les vrais négatifs, FP les faux positifs, et FN les faux négatifs.

2. L'approche précédente n'est pas convenable si on évalue des systèmes qui utilisent des représentations sémantiques différentes. L'évaluation de bout-à-bout résout ce problème. Par exemple, dans le projet de création du corpus d'évaluation ATIS par DARPA (Hemphill *et al.*, 1990), on vise à évaluer les systèmes en fonction des requêtes SQL qu'ils génèrent sur la base des données. Donc les systèmes reconnaissent les énoncés, les convertissent en représentation sémantique, et génèrent les requêtes SQL qui sont comparées aux requêtes SQL de référence créées manuellement.

Dans le cas où il ne s'agit pas seulement d'un système de compréhension automatique de la langue mais d'un système complet de dialogue, plusieurs métriques sont à prendre en compte comme par exemple dans le système d'évaluation PARADISE (Walker *et al.*, 2001). PARADISE a été utilisé dans le projet DARPA COMMUNICATOR pour évaluer neuf systèmes d'information et de planification de l'itinéraire pour les voyages. Le corpus était composé de 662 dialogues issus de ces 9 systèmes. Chaque dialogue consistait en un enregistrement, un fichier de trace (log file en anglais) qui enregistrait les sorties des étapes intermédiaires (par exemple, la parole reconnue), une transcription manuelle, et les résultats du sondage de l'utilisateur. Les métriques récoltées étaient les suivantes (avec leurs noms en anglais dans les parenthèses) :

1. L'efficacité du dialogue : la durée de la tâche (task duration), la durée totale (total elapsed time), le nombre des tours de dialogue de l'utilisateur/système, le nombre des tours de dialogue par tâche, la durée des tours de dialogue, le nombre de mots pour un tour de dialogue du système.
2. La qualité du dialogue : Le taux de bonne reconnaissance en mots (word accuracy), le taux de bonne reconnaissance en phrases (sentence accuracy), le temps d'attente (response latency), nombre de fois où il y avait des interruptions(number of overlaps).
3. La complétude par rapport à la tâche (task success) annotée manuellement pour chaque dialogue.
4. La satisfaction de l'utilisateur (user satisfaction) : mesurée à l'aide d'un questionnaire comportant 5 questions portant sur le degré de difficulté d'accomplissement de la tâche, le degré de la prédictibilité du comportement du système, la performance de la génération de parole, et le souhait d'utiliser le système dans le futur.

Les 3 premières métriques sont objectives alors que la dernière est subjective.

En faisant varier les modèles lors de l'évaluation, l'étude a relevé quatre mesures objectives qui prédisent 37% de la variation de la satisfaction de l'utilisateur telle qu'elle est calculée selon le sondage : la complétude par rapport à la tâche, la durée de la tâche, la durée du tour de dialogue du système et le taux de bonne reconnaissance en mots (on fait la somme de ces facteurs). Puis, 3 métriques ont été ajoutées : sur la prédiction des actes de dialogue (par exemple, « demander les informations », « offrir »), la prédiction des slots (ici appelés sous-tâches ; par exemple, la destination, la date) et de le domaine de la conversation (par exemple, *about_task* marque les énoncés sur l'accomplissement de la tâche, alors que *about_communication* marque les énoncés du système qui confirment la compréhension de ce que l'utilisateur vient de dire). Après l'ajout de ces métriques, la prédiction monte à 42% ce qui est une amélioration importante. Les auteurs concluent que les systèmes d'évaluation des systèmes dialogiques pourraient profiter de l'intégration de ces dernières métriques.

L'étude de (Pietquin et Hastie, 2013) énumère plusieurs métriques possible pour l'évaluation des systèmes qui prédisent l'acte de langage (par exemple, « inform ») à partir des dialogues réels annotés). Mais on peut supposer que ces métriques peuvent également être appliqués aux slots (slot-labels et slot-values).

Parmi ces métriques, on peut relever la divergence/dissimilitude de Kullback-Leibler (KL). La divergence de Kullback-Leibler calcule une seule valeur scalaire à partir de la comparaison de la distribution des actes de langage prédits avec la distribution des actes de langage cibles. Il faut rappeler que dans le slot-filling, les actes de langage sont synonymes d'intentions ; mais on peut appliquer la même métrique sur les slots. La formule est la suivante :

$$D_{KL}(P||Q) = \sum_{i=1}^{M} p_i log(\frac{p_i}{q_i})$$

où p_i est la fréquence d'un acte de langage a_i dans la distribution P ; et q_i est la fréquence d'un acte de langage a_i dans la distribution Q. (une des distributions est la prédiction, l'autre est la cible).

La divergence KL n'est pas une mesure de distance car $D_{KL}(P||Q) \neq D_{KL}(Q||P))$. Pour remédier à cela, la dissimilitude KL a été introduite :

$$DS_{KL}(P||Q) = \frac{D_{KL}(P||Q)+D_{KL}(Q||P)}{2}$$

Un désavantage de cette métrique est qu'elle accorde plus d'importance à la dissimilitude des distributions qu'à leurs variances.

Une autre métrique est la perplexité. Elle est utilisée pour comparer les modèles probabilistes, surtout les modèles langagiers.

L'intérêt de cette métrique est que si le modèle pour la prédiction des actes de langage est efficace, il prédira une forte possibilité pour les séquences de actes de langage que l'on lui donne dans le cadre du test. La perplexité est ainsi définie comme :

$$PP = 2^{\sum_{i=0}^{N} \frac{1}{N} log_2 p_m(x_i)}$$

où x_i est un exemple pris du corpus de test contenant N exemples, et p_m est la probabilité de x_i selon le modèle.

2.3 Les corpus

2.3.1 Généralités

Annotation des corpus

Un corpus utilisé pour l'apprentissage supervisé doit être annoté. Avant de commencer l'annotation, il faut d'abord définir les caractéristiques à prendre en compte par l'algorithme d'apprentissage en utilisant les observations sur les données à traiter (Resnik et Lin, 2010). Leur nombre devrait être suffisant pour modéliser le domaine afin que le système puisse bien généraliser. Si l'annotation peut se faire à l'aide de XML, le modèle décrit ci-dessus peut être représenté par DTD. Par exemple, dans la tâche d'annotation morphologique <noun>, <verb>, <adj>, le DTD pourrait être du type :

<!ELEMENT noun (#PCDATA) >
<!ELEMENT adj (#PCDATA) >
<!ELEMENT verb (#PCDATA) >
<!ATTLIST verb tense (past | present | future | none) #IMPLIED >

Découpage des corpus

La plupart du temps, le développement d'un modèle sous-entend la répartition des données en jeux de la manière suivante (Resnik et Lin, 2010) :

1. Les données d'entraînement (Train). Pour l'apprentissage supervisé, les éléments d'entrée sont donnés avec leurs étiquettes correspondantes (sortie désirée). Ces étiquettes sont souvent issues d'une annotation manuelle.
2. Les données de développement (Dev). Certains systèmes d'apprentissage comprennent des paramètres qui influencent leur performance. Par exemple, si un système d'étiquetage choisit sa sortie en se basant sur un vote pondéré $\lambda_i f_j(\text{entrée})$, où chaque f_j est une méthode différente de prédiction et les λ_i sont des poids pour les différentes méthodes. Au lieu de choisir arbitrairement des poids ou des paramètres, il est courant de réserver un certain sous-ensemble des données d'apprentissage en tant que les données de développement. Grâce à ce sous-ensemble, on fait une recherche de bonnes valeurs pour λ_i, soit de manière ad hoc manuelle, soit en utilisant une technique d'optimisation telle que l'espérance-maximisation. Dans les deux cas, les performances sur les données de développement aident à optimiser le choix des paramètres.
3. Les données de test de développement (Dev-test). Parfois, un ou plusieurs ensembles de données sont également réservés pour utiliser dans l'évaluation intermédiaire du cycle du développement du système et son amélioration.
4. Les données de test (Test). Ce jeu comprend les données qui seront utilisées pour évaluer la performance du système après le développement, c'est-à-dire pour l'évaluation finale.

Il est courant de réserver autant de données que possible pour l'entraînement et le développement. Par exemple, on pourrait diviser les données disponibles respectivement en 70%, 20% et 10% pour l'entraînement, développement (Dev-test n'étant pas prévu dans ce cas), et des données de test. Ces données ne doivent pas avoir des échantillons en commun.

2.3.2 Corpus existants

Comme nous venons de le dire, pour entraîner un système, il est nécessaire de disposer d'un corpus d'apprentissage. Un état de l'art des corpus existants est fait dans (Serban *et al.*, 2015). Nous allons décrire cinq corpus dont un est en anglais (ATIS) et les autres - MEDIA, PORT-MEDIA, SWEET-HOME et VocADom - en français. Les corpus SWEET-HOME et VocADom relèvent de notre domaine, l'interaction dans l'habitat intelligent.

ATIS

Un des corpus les plus anciens pour apprendre les modèles en compréhension automatique/génération est ATIS (Air Travel Information System) (Hemphill *et al.*, 1990). Ce corpus en anglais est un de premiers corpus de communication homme-machine. Il contient des interactions de 40 minutes environ chacune entre des participants humains et un système de réservation de billets d'avion opéré par des humains en magicien d'Oz (Wizard of Oz). Le nombre total des énoncés est 1041.

Les énoncés dans ATIS sont représentées de la façon suivante : chaque énoncé a un but (*goal* en anglais, qui est équivalent de l'intention) et des slots complétés par les informations de la phrase. Un exemple d'énoncé annoté correspondant à ce corpus (Tur *et al.*, 2010) est affiché sur le tableau 2.1.

Utterance	How much is the cheapest flight from Boston to New York tomorrow morning ?
Goal	Airfare
Cost_Relative	cheapest
Depart_City	Boston
Arrival_City	New York
Depart_Date_Relative	tomorrow
Depart_Time_Period	morning

TABLE 2.1 – Exemple d'un énoncé annoté du corpus ATIS (Tur *et al.*, 2010)

Le corpus contient 16 intentions, telles que *flight reservation*. Mais la distribution des intentions n'est pas égale : l'intention la plus fréquente (*flight reservation*) représente à elle seule 70% des intentions.

Il est évident que ce corpus seul n'est pas suffisant pour apprendre un système qui soit capable de traiter des tâches plus complexes qu'une simple requête d'informations sur les vols, et qui ne soit pas biaisé (over-tuned en anglais) envers un type de données particulier (Tur *et al.*, 2010). Pour cela, il faut utiliser d'autres corpus.

MEDIA et PORTMEDIA

En ce qui concerne le français, il existe le corpus français PORTMEDIA qui a été également collecté à partir d'un service téléphonique pour la réservation des billets lors du festival d'Avignon de 2010 (Lefèvre *et al.*, 2012). Il contient 700 dialogues enregistrés et annotés sémantiquement. Ce corpus fait suite à la création du corpus français MEDIA (Bonneau-Maynard *et al.*, 2009) créé à partir d'un service téléphonique d'informations touristiques. Les deux corpus ont été traité suivant les mêmes principes décrits ci-après.

L'annotation a été faite de la façon suivante : les tours de dialogue ont été découpés en segments, en présumant que chaque segment exprime un seul acte de langage. Puis, chaque segment a été annoté.

Les notions basiques de la représentation sémantique de ce corpus sont : les attributs (par exemple, *ville*), la valeur des attributs (par exemple, *Paris*) et le mode qui peut être positif, négatif, interrogatif ou optionnel pour désambiguïser les phrases telles que *pas à Paris, à Nancy*. Une autre notion est celle du spécifiant (specifier en anglais) qui est le lien qu'on peut créer entre deux attributs, par exemple, si on a une phrase *je voudrais que le prix soit inférieur à 150 euros*, on lie l'attribut *prix* avec l'attribut *entier* (dont la valeur est 150) et avec l'attribut *monnaie* (dont la valeur est *euro*). Cela permet de représenter les relations hiérarchiques existants dans la phrase. En total, le corpus PORTMEDIA compte 83 attributs, 19 spécifiants et 4 modes.

Si on compare cette représentation avec celle basée sur les intentions et les slots, les attributs de MEDIA et PORTMEDIA sont la fusion de ces concepts car un attribut peut définir une intention. Dans ce cas, cet attribut est appelé *command-task*. Sur la figure 2.2, on peut voir un exemple de l'annotation de la phrase *Je voudrais faire une réservation pour le trente et un mai deux jours deux nuits à Paris mais dans un hôtel qui se trouverait près de la place de la Bastille s'il vous plaît pour six chambres individuelles* extraite du corpus MEDIA.

Original :

Je voudrais faire une réservation pour le trente et un mai deux jours deux nuits à Paris mais dans un hôtel qui se trouverait près de la place de la Bastille s'il vous plaît pour six chambres individuelles

Annoté :

Segment	Mode/attribut :valeur
Je voudrais faire une réservation	+/command-task :reservation
pour le trente et un mai	+/time-date : 05/31
deux jours deux nuits	+/stay-nbNight-reservation : 2
à Paris	+/location-city : paris
mais	+/connectProp : addition
dans un hôtel	+/DBObject : hotel
qui se trouverait près de	+/location-relativeDistance-hotel : near
la place de la Bastille	+/location-street-hotel : bastille
s'il vous plaît pour six	+/number-room-reservation : 6
chambres individuelles	+/room-type : single

TABLE 2.2 – Exemple d'une annotation de la phrase dans le corpus MEDIA (Bonneau-Maynard *et al.*, 2009)

SWEET-HOME

Les trois corpus décrits ci-dessus ne relèvent pas du domaine que nous traitons dans ce travail, la commande vocale de l'habitat intelligent. Pourtant, il existe un corpus SWEET-HOME qui a été recueilli dans un habitat intelligent DOMUS, équipé avec des microphones qui enregistrent la parole et des capteurs qui fournissent les informations sur la localisation et l'activité de l'utilisateur (Vacher *et al.*, 2014).

Les participants prononçaient des commandes vocales choisies dans un jeu défini par les expérimentateurs.Ces commandes se conformaient à la grammaire suivante (les participants n'avaient eux-même accès à la grammaire) :

basicCmd = key initiateCommand object | key emergencyCommand
key = "Nestor"
initiateCommand = "ouvre" | "ferme" | "baisse" | "éteins" | "monte" | "allume" | "descend" | "appelle" " | "donne"
emergencyCommand = "au secours" | "à l'aide"
object = [determiner] (device | person | organisation)
determiner = "mon" | "ma" | "l'" | "le" | "la" | "les" | "un" | "des" | "du"
device = "lumière" | "store" | "rideau" | "télé" | "télévision" | "radio" | "heure" | "température"
person = "fille" | "fils" | "femme" | "mari" | "infirmière" | "médecin" | "docteur"
organisation = "samu" | "secours" | "pompiers" | "supérette" | "supermarché"

Les commandes ont été exécutées par un système domotique automatisé (ou par un Magicien de Oz en cas de problème).

Le corpus résultant comprend 4 parties qui comprennent les enregistrements de l'interaction des participants avec l'habitat. Elle ont des caractéristiques suivantes :

1. Groupe interagissant avec l'habitat. Les enregistrements sont reliés avec les données des capteurs indiquant la localisation du locuteur (pour pouvoir développer les systèmes prenant en compte le contexte). Participants : 21 personnes. 1779 énoncés.
2. Groupe faisant des enregistrements de parole à distance, c'est-à-dire, capturés par les microphones dans le plafond, pas par les microphones portables (pour pouvoir développer les systèmes de reconnaissance de parole à distance utilisable dans l'habitat intelligent). Dans le cas il n'y a eu ni magicien d'Oz ni interaction avec le système. Participants : 23 personnes. 5520 énoncés.
3. Premier groupe de participants interagissant avec le système : 16 personnes sans handicap. 993 énoncés.
4. Second groupe de participants interagissant avec le système : 6 personnes âgées, 5 personnes ayant des troubles de vision. 907 énoncés.

Les enregistrements ont été transcrits et annotés avec les informations sur les paroles prononcées, l'activité et la localisation du locuteur à un moment donné. Les chercheurs constatent que les résultats obtenus ne sont pas suffisants pour développer un système capable de fonctionner dans les conditions réelles mais restent intéressants vu le manque des corpus de ce type.

VocADom

Un autre corpus relevant du domaine de l'habitat intelligent est le corpus Vocadom recueilli dans les conditions réelles dans Amiqual4Home (Vacher, 2018). C'est un habitat intelligent équipé avec des outils multimédia, des capteurs pour la consommation d'eau et d'énergie et des microphones. La cuisine et le salon sont sur le rez-de-chaussée alors que la chambre et la salle de bain sont à l'étage. Le plan de la maison est dans l'annexe A.

L'enregistrement des données de tous les capteurs et le contrôle des outils est fait à partir d'une salle de contrôle (la régie) à proximité de l'habitat intelligent. Les commandes des participants ont été exécutées par un magicien d'Oz.

Chaque participant a joué des activités de la vie quotidienne pendant environ une heure. Dans la première partie de l'expérimentation, les participants n'étaient pas contraints par la grammaire des ordres (pour permettre plus de variabilité dans les ordres). On leur a donné des images représentants des scénarios. La figure 2.16 montre une de ces images.

En ce qui concerne la deuxième partie de l'expérimentation, on a donné aux participants une liste des énoncés à lire (pas tous les énoncés étaient des commandes), par exemple :

je ferme le verrou
non merci bien
vocadom baisser lumière
diminuer le chauffage
téraphim arrête la radio de la salle de bain

Le format du corpus résultant est le suivant (les balises indiquent les repères temporaires) :

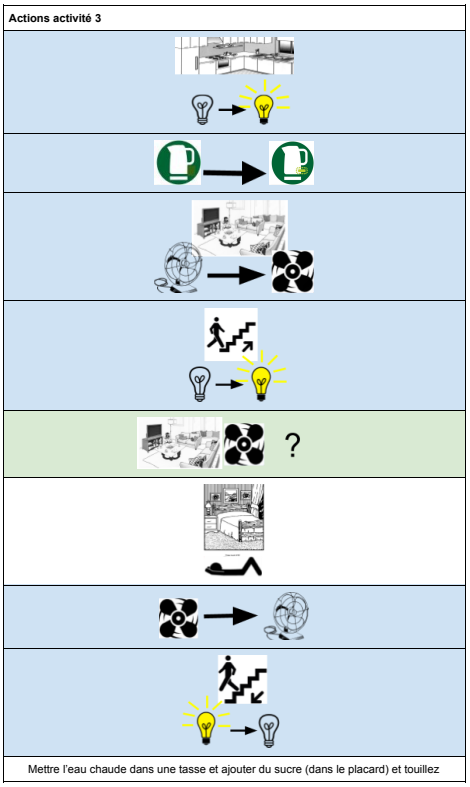


FIGURE 2.16 – L'image utilisée pour éliciter des ordres dans le cadre du recueil du corpus VocADom

<Sync time="2780.720000"/>
c'est ici
<Sync time="2781.572000"/><Sync time="2783.472000"/>
vocadom baisse le store du bureau
<Sync time="2785.348000"/><Sync time="2786.817"/>
dis hestia stoppe les stores du rez-de-chaussée
<Sync time="2789.316000"/>

Il faut faire une remarque sur les mots-clés utilisés ; c'est-à-dire les mots utilisés pour adresser au système. Dans le cadre du projet VocADom, il a été décidé qu'une commande vocale doit toujours commencer par un mot clé, ce qui déclenche les étapes suivantes de traitement. C'est pourquoi les mots clés devraient être choisis avec prudence. Il faut aussi tenir en compte que la qualité du parole se détériore avec l'âge. Selon (Aman, 2014), les voyelles les mieux reconnues par les systèmes de reconnaissance du parole (pour tous les locuteurs) sont i, y, u, ɛ, alors que les consonnes les plus reconnues sont s, ʃ, m, ʁ.

Les noms qui respectent ces contraintes ont été choisis automatiquement à partir d'un dictionnaire et puis filtrés, ce qui a donné la liste suivante : *téraphim, ulysse, ichefix, chanticou, vocadom, écirrus, hé cirrus, allô cirrus, allô messire, dis vesta, dis hestia, dis bérénio, dis téraphim, dis vocadom, minouche.* (Rossato *et al.*, 2018)

Le nombre total des commandes du corpus est 1646.

Il est intéressant de souligner que si le corpus SWEET-HOME ne contient que des énoncés lus, VocADom lui, contient des énoncés spontanés et lus. Tous les deux ne sont pas annotés sémantiquement.

2.4 Travaux préliminaires dans le cadre du projet VocADom

Le problème de la compréhension automatique du langage dans le cadre du projet ANR VocADom a déjà fait l'objet d'un travail initial par Stefania Raimondo (Raimondo, 2017) qui a abouti à la création de la grammaire de génération que nous comptons améliorer dans le cadre de notre travail.

Dans ce but, une grammaire a été écrite avec un script Python qui génère les ordres (sous forme textuelle) annotés à partir de cette grammaire. Ceci a permis de générer un corpus artificiel. Le script Python est basé sur la classe FeatStruct (Feature-Based Grammar, ou grammaire des attributs) de Natural Language ToolKit (NLTK) (Bird *et al.*, 2009).

Un exemple d'ordre généré est montré ci-dessous.

KEYWORD allume la radio dans le salon 'action' : 'TURN_ON', 'device' : 'music_player', 'intent' : 'set_device', 'location-room' : 'living_room'

Comme on peut voir, l'intention ici est *set_device*, avec les slots *device*, *action* et *location*. Au total, la grammaire développée a 11 intentions, telles que :

Contact : « appelle ma fille »
Set_device : « ouvre le porte du salon un peu »
Set_device_group : « ouvre les portes du salon »
Set_device_property : « augmente le volume de la radio »
Set_device_group_property : « diminue le volume des radios »
Set_room_property : « augmente le température du salon »
Check_device : « est-ce que la porte du salon est ouverte ? »
Check_device_group : « est-ce que la télé est allumées ? »
Check_room_property : « est-ce qu'il y a quelqu'un dans le salon ? »
Check_world_property : « quel temps fait-il ? »

On peut voir que l'on a 3 catégories d'intention : *contact, set* et *check*. En ce qui concerne les slot-labels et slot-values, ils sont énumérés dans la table 2.3.

La génération des ordres a été implémentée à partir de représentations sémantiques contenant le type d'intention avec les slots correspondants, par exemple, pour générer un ordre *Allume la lumière*, on utilise une représentation où l'intention est *set_device*, et les slots sont *action* et *device*. On crée donc une règle où *Dact_set_device* est un prédicat qui renvoie vers les prédicats qui définissent les slots :

Dact_set_device[ACTION= ?s,Location= ?l, Device= ?d] ->
Slot_action[ACTION= ?s]
Slot_device[ALLOWABLE_ACTION= ?s, Location= ?l, Device= ?d, ARTTYPE= ?a]

Prenons comme exemple le prédicat définissant le slot *Slot_action*.

TABLE 2.3 – Grammaire : *slot-labels* et *slot-values*

Slot-category	Slot-label	Slot-values
action	action	turn_on, turn_off, open, close, raise, lower, turn_up, turn_down, play, stop, pause, change, set_to...
device-setting	device-setting	on, off, opened, closed
device	device	radio, light, door, window, cupboard, drawer, stove, fridge, dishwasher, washing_machine, range_hood, heating, shutter, tap, tv_music_player, music_player, kettle, vacuum, fan
	device-component	temperature, location, volume, channel, color
	device-property	location
value	value-quantity-qualitative	a_bit, a_lot
	value-numeric-relative	5 (exemple : « by 5 degrees »)
	value-numeric	1, 2.5 ...
	value-unit	degree_celcius
	value-artist	<n'importe quelle chaîne de caractères>
	value-genre	<n'importe quelle chaîne de caractères>
	value-album	<n'importe quelle chaîne de caractères>
	value-color	blue, red ...
location	location-room	living_room, corridor ...
	location-floor	0, 1 ...
	location-house	house
	location-relative-to	above, below, left_of, right_of, next_to, inside, outside, under, on_top_of
	location-relative	right, left, upper, lower, first/second/..., closest, furthest
	location-relative-to-object	sofa, table, etc ...
person	person-relation	mother, brother, son, husband, friend, colleague ...
	person-name	<any string >
	person-occupation	doctor, nurse
organization	organization	emergency, supermarket, fire_brigade
device_property	device_property	location, volume
room_property	room_property	temperature, presence
world_property	world_property	time, weather

Slot_action[ACTION= ?s] -> Action[ACTION= ?s]

Le prédicat Action ramène au prédicat SpecAction qui spécifie quels verbes peuvent être utilisés dans le sens *TURN_ON*.

Action[ACTION= ?s] -> SpecAction[ACTION= ?s]
SpecAction[:ACTION=(TURN_ON)] -> V[SEM=allumer]

Puis, on utilise un vocabulaire pour passer d'une représentation sémantique à un mot. La sémantique *allumer* est représentée par deux mots dans le vocabulaire :

V[AGR=[TENSE=imperatif,PERS=2,PLUR=sing, VOWEL=voy], VT=transitive, SEM=allumer] -> 'allume'
V[AGR=[TENSE=imperatif,PERS=2,PLUR=plur, VOWEL=voy], VT=transitive, SEM=allumer] -> 'allumez'

Les informations grammaticales données dans l'attribut AGR s'assurent que l'on produit des phrases respectant les accords entre l'article et le nom et dans les autres cas.

De la même façon, on traite *Slot_device* pour obtenir la phrase *Allume la lumière*. De cette façon, on obtient un corpus des ordres annotés. Le nombre de phrases générées est 62.

En conclusion, ce travail préliminaire a été la première étape de la conception d'un corpus artificiel qui peut être utilisé pour apprendre un modèle de la compréhension de la langue pour les ordres vocaux dans l'habitat intelligent. Pourtant, on peut constater que ce corpus artificiel est extrêmement petit et qu'il nécessite un enrichissement sémantique et syntaxique très important. Étant donné sa taille, il n'est pas utilisable pour entraîner un système qui pourrait être évalué sur

un corpus réaliste comme VocADom.

Chapitre 3

Problématique et méthode

Nous avons montré les défis principaux concernant le domaine du développement des systèmes domotiques pour les personnes âgées : le besoin d'adapter le système à la grammaire propre au locuteur, en prenant en compte les particularités de la parole des personnes âgées. Nous avons conclu que les méthodes statistiques sont plus adaptées pour cette tâche que les méthodes par règles, ces dernières n'étant pas suffisamment flexibles.

Pourtant les méthodes statistiques demandent des corpus suffisamment grands - de plusieurs milliers de phrases - pour apprendre un système de compréhension automatique de la langue. Nous n'avons que 2 petits corpus SWEET-HOME et VocADom à notre disposition. Le problème du manque des données d'entraînement peut être résolu par la génération d'un corpus artificiel à partir de la grammaire existante. Dans ce but, on peut prendre la grammaire existante de génération décrite dans la section 2.4. Cette grammaire ne génère que 62 phrases annotées et devrait être enrichie sémantiquement et syntaxiquement. On pourrait faire s'inspirer des corpus réalistes pour y puiser les éléments caractéristiques du langage des personnes âgées : les caractéristiques syntaxiques comme des tournures de politesse, mais aussi les éléments sémantiques comme des synonymes des mots déjà existants dans la grammaire (*supérette* pour *supermarché* et d'autres) etc.

Pour évaluer notre corpus, nous avons décidé d'apprendre quelques modèles neuronaux sur ce corpus, et puis les évaluer sur le corpus réaliste VocAdom.

Il faut s'assurer que l'on choisit les méthodes d'apprentissage les plus adaptées à la tâche de la compréhension automatique, c'est-à-dire, la détection des intentions et le remplissage des slots. Pour faire cela, il faut comparer plusieurs modèles. Nous allons utiliser Tri-CRF, att-RNN, RASA et Tf-seq2seq. Les trois premiers modèles sont l'état de l'art dans ce domaine mais ils ont des inconvénients suivants. Tri-CRF et att-RNN requièrent l'entraînement de deux modèles séparés : un modèle pour la prédiction des intentions et slot-labels, l'autre pour la prédiction des slot-values. RASA requiert trois modèles - pour l'intention, slot-labels et slot-values. En plus, Tri-CRF, att-RNN et RASA requièrent les données alignées, qui ne sont pas toujours disponibles. Tf-seq2seq a comme avantage le fait qu'il ne requiert qu'un seul modèle et ne demande pas de données alignées. Un autre aspect dans lequel il est différent à trois modèles précédant, est qu'il n'a pas de séparation explicite entre les intentions et les slots. Il est intéressant de savoir si malgré tous ces particularités, Tf-seq2seq peut être compétitif par rapport à trois autres modèles.

Notre démarche peut être divisée en deux grandes étapes.

D'abord, nous allons générer et évaluer un corpus artificiel à l'aide des modèles Tri-CRF, att-RNN, RASA et Tf-seq2seq. Pour pouvoir entraîner les modèles sur le corpus artificiel, sa grammaire devrait être encore enrichie du point de vue sémantique et syntaxique. Le nombre et

les noms des intentions, slots-labels et slot-values seront révisés. Le corpus devrait être rendu en format convenable pour chaque modèle. Aussi, nous aurons besoin des corpus existants réalistes VocADom et SWEET-HOME qui devraient être annotés, normalisés et rendus en formats nécessaires. Ensuite Tri-CRF, att-RNN, RASA seront entraînés sur le corpus artificiel et testés sur le corpus réaliste VocADom. Tf-seq2seq sera entraîné en mode mot sur le corpus artificiel avec l'addition du corpus réaliste SWEET-HOME. Ce dernier est extrêmement petit en comparaison avec le corpus artificiel mais nous l'avons ajouté pour augmenter la variation lexicale des données d'entraînement. Puis ce modèle sera également testé sur le corpus réaliste VocADom. Le test sur VocADom nous permettra de savoir quelle pourrait être la performance d'un système appris sur notre corpus artificiel dans des conditions réalistes. On va également comparer ces résultats avec les résultats des 4 modèles (RASA, Tri-CRF, att-RNN, Tf-seq2seq) appris et testés sur PORTMEDIA, pour être sûr que nos conclusions sont tirées sur la base des limitations des corpus plutôt que les limitations des modèles eux-mêmes. L'entraînement de RASA, Tri-CRF et att-RNN sera fait par notre collègue Thierry Desot.

La deuxième étape consiste à étudier la performance de Tf-seq2seq et elle contient deux sous-étapes.

La première sous-étape consiste à étudier sa performance en variant la mode : mot et caractère, et la tâche : la prédiction des intentions et slots, et la prédictions des slots seulement. L'étude des performances en mode mot vs mot caractère nous permettra de savoir quelle mode est plus convenable pour le corpus avec le vocabulaire d'une certaine taille. Nous allons utiliser un modèle Tf-seq2seq en mode caractère sur le corpus artificiel avec l'addition du SWEET-HOME et puis le tester sur VocADom. Puis nous allons comparer ces résultats avec ceux du modèle à mots appris à la sous-étape précédente, sur ce même corpus.

Le deuxième sous-étape consiste à savoir si l'absence de la séparation des intentions et slots a une influence négative sur la performance de Tf-seq2seq. Nous allons apprendre Tf-seq2seq (en mode mot et caractère) sur le corpus PORTMEDIA dont tous les intentions sont supprimées. La tâche donc sera réduite à la prédiction des slot-labels et slot-values. La comparaison de ces résultats avec les résultats du modèle appris sur PORTMEDIA avec intentions (en mode mot) à l'étape précédente, nous permettra de supposer que l'intention nuit ou aide à la prédiction des slots.

Il faut rappeler que le corpus VocADom est un corpus des énoncés capturés dans une expérimentation réaliste (habitat intelligent Amiqual4home) qui comprend des énoncés spontanés (elicités par des images) et des énoncés lus. Le corpus SWEET-HOME est un corpus d'énoncés lus, également enregistré dans un habitat intelligent réaliste (DOMUS). Les deux corpus sont décrits dans la section 2.3.2. Par le corpus artificiel, on comprend le corpus généré à partir de la grammaire qui est décrit dans la chapitre 2.4.

Les étapes décrites ci-dessus feront l'objet de la prochaine section.

Chapitre 4

Génération et évaluation du corpus artificiel

Dans cette section, il s'agit de la génération du corpus artificiel et son évaluation sur les modèles états de l'art Tri-CRF, RASA et att-RNN, aussi bien que Tf-seq2seq, modèle séquence-vers-séquence. En accord avec les objectifs décrits dans la problématique, notre travail consistera en étapes suivantes :

1. La modification de la grammaire générant le corpus artificiel ;
2. L'annotation des corpus réalistes VocADom et SWEET-HOME ;
3. La comparaison de tous les corpus utilisés : l'artificiel, VocADom, SWEET-HOME et PORTMEDIA ;
4. L'évaluation du corpus artificiel avec Tri-CRF, RASA, att-RNN et Tf-seq2seq.

Ces étapes seront décrites plus en détails dans les sections qui suivent. Il faut remarquer que la description de l'entraînement de Tf-seq2seq est plus détaillée et elle est faite dans une section séparée car l'entraînement de RASA, Tri-CRF et att-RNN a été fait par notre collègue Thierry Desot, et l'entraînement de Tf-seq2seq par nous.

4.1 La modification de la grammaire du corpus artificiel

Dans cette section, il s'agit de la modification de la grammaire existante qui génère le corpus artificiel des commandes. Ci-dessous sont énumérés quelques modifications apportées : la révision de l'intention du type *check* et *get*, l'enrichissement sémantique et syntaxique, le homogénéisation de la langue de la grammaire, l'ajout de KEYWORD qui est sensé de substituer le mot clé avec lequel on s'adresse au système. Finalement, on génère le corpus annoté en formats traitables par RASA, Tri-CRF et att-RNN.

4.1.1 Redéfinition des intentions

Au départ, les phrases du type *Quelle heure est-t-il* ont été annotées avec l'intention *check_world_property*, et les phrases du type *Est-ce que les lumières du salon sont éteintes* avec l'intention *check_device_property*. Nous avons décidé de faire deux intentions distinctes : du type *get* et *check*. Ainsi, les intentions du type *get* sont utilisées dans le cas d'une question ouverte, comme *Quelle heure est-t-il*. Donc on l'a annotée avec l'intention *get_world_property*. Les intentions

du type *check* sont utilisées dans les questions du type oui-non, comme *Est-ce que les lumières du salon sont éteintes*. On l'a annoté avec l'intention *heck_device_property*.

4.1.2 Enrichissement sémantique

Il y avait beaucoup de vocabulaire qui était présent dans le fichier vocabulaire mais qui était absent des commandes générées, par exemple : les mots *store, porte, frigo*. Pour chaque mot, nous avons déterminé à quelle étape de réécriture il y avait le problème et nous avons corrigé/ajouté les règles nécessaires.

De plus, au départ, seulement les phrases les plus grandes possibles ont été générées, du type :
descends le volet du salon du rez-de-chaussée de la maison

Mais nous avons fait en sorte que la grammaire génère également :

descends le volet
descends le volet du salon
descends le volet du salon du rez-de-chaussée
descends le volet du salon du rez-de-chaussée de la maison

On a déterminé quelques synonymes qui pourraient être ajoutés dans le vocabulaire, par exemple, *la supérette* pour *le supermarché*, *stopper* et *éteindre* pour *arrêter*. Aussi, les mots précisant la localisation comme *la porte d'entrée* (au lieu de *la porte*) *lumière plafonnier, lampe de chevet,* et les phrases telles que *baisse l'intensité de la lumière* au lieu de *baisse la lumière*.

On a aussi ajouté un mot clé au début des phrases générées (*KEYWORD allume la lumière*). KEYWORD est censé de substituer le mot clé que l'on utilise quand on s'adresse au système.

4.1.3 Enrichissement syntaxique

En accord avec les études sur les particularités de l'interaction des personnes âgées avec l'habitat intelligent décrites dans la section 2.1, nous avons ajouté quelques constructions syntaxiques, notamment, l'ajout des constructions de politesse : *s'il te (vous) plaît* au début/fin des phrases, les constructions du type *Tu peux allumer ..., Est-ce que tu peux allumer ...*, les infinitifs : *ouvrir les fenêtres*.

Nous avons remarqué que dans le corpus réaliste, il n'y a pas de consistance en ce qui concerne l'usage des articles. Ils sont : parfois définis (*appeler l'infirmière*), parfois indéfinis (*appelle un docteur*), parfois partitifs (*du secours s'il vous plaît*) parfois absents (*nestor appeler secours*). Nous avons donc généré les phrases annotées avec intention *contact* avec les articles défini, indéfini et absent.

4.1.4 Homogénéisation de la langue de la grammaire

Initialement, il y avait une utilisation conjointe de termes anglais et français. Nous avons donc décidé d'utiliser uniquement l'anglais pour des raisons d'homogénéité et de communication, notamment pour faciliter la mise en accès libre. Par exemple, la règle :

SpecDeviceFour[Location=[Room=cuisine], AGR=[PLUR=sing,GENDER= ?g,VOWEL= ?v], ID=four_cuisine] -> WordDeviceFour[AGR= ?a]

a été traduite en :

SpecDeviceOven[Location=[Room=kitchen],AGR=[PLUR=sing,GENDER= ?g,VOWEL= ?v], ID=oven_kitchen] -> WordDeviceOven[AGR= ?a]

4.1.5 Ajout de KEYWORD

Le corpus a été généré d'abord dans le format montré ci-dessus (la phrase *KEYWORD tu peux fermer le store* annotée) :

KEYWORD tu peux fermer le store
{'action' : 'CLOSE', 'device' : 'blind', 'intent' : 'set_device', 'location-room' : 'living_room'}

Ici *KEYWORD* est censé de remplacer le nom par lequel on s'adresse au système.

4.1.6 Génération

A partir du format ci-dessus, nous avons dû générer trois versions de ce corpus en formats traitables par RASA, Tri-CRF et att-RNN ; ce qui a inclus l'alignement des slots avec les segments de la phrase. (Tf-seq2seq ne requiert pas d'alignement). Le script qui gère la génération a été modifié pour pouvoir aligner les slots avec le texte. Ci-dessous est la phrase *KEYWORD tu peux fermer le store* annotée dans le format RASA. Les valeurs de *start* et *end* indiquent la position du segment auquel le slot est associé :


```
{ "text" : "KEYWORD tu peux fermer le store"",
"entities" : [
    {
    "start" : 16,
    "end" : 22,
    "entity" : "action",
    "value" : "CLOSE",
    "text" : "fermer",
    },
    {
    "start" : 23,
    "end" : 31,
    "entity" : "device",
    "value" : "blind",
        "text" : "le store",
    }
],
"intent" : "set_device"
```

}

Tri-CRF et att-RNN utilisent le format BIO qui contient les mêmes informations. Nous avons donc converti le corpus en ce format. Ci-dessus est l'exemple d'une phrase en BIO, annotés avec l'intention et les slot-labels :

contact KEYWORD appelez l' ambulance
NONE word=KEYWORD word-1=<s> word+1=appelez word+2=l'
action-B word=appelez word-2=<s> word-1=KEYWORD word+1=l' word+2=ambulance
organization-B word=l' word-2=KEYWORD word-1=appelez word+1=ambulance word+2=</s>
organization-I word=ambulance word-2=appelez word-1=l' word+1=</s>

Ce format sous-entend l'alignement de l'ensemble de la phrase avec une intention (la première ligne) et l'alignement du chaque mot avec un slot-label. Le slot-label peut être du type xxx-B ou xxx-I où xxx est le nom du slot, B signifie que le slot en question commence à partir de ce mot, I signifie que le slot en question est étendu sur ce mot, donc il est associé à plus d'un mot (comme *organization* dans l'exemple ci-dessus). Si le mot n'est pas associé avec un slot, il est aligné avec slot-label *NONE*. Chaque mot est également annoté à un mot précédent et deux mots suivants.

Ci-dessus est l'exemple de la même phrase en BIO, annotés avec l'intention et les slot-values :

contact KEYWORD appelez l' ambulance
NONE word=KEYWORD word-1=<s> word+1=appelez word+2=l'
action-B word=appelez word-2=<s> word-1=KEYWORD word+1=l' word+2=ambulance
organization-B word=l' word-2=KEYWORD word-1=appelez word+1=ambulance word+2=</s>
organization-I word=ambulance word-2=appelez word-1=l' word+1=</s>

Au total, 26082 phrases annotées ont été générées pour la première version et 42195 phrases pour la deuxième. La différence entre les deux versions est que nous avons fait des changements dans les attributs *location* (localisation) des *devices*. La grammaire de départ avait un fichier de spécification où chaque device a été attribué une chambre. Par exemple, *douche* a été attribué la chambre *bathroom* (salle de bain) et l'étage (rez-de-chaussée ou l'étage). Nous avons supprimé la contrainte de l'étage ce qui nous a produit plus de phrases. La contrainte de la chambre a été laissée en place pour éviter la génération des phrases du type : *allume la four de la salle de bain*. Le corpus artificiel étant prêt, nous sommes passés à l'annotation des corpus existants VocADom et SWEET-HOME, décrite dans la section suivante.

4.2 Annotation des corpus VocADom et SWEET-HOME

Dans cette section, il s'agit de l'annotation des corpus réalistes VocADom et SWEET-HOME. La grammaire de génération nous a servi comme le schéma d'annotation.

Dans le cadre des travaux préliminaires du projet VocADom, il a été fait une interface qui facilite l'annotation du corpus VocADom (cf. 4.1).

Pour annoter une phrase, il faut cliquer dessus ; et remplir les champs qui apparaissent.

Comme le travail d'annotation est fait manuellement, cela prend beaucoup de temps. Nous avons élaboré un script qui fait la pré-annotation du corpus avec les intentions suivantes :

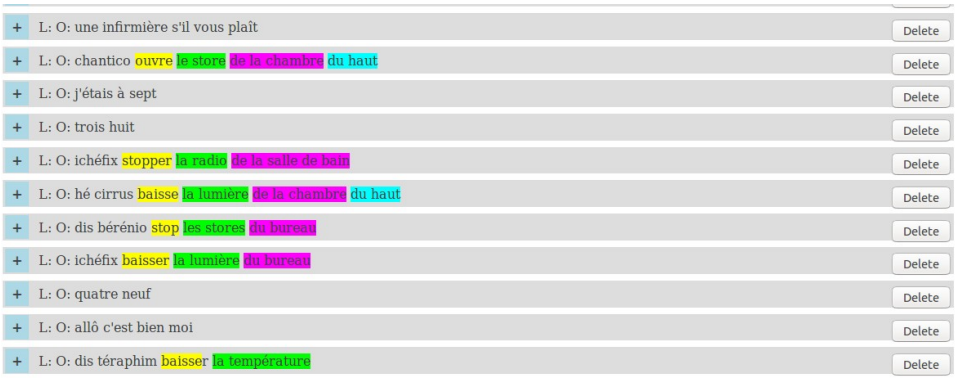


FIGURE 4.1 – L'interface pour annotation

set_device et *set_device_group* (slots *action, device), check_device* et *check_device_group* (slots *device*, *device_setting*) et *check_room_property* (slots *action*, *room_property*). Le slot optionnel *location* peut être appliqué dans tous les cas. L'outil est basé sur les expressions régulières qui matche les mots (ex. *éteins*) quand ils commencent avec des chaînes spécifiées (ex. *étei*). Après cette pré-annotation, nous avons annoté manuellement tout ce qui n'a pas été traité et corrigé les erreurs d'annotation. Puisque la plupart des intentions du corpus sont parmi les intentions mentionnées ci-dessus, cet outil nous a économisé du temps.

Le résultat d'annotation est le même format RASA que nous avons utilisé pour notre corpus artificiel, avec quelques différences non-significatives. Ci-dessous est la phrase *diminue la lumière* annotée :


```
{"text" : "diminue la lumière",
"sync_trs" : "1232.753",
"entities" : [
    {
    "start" : 0,
    "end" : 7,
    "entity" : "action",
    "value" : "TURN_DOWN",
    "text" : "diminue",
    "color" : "#FFFF00"
    },
    {
    "start" : 8,
    "end" : 18,
    "entity" : "device",
    "value" : "light",
    "text" : "la lumière",
    "color" : "#00FF00"
    }
],
"intent" : "set_device"
}
```

Cette phrase est annotée avec l'intention et les slots (sous la liste *entities*). Chaque slot est représenté par un slot-label (le champ *slot*) et un slot-value (*slot value*), son texte correspondant (*text*) et l'index de début (*start*) et de fin (*end*) de ce slot dans la chaîne du texte. On indique aussi la couleur *color* qui est utilisé dans l'interface de l'annotation. Le champ *sync_trs* indique le temps de l'enregistrement est n'est pas en soi important. Les champs *color* et *sync_trs* n'ont pas été utilisés par les modèles donc ils n'ont pas apporté du bruit.

Le corpus a été normalisé, c'est-à-dire, traité automatiquement pour remplacer tous les mots clés énumérés dans la section 2.3.2 et utilisés pour s'adresser au système, par *KEYWORD*. Les fautes de reconnaissance de parole ont été prises en compte - par exemple, les mots-clés *vocadom* et *vocadum* ont été traités tous les deux. Les fautes d'orthographe ont été corrigées aussi.

Le corpus résultant contient 1646 phrases annotées, sans compter les phrases sans intention (tout ce que n'était pas une commande).

Le corpus SWEET-HOME a été annoté de la même manière que le corpus VocADom. Il contient 727 phrases annotées, sans compter les phrases sans intention. Le corpus SWEET-HOME est donc plus petit. D'autres différences de ces deux corpus, aussi bien que le corpus artificiel et PORTMEDIA seront décrites dans la section suivante.

4.3 Comparaison du corpus artificiel, VocADom, SWEET-HOME et PORTMEDIA

Cette section parle des différences entre les corpus en fonction de la taille en phrases, et la taille du vocabulaire, du nombre et de la fréquence des intentions, slot-labels et slot-values. Il est important que les intentions *none*, c'est-à-dire toutes les phrases qui n'étaient pas une commande, ont été supprimées des corpus VocADom et SWEET-HOME, parce que le corpus artificiel ne comporte pas de phrases sans intention.

Le tableau 4.1 représente les paramètres du corpus artificiel (deux versions), et des corpus réalistes VocADom et SWEET-HOME. La première version du corpus artificiel a été utilisée pour entraîner les modèles Tri-CRF, RASA et att-RNN, et la deuxième (avec une addition du SWEET-HOME) pour entraîner Tf-seq2seq.

TABLE 4.1 – Comparaison du corpus artificiel, VocADom, SWEET-HOME et PORTMEDIA.

Paramètres	Artificiel1	Artificiel2	VocADom	SWEET-HOME	PORTMEDIA
Taille, phrases	26082	42195	1646	727	18026
Taille du vocabulaire, mots	156	157	285	120	3062
Taille du vocabulaire, caractère	43	43	44	42	71
Nombre d'intentions	8	8	6	7	4
Nombre de slot-labels	16	16	11	7	32
Nombre de slot-values	60	60	44	24	378

On peut conclure que même si le corpus artificiel est beaucoup plus grand et plus riche en intentions, slot-labels et slot-values, il est moins riche en vocabulaire que le corpus réaliste VocADom. Il faut remarquer que SWEET-HOME est encore moins riche, mais il est beaucoup plus petit.

Une autre manière de comparer les deux corpus est de comparer des modèles de langages correspondants. Nous avons fait un modèle de langage 3-gram appris sur le corpus artificiel est

testé sur le corpus VocADom. Son perplexité est 134 (en ne tenant pas compte du marqueur $< s >$) ce qui est assez élevé pour un vocabulaire aussi petit. Le nombre de mots hors vocabulaire est important, avec 206 mots absents dans le corpus artificiel.

Le corpus PORTMEDIA est un corpus réaliste assez grand, avec un vocabulaire riche. Il ne contient que 4 intentions, en plus, 1509 phrases sur le total de 2006 sont annotées avec l'intention *none* (l'absence d'intention) ce qui crée une sur-représentation importante d'une intention par rapport aux autres. Par contre, le nombre de slot-labels et slot-values est beaucoup plus grand que dans les autres corpus.

PORTMEDIA n'est pas seule à avoir une représentation inégale des éléments (intentions, slot-labels et slot-labels) ; cette caractéristique est propre à tous les corpus utilisés dans ce travail. Par exemple, l'intention la plus fréquente du corpus artificiel est *set_device* avec 17375 occurrences, alors que l'intention *get_world_property* ne compte que 10 occurrences. Ce grand décalage est également présent parmi les slot-labels et slot-values. Les occurrences pour chaque intention, slot-label et slot-value pour tous les corpus sont données dans l'annexe B.

Vu que les corpus ont été préparés, nous avons pu passer à l'entraînement des modèles, décrit dans la section suivante.

4.4 L'évaluation du corpus artificiel

4.4.1 L'entraînement de RASA, Tri-CRF, att-RNN

Pour comprendre comment un système de compréhension automatique appris sur notre corpus artificiel va fonctionner dans les conditions réelles, dans un premier temps, 3 modèles états de l'art ont été utilisées : le champ aléatoire conditionnel triangulaire (Tri-CRF), le réseau de neurones avec attention (att-RNN) ; aussi bien que l'outil commercial open source, Rasa. Tri-CRF, att-RNN et RASA sont décrits dans la section 2.2.4.

D'abord il faut expliquer en quoi consiste la tâche de la compréhension automatique que nous envisageons et en quoi elle est différente de l'approche traditionnelle. Prenons comme exemple la phrase *allume la lumière*. Elle devrait être annotée avec l'intention *set_device*, le slot-label *action* avec son slot-value *turn_on*, et le slot-label *device* avec son slot-value *light*. Ainsi, la compréhension automatique dans notre projet comprends trois sous-taches, notamment la prédiction de l'intention (un seul élément), la prédiction de la séquence des slot-labels et la prédiction de la séquence des slot-values.

Cela est différent de l'approche traditionnelle qui sous-entend l'annotation de cette phrase avec l'intention (par exemple, *set_device*) et une séquence des slot-labels *turn_on* et *device*. Dans cette approche, les slot-values ne sont que les segments du texte associés aux slot-labels (*allume* et *lumière* respectivement). Tous les 3 modèles décrits ici sont conçus pour prédire l'intention et une séquence des slot-labels, ce qui n'est pas suffisant pour nous. C'est pourquoi nous devons entraîner un modèle séparé - que ce soit Tri-CRF, RASA ou att-RNN - pour la taĉhe de prédiction des slot-values.

Maintenant il faut décrire les différences des modèles en leur façon de prédire et la façon d'associer les slot-labels/values à la séquence d'entrée.

Le modèle RASA peut prédire une séquence d'éléments dont chacun est associé à un segment de la séquence d'entrée. Cela veut dire que RASA contient trois modèles séparés pour prédire intention, slot-labels et slot-values pour chaque phrase. Les slot-labels/values sont associés aux morceaux du texte, alors que l'intention est associée à l'ensemble de la phrase.

Le modèle Tri-CRF peut prédire simultanément une intention associée à l'ensemble de la séquence d'entrée et une séquence des éléments dont chacun est associé à un segment de la séquence de l'entrée. Cela veut dire qu'il peut prédire une intention et une séquence des slots-labels, ou une intention et une séquence des slot-values. Ainsi, il faut entraîner 2 modèles Tri-CRF séparés : un modèle pour prédire intention et slot-labels, l'autre pour prédire les slot-values.

Le modèle att-RNN peut prédire simultanément une intention associée à l'ensemble de la phrase et une séquence d'éléments de la même longueur que la séquence d'entrée ; c'est-à-dire, chaque mot est associé à un élément. Comme dans le cas avec Tri-CRF, il faut entraîner 2 modèles Tri-CRF séparés : un modèle pour prédire l'intention et les slot-labels, l'autre pour prédire les slot-values. La différence est que dans att-RNN, les slot-labels/values sont associés à chaque mot (il existe un slot-label/value zero pour exprimer que le mot n'est pas attribué de slot-label/value), alors que dans Tri-CRF, les slot-labels/values sont associés aux morceaux du texte (comme dans RASA).

Pour tous les trois modèles, la prédiction de l'intention est une tâche de classification, et la prédiction des slot-labels/values est une prédiction de séquence.

Il faut aussi remarquer que tandis que Rasa utilise les word-embeddings pré-appris, Tri-CRF et att-RNN les apprennent pendant l'apprentissage.

La démarche était suivante. D'abord, nous avons entraîné les 3 modèles - Tri-CRF, att-RNN et Rasa - sur le corpus artificiel (version 1) et puis nous les avons testés sur le corpus réaliste VocADom. Après l'extraction des métriques, pour pouvoir conclure que les limites de performance sont dues aux limitations du corpus artificiel utilisé en entraînement plutôt qu'aux limitations des modèles eux-mêmes, nous avons appris 3 modèles - également Tri-CRF, att-RNN et Rasa - sur le corpus PORTMEDIA et nous les avons testés sur le même corpus. Nous avons donc pu comparer les métriques de ces 2 groupes de modèles. L'évaluation des modèles a été faite sur 3 tâches : la classification de l'intention, des slot-labels et slot-values.

La table 4.2a montre les résultats pour les modèles entraînés et testés sur PORTMEDIA, divisé en corpus apprentissage (90%) et test (10%). Les meilleures résultats pour chaque tâche - prédiction des intentions, slot-labels et slot-values - sont en gras. Les 3 méthodes montrent une performance similaire (la différence est de 5% de f-score au maximum). Att-RNN a montré les meilleurs performances pour les 3 tâches ce qui n'est pas surprenant car les réseaux neuronaux généralisent mieux et le corpus est suffisamment grand. Ces résultats sur PORTMEDIA montrent le niveau de performance qu'on peut attendre quand les modèles de compréhension automatique sont entraîné sur un corpus qui est idéalement proche du corpus de test.

La table 4.2b montre les résultats pour les modèles entraînés sur les données artificielles qui ont été réparties aléatoirement entre apprentissage (90%) et développement (10%), et puis testé sur le corpus réaliste VocADom. Il faut remarquer que les performances sur les données de développement (artificiels) étaient presque parfaites à cause de la grande homogénéité de ce corpus ; ce qui n'est pas le cas avec le corpus naturel de test. Encore une fois, c'est att-RNN qui a montré la meilleure performance sur les intentions, mais sur les slot-labels et slot-values, c'est RASA qui a été plus performant. Cela peut être dû au fait que contrairement à Tri-CRF et à att-RNN, Rasa utilise des word-embeddings pré-appris sur des données externes ce qui lui permet de prendre en compte les mots hors vocabulaire (c'est-à-dire, les mots qui sont présents dans le corpus de test et qui ne sont pas présents dans le corpus d'entraînement).

On peut conclure que les modèles entraînés sur le corpus artificiel et testés sur le corpus VocADom ont montré des performances moins bonnes ce qui n'est pas surprenant. Après avoir évalué RASA, Tri-CRF et att-RNN, nous avons passé à l'entraînement et test de Tf-seq2seq, décrit dans la section suivante.

TABLE 4.2 – Résultats de l'apprentissage de att-RNN, Tri-CRF et RASA sur les jeux de données PORTMEDIA et VocADom pour les intentions, slots-labels et slot-values

Model	Précision	Rappel	F1
Att-RNN-Intention	**97.56**	**97.56**	**97.56**
Tri-CRF-Intention	96.42	96.43	96.36
Rasa-Intention	92.20	92.52	92.26
Att-RNN-Labels	**95.96**	**96.36**	**96.11**
Tri-CRF-Labels	95.31	95.74	95.39
Rasa-Labels	95.17	94.22	94.16
Att-RNN-Values	**94.85**	**95.73**	**95.08**
Tri-CRF-Values	92.01	93.49	92.32
Rasa-Values	93.94	93.73	93.34

(a) Performances des modèles entraînés et testés sur PORTMEDIA

Model	Précision	Rappel	F1
Att-RNN-Intention	**93.77**	**90.28**	**91.30**
Tri-CRF-Intention	84.11	79.47	76.36
Rasa-Intention	90.48	71.39	76.57
Att-RNN-Labels	69.19	66.24	66.09
Tri-CRF-Labels	77.28	52.65	60.64
Rasa-Labels	**85.72**	**73.54**	**79.03**
Att-RNN-Values	43.02	30.51	35.00
Tri-CRF-Values	51.33	25.52	33.51
Rasa-Values	**68.56**	**56.73**	**61.95**

(b) Performances des modèles entraînés sur le corpus artificiel et testés sur Vocadom

4.4.2 Entraînement du Tf-seq2seq

Dans cette section, il s'agit de Tf-seq2seq qui a été un autre modèle que nous avons utilisé pour évaluer notre corpus artificiel (version 2). Tf-seq2seq (Britz *et al.*, 2017) est décrit dans la section 2.2.4. Contrairement à Tri-CRF, RASA et att-RNN, il a comme avantage le fait qu'il ne demande pas de données annotées et qu'il ne requiert qu'un seul modèle pour les intentions, slot-labels et slot-values. En plus, il n'a pas de séparation entre les intentions et les slots. Il serait donc intéressant de savoir si il peut néanmoins se montrer compétitif.

D'abord, nous allons décrire la conversion des corpus en format traitable par Tf-seq2seq, puis leur répartition en Train, Dev et Test. Ensuite nous allons spécifier les paramètres de configuration du modèle. Finalement nous allons apprendre deux modèles Tf-seq2seq en mode mot : un modèle sera appris et testé sur le corpus PORTMEDIA, l'autre sera appris sur le corpus artificiel (version 2) avec une addition de SWEET-HOME et testé sur le corpus réaliste VocADom. Nous avons décidé d'ajouter le corpus SWEET-HOME au corpus artificiel car ce dernier est trop pauvre en lexique. (Nous n'avons pas ajouté SWEET-HOME dans l'entraînement de Tri-CRF, att-RNN, RASA car ce corpus n'était pas annoté pour ce moment). En tous cas, SWEET-HOME ne contient que 727 phrases ce qui est beaucoup plus petit que le corpus artificiel contenant 42195 phrases.

Finalement on va convertir la prédiction et la cible du test en format traitable par l'outil d'évaluation (le même outil que nous avons utilisé avec les autres modèles) et évaluer la performance. Ces étapes seront détaillées ci-dessous.

Modification et pré-traitement des corpus

Comme Tf-seq2seq est basé sur RNN du type séquence-en-séquence, nous avons dû convertir le corpus artificiel, SWEET-HOME, VocADom et PORTMEDIA en format convenable. Au départ, les deux corpus ont été en format RASA. Nous avons choisi le format utilisé dans le corpus E2E (Novikova *et al.*, 2017) comme le format de départ. Le format résultant est :

Source :
KEYWORD éteins la bouilloire
Cible :

intent[set_device], action[turn_off], device[kettle]

Nous avons décidé de séparer chaque élément, y compris les crochets et virgules par espace pour apprendre les slot-labels et slot-values séparément. Nos corpus ont été donc convertis en 2 fichiers : de source et de cible :

Source :
KEYWORD éteins la bouilloire

Cible :
intent [set_device] , action [turn_off] , device [kettle]

Nous avons également remplacé les mots clés dans Vocadom par *KEYWORD*, corrigé les erreurs orthographiques, et mis la source et la cible en minuscules (sauf *KEYWORD*). Comme on peut voir dans cet exemple, les intentions ne sont pas différenciées des slots. Ce corpus a été divisé en données d'entraînement (90%) et du développement (10%).

Il faut remarquer que dans nos corpus, il n'y a pas d'ordre défini des slots, donc on peut bien rencontrer les deux représentations ci-dessus :

intent [set_device] , action [turn_off] , device [kettle]
device [kettle] , intent [set_device] , action [turn_off]

Pour tous nos corpus - artificiel (version2), SWEET-HOME, VocADom et PORTMEDIA - nous avons décidé de mettre tous les slots dans la cible dans le même ordre : *intent, action, device* etc. Nous avons mis l'intention en premier car si le décodeur prédit d'abord l'intention (ce qui est une tâche assez facile car le nombre des intentions est restreint), il est peut-être plus facile pour le décodeur de prédire le premier slot. De la même façon, la prédiction du second slot est facilité par le fait que le premier slot est prédit, etc. Côté intuitif, chaque type d'intention est implicitement associé à un certain ensemble des slots, comme par exemple *set_device* qui va toujours avec les slots *action* et *device*, donc le fait de placer l'intention en premier peut aider à la prédiction des slots.

Le méchanisme d'attention peut aussi profiter d'ordonnancement car le volume d'attention dépend, entre autres, de la localisation du slot donné dans la séquence. Ainsi, si un slot se trouve dans la même position la plupart du temps, la tâche du méchanisme d'attention sera facilitée.

Nous avons fait le vocabulaire des mots avec l'outil fourni par Tf-seq2seq, avec l'espace comme séparateur. Ci-dessous est l'exemple du vocabulaire des mots pour la source :

la 33962
de 26859
KEYWORD 23316

Ci-dessus est la cible :

] 187149
[187149
, 153029
intent 34120

Répartition des données

Les données d'entraînement comportent 80% du corpus artificiel (version 2) et 50% du corpus SWEET-HOME (le pourcentage du corpus composant, pas des données résultantes), ce qui fait 33756 lignes du corpus artificiel et 364 du corpus SWEET-HOME. Au total cela fait 34120 lignes.

Les données de développement ont été utilisées seulement pour pouvoir suivre le score bleu lors de l'apprentissage. Ces données consistent de 20% du corpus artificiel (8439 lignes) et 50% du corpus SWEET-HOME (363 lignes) ce qui fait 8802 lignes au total.

Le corpus de test a été VocADom (1646 lignes).

Comparaison du corpus de l'entraînement (Artificiel + SWEET-HOME) avec le corpus de test (VocADom)

Pour pouvoir mieux interpréter les résultats plus tard, nous avons extrait quelques statistiques sur le corpus résultant de l'entraînement (Train) et le corpus de test (Test). La table 4.3 montre la taille du vocabulaire de chacun. Le vocabulaire des mots du Test est plus riche, comme on pouvait s'y attendre.

TABLE 4.3 – Comparaison du corpus d'entraînement (artificiel2 + SWEETHOME) et du corpus de test (VocADom)

Paramètres	Train	Test
Taille du vocabulaire, mots	196	285
Taille du vocabulaire, caractère	44	44

Ci-dessous sont les mots parmi les mots les plus fréquents qui sont présents dans Test mais absents dans Train, avec leurs fréquences correspondantes.

haut 183
bas 179
hé 124
d'en 97
premier 80
toilettes 75
fermés 50
et 43
é 20
chauffage 17
éteint 16
l'étage 16
heu 13
éteinte 10
descend 9
est-elle 9
toutes 8

éh 8

On peut supposer que les phrases du type *allume la lumière en haut, allume la lumière d'en haut* et *allume la lumière au premier étage* ne seront pas traitées correctement lors de l'évaluation sur Test car les mots *haut, d'en, premier* sont parmi les mots hors vocabulaire. Même la phrase *allume la lumière de l'étage* qui est bien présent dans le Train, peut provoquer des problèmes car dans le Train, il y a un espace entre l'article défini et le mot *étage* ; c'est pourquoi le mot *l'étage* est parmi les mots hors vocabulaire. Les mots courants tels que *toilettes, chauffage* et d'autres ne seront pas bien traités non plus. En plus, les interjections *hé, é, éh* vont aussi peut-être perturber la compréhension.

En ce qui concerne la cible, le tableau dans l'annexe D montre la distribution des slot-values par chaque slot-label dans les deux corpus. La distribution est représenté comme une proportion : le nombre d'occurrence d'un slot-value donné dans Train divisé par son nombre d'occurrence dans Test. Il est évident que les distributions sont très différents. Le déséquilibre est particulièrement important au niveau des intentions. Par exemple, l'intention *check_device* n'apparaît que 2 fois dans Train mais 64 fois dans Test. Par contre, *set_device* est sur-représenté dans Train : 17556 occurrences contre 926 dans Test. On peut supposer que le système sera biaisé vers les éléments sur-représentés donc la performance sera faible sur tous les éléments peu représentés dans le corpus d'entraînement.

Paramètres des modèles

Le modèle utilisé est un GRU RNN bidirectionnel du type encodeur-decodeur avec attention. Le nombre d'unités d'embeddings est 128, le nombre d'unités de l'encodeur est 128, même nombre pour le décodeur. Algorithme d'optimisation (gradient) est Adam, le taux d'apprentissage est 0.0001. La taille de batch est 32. Tous ces paramètres sont des paramètres par defaut de Tf-seq2seq.

Le nombre de pas (steps) d'apprentissage a été fixé à 150 000. Étant donné que la taille de batch est 32, cela veut dire que à chaque pas d'apprentissage, on utilise 32 exemples (lignes du corpus dans notre cas) à la fois. Notre corpus de Train a 34 120 exemples donc une itération sur le corpus (epoch en anglais) comprend 34 120 / 32 = 1066 pas environ. Si on fixe le nombre de pas à 150 000, cela veut dire qu'on itère sur le corpus 150 000 / 1 066 = 140 fois environ.

En ce qui concerne la taille de la séquence de l'entrée/sortie, elle est différente pour les corpus différents. Nous avons fait les statistiques sur la taille moyenne (en caractères) des lignes de la source et de la cible dans la partie Test de PORTMEDIA et dans VocADom. Nous avons pris la quartile et et nous l'avons arrondie à un nombre divisible par 50. Ainsi, pour le modèles entraîné sur PORTMEDIA, la taille de séquence d'entrée est 150 et la séquence de sortie est 50. Pour les modèles entraînés sur le corpus artificiel, les chiffres sont 50 et 100.

Il est nécessaire de rappeler que Tf-seq2seq a été lancé en mode mots.

Pré-traitement des prédictions et des cibles et l'évaluation

Pour pouvoir évaluer les résultats, nous avons utilisé l'outil existant (Raimondo, 2017) ; qui prend les données en format binaire, leur forme string étant :

'words [str], 'true-label' : str_or_int, 'predicted-label-conf' : ['name' : str_or_int, 'confidence' :float],

Dans notre cas, il n'y a pas d'alignement des slots avec les mots, ni de valeur de confidence ; donc la sortie de notre script est du type :

'words' : [], 'true-label' : "device", 'predicted-label-conf' : ['name' : "device", 'confidence' :1.0]

Ce format sous-entend l'alignement de chaque slot prédit avec son slot référence correspondant, alors que la prédiction de seq2seq et la référence étaient de format où chaque séquence des slots correspond à une phrase :

Prédiction : intent [set_device] , action [turn_on] , device [light] , location-room [kitchen]
Référence : intent [check_device] , action [turn_on] , device [light]

Notre outil aligne les éléments un-par-un, c'est-à-dire, chaque slot-label (exemple : *device*) de la prédiction est alignée avec son slot-label cible correspondant, chaque slot-value (exemple : *light*) de la prédiction est alignée avec sa slot-value cible, et l'intention (exemple : *set_device*) de la prédiction est alignée avec son intent cible (exemple : *check_device*). La condition pour l'alignement est que les slots-labels coïncident.

On peut voir que le slot *location-room[kitchen]* est absent dans la cible ; dans ce cas, on l'aligne avec une cible qui est une chaîne vide. Cela est aussi vrai inversement : si un slot est présent dans la référence mais absent dans la prédiction, on aligne la référence avec une chaîne vide.

De telle façon, nous faisons 3 fichiers séparés pour les intentions, slot-labels et slot-values. Ces fichiers peuvent être ensuite traités par l'outil qui extrait les métriques.

Résultats

Il faut remarquer que le modèle appris sur PORTMEDIA a été évalué à 150 000 pas, alors que le modèle appris sur le corpus artificiel a été évalué à 50 000 pas car elle a atteint son maximum de performance.

La table 4.4a montre les résultats pour Tf-seq2seq entraîné et testé sur PORTMEDIA, et la table 4.4b montre les résultats pour Tf-seq2seq entraîné et testé sur le corpus artificiel avec l'addition du corpus SWEET-HOME.

TABLE 4.4 – Résultats de l'apprentissage du Tf-seq2seq en mode mots sur PORTMEDIA et le corpus artificiel avec l'addition du corpus SWEET-HOME pour les intentions, les slots-labels et les slot-values

Model	Précision	Rappel	F1
Intention	97.08	97.11	97.08
Labels	63.55	65.36	64.21
Values	58.51	58.95	58.06

(a) Performances du Tf-seq2seq entraîné et testé sur PORTMEDIA en mode mots

Model	Précision	Rappel	F1
Intention	95.37	94.59	94.74
Labels	48.95	55.27	51.06
Values	37.75	35.52	34.95

(b) Performances du Tf-seq2seq entraîné sur le corpus artificiel (avec une addition du corpus SWEET-HOME) et testé sur Vocadom en mode mots

Les résultats beaucoup plus détaillés, avec et sans pondération par le nombre d'instances, pour chaque intention, slot-label et slot-value sont présentés dans l'annexe C.

Ces résultats, aussi bien que les résultats de Tri-CRF, RASA et Att-RNN seront discutés dans la section 7. Maintenant nous pouvons passer à la seconde étape de notre travail qui comprend la comparaison des performances Tf-seq2seq pour les modes et tâches différents.

Chapitre 5

L'étude de Tf-seq2seq : différents modes et tâches

Dans cette section, il s'agit de l'étude des performances de Tf-seq2seq, modèle séquence-vers-séquence. En accord avec les objectifs décrits dans la problématique, notre travail consistera en l'étude des aspects suivants :

1. L'influence du mode - mot vs caractère - sur la performance de Tf-seq2seq ;
2. L'influence de la tâche - prédiction des intentions et des slots vs la prédiction des slots seulement - sur la performance de Tf-seq2seq.

5.1 Mots vs caractères

Dans cette sous-section, il s'agit de l'entraînement de Tf-seq2seq sur le corpus artificiel avec SWEET-HOME en en mode caractère. Puis nous allons comparer ces résultats avec ceux du modèle appris à l'étape précédente, sur ce même corpus en mode mot, pour savoir quel mode est mieux pour le corpus en question.

D'abord nous avons pré-traité les données pour pouvoir y lancer Tf-seq2seq en mode caractère. Le corpus de départ a été dans le même format que dans l'expériment précedant (4.4.2). Dans la cible, nous avons supprimé tous les espaces car avec l'approche caractère-par-caractère, le problème de séparation des slot-labels et slot-values ne se pose pas. En plus, la suppression des espaces inutiles accéléré l'apprentissage.

Puis nous avons mis les espaces séparateurs qui forcent Tf-seq2seq a considérer les caractères comme des mots séparés. Ceci est donc le format de la cible en mode caractère :

Cible :
i n t e n t [s e t _ d e v i c e] , a c t i o n [t u r n _ o f f] , d e v i c e [k e t t l e]

Cela est le format de la source en mode caractère :

Source :
K E Y W O R D é t e i n s l a b o u i l l o i r e

Le modèle à caractères utilise les mêmes paramètres que les modèles précédents, sauf le nombre de pas d'apprentissage fixé à 50 000. En ce qui concerne la taille de la séquence de l'entrée/sortie, nous avons fait les mêmes calculs que avec deux modèles précédents pour fixer la taille de la séquence d'entrée/sortie à 100 et 150 respectivement.

La table 5.1a montre les résultats pour Tf-seq2seq entraîné sur l'artificiel avec l'addition de SWEET-HOME en mode mot et testé sur VocADom (le même modèle que sur 4.4b), et la table 5.1b montre les résultats pour Tf-seq2seq sur ce même jeu de données en mode caractère.

TABLE 5.1 – Résultats de l'apprentissage du Tf-seq2seq entraîné sur le corpus artificiel avec l'addition de SWEET-HOME et testé sur VocADom en mode mot et caractère

Model	Précision	Rappel	F1
Intention	95.37	94.59	94.74
Labels	48.95	55.27	51.06
Values	37.75	35.52	34.95

(a) Performances du Tf-seq2seq en mode mot

Model	Précision	Rappel	F1
Intention	72.88	72.78	72.31
Labels	74.18	68.80	67.33
Values	51.23	38.81	41.00

(b) Performances du Tf-seq2seq en mode caractère

Ces résultats seront discutés plus tard dans la section 7.

5.2 Prédiction des intentions et slots vs slots seulement

Dans cette sous-section, il s'agit de l'entraînement et le test de Tf-seq2seq sur PORTMEDIA sans intention pour savoir si l'absence de la séparation explicite des intentions et slots nuit ou améliore la performance de Tf-seq2seq.

Nous allons entraîner deux modèles - à mots et à caractères. Le nombre de pas d'apprentissage fixé à 150 000 pour le modèle à mots et 50 000 pour le modèle à caractères. La taille de la séquence de l'entrée/sortie est 150 et 50 respectivement pour les mots, 300 et 100 respectivement pour les caractères.

La table 5.2a montre les résultats pour Tf-seq2seq entraîné et testé sur PORTMEDIA sans intention en mode mot, et la table 5.2b montre les résultats pour Tf-seq2seq sur ce même jeu de données en mode caractère.

TABLE 5.2 – Résultats de l'apprentissage du Tf-seq2seq entraîné et testé sur PORTMEDIA sans intention en mode mot et caractère

Model	Précision	Rappel	F1
Labels	66.46	66.54	66.09
Values	55.36	55.20	54.40

(a) Performances du Tf-seq2seq en mode mot sur PORTMEDIA sans intention

Model	Précision	Rappel	F1
Labels	66.01	62.28	63.42
Values	55.38	52.73	53.22

(b) Performances du Tf-seq2seq en mode caractère sur PORTMEDIA sans intention

Ces résultats seront discutés plus tard dans la chapitre 7.

Chapitre 6

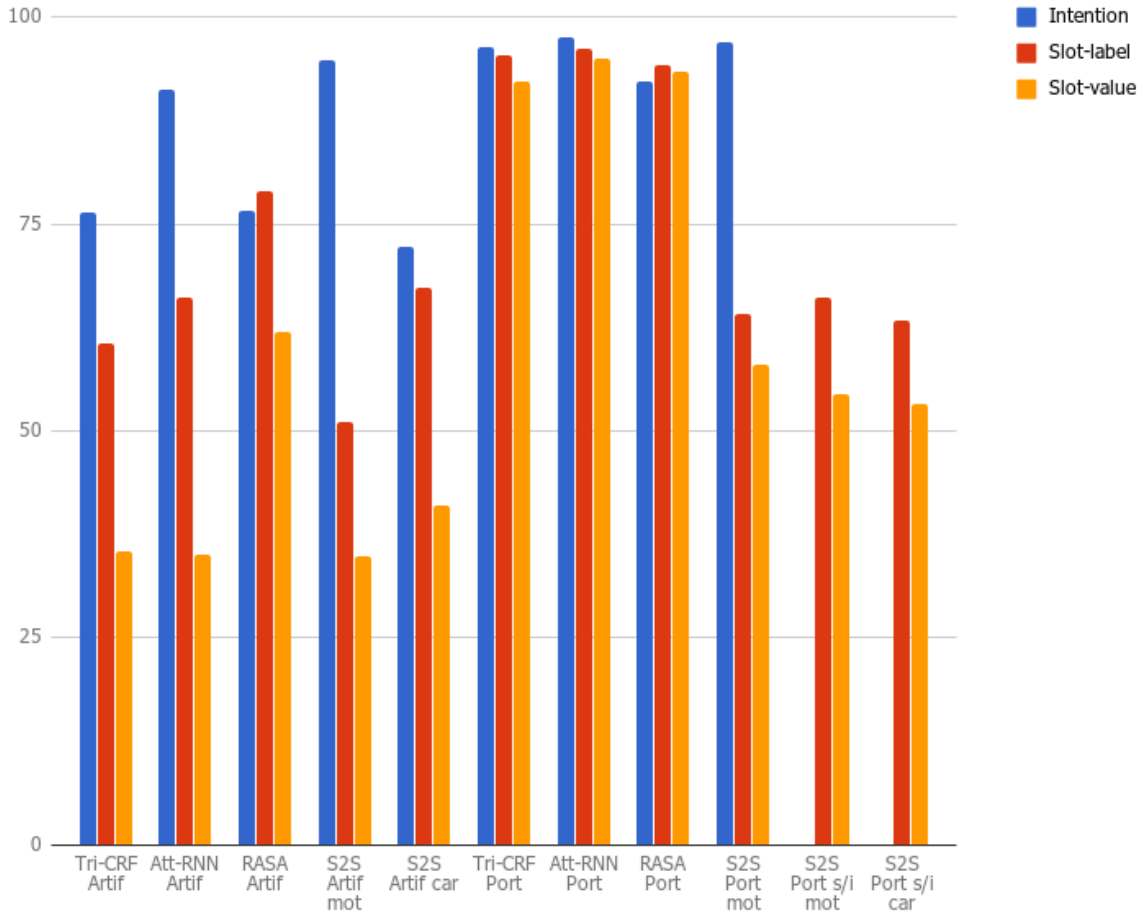
Récapitulatif des résultats de tous les modèles

Pour rendre les résultats de tous les modèles utilisés dans ce travail en forme plus visuel, nous les avons représentés dans la table 6.1 aussi bien que sur le diagramme 6.1.

TABLE 6.1 – La performance de tous les modèles utilisés dans ce travail.

Corpus train	Corpus test	Modèle	Intention	Slot-label	Slot-value
Artificiel, version 1	VocADom	Tri-CRF	76.36	60.64	35.51
Artificiel, version 1	VocADom	Att-RNN	91.30	66.09	35.00
Artificiel, version 1	VocADom	RASA	76.57	79.03	61.95
Artificiel, version 2 + SWEET-HOME	VocADom	Tf-seq2seq, mot	94.74	51.06	34.95
Artificiel, version 2 + SWEET-HOME	VocADom	Tf-seq2seq, car	72.31	67.33	41.00
PORTMEDIA	PORTMEDIA	Tri-CRF	96.36	95.39	92.32
PORTMEDIA	PORTMEDIA	Att-RNN	97.56	96.11	95.08
PORTMEDIA	PORTMEDIA	RASA	92.26	94.16	93.34
PORTMEDIA	PORTMEDIA	Tf-seq2seq, mot	97.08	64.21	58.06
PORTMEDIA sans intention	PORTMEDIA sans intention	Tf-seq2seq, mot		66.09	54.40
PORTMEDIA sans intention	PORTMEDIA sans intention	Tf-seq2seq, car		63.42	53.22

FIGURE 6.1 – La performance de tous les modèles utilisés dans ce travail. Tf-seq2seq est marqué comme « S2S ». Les modèles entraînés sur le corpus artificiel (avec ou sans SWEET-HOME) et testés sur VocADom sont marqués comme « Artif », les modèles entraînés et testés sur PORT-MEDIA sont marqués comme « Port ». Le mode de Tf-seq2seq est marqué comme « mot » ou « car » pour les caractères. Les deux derniers modèles sont entraînés et testés sur le corpus PORTMEDIA sans intention.



Chapitre 7

Discussion

D'abord il est nécessaire de rappeler que les deux corpus utilisés dans ce mémoire - un grand corpus de données conversationnelles réelles PORTMEDIA et le corpus artificiel - ont des différences importants. Premièrement, PORTMEDIA contient beaucoup d'intentions *none* (qui veut dire l'absence de l'intention) ce qui a amélioré les performances (la précision) des modèles sur les intentions. Le corpus artificiel ne contenait pas des intentions *none*, et les corpus VocADom et SWEET-HOME ont été traités pour supprimer les phrases dont l'intention est *none*. Deuxièmement, PORTMEDIA contenait seulement 4 types d'intentions, alors que le corpus artificiel en contenait 7 ; donc la tâche de prédiction est plus difficile dans le cas du corpus artificiel.

Tri-CRF, RASA, Att-RNN et Tf-seq2seq avaient tous les quatre rendu de bon résultats sur PORTMEDIA, mais ils ont été bien moins performants appris sur le corpus artificiel et testés sur le corpus réaliste VocADom. Ceci montre la difficulté à prendre en charge la diversité d'un corpus réaliste par rapport à celle d'un corpus artificiel. Une raison possible est que le corpus réaliste a été enregistré avec des participants naïfs et qu'il contient des variations significatives de vocabulaire et de syntaxe par rapport au corpus artificiel : les répétitions, les disfluences et les interjections (ex. *euh*) qui conduisent à des énoncés syntaxiquement différents de ceux du corpus artificiel. En particulier, les mots clés apparaissent à des positions différentes (ex. *Allume la lumière KEYWORD* au lieu de *KEYWORD allume la lumière*). En plus, le vocabulaire du corpus réaliste est beaucoup plus élevé : 336 mots contre 155 mots du corpus artificiel.

Pour tous les modèles - appris sur PORTMEDIA ou sur le corpus artificiel - les résultats sur la prédiction des intentions étaient satisfaisants, mais cela est partiellement dû au fait que le nombre des intentions possibles est assez bas.

Par contre, les résultats sur le remplissage des slots (slot-labels, slot-values) étaient beaucoup plus bas. La raison pour cela peut être le fait que le nombre des slot-labels et slot-values est beaucoup plus important que celui des intentions. En plus, certains mots avec leurs slot-labels et slot-values correspondants ont été sous-représentés dans le corpus d'apprentissage artificiel (comme par exemple, le mot *témperature* avec le slot-label correspondant *room-property* et sa valeur *temperature*). Dans ce cas, les modèles ont montré des performances proches ou égale à zero car ils ont tendance à prédire des slot-labels et slot-values plus courants. Le problème de sous-représentation des éléments est évidente si nous regardons les résultats pondérés (par le nombre de chaque élément) car ils sont beaucoup plus bas que les résultats non-pondérés. Cela veut dire que les résultats non-pondérés sont bons partiellement grâce au fait que certains éléments sont simultanément sur-représentés dans le corpus de l'entraînement (ce qui améliore la performance du système sur ces éléments) et de test (ce qui améliore le score moyen non-pondéré pour tous les éléments).

Parmi tous les modèles, RASA a montré la meilleure performance sur la tâche difficile de la prédiction des slot-labels et slot-values, ce qui peut être expliqué par le fait que ses word-embeddings ont été pré-appris sur un grand jeu de données externe (Wikipedia, Open Subtitles et d'autres), donc la sous-représentation de certains éléments dans le corpus artificiel avait moins d'influence négative que pour les autres modèles.

En ce qui concerne Tf-seq2seq, il faut tout d'abord remarquer qu'il est entraîné sur la version du corpus artificiel qui est deux fois plus grande que la version utilisée pour les trois modèles précédentes ; en plus, nous avons ajouté SWEET-HOME pour assurer plus de variabilité lexicale. En tenant compte de ces différences, il est néanmoins intéressant de comparer les performances de Tf-se2seq avec les modèles états de l'art. Tf-seq2seq a montré une performance équivalente à celles de RASA, att-RNN et Tri-CRF, malgré le fait que ces 3 modèles utilisent l'alignement des slots avec le segments du texte de la phrase d'entrée, alors que Tf-seq2seq ne l'utilise pas , et le fait que Tf-seq2seq n'utilise qu'un seul modèle pour toutes les tâches. En fait, Tf-seq2seq a montré les meilleures résultats parmi tous les modèles étudiés sur la tâche de la prédiction de l'intention.

Quant aux différents modes de Tf-seq2seq, nous avons trouvé que la performance du modèle à mots est mieux sur les intentions mais elle est pis sur les slot-labels et slot-values en comparaison avec le modèle à caractères. Cela peut être dû au fait que le modèle à caractères est capable de gérer plusieurs formes différents des mots, pas seulement ceux qui sont dans son corpus d'entraînement. Donc il est plus utile dans le cas du vocabulaire important ; et la prédiction des slot-labels et slot-values demande un vocabulaire important. Il est nécessaire de remarquer que le modèle à caractères apprend l'orthographe de chaque mot ; mais qu'il n'y avait pas beaucoup de cas de intentions/slots mal ortographiés.

On peut comparer nos résultats à l'étude (Agarwal et Dymetman, 2017) qui utilise Tf-seq2seq en mode caractères sur le corpus E2E, pout la tâche de génération, pas de compéhension automatique. Côté positif, ils ont trouvé que le modèle n'a jamais « halluciné », c-est-à-dire, n'a pas produit ce que ne devrait pas être produit, et a produit très peu de répétitions. Côté négatif, il y avait parfois des omissions. Dans notre cas, le modèle n'a pas halluciné mais il a produit beaucoup de substitutions des slot-values, par exemple, *action[lower]* au lieu de *action[turn_on]*.

Finalement, en ce qui concerne la comparaison des performances de Tf-seq2seq sur la tâche de prédiction des intentions avec slots et la prédiction des slots uniquement, il s'avère que la présence de l'intention n'a pas beaucoup d'influence sur la prédiction des slots.

Chapitre 8

Perspectives de la recherche

Ces expérimentations montrent les limites de l'approche basée sur la génération d'un corpus artificiel lorsqu'il faut prendre en compte la variation de vocabulaire et de syntaxe. Suite à cela, il faudrait ajouter plus de vocabulaire et constructions syntaxiques, en prenant des corpus réalistes comme exemple.

Un autre problème important du corpus artificiel est la sur-représentation des quelques intentions/slots et la sous-représentation d'autres. Suite à cela, les modèles sont biaisés vers les éléments sur-représentés. Ainsi ils montrent de bons résultats sur l'intention *set_device* et le slot *device* car ils sont parmi les éléments les plus nombreux dans le corpus ; alors que les autres intentions/slots ont un score beaucoup plus bas. On pourrait résoudre ce problème en s'assurant que chaque intention/slot est représenté de façon plus ou moins égale. Une façon de le faire et de générer un corpus où chaque intention/slot a plus ou moins la même fréquence. Mais il y a une façon de le faire avec le corpus existant. On peut prendre ce corpus et en extraire un sous-corpus à représentation égale - supposons que chaque intention/slot ne devrait pas apparaître plus de 1000 fois. Nous aurions dû supprimer les phrases au-délà de cette limite, ce qui va nous donner un corpus beaucoup plus petit. Après un nombre donné de steps de l'apprentissage, nous pourrions substituer une partie de ce corpus par des phrases supprimées à la dernière étape, toujours en assurant l'équivalence de représentation ; et puis apprendre ce même modèle sur ce corpus. On répète cette substitution autant de fois que c'est nécessaire. Étant donné que dans l'ensemble du corpus de départ, il y a des intentions/slots plus fréquents que les autres, les sous-corpus d'apprentissage partagerons les mêmes phrases avec les intentions/slots peu fréquents (car on n'a rien avec quoi les substituer) mais ils aurons des phrases plus ou moins variées avec les intentions/slots fréquents (car on a un grand choix des telles phrases). De telle façon, même si la variation lexicale de la source annotée avec des intentions/slots peu fréquents reste pauvre, le modèle appris ne serait pas biaisé envers les intentions/slots fréquents car chaque sous-corpus observe la représentation égale.

Une solution possible pour le problème du manque de variabilité lexicale et syntaxique est proposée dans (Manishina *et al.*, 2016), notamment, l'integration des synonymes et paraphrases obtenus automatiquement des sources externes.

De plus, les algorithmes d'apprentissage devront aussi être consolidés afin d'induire des modèles plus robustes en présence de mots inconnus. Une des solutions les plus prometteuses pourrait être d'insérer des word-embeddings pré-appris sur un jeu de données externe assez grand (Wikipedia, Open Subtitles etc.) comme c'était dans le cas de RASA qui a montré une bonne performance sur des slot-labels et slot-values.

Un autre moyen d'augmenter le vocabulaire des modèles de compréhension de la langue dans l'habitat intelligent est d'apprendre le modèle d'abord sur un grand corpus réaliste exis-

tant, même si il provient d'un autre domaine, et puis apprendre ce modèle sur le corpus artificiel du domaine. Dans notre cas, nous pourrions apprendre chaque modèle d'abord sur PORTMEDIA et puis sur le corpus artificiel. De telle façon, les modèles pourraient apprendre les word-embeddings du nombre de mots plus important.

Les études (Zhang et Wang, 2016) et (Liu et Lane, 2017) ont trouvé que les algorithmes qui prédisent l'intention et slots simultanément fonctionnent mieux car les deux tâches sont fortement corrélées. Dans ce travail, seuls att-RNN et Tri-CRF le font simultanément. Peut-être RASA pourrait améliorer sa performance si on y ajouterait une possibilité d'interaction entre les états cachés représentant l'intention et les états cachés représentant les slots (slot-labels et slot-values). En ce qui concerne Tf-seq2seq, il ne produit qu'une seule séquence où les intentions et slots ne sont pas différenciés, donc il ne peut pas intégrer la notion des intentions, slot-labels et slot-values de la même façon que les 3 modèles décrits.

Une autre perspective de recherche concerne la méthode des expérimentations décrits dans ce travail qui ont ses limites. Nous avons déjà remarqué que notre comparaison des performances de trois modèles états de l'art avec Tf-seq2seq est faite en tenant compte du fait que Tf-seq2seq a été entraîné sur un corpus artificiel plus grand que celui utilisé par les trois modèles ; et il a été enrichi avec le corpus réaliste SWEET-HOME. L'expérimentation devrait être plutôt faite sur le même corpus pour pouvoir tirer les conclusions.

Il paraît également important de comparer les résultats de Tf-seq2seq sur le même corpus en formats différents. Dans notre cas, nous avons séparé les slot-labels des slot-values par espaces (les intentions étant traitées comme des slots ordinaires) pour que le modèle les apprend séparément. Pourtant, cela a augmenté le vocabulaire des mots. Par exemple, la séquence *intent [set_device]* a le vocabulaire *intent, [, set_device,]*. Sans séparation, la séquence aurait la forme *intent[set_device]* et son vocabulaire serait *intent[set_device]*. Il est probable que la séparation des slot-labels et slot-values peut être utile pour les corpus où les mêmes slot-values peuvent appartenir aux slot-labels différents ; ce qui n'est pas notre cas.

Enfin on peut remarquer que les approches statistiques sont efficaces pour traiter les structures plates (c'est-à-dire, sans slots imbriqués) mais pour l'étape de décision, parfois on a besoin de la structure hiérarchique. Par exemple, au lieu de : *[device=light, location=kitchen]*, on pourrait générer : *[device=[name=light, location=kitchen]]*. Cela peut être utile si on décide à intégrer le traitement de deux ordres à la fois, par exemple *allume la lumière et la four de la cuisine*, où nous aurions deux slots *locations* distincts à l'intérieur des slots *devices* correspondants à la lumière et la four.

(Dusek et Jurcícek, 2016) décrit un modèle séquence-en-séquence pour la génération des phrases à partir des arbres syntaxique décorés comme celui présenté sur la figure 8.1

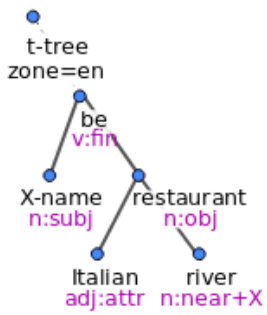


FIGURE 8.1 – L'arbre syntaxique décoré correspondant à la phrases *X-name is an Italian restaurant by the river* (Dusek et Jurcícek, 2016)

Cet arbre correspond à la phrases “X-name is an Italian restaurant by the river”. La représentation arborescente est d’abord convertie dans une chaîne des caractères à l’aide des parenthèses :

(<root> <root> ((X-name n :subj) be v : fin ((Italian adj :attr) restaurant n :obj (river n :near+X))))

Cette représentation est passé au modèle LSTM RNN du type seq2seq, encoder-decoder avec attention, qui produit une séquence des mots correspondant à cet arbre, c’est-à-dire, la phrase. Pour pénaliser le système pour les sorties qui manquent ou ajoutent les informations, ils utilisent un LSTM RNN classifieur qui prend en entrée cette phrase prédite et qui détermine s’il y a des omissions ou ajouts des slots par rapport à la représentation sémantique de source.

Nous pourrions appliquer cette approche dans le sens inverse pour la compréhension automatique ; c’est-à-dire, entraîner un système qui convertit les phrases en arbres syntaxiques décorés qui sont assez proches de la représentation sémantique par slots.

Chapitre 9

Conclusion

Le développement de systèmes de la compréhension automatique d'ordres vocaux pour le système VocADom est rendu difficile par le manque de données d'apprentissage pour ce système. Dans ce but, la grammaire de génération du corpus des ordres annotés a été créée pour le domaine de l'habitat intelligent. Ce corpus a été généré à l'aide du module NLTK de Python, notamment la classe FeatStruct (Feature-Based Grammar, ou grammaire des attributs). Ce mémoire a décrit entre autres, les améliorations que nous avons apportées à cette grammaire. Si au début, elle ne générait que 62 ordres annotées, nous avons augmenté ce nombre d'abord à 26082 (la première version du corpus artificiel) et finalement à 42195 (deuxième version).

Ce corpus artificiel a été évalué en entraînant avec lui quatre modèles - Tri-CRF, att-RNN, RASA (appris sur la première version du corpus artificiel) et Tf-seq2seq (appris sur la deuxième version du corpus artificiel avec une addition d'un petit corpus réaliste SWEET-HOME de 727). Les quatre modèles ont ensuite été testés sur un petit corpus réaliste VocADom. Ces modèles ont ensuite été comparés aux modèles entraînés et testés sur le corpus de données conversationnelles réelles PORTMEDIA. Cette comparaison nous a permis de savoir que la performance des modèles entraînés sur le corpus artificiel est due aux limitations du corpus artificiel plutôt qu'aux limitations des modèles eux-mêmes.

Tous les modèles entraînés et testés sur PORTMEDIA avaient rendu de bon résultats mais les modèles entraînés sur le corpus artificiel et testés sur le corpus réaliste VocADom ont été bien moins performants. Ceci montre la difficulté à prendre en charge la diversité d'un corpus réel par rapport à celle d'un corpus artificiel. Le corpus artificiel manque du vocabulaire et constructions syntaxiques, qui pourraient être ajoutées en s'inspirant des corpus réalistes. Des ressources externes pourraient aussi y être ajoutées pour enrichir le vocabulaire. La distribution des intentions/slots peut également jouer un rôle important car les modèles montrent de mauvais résultats sur les éléments qui sont sous-représentés dans le corpus d'entraînement.

Tous les quatre modèles étudiés ont montrés des résultats comparables. Pourtant contrairement à Tri-CRF, RASA et att-RNN, Tf-seq2seq ne requiert pas que l'apprentissage d'un seul modèle et ne demande pas de données alignées, donc il peut être utilisé dans les conditions d'absence de données alignées et du manque du temps d'entraînement.

La performance sur la prédiction des intentions est assez bonne pour les quatre modèles, mais cela peut être dû au fait que le nombre d'intentions est très restreint (4 dans PORTMEDIA) contrairement au nombre des slot-labels et slot-values (jusqu'à 378 dans PORTMEDIA).

Sur la tâche difficile de prédiction des slot-labels et slot-values, RASA a montré la meilleure performance. Cela peut être dû au fait qu'il utilise des word-embeddings pré-appris sur les grandes ressources externes ce qui assure le meilleure traitement des mots hors vocabulaire du corpus d'entraînement. Il est possible que les algorithmes utilisés dans la compréhension de

la langue automatique pourraient profiter de l'intégration de ces mécanismes.

Nous avons aussi découvert que la présence de l'intention semble ne pas avoir d'influence sur la prédiction des slots par Tf-seq2seq. En ce qui concerne son mode mot vs caractère, la performance du modèle à caractères est pis sur les intentions mais elle est mieux sur les slot-labels et slot-values.

Bibliographie

ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y. et ZHENG, X. (2015). TensorFlow : Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

AGARWAL, S. et DYMETMAN, M. (2017). A surprisingly effective out-of-the-box char2char model on the e2e nlg challenge dataset. *In Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 158–163. Association for Computational Linguistics.

AMAN, F. (2014). *Reconnaissance automatique de la parole de personnes âgées pour les services d'assistance à domicile. Traitement du signal et de l'image.* Thèse de doctorat, Université de Grenoble, École doctorale MSTII.

ASADI, A. et MAKHOUL, R. S. J. (1991). Automatic modeling for adding new words to a large-vocabulary continuous speech recognition system. *In Proceedings of the Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference*, ICASSP '91, pages 305–308, Washington, DC, USA. IEEE Computer Society.

BILLI, R., CANAVESIO, F. et RULLENT, C. (1998). Automation of telecom italia directory assistance service : field trial results. *In Interactive Voice Technology for Telecommunications Applications, 1998. IVTTA '98. Proceedings. 1998 IEEE 4th Workshop*, pages 11–16.

BIRD, S., KLEIN, E. et LOPER, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st édition.

BONNEAU-MAYNARD, H., QUIGNARD, M. et DENIS, A. (2009). Media : a semantically annotated corpus of task oriented dialogs in french. results of the french media evaluation campaign. *Language Resources and Evaluation*, 43(4):329–354.

BORDES, A. et WESTON, J. (2016). Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

BRAUN, D., HERNANDEZ-MENDEZ, A., MATTHES, F. et LANGEN, M. (2017). Evaluating natural language understanding services for conversational question answering systems. *In Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*.

BRITZ, D., GOLDIE, A., LUONG, T. et LE, Q. (2017). Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints*.

BUTZBERGER, J., MURVEIT, H., SHRIBERG, E. et PRICE, P. (1992). Modeling spontaneous speech effects in large vocabulary speech recognition applications. *In In Speech and Natural Language Workshop*. Morgan Kaufmann.

CALLEJAS, Z. et LÓPEZ-CÓZAR, R. (2009). Designing smart home interfaces for the elderly. *SIGACCESS Accessibility and Computing*, 95:10–16.

CHAN, M., ESTÈVE, D., ESCRIBA, C. et CAMPO, E. (2008). A review of smart homes-present state and future challenges. *Computer Methods and Programs in Biomedicine*, 91(1):55–81.

CHAN, W., JAITLY, N., LE, Q. V. et VINYALS, O. (2015). Listen, attend and spell. *CoRR*, abs/1508.01211.

CHO, K., van MERRIENBOER, B., GÜLÇEHRE, Ç., BOUGARES, F., SCHWENK, H. et BENGIO, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

CHUNG, J., GÜLÇEHRE, Ç., CHO, K. et BENGIO, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

CORE, M. G. et ALLEN, J. F. (1997). Coding dialogs with the damsl annotation scheme.

CORTES, C. et VAPNIK, V. (1995). Support-vector networks. *In Machine Learning*, pages 273–297.

CRYSTAL, D. (2011). *A Dictionary of Linguistics and Phonetics*. The Language Library. Wiley.

DUSEK, O. et JURCÍCEK, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *CoRR*, abs/1606.05491.

GURNEY, K. (1997). *An Introduction to Neural Networks*. Taylor & Francis, Inc., Bristol, PA, USA.

HE, Y. et YOUNG, S. (2003). A data-driven spoken language understanding system. *In 2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*, pages 583–588.

HEMPHILL, C. T., GODFREY, J. J. et DODDINGTON, G. R. (1990). The atis spoken language systems pilot corpus. *In Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 96–101, Stroudsburg, PA, USA. Association for Computational Linguistics.

HETHERINGTON, I. L. et ZUE, V. W. (1993). New words : implications for continuous speech recognition. *In Third European Conference on Speech Communication and Technology, EUROSPEECH 1993*.

HUANG, L., SIL, A., JI, H. et FLORIAN, R. (2017). Improving slot filling performance with attentive neural networks on dependency structures. *CoRR*, abs/1707.01075.

JEONG, M. et LEE, G. G. (2008). Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1287–1302.

JI, Y., HAFFARI, G. et EISENSTEIN, J. (2016). A latent variable recurrent neural network for discourse relation language models. *CoRR*, abs/1603.01913.

KALCHBRENNER, N. et BLUNSOM, P. (2013). Recurrent convolutional neural networks for discourse compositionality. *CoRR*, abs/1306.3584.

KOSKELA, T. et VÄÄNÄNEN-VAINIO-MATTILA, K. (2004). Evolution towards smart home environments : Empirical evaluation of three user interfaces. *Personal Ubiquitous Comput.*, 8(3-4):234–240.

LAFFERTY, J. D., MCCALLUM, A. et PEREIRA, F. C. N. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data. *In Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

LEFÈVRE, F., MOSTEFA, D., BESACIER, L., ESTÈVE, Y., QUIGNARD, M., CAMELIN, N., FAVRE, B., JABAIAN, B. et ROJAS-BARAHONA, L. M. (2012). Leveraging study of robustness and portability of spoken language understanding systems across languages and domains : the PORTMEDIA corpora. *In LREC*, pages 1436–1442.

LIU, B. et LANE, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *CoRR*, abs/1609.01454.

LIU, B. et LANE, I. (2017). An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *CoRR*, abs/1708.05956.

MANISHINA, E., JABAIAN, B., HUET, S. et LEFEVRE, F. (2016). Automatic corpus extension for data-driven natural language generation. *In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

MESNIL, G., DAUPHIN, Y., YAO, K., BENGIO, Y., DENG, L., HAKKANI-TUR, D., HE, X., HECK, L., TUR, G., YU, D. et ZWEIG, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. et DEAN, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

MÖLLER, S., GÖDDE, F. et WOLTERS, M. (2008). Corpus analysis of spoken smart-home interactions with older users. *In* CALZOLARI, N., CHOUKRI, K., MAEGAARD, B., MARIANI, J., ODIJK, J., PIPERIDIS, S. et TAPIAS, D., éditeurs : *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

NOVIKOVA, J., DUŠEK, O. et RIESER, V. (2017). The E2E dataset : New challenges for end-to-end generation. *In Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. arXiv :1706.09254.

PIERACCINI, R., TZOUKERMANN, E., GORELOV, Z., LEVIN, E., GAUVAIN, J.-L., LEE, C.-H. et WILPON, J. G. (1992). A speech understanding system based on statistical representation of semantics. *In International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 193–196.

PIETQUIN, O. et HASTIE, H. (2013). A survey on metrics for the evaluation of user simulations. *Knowledge Engineering Review*, 28(01):59–73.

PORTET, F., VACHER, M., GOLANSKI, C., ROUX, C. et MEILLON, B. (2013). Design and evaluation of a smart home voice interface for the elderly : Acceptability and objection aspects. *Personal Ubiquitous Comput.*, 17(1):127–144.

RAIMONDO, S. (2017). Natural language understanding for smarthomes. internship report.

RAUX, A., LANGNER, B., BOHUS, D., BLACK, A. W. et ESKÉNAZI, M. (2005). Let's go public ! taking a spoken dialog system to the real world. *In INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech, Communication and Technology, Lisbon, Portugal, September 4-8, 2005*, pages 885–888.

RAVURI, S. V. et STOLCKE, A. (2015). Recurrent neural network and lstm models for lexical utterance classification. *In INTERSPEECH*, pages 135–139. ISCA.

RESNIK, P. et LIN, J. (2010). *Evaluation of NLP Systems. In The Handbook of Computational Linguistics and Natural Language Processing*. Wiley Blackwell.

ROSSATO, S., VACHER, M. et PORTET, F. (2018). Documentation interne du projet ANR VocADom.

RUDNICKY, A. I., THAYER, E. H., CONSTANTINIDES, P. C., TCHOU, C., SHERN, R., LENZO, K. A., XU, W. et OH, A. (1999). Creating natural dialogs in the carnegie mellon communicator system. *In Sixth European Conference on Speech Communication and Technology, EUROSPEECH 1999, Budapest, Hungary, September 5-9, 1999*.

SENEFF, S. (1992). Tina : A natural language system for spoken language applications. *Comput. Linguist.*, 18(1):61–86.

SERBAN, I. V., LOWE, R., HENDERSON, P., CHARLIN, L. et PINEAU, J. (2015). A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1512.05742.

STUBBS, A. et PUSTEJOVSKY, J. (2012). *Natural Language Annotation for Machine Learning*. O'Reilly Media.

SUTSKEVER, I., VINYALS, O. et LE, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

TRAN, Q. H., ZUKERMAN, I. et HAFFARI, G. (2017). Preserving distributional information in dialogue act classification. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, (EMNLP) 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2151–2156.

TRAUM, D. R. et HEEMAN, P. A. (1996). Utterance units in spoken dialogue. *In* MAIER, E., MAST, M. et LUPERFOY, S., éditeurs : *Dialogue Processing in Spoken Language Systems, ECAI'96 Workshop, Budapest, Hungary, August 13, 1996, Revised Papers*, volume 1236 de *Lecture Notes in Computer Science*, pages 125–140. Springer.

TUR, G. et DE MORI, R. (2011). *Spoken Language Understanding Systems for Extracting Semantic Information from Speech*. Wiley.

TUR, G., HAKKANI-TÜR, D. et HECK, L. (2010). What's left to be understood in ATIS ? *In IEEE Workshop on Spoken Language Technologies*.

VACHER, M. (2018). Projet ANR VocADom - Livrable C5.1 - Corpus sonore et multimodal d'évaluation. Rapport technique, CNRS.

VACHER, M., CAFFIAU, S., PORTET, F., MEILLON, B., ROUX, C., ELIAS, E., LECOUTEUX, B. et CHAHUARA, P. (2015). Evaluation of a context-aware voice interface for ambient assisted living : qualitative user study vs. quantitative system evaluation. *ACM Transactions on Accessible Computing*, 7(2):5 :1–5 :36.

VACHER, M., LECOUTEUX, B., CHAHUARA, P., PORTET, F., MEILLON, B. et BONNEFOND, N. (2014). The Sweet-Home speech and multimodal corpus for home automation interaction. *In The 9th edition of the Language Resources and Evaluation Conference (LREC)*, pages 4499–4506, Reykjavik, Iceland.

WALKER, M. A., PASSONNEAU, R. et BOLAND, J. E. (2001). Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. *In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 515–522, Stroudsburg, PA, USA. Association for Computational Linguistics.

WALLACE, R. S. (2009). *The Anatomy of A.L.I.C.E.*, pages 181–210. Springer Netherlands, Dordrecht.

WANG, Y.-Y., DENG, L. et ACERO, A. (2005). Spoken language understanding — an introduction to the statistical framework. *IEEE Signal Processing Magazine*, 22/5:16–31.

WARD, W. (1989). Modelling non-verbal sounds for speech recognition. *In Proceedings of the Workshop on Speech and Natural Language*, HLT '89, pages 47–50, Stroudsburg, PA, USA. Association for Computational Linguistics.

WARD, W. (1990). The cmu air travel information service : Understanding spontaneous speech. *In Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 127–129, Stroudsburg, PA, USA. Association for Computational Linguistics.

WARD, W. (1991). Understanding spontaneous speech : the phoenix system. *In [Proceedings] ICASSP 91 : 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 365–367 vol.1.

WEIZENBAUM, J. (1966). Eliza&mdash ;a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

WILLIAMS, J. D., RAUX, A. et HENDERSON, M. (2016). The dialog state tracking challenge series : A review. *D&D*, 7(3):4–33.

yi WANG, Y. et ACERO, A. (2006). Discriminative models for spoken language understanding. *In in ICSLP*.

YOUNG, S., GAŠIĆ, M., THOMSON, B. et WILLIAMS, J. D. (2013). Pomdp-based statistical spoken dialog systems : A review. *Proceedings of the IEEE*, 101(5):1160–1179.

YU, Z., BLACK, A. W. et RUDNICKY, A. I. (2017). Learning conversational systems that interleave task and non-task content. *CoRR*, abs/1703.00099.

YU, Z., BOHUS, D. et HORVITZ, E. (2015). Incremental coordination : Attention-centric speech production in a physically situated conversational agent. *In Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 402–406.

ZHANG, X. et WANG, H. (2016). A joint model of intent determination and slot filling for spoken language understanding. *In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2993–2999. AAAI Press.

ZUE, V., SENEFF, S., POLIFRONI, J., PHILLIPS, M. S., PAO, C., GOODINE, D., GODDEAU, D. et GLASS, J. R. (1994). PEGASUS : A spoken dialogue interface for on-line air travel planning. *Speech Communication*, 15(3-4):331–340.