La compréhension de la parole dans les systèmes de dialogues humain-machine à l'heure des modèles pré-entraînés

Valentin Pelloin

THÈSE DE DOCTORAT

De
LE MANS UNIVERSITÉ
Sous le sceau de
LA COMUE ANGERS-LE MANS

ÉCOLE DOCTORALE N° 641
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Valentin PELLOIN

La compréhension de la parole dans les systèmes de dialogues humain-machine à l'heure des modèles pré-entraînés

Thèse présentée et soutenue à Le Mans, le 24 janvier 2024
Unité de recherche : Laboratoire d'Informatique de l'Université du Mans
Thèse N° : 2024LEMA1002

Rapporteurs avant soutenance :

Christophe CERISARA	Chargé de recherche (HDR)	LORIA, Nancy
Benoit FAVRE	Professeur des universités	LIS, Marseille

Composition du Jury :

Président :	Benoit FAVRE	Professeur des universités	LIS, Marseille
Examinateurs :	Christophe CERISARA	Chargé de recherche (HDR)	LORIA, Nancy
	Géraldine DAMNATI	Ingénieure de recherche	Orange Labs, Lannion
	Richard DUFOUR	Maître de conférences (HDR)	LIA, Avignon
	Benoit FAVRE	Professeur des universités	LIS, Marseille
	Sophie ROSSET	Directrice de recherche	LISN, Orsay
Dir. de thèse :	Sylvain MEIGNIER	Professeur des universités	LIUM, Le Mans
Encadrants :	Nathalie CAMELIN	Maîtresse de conférences	LIUM, Le Mans
	Antoine LAURENT	Professeur des universités	LIUM, Le Mans

Remerciements

Ce sont ces dernières phrases qui se retrouvent les premières au sein de ce manuscrit. Elles en sont néanmoins, à mon avis, les plus importantes. À travers celles-ci, je souhaite remercier toutes les personnes ayant contribué, de près comme de loin, à la réussite de mes travaux exposés ci-dessous.

Tout d'abord, Sylvain, je te remercie, à la fois pour ton expertise et également ta bienveillance. Tes questions ont toujours su m'interroger de manière pertinente et me permettre de m'améliorer. Antoine, tes compétences techniques et de recherche ont su me guider durant ces trois dernières années (merci également pour la gestion de la machine à café !). Enfin, comment ne pas te remercier Nathalie ? Sans toi, cette thèse n'aurait pas été possible, et pour plusieurs raisons. Tu as tout d'abord accordé en moi ta confiance dans la réalisation de ces travaux. Ton expertise sur le domaine, ainsi que tes scripts Perl (*ouch*) ont été cruciaux. Ensuite, tu as toujours été là dans les moments de doute et tu as su préserver ma motivation. Merci à vous d'avoir su me faire confiance.

Durant mes travaux, j'ai eu l'immense honneur de travailler en collaboration avec diverses personnes. Antoine (C), tu as soutenu ta thèse peu de temps après le début de la mienne, mais tes conseils et tes fichiers pré-traités m'ont permis de rapidement entrer dans le vif du sujet. J'ai également eu la chance de collaborer avec de nombreuses personnes associées au LIA : Gaëlle, Renato, Salima, Yannick. Collaborer avec vous a été très enrichissant et une vraie opportunité pour moi. Sahar, merci de toujours avoir répondu à mes questions. Tes modèles ont bien souvent été meilleurs que les miens, mais c'est grâce à cela si j'ai tant su me remettre en question. Merci par ailleurs à mes nouveaux collègues de l'INA pour la confiance dont vous m'avez apporté durant ces derniers mois.

J'ai eu la chance de participer, de près comme de loin, aux workshops JSALT 2022 et 2023. J'y ai passé de très bons moments, tant dans les collaborations scientifiques, que dans les séances cinéma, karaoké, parties de pingpong, billard, randonnées ou canoë. Je ne vais malheureusement pas pouvoir tous·tes vous citer, mais je tiens à remercier chaleureusement Gaëlle (encore !), Harshita, Pablo, Patricia, Peter, Themos, Thomas, Victoria pour avoir rendu ces événements formidables. Ces événements n'auraient pu se produire sans l'immense travail réalisé par Anthony, Emmanuelle et aussi Sanjeev, merci à vous.

Le LIUM a été comme une deuxième (troisième ?) maison pour moi, merci à vous tous·tes pour l'accueil. Ça a été un vrai plaisir de travailler avec vous. C'est grâce à Anne-Cécile, Etienne et Grégor que le LIUM peut fonctionner tel qu'il est aujourd'hui. Alors comment ne pas vous remercier tout particulièrement ? (et s.v.p, pardonnez-moi pour toutes mes demandes compliquées et souvent à la dernière minute...)

L'ambiance au labo était absolument formidable, et c'est aussi en grande partie grâce aux doctorant·e·s. Albane, j'ai adoré pouvoir discuter de tout, mais aussi de rien avec toi. On a pu débattre sur tellement de choses, c'était génial (et j'ai été ravi de pouvoir faire quelques figures Ti*k*Z en plus pour toi !). Martin, tu nous as tous·tes fait profiter, parfois contre notre gré, de tes talents en percussions, mais aussi récemment en Otamatone. Simon, tes connaissances linguistiques m'ont toujours épaté. Théo, voisin de bureau, on t'a adopté avec plaisir quand tu nous as rejoints au LIUM. J'ai adoré assister à tes concerts avec Martin. Thibault le dino, voisin de la porte d'à côté, mais également coloc' de Baltimore : on a eu de nombreuses conversations super intéressantes. Mais j'ai particulièrement apprécié de pouvoir partager avec toi les montagnes russes à Orlando, la visite de Cape Canaveral, sans oublier le trajet en train jusqu'à New York. Enfin, Thibault le vieux. Nos premiers travaux ensemble commencent à dater désormais : que le temps passe vite ! Ça a été un plaisir de bosser avec toi. Je continue cependant à être perplexe vis-à-vis de ta recette de beignets de nouilles... Heureusement, tu fais, je pense, les meilleures crêpes qui peuvent exister en France.

Sans ma famille, cette thèse n'aurait sans doute jamais vu le jour. Maman, Papa, merci de m'avoir supporté et soutenu durant toutes ces années d'études. Merci aussi pour les bons petits plats ramenés les dimanches soir ! Romain, Nolwenn, Noah, Léo, Candice, Julia, cette petite famille qui est devenue de plus en plus nombreuse aux fils des années a su m'apporter bonheur et joie tout au long de cette thèse.

Je tiens également à remercier la personne, qui, se reconnaîtra à travers ces lignes, et qui a toujours su être à mes côtés durant ces dernières années, dans les moments de joie comme dans les moments de doute. Merci à toi.

À tous et à toutes, je vous dis : merci !

À mes grands-parents,

TABLE DES MATIÈRES

Introduction **15**

I Contexte et état de l'art 21

1 L'apprentissage profond 23

1.1 Architectures . . . 25
1.1.1 Les réseaux de neurones à propagation avant . . . 25
1.1.2 Les réseaux récurrents . . . 27
1.1.3 *Connectionist Temporal Classification* . . . 30
1.1.4 Les mécanismes d'attention . . . 31
1.1.5 Les transformers . . . 33
1.2 Méthodes d'apprentissage . . . 36
1.2.1 L'apprentissage supervisé . . . 37
1.2.2 L'apprentissage auto-supervisé . . . 37
1.2.3 Techniques d'optimisation . . . 39
1.2.4 Apprentissage *from scratch*, *finetuning* et x-*shot* . . . 42
1.2.5 Optimisation des hyper-paramètres . . . 43
1.3 Conclusion . . . 43

2 La reconnaissance automatique de la parole 47

2.1 Principe général . . . 48
2.2 Modélisation acoustique : $P(X \mid Y)$ . . . 50
2.2.1 Représentations d'entrée . . . 50
2.2.2 Les modèles de Markov cachés . . . 52
2.2.3 Les modèles neuronaux . . . 54
2.3 Modélisation de la langue : $P(Y)$ . . . 55
2.3.1 Modélisation n-grammes . . . 57
2.3.2 Modélisation neuronale . . . 58
2.3.3 *Tokenization* du texte . . . 59
2.4 Décodage, prédiction et évaluation . . . 60

2.4.1 Intégration du modèle de langage au modèle acoustique 60
2.4.2 Décodage : glouton ou faisceau ? 61
2.4.3 Évaluation des transcriptions 62
2.5 Corpus 64
2.6 Conclusion 66

3 La compréhension de la parole 69
3.1 Représentations sémantiques 71
3.1.1 Représentation à base de *frames* 71
3.1.2 Représentation en segments sémantiques 72
3.1.3 Représentation générale par classes 72
3.2 Compréhension de la parole : quelles applications ? 73
3.2.1 Routage et détection de l'intention 74
3.2.2 Résumé automatique du discours 74
3.2.3 Recherche vocale 74
3.2.4 Systèmes de questions-réponses 75
3.2.5 Systèmes de dialogue humain-machine 75
3.3 Les systèmes de compréhension 77
3.3.1 Systèmes NLU 77
3.3.2 Systèmes SLU 79
3.4 Méthodologie : formats et évaluation 81
3.4.1 Formats 81
3.4.2 Les métriques d'évaluation 83
3.5 Les corpus de compréhension de la parole 84
3.5.1 ATIS et MultiAtis++ 84
3.5.2 MultiWOZ 86
3.5.3 SpokenWOZ 86
3.5.4 MEDIA 87
3.5.5 PortMEDIA 88
3.5.6 Récapitulatif 89
3.6 Conclusion 90

II Contributions 93

4 Architectures récurrentes pour la compréhension *end-to-end* 95

4.1 Une architecture encodeur-décodeur 96
4.1.1 Protocole expérimental 96
4.1.2 Extraction des valeurs associées aux concepts 99
4.1.3 Détail de l'architecture 100
4.1.4 Résultats obtenus 103
4.2 Hybridation des systèmes *end-to-end* et cascade 107
4.2.1 Pourquoi une architecture hybride ? 108
4.2.2 Détail de l'architecture employée 109
4.2.3 Résultats obtenus 112
4.3 Conclusion 114

5 Architectures *transformers* pour la compréhension de la parole 117
5.1 Architectures SSL *end-to-end* et cascade 118
5.1.1 Architectures 119
5.1.2 Protocole d'apprentissage et d'évaluation 122
5.1.3 Résultats de l'architecture *baseline* cascade 124
5.1.4 Résultats de l'architecture *end-to-end* 125
5.1.5 Discussions 126
5.2 Analyse des erreurs 127
5.2.1 Systèmes étudiés 127
5.2.2 Distribution des erreurs par concept 128
5.2.3 Évaluation de la transcription sur les mots supports de concepts . . 131
5.3 Conclusion 133

6 Utiliser les modèles pré-entraînés de manière auto-supervisée 137
6.1 Adapter les modèles de langage aux spécificités de la parole 138
6.1.1 FlauBERT et FlauBERT-Oral 139
6.1.2 Utilisation pour la compréhension de la parole 140
6.2 Les modèles SSL multimodaux et multilingues 143
6.2.1 Représentations 143
6.2.2 Protocole d'évaluation *zero-shot* des représentations 144
6.2.3 Résultats 146
6.2.4 Crosslingue et crossmodal : conclusion et perspectives 148
6.3 Utilisation de modèles conversationnels pour la compréhension 149
6.3.1 Détection des concepts MEDIA 150
6.3.2 Compréhension globale du dialogue 154

6.4 Conclusion 158

Conclusion et perspectives **161**

Références personnelles **167**

Références **169**

Liste des tableaux **193**

Liste des figures **195**

Acronymes **199**

Introduction

Depuis longtemps, la parole est perçue comme une interface de communication idéale entre les machines et les humains. Dans de nombreuses œuvres de fiction, que ce soit avec Hal-9000 (Clarke 1968), C-3PO (Lucas 1977), *la machine* (Nolan 2011), ou d'autres, les humains dialoguent naturellement avec des machines comme ils le feraient avec d'autres humains. Pour la plupart des êtres humains, la parole est la forme la plus naturelle de communication. Cependant, la parole n'est pas l'interface idéale pour un système automatique. La parole est une source de données qui n'est pas formelle et qui est sujette à de nombreuses variations : langue, accent, vocabulaire, bruit, hésitations, répétitions, etc. Le traitement et la compréhension *automatique* d'une telle source de données sont ainsi des tâches ardues. Dans ces travaux, nous nous intéressons à ces tâches dans le cadre des conversations orales entre un humain et une machine.

La compréhension de la parole consiste à extraire une représentation sémantique du sens du message émis par un·e utilisateur·e. Cette représentation sera ensuite utilisée par une application dans un cadre particulier : routage d'appel, résumé automatique, systèmes de recherche vocale, ou systèmes de dialogues humain-machine par exemple. Nos travaux de thèse se concentrent sur l'extraction de représentations sémantiques dans ce dernier cadre et plus particulièrement dans celui de la réservation de chambres d'hôtels et d'informations touristiques par téléphone, au travers des corpus MEDIA (Devillers et al. 2004) et PortMEDIA (Lefèvre et al. 2012).

Traditionnellement, la compréhension de la parole est réalisée en deux temps, grâce à une cascade de systèmes. Une première étape de reconnaissance automatique permet de transcrire la parole en une suite de mots (ou de caractères). Un second module est alors utilisé pour extraire les représentations sémantiques à partir de cette séquence de mots. Ces deux modules peuvent être construits avec des réseaux de neurones, qui sont les architectures d'intelligence artificielle les plus populaires. Cependant, il y a quelques années, avec le progrès des architectures neuronales et grâce à de grandes quantités de données disponibles, des systèmes unifiés (dits «*end-to-end*») ont vu le jour. Ces systèmes *end-to-end* génèrent directement les représentations sémantiques à partir du signal de parole.

De nouveaux modèles ont depuis vu le jour : les *transformers*. Il s'agit de modèles

neuronaux faisant l'usage de mécanismes d'attention pour traiter des séquences. Ils possèdent généralement un très grand nombre de paramètres. Pour apprendre ces nouveaux modèles, des techniques de pré-apprentissage auto-supervisé sont généralement employées. Ce pré-apprentissage est effectué sur des données sans annotation : par exemple de la parole sans transcription ou des phrases sans annotation sémantique. Ces corpus ne requièrent pas d'annotation humaine manuelle, c'est pourquoi ces données sont disponibles en très grande quantité. Les modèles peuvent ensuite être adaptés pour différentes tâches. Ils sont désormais état-de-l'art en compréhension automatique de la parole avec un fonctionnement en cascade : un premier modèle *transformer* est utilisé pour transcrire, puis un deuxième l'est pour extraire les représentations sémantiques.

Le choix d'une architecture cascade plutôt que *end-to-end* semble donc être une question encore sans réponse définitive. Chaque architecture possède ses avantages : une architecture cascade peut bénéficier des modèles *transformers* et du pré-entraînement massif qu'il a subi. Cependant, ces architectures sont sujettes à une amplification des erreurs de transcription dans le second module. Notre objectif est d'étudier ces types d'architecture dans le cadre de la compréhension de la parole. Nous essayons néanmoins d'aller au-delà de cette séparation entre *end-to-end* et cascade, en proposant des nouvelles architectures hybrides. Nous développons ainsi différents systèmes de compréhension cascade, *end-to-end* et hybride, avec et sans utilisation des modèles *transformers* pré-entraînés.

Par ailleurs, depuis quelques mois, certains outils d'intelligence artificielle sont connus auprès du grand public. En traitement de la langue, de nouveaux *chatbots* ont vu le jour. Ces nouveaux systèmes sont issus de modèles de langage pré-entraînés de manière auto-supervisée, et sont ensuite adaptés sur des données conversationnelles. Les humains peuvent dialoguer à l'écrit avec ces systèmes. Ces nouveaux systèmes questionnent les perspectives envisageables pour la compréhension de la parole. Pourraient-ils être directement utilisés pour résoudre des tâches telles que celles associées aux corpus MEDIA et PortMEDIA ?

Contexte de la thèse

Les travaux réalisés dans cette thèse s'inscrivent dans le cadre du projet ANR AISSPER (*Artificial Intelligence for Semantically controlled SPEech undeRstanding*). Le projet vise à développer des nouveaux modèles de compréhension de la parole au niveau de la phrase et du document, en utilisant notamment des mécanismes d'attention. Les travaux réalisés se sont également intégrés dans deux autres projets auxquels le LIUM a participé : le projet BigScience et le *workshop* JSALT 2022. Le projet BigScience avait comme objectif

l'étude et la création de larges modèles de langue. Le projet dans lequel l'équipe du LIUM a rejoint le *workshop* JSALT 2022 portait sur la traduction multimodale et multilingue, avec une focalisation sur les langues peu dotées en ressources.

Organisation du document

Ce document est organisé en deux parties, chacune divisée en trois chapitres.

La première partie situe cette thèse dans son contexte, avec un état de l'art.

- Le premier chapitre réalise un état de l'art de l'apprentissage profond. Nous présentons différentes structures de réseaux de neurones, ainsi que des méthodes pour apprendre ces mêmes réseaux.
- Le second chapitre décrit la tâche de reconnaissance automatique de la parole. Nous y présentons différents modèles et architectures utilisés pour résoudre cette tâche.
- Dans le troisième chapitre, nous décrivons le domaine de la compréhension automatique (de la langue et de la parole). Nous présentons différents modèles utilisés pour la compréhension du dialogue entre un humain et une machine, et nous décrivons les corpus habituellement utilisés.

Nous présentons dans la seconde partie les contributions apportées par nos travaux.

- Le chapitre quatre présente des premières architectures neuronales pour la compréhension de la parole sur les corpus MEDIA et PortMEDIA. Il s'agit d'architectures récurrentes. Une première architecture *end-to-end* est développée, puis une version hybride cascade-*end-to-end* est construite.
- Dans le cinquième chapitre, nous présentons des architectures de compréhension basées sur les modèles *transformers*. Il s'agit de modèles qui ont été pré-entraînés sur des données de parole ou de texte, sans aucune autre annotation. Nous développons un système cascade, et différentes versions *end-to-end*, ainsi qu'hybrides. Nous présentons également dans ce chapitre des analyses des erreurs produites par ces systèmes, afin de tenter de comprendre quelles seraient les perspectives d'amélioration.
- Enfin, nous présentons dans le sixième chapitre des travaux où nous utilisons différents modèles *transformers* pré-appris de manière auto-supervisée pour la compréhension de la langue. Nous présentons un premier système utilisant un modèle de langage adapté à la langue parlée. Nous évaluons ensuite des systèmes multilingues et multimodaux pour la compréhension. Enfin, nous explorons l'usage des modèles *chatbots* pour notre tâche de compréhension.

Première partie

Contexte et état de l'art

Chapitre 1

L'apprentissage profond

Sommaire

1.1 Architectures 25
1.1.1 Les réseaux de neurones à propagation avant 25
1.1.2 Les réseaux récurrents 27
1.1.3 *Connectionist Temporal Classification* 30
1.1.4 Les mécanismes d'attention 31
1.1.5 Les transformers 33
1.2 Méthodes d'apprentissage 36
1.2.1 L'apprentissage supervisé 37
1.2.2 L'apprentissage auto-supervisé 37
1.2.3 Techniques d'optimisation 39
1.2.4 Apprentissage *from scratch*, *finetuning* et *x-shot* 42
1.2.5 Optimisation des hyper-paramètres 43
1.3 Conclusion 43

L'une des premières briques essentielles en informatique, et en IA (Intelligence Artificielle) fut apportée par Church (1936) et Turing (1937) dans leurs travaux sur la calculabilité. Cette brique, par la suite nommée *thèse de Church-Turing*, indique que n'importe quel raisonnement mathématique peut être calculé mécaniquement, en manipulant des nombres binaires. À la suite de ces travaux, certains ont commencé à reproduire des caractéristiques neuronales du cerveau avec des fonctions mathématiques (McCulloch et Pitts 1943). Les architectures neuronales et leurs utilisations ont évolué avec le temps. Désormais, en 2023, l'intelligence artificielle est présente partout dans nos vies connectées. À l'heure où les moteurs de recherche reposent désormais sur des IA, où les plateformes de *e-commerce*, de vidéo à la demande utilisent des IA de recommandation, où des IA de génération d'images, de vidéo, de son, et de dialogue (*chatbot*) sont connues et tendent à être utilisées par le grand public, nous nous intéressons dans nos travaux aux intelligences artificielles pour la compréhension de la parole. Mais pour cela, nous devons d'abord revenir sur les différentes architectures d'intelligence artificielle qui existent et qui peuvent être utilisées pour cette tâche.

Deux courants ont vu le jour au sein de ce domaine de l'intelligence artificielle. Nous les représentons sous forme de diagramme d'Euler en Figure 1.1. L'intelligence artificielle

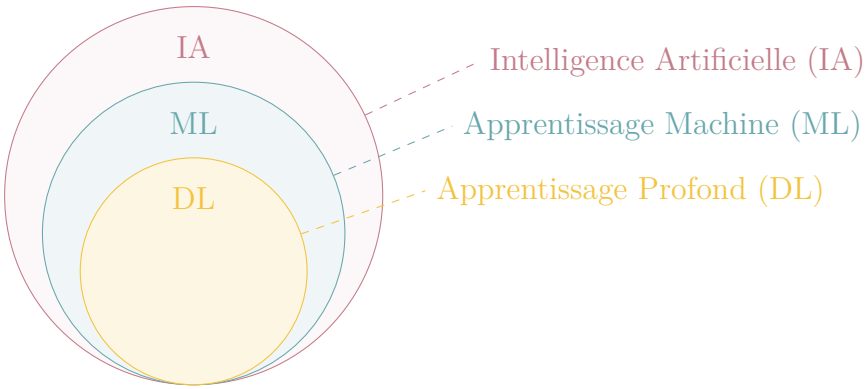


Figure 1.1 – Diagramme d'Euler de l'intelligence artificielle.

peut se définir comme un ensemble de procédés conçus pour imiter l'intelligence (animale, dont les humains). L'apprentissage machine (ML – *Machine Learning*) consiste en des modèles d'intelligence artificielle qui sont capables d'apprendre des connaissances à partir d'un grand nombre de données, sans avoir à définir un jeu de règles complexes pour imiter l'intelligence. Enfin, l'apprentissage profond (DL – *Deep Learning*) correspond aux modèles capables d'apprendre des connaissances, également à partir de données, mais en simulant le comportement de nos cerveaux. Le *Deep Learning* utilise pour cela des réseaux de neurones

artificiels.

Ces trois domaines sont très vastes. Dans ce chapitre, nous nous focalisons sur un ensemble de méthodes d'apprentissage profond. Nous présentons différentes architectures puis des méthodes d'apprentissage.

1.1 Architectures

L'apprentissage profond est réalisé à partir de réseaux de neurones, composés eux-mêmes de briques élémentaires, généralement appelées neurones. Ces neurones sont disposés sous forme de couches (*layers*). Nous décrivons dans cette section les principales architectures, historiques ou actuellement utilisées, permettant de réaliser ces couches.

1.1.1 Les réseaux de neurones à propagation avant

Les réseaux de neurones à propagation avant sont les premiers réseaux de neurones ayant été créés. La notation de propagation avant (*feedfoward neural network* en anglais) désigne le fait que les données ne circulent que dans un sens au sein de ce réseau, de l'entrée vers la sortie. Contrairement aux réseaux récurrents (section 1.1.2), les réseaux à propagation avant ne disposent pas de boucles.

Le *Perceptron*

À l'origine des réseaux de neurones se trouve le *Perceptron*, présenté par Rosenblatt (1958). Il s'agit d'un seul neurone, permettant la classification binaire d'éléments. Il permet de résoudre certains problèmes linéairement séparables tels que la classification d'éléments de jeux de données simples (par exemple certaines classes du corpus Iris (Fisher 1936)). Le perceptron prend en entrée n éléments (des nombres). Le nombre d'éléments en entrée n est fixe, et propre à chaque perceptron. Ensuite, chaque entrée se voit multipliée par un poids associé w_i $(w_1, w_2, \ldots, w_n) \in W$. La somme de ces résultats est calculée. Enfin, une fonction d'activation est employée, afin de décider la sortie (0 ou 1). Au moment de l'apprentissage du perceptron, les poids W sont mis à jour afin de corriger la sortie par rapport à celle attendue. Pour cela, un algorithme d'optimisation, tel que la descente de gradient est utilisé (présentés en section 1.2.3). Ce fonctionnement est schématisé en Figure 1.2.

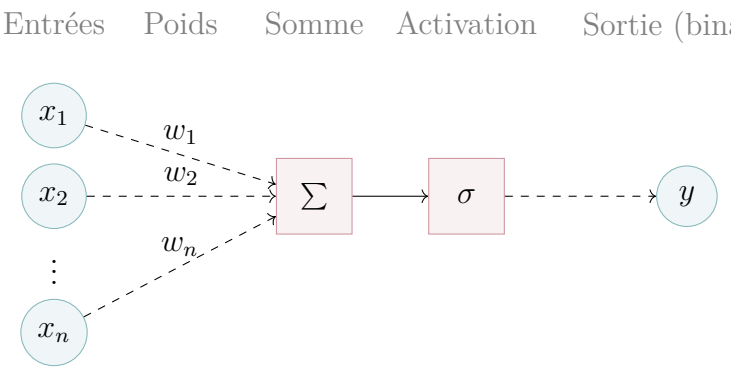


FIGURE 1.2 – Schéma conceptuel du *Perceptron*.

Le *Multilayer Perceptron*

Le principal inconvénient du perceptron est qu'il ne peut pas résoudre de problèmes non linéaires. Le *Multilayer Perceptron*, ou MLP (ROSENBLATT 1961) est une extension du perceptron classique, qui permet de résoudre des problèmes non séparables linéairement. Il est composé d'au moins trois couches : une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie. Chaque couche reprend le fonctionnement de perceptrons. Il s'agit d'un réseau complètement connecté, dans le sens où tous les neurones sont connectés ensemble (à la couche précédente et suivante). Nous présentons en *Figure* 1.3 le schéma de fonctionnement du MLP. Dans un tel modèle, le nombre de couches, et la taille de celles-ci dépendent de la tâche à réaliser. Il faut pour cela procéder de manière empirique pour trouver le bon nombre de paramètres de manière à obtenir des résultats corrects, en évitant le sur-apprentissage et une complexité algorithmique trop élevée.

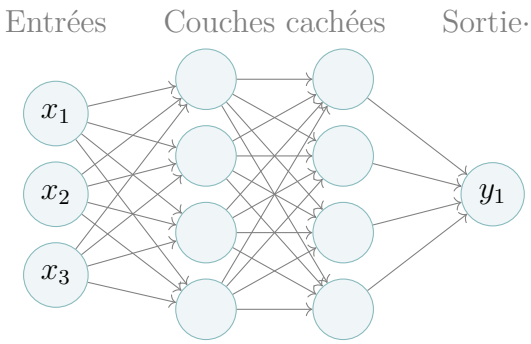


FIGURE 1.3 – Schéma conceptuel du *Multilayer Perceptron*.

Formellement, soit un réseau composé de L couches. Il prend en entrée les valeurs $(x_1, x_2, \ldots, x_n) \in X$. Des poids $W_{l-1,l}$ indiquent la transformation effectuée entre la couche $l-1$ et la couche l. H_l correspond aux vecteurs de sortie de la couche l. Des biais B_l sont

ajoutés à chaque couche l, qui ne dépendent pas de l'entrée X ni du vecteur H_{l-1}. La sortie du réseau est donnée par le résultat de la dernière couche H_L. Des fonctions d'activation f sont utilisées sur les résultats des couches cachées. La sortie H_l d'une couche l est calculée comme suit :

$$H_l = \begin{cases} f\left(W_{l-1,l} \cdot H_{l-1} + B_l\right), & \text{si } l > 1 \\ f\left(W_{l-1,l} \cdot X + B_1\right), & \text{sinon} \end{cases} \tag{1.1}$$

Fonctions d'activation

Les fonctions d'activation sont un ensemble de fonctions utilisées dans les réseaux de neurones. Ces fonctions sont utilisées entre des couches du modèle pour transformer la sortie de la couche précédente. L'utilisation de fonctions non linéaires dans les réseaux de neurones permet d'améliorer l'apprentissage de ceux-ci (Dubey et al. 2022). Parmi les fonctions d'activation les plus utilisées, nous pouvons citer la tangente hyperbolique (tanh), la fonction sigmoïde (σ), ou la fonction de redressement (ReLU). Ces fonctions associent, une valeur pour toute entrée x dans $]-\infty, +\infty[$. Les valeurs de la tangente hyperbolique et de la sigmoïde sont bornées respectivement dans $]-1, 1[$ et $]0, 1[$, tandis que ReLU est bornée dans $[0, +\infty[$.

La fonction softmax est également utilisée dans les réseaux de neurones, mais généralement en sortie des modèles. Celle-ci permet de convertir le vecteur de sortie du modèle en une distribution de probabilité.

1.1.2 Les réseaux récurrents

Les réseaux de neurones *feedforward* décrits précédemment sont particulièrement adaptés à des tâches de classification «globale», où nous ne traitons pas des données séquentielles. Cependant, lorsque nous devons traiter des données sous forme de séquence, souvent de longueurs variables, telles qu'un signal audio, ou une phrase par exemple, il est généralement judicieux d'utiliser un réseau de neurones récurrent, comportant des boucles. Le réseau de neurones récurrent (RNN pour *Recurrent Neural Network*), pour la première fois décrit par Rumelhart et al. (1986), introduit de la récurrence et permet ainsi de conditionner la sortie à l'instant t_n en fonction des sorties précédentes (de t_0 jusqu'à t_{n-1}). Dans le cas d'un réseau bidirectionnel, le conditionnement est également effectué sur les sorties suivantes.

En pratique, le RNN permet surtout de traiter des séquences de longueur variable, là

où le MLP est figé – si n est le nombre d'entrées configuré à la création du modèle, il doit traiter exactement ces n éléments. Pour traiter un nombre d'entrées variable, un RNN est composé de cellules. Une telle cellule est répliquée pour chaque entrée x_t de la séquence. Elle génère une représentation cachée h_t à chaque instant t, qui pourra être utilisée par la suite du réseau. En fonction de la tâche, toutes les représentations cachées pourront être utilisées, ou bien seulement celle de la dernière cellule.

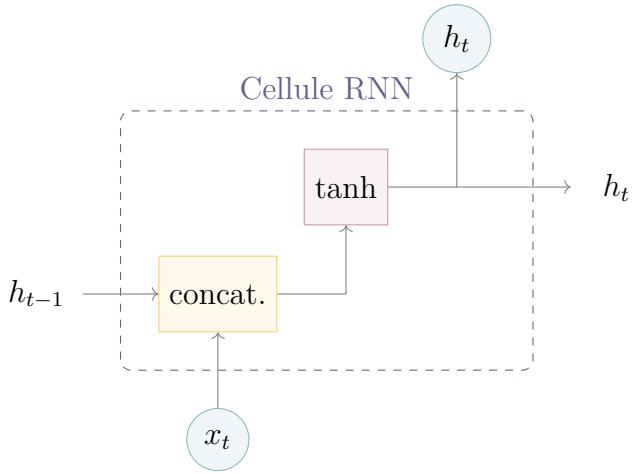


FIGURE 1.4 – Schéma conceptuel d'une cellule d'un RNN.

Un schéma d'une cellule d'un RNN est présent en FIGURE 1.4. Des matrices de poids sont utilisées pour transformer les entrées de la cellule x_t et h_{t-1} et la représentation de sortie h_t, grâce à un produit matriciel. Ce sont ces paramètres qui sont appris par le modèle.

Le LSTM (*Long Short-Term Memory*) est une architecture récurrente proposée par HOCHREITER et SCHMIDHUBER (1997), qui permet, par rapport à un RNN standard, de mieux conserver l'information au fil du temps (***long*** *short-term memory*). Pour cela, le LSTM utilise des «portes» qui permettent de contrôler le flux de l'information le long de la séquence, à l'aide d'une porte d'entrée, de sortie, et d'oubli. Grâce à la porte d'oubli, le neurone peut moduler l'ajout de nouvelles informations et la conservation de l'historique. Nous présentons en FIGURE 1.5 une cellule type d'un LSTM. La mémoire des cellules se propage grâce au vecteur C $(C_0, \ldots, C_{t-1}, C_t)$. En plus de la fonction d'activation tanh, la fonction sigmoïde est utilisée (σ).

Les GRU (*Gated Recurrent Unit*) sont un autre type de cellules utilisables pour construire un RNN. Ceux-ci sont plus légers que les LSTM, car ils ne contiennent que deux

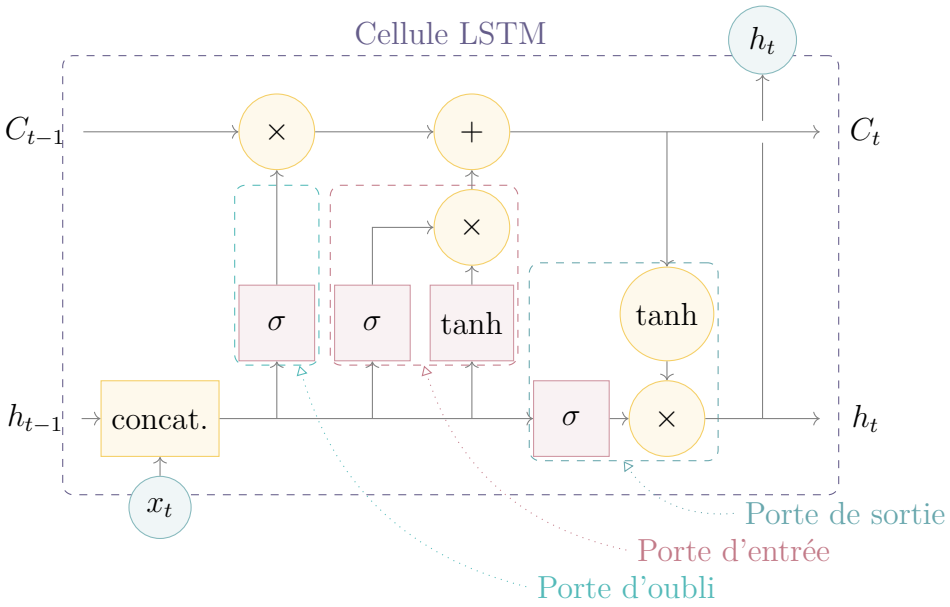


FIGURE 1.5 – Schéma conceptuel d'une cellule d'un LSTM.

portes, une de remise à zéro, et une de mise à jour. Ils ont été introduits plus récemment par CHO et al. (2014).

Une architecture encodeur-décodeur. Certaines tâches nécessitent de générer en sortie des séquences de longueur variables, à partir d'une séquence d'entrées elle-même de longueur variable. Cette tâche, dénommée «séquence vers séquence», est commune à de nombreux domaines : reconnaissance de la parole, reconnaissance de caractères, traduction automatique, etc. Pour, cela, une architecture encodeur-décodeur est souvent employée. Un encodeur est construit à partir d'une ou plusieurs couches récurrentes. Un décodeur est, lui aussi, construit à partir d'une ou plusieurs couches récurrentes. Le vecteur d'état caché (h_t) de la dernière cellule de l'encodeur est conservé, et utilisé en entrée de la première cellule du décodeur. Toute l'information est donc encodée dans un vecteur de taille fixe, et le décodeur le décode de façon inverse pour produire la séquence attendue. Pour réduire la perte d'information au sein du vecteur encodé, l'encodeur peut être transformé en encodeur bidirectionnel. Pour cela, la séquence est lue dans un sens, puis dans l'autre, et les deux vecteurs résultants sont concaténés. Le premier modèle encodeur-décodeur neuronal moderne a été présenté par CHO et al. (2014). Nous présentons en FIGURE 1.6 un schéma d'une telle architecture encodeur-décodeur utilisant des réseaux récurrents. Dans ce schéma, trois sorties sont générées à partir de quatre entrées. Les vecteurs h_0^E et h_0^D, correspondant respectivement à l'encodeur et au décodeur, sont typiquement initialisés

avec des zéros.

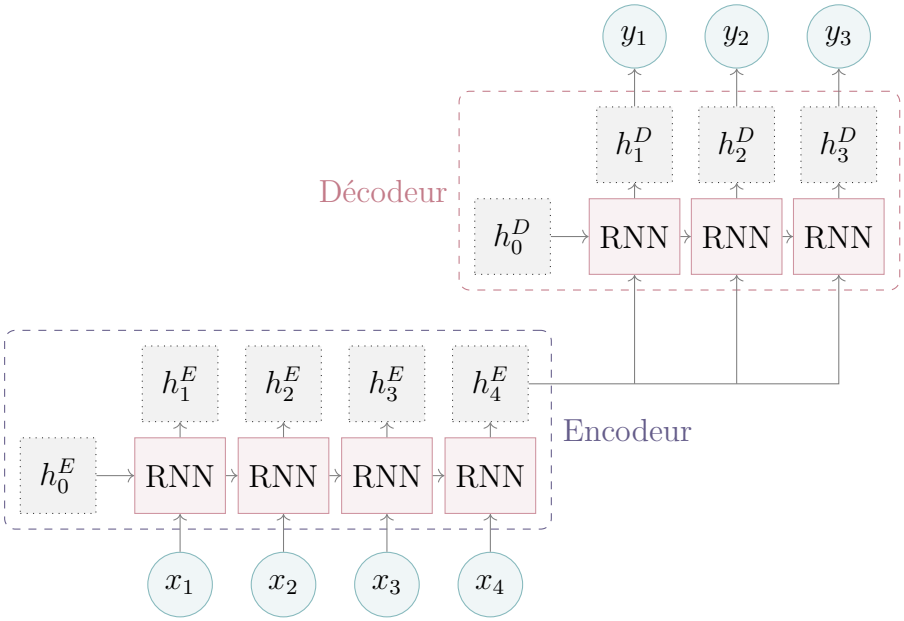


FIGURE 1.6 – Schéma conceptuel d'un encodeur-décodeur utilisant des cellules RNN.

L'utilisation de LSTM ou de GRU permet de mieux conserver l'historique lors de longues séquences par rapport à un RNN classique. Cependant, avec de très longues séquences, même ces cellules perdent de l'information. Cela crée un goulot d'étranglement, où toutes les informations doivent être compressées dans un vecteur de taille fixe, et les résultats, en fonction de la tâche, peuvent se trouver dégradés. Toujours en utilisant les RNN, deux courants existent afin de contrer cet effet : le CTC (section 1.1.3) et les mécanismes d'attention (section 1.1.4). Cependant, les deux courants ne sont pas exclusifs, et dans la littérature, certains modèles combinent les deux approches dans un mode dit *hybride* (WATANABE et al. 2017 ; DESOT et al. 2019).

1.1.3 *Connectionist Temporal Classification*

Le CTC (*Connectionist Temporal Classification*) désigne un algorithme permettant de déterminer l'alignement entre une séquence d'entrée X d'un modèle et une séquence de sortie Y. La longueur de X doit être supérieure ou égale à celle de la séquence de sortie Y. Cet algorithme, introduit par GRAVES et al. (2006), repose sur la prédiction d'un caractère en sortie pour chaque entrée. La tâche «séquence vers séquence» est de cette façon transformée en une tâche de classification. Le CTC fusionne ensuite les sorties dupliquées entre elles. Pour autoriser deux sorties identiques à la suite (par exemple, dans le cas de la

reconnaissance de la parole, le caractère `r` est double dans le mot `r·é·c·u·r·r·e·n·t`), une sortie additionnelle est ajoutée au système. Toujours dans le cas de la reconnaissance de caractères, un caractère vide, noté ϵ, est employé. Un exemple est donné en FIGURE 1.7, avec la séquence de sortie du modèle correspondant à `r·r·é·é·é·`ϵ`·c·c·u·r·r·`ϵ`·r·e·e·e·e·n·t·t`. Une fois l'algorithme du CTC appliqué, le mot `r·é·c·u·r·r·e·n·t` est retrouvé.

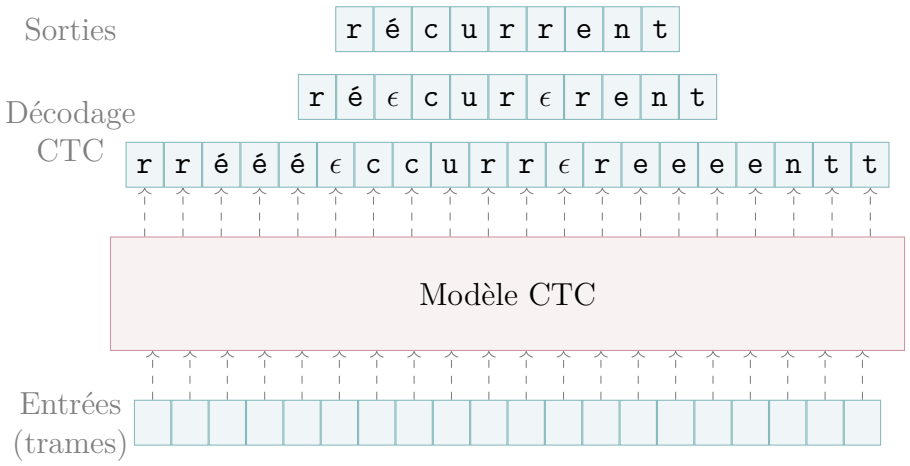


FIGURE 1.7 – Principe du décodage CTC, avec un exemple pour la transcription de la parole.

Le CTC désigne également la fonction de coût employée pour apprendre un RNN utilisant cet algorithme CTC. Cette fonction de coût doit calculer l'ensemble des alignements possibles. Ce calcul est optimisé grâce à un algorithme de programmation dynamique (GRAVES et al. 2006), de façon similaire à l'algorithme *forward-backward* employé pour les modèles de Markov cachés (présentés en section 2.2.2 du chapitre suivant).

1.1.4 Les mécanismes d'attention

Les mécanismes d'attention permettent eux aussi de réduire le goulot d'étranglement des modèles encodeur-décodeur. Pour cela, le mécanisme d'attention est ajouté entre l'encodeur et le décodeur. Le mécanisme d'attention neuronal, proposé par BAHDANAU et al. (2015) pour la traduction automatique, permet au modèle de réaliser un alignement entre la séquence de vecteurs encodée et la séquence décodée. Soient x_j le $j^{\text{ème}}$ élément de la séquence d'entrée X, et y_i le $i^{\text{ème}}$ élément de la séquence décodée de sortie Y. Le but est de trouver le score d'alignement entre x_j et y_i. En suivant cette notation, h_j^E correspond au vecteur d'état caché de l'encodeur au moment j. Pour chaque sortie i, un vecteur de contexte c_i est calculé, comme la somme pondérée entre tous les vecteurs encodés h_j^E et

un poids d'alignement entre j et i :

$$c_i = \sum_{j=1}^{|X|} \alpha_{ij} h_j^E \tag{1.2}$$

Ce poids d'alignement α_{ij} est calculé avec la SOFTMAX du score d'alignement, lui-même obtenu avec un réseau de neurones à propagation avant (NN-ALIGNEMENT) :

$$\alpha_{ij} = \text{SOFTMAX}\left(\text{NN-ALIGNEMENT}(h_{i-1}^D, h_j^E)\right) \tag{1.3}$$

Ce réseau de neurones NN-ALIGNEMENT prend en entrée l'état caché du décodeur au moment précédent (h_{i-1}^D), et l'état caché actuel de l'encodeur (h_j^E). L'encodeur, le mécanisme d'attention et le décodeur sont appris ensemble, grâce à la fonction d'optimisation du réseau de neurones.

In fine, le décodeur génère les sorties à l'instant i à partir du vecteur de contexte c_i et de la sortie cachée h_{i-1}^D. Nous présentons en FIGURE 1.8 une schématisation d'un modèle encodeur-décodeur avec mécanisme d'attention. Dans l'exemple de ce schéma, trois sorties sont générées à partir de quatre entrées.

Contrairement au CTC (section 1.1.3), l'alignement produit par un mécanisme d'attention n'est pas monotonique : à l'instant i, il est possible d'*attendre* n'importe quel élément j. Dans le cas du CTC, l'alignement entre l'entrée X et la sortie Y conserve l'ordre. De plus, l'alignement fourni par un mécanisme d'attention n'est pas strict, une sortie x_j peut être alignée à plusieurs entrées y_i, de façon pondérée.

Ces deux caractéristiques sont particulièrement utiles dans le cas de la traduction automatique : entre deux langues, l'alignement n'est pas toujours monotonique, et plusieurs mots peuvent correspondre à un seul dans une autre langue, ou inversement. Par exemple : «*le chien blanc est à côté de la piscine*» se traduit en anglais par «*the white dog is next to the swimming pool*». Le mot «piscine» se traduit en deux mots «swimming pool», tandis que l'ordre des mots «chien blanc» est inversé en anglais. Au premier abord, ces caractéristiques ne sont pas nécessairement utiles pour la transcription de la parole, où les mots suivent forcément l'ordre de la parole. Cependant, des résultats ont montré l'intérêt de cet alignement, plutôt que celui du CTC (BAHDANAU et al. 2016).

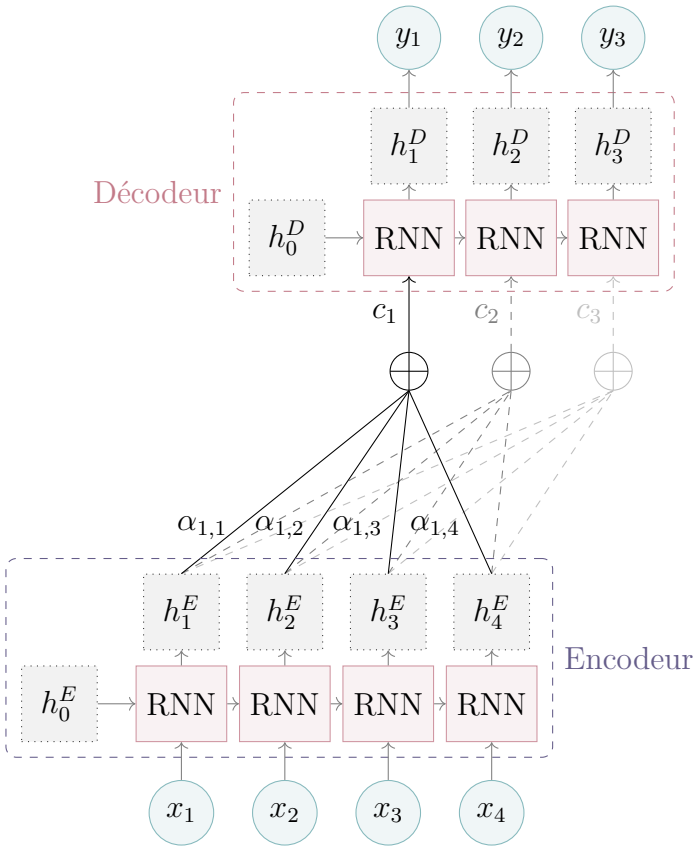


FIGURE 1.8 – Schéma conceptuel d'un encodeur-décodeur avec mécanisme d'attention.

1.1.5 Les transformers

Attention is all you need

VASWANI et al. (2017)

En traduction automatique, les systèmes état de l'art ont rapidement été conquis par les modèles encodeur-décodeur avec des mécanismes d'attention. Une architecture, nommée le *transformer*, proposée par VASWANI et al. (2017), est venue initier un nouveau courant dans le domaine de l'apprentissage profond. Dans cette architecture, l'encodeur comme le décodeur sont composés uniquement de modules d'attention empilés, et de couches linéaires. Ces encodeurs et décodeurs ne contiennent plus de RNN. Cette méthode de construction de modèles sans RNN, avec principalement que des mécanismes d'attention avait été utilisée auparavant par PARIKH et al. (2016) pour une tâche d'implication textuelle entre un texte et une hypothèse.

Dans le transformer, l'encodeur et le décodeur sont remplacés par de multiples couches

d'attention, dites de *self-attention*. Les mécanismes vus en section 1.1.4 portent leur attention sur des séquences provenant d'autres modules : au moment du décodage, ils portent sur la séquence encodée. La self-attention fut introduite par Yang et al. (2016) pour une tâche de classification de documents et Cheng et al. (2016) pour de nombreuses tâches, telles que la modélisation de la langue. Celle-ci porte l'attention sur la séquence elle-même, provenant du même réseau. Dans le transformer original, des modules de self-attention sont utilisés dans les couches de l'encodeur, et dans les couches du décodeur. Des modules d'attention standard, également appelés *cross-attention*, sont également utilisés entre l'encodeur et le décodeur.

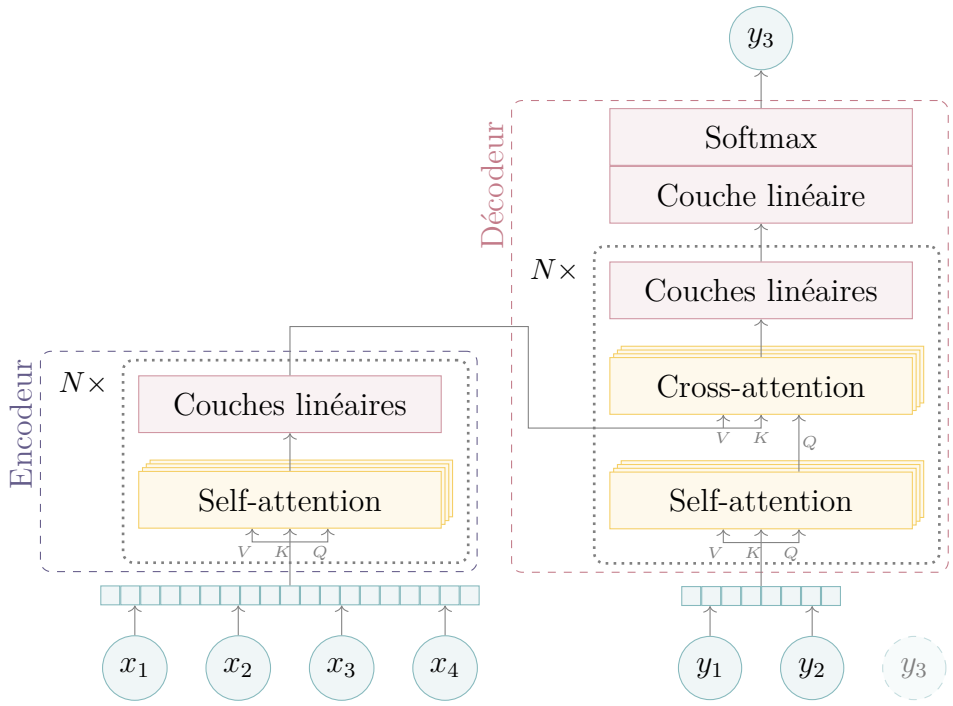


Figure 1.9 – Schéma conceptuel d'un modèle transformer encodeur-décodeur générant une séquence à partir de quatre éléments d'entrée.

Nous présentons en Figure 1.9 un schéma d'exemple simplifié du transformer de Vaswani et al. (2017). Dans celui-ci, les modules composés de self-attention, de cross-attention, de couches linéaires de l'encodeur et du décodeur sont répétés N fois ($N = 6$ pour Vaswani et al. (2017)). De plus, ces modules de self et cross-attention sont dupliqués et mis en parallèle dans ce qui s'appelle des têtes d'attention. Chaque tête d'attention permet de porter l'attention à des informations représentées dans des sous-espaces différents. Le décodeur fonctionne de façon itérative, et prend en entrée les dernières sorties générées (y_1 et y_2 au moment de générer y_3 par exemple). Toutes les entrées de ces modèles

sont représentées à travers des *embeddings* contextuels et positionnels : contextuels, car l'*embedding* d'un mot est mis en contexte par rapport aux autres mots dans la phrase, et positionnels, car la position du mot dans la phrase est présente dans cet *embedding*.

L'avantage majeur par rapport aux encodeurs/décodeurs récurrents est au niveau calculatoire. Avec un RNN, il est impossible de paralléliser le calcul, du fait du traitement séquentiel des données. Avec un module de self-attention, tous les calculs peuvent être faits indépendamment, et donc parallélisés, pour réduire le temps d'apprentissage. En contrepartie de cette optimisation du temps d'apprentissage, cela a permis à la communauté scientifique de créer des modèles transformers avec un plus grand nombre de paramètres, et ainsi d'améliorer les résultats dans de nombreux domaines (Devlin et al. 2019 ; Raffel et al. 2020 ; Baevski et al. 2020b ; Babu et al. 2022 ; Scao et al. 2022 ; Touvron et al. 2023a), y compris en dehors du TAL (Khan et al. 2022).

La notation *Query*, *Key*, *Value* est régulièrement employée pour généraliser et formaliser le calcul des mécanismes d'attention. Avec ce formalisme, Q, K et V correspondent à trois vecteurs en entrée d'un module d'attention. Les scores obtenus permettent de pondérer les *values* (V) à partir de la correspondance entre la *query* (Q) et les *keys* (K). La sortie correspond à la somme pondérée des valeurs avec une fonction de compatibilité entre la *query* et la *key*. Cette somme pondérée est souvent calculée comme un produit scalaire pour réaliser l'opération plus rapidement sur GPU (éq. 1.4).

$$\begin{aligned} \text{att}(Q, K, V) &= \sum V \times \text{compatibilité}(Q, K) \\ &= V \cdot \text{compatibilité}(Q, K) \end{aligned} \tag{1.4}$$

- Dans le cas du mécanisme d'attention standard de Bahdanau et al. (2015), présenté en section 1.1.4, on peut reformuler le calcul de la sorte :

$$\begin{aligned} Q &= h_{i-1}^{D} \\ K &= h_{j}^{E} \\ V &= h_{j}^{E} \\ \text{compatibilité}(Q, K) &= \textsc{softmax}\left(\textsc{nn-alignement}(Q, K)\right) \end{aligned} \tag{1.5}$$

- Dans le cas de la self-attention de Vaswani et al. (2017), la méthode utilisée est le *scaled dot-product*. Q, K et V sont formés à partir de la même représentation d'une

séquence (H^E), en utilisant trois matrices de transformation apprises par le modèle (W^q, W^k, et W^v) :

$$\begin{aligned} Q &= H^E \times W^q \\ K &= H^E \times W^k \\ V &= H^E \times W^v \\ \text{compatibilité}(Q, K) &= \textsc{softmax}\left(\frac{QK^T}{\sqrt{|K|}}\right) \end{aligned} \tag{1.6}$$

- Dans le cas de la cross-attention de Vaswani et al. (2017), l'information de la *query* provient de la représentation d'une autre séquence (ici, du décodeur : H^D) :

$$\begin{aligned} Q &= H^D \times W^q \\ K &= H^E \times W^k \\ V &= H^E \times W^v \\ \text{compatibilité}(Q, K) &= \textsc{softmax}\left(\frac{QK^T}{\sqrt{|K|}}\right) \end{aligned} \tag{1.7}$$

Vaswani et al. (2017) emploient plusieurs têtes d'attention (*multi-head attention*), qui correspondent à plusieurs modules d'attention indépendants, réalisés en parallèle, et concaténés. Cette technique permet d'augmenter la capacité de traitement du réseau pour les représentations en entrée. Afin de limiter la complexité, et le temps de calcul, ils réduisent les dimensions des représentations cachées du modèle.

1.2 Méthodes d'apprentissage

L'apprentissage de réseaux de neurones se fait à partir de jeux de données, qui tendent à devenir de plus en plus grands. L'apprentissage se fait soit à partir de jeux de données avec annotation (apprentissage supervisé) ou sans (apprentissage non supervisé). Il est cependant possible de réaliser un apprentissage auto-supervisé, en appliquant des méthodes d'apprentissage supervisé à des corpus sans annotations.

Depuis l'invention du réseau de neurones, de nombreuses méthodes ont été utilisées afin de les apprendre. Cependant, le fonctionnement général de cet apprentissage est resté inchangé. Ces nouvelles méthodes permettent d'améliorer les résultats, de réduire les temps d'apprentissage, ou d'optimiser l'emploi des ressources de calcul utilisées.

1.2.1 L'apprentissage supervisé

L'apprentissage supervisé consiste à apprendre un modèle en lui donnant des couples (entrée, sortie) du corpus d'apprentissage. Le modèle doit alors prédire la sortie à partir de cette entrée. L'erreur du modèle sur ces exemples est alors utilisée pour ajuster les paramètres de celui-ci. Cette technique d'apprentissage est à la base de nombreuses tâches en apprentissage automatique et en TAL notamment. Nous pouvons citer la classification de documents : prédire une classe à partir de *features* d'entrée. La reconnaissance de la parole utilise également l'apprentissage supervisé : déterminer une séquence de caractères ou de mots à partir d'un signal de parole.

1.2.2 L'apprentissage auto-supervisé

Apprendre de larges modèles, tel que les transformers nécessite une grande quantité de données. Cela est surtout le cas avec les modèles comportant un nombre de paramètres important. Cependant, dans de nombreux domaines, la quantité de données annotées pour apprendre ces modèles pour une tâche en particulier n'est pas à la hauteur de la taille de ces modèles. En revanche, il existe souvent de larges corpus non annotés. Nous pouvons par exemple citer CommonCrawl [1] qui contient plus de 428To de texte, ou Libri-Light (Kahn et al. 2020) qui contient 60k heures d'audio non transcrites. Afin de faire usage de ces connaissances supplémentaires, un pré-apprentissage auto-supervisé peut être employé. Un premier modèle est généralement appris avec les données sans annotation, puis ce modèle est *finetuné* sur les données avec annotations correspondantes à la tâche cible. Le *finetuning* est présenté en section 1.2.4.

À première vue, l'apprentissage d'un modèle à partir de données non annotées, c'est-à-dire sans annotation ni tâche concrète à réaliser, peut paraître déroutant. Cependant, en TAL, des tâches ont été inventées afin de contourner le problème. Il s'agit généralement de «cacher» un élément de la séquence d'entrée, au hasard, et de demander au modèle de prédire cet élément manquant. En texte, cela correspond au MLM (*Masked Language Modeling*), qui consiste à prédire un mot caché au préalable dans la phrase. Pour l'audio, la même procédure est appliquée sur un court morceau du signal en entrée. Ces techniques d'apprentissage sont résumées à travers l'acronyme SSL (*Self-Supervised Learning*). Un modèle SSL est ainsi un modèle ayant été (pré)-appris sur une grande masse de données avec une technique non supervisée.

Nous présentons en Figure 1.10 un exemple d'apprentissage d'un modèle SSL en mode

1. `https://commoncrawl.org/`

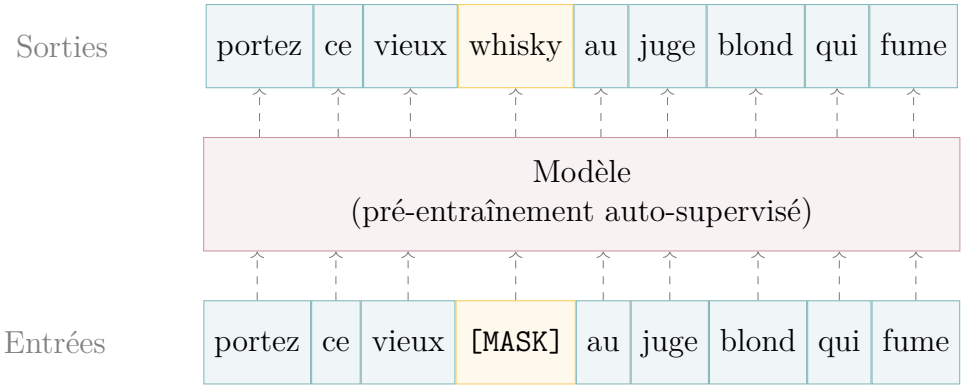


FIGURE 1.10 – Principe du pré-apprentissage auto-supervisé (SSL).

texte, avec la phrase «portez ce vieux whisky au juge blond qui fume». Le mot «whisky» a été caché – remplacé par l'unité `[MASK]` –, et le modèle doit le retrouver.

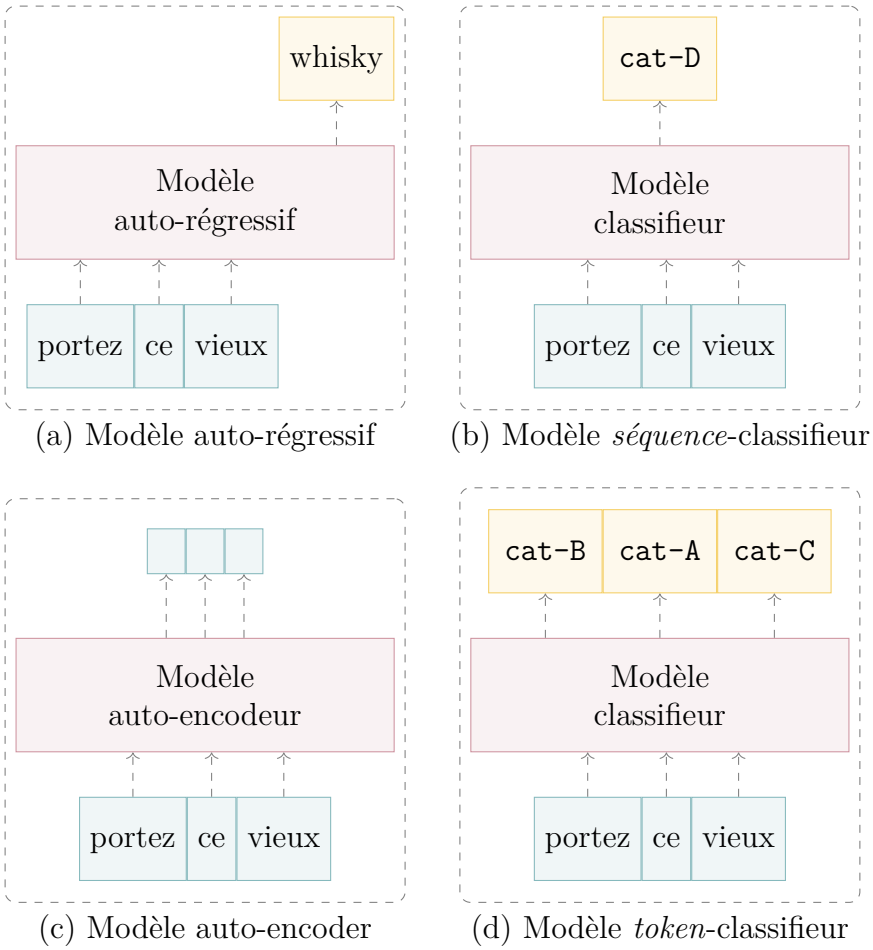
Une fois un tel modèle pré-appris, il faut le *finetuner* pour une tâche souhaitée. Cela consiste généralement à modifier le modèle, puis à reprendre l'apprentissage sur les données annotées.

En fonction des tâches, il est possible de modifier l'architecture même du modèle afin de produire une sortie adéquate avec l'objectif. L'architecture peut ainsi être configurée pour de la classification, de la génération d'éléments, ou de la génération de représentations cachées par exemple. Nous présentons en FIGURE 1.11 quatre exemples de configuration et d'utilisation d'un modèle SSL une fois la phase de pré-apprentissage auto-supervisé terminée.

Le modèle auto-régressif (a), également parfois appelé modèle *causal* est utilisé notamment par les modèles de langage génératifs pour compléter du texte. À partir d'une séquence d'éléments donnés, le modèle génère l'élément suivant. En exécutant cette étape plusieurs fois, il est possible de générer de longues séquences. Cette configuration est par exemple utilisée pour la traduction de textes par BROWN et al. (2020).

Le modèle de classification de séquence (b) catégorise simplement toute la séquence en suivant le schéma de classification utilisé pendant l'apprentissage. DEVLIN et al. (2019) utilisent des modèles SSL de la sorte pour une tâche de classification de phrases.

Un modèle auto-encodeur (c) est généralement utilisé conjointement avec un deuxième modèle, un décodeur. L'auto-encodeur génère une représentation de la séquence d'entrée dans un espace latent, qui sera utilisée par le décodeur pour générer une sortie, généralement une séquence. C. WANG et al. (2021b) utilisent par exemple un modèle auto-encodeur pour la tâche de traduction automatique de la parole, avec un modèle transformer comme



(a) Modèle auto-régressif (b) Modèle *séquence*-classifieur

(c) Modèle auto-encoder (d) Modèle *token*-classifieur

FIGURE 1.11 – Quatre exemples de configuration possibles pour les modèles SSL une fois la phase de pré-apprentissage terminée. Les exemples portent sur un modèle prenant du texte comme entrée (modèle de langage).

décodeur.

Enfin, il est possible de construire un modèle de classification de chacun des éléments de la séquence d'entrée (d). Ce modèle, *token*-classifieur, est par exemple utilisé par DEVLIN et al. (2019) pour la reconnaissance d'entités nommées.

1.2.3 Techniques d'optimisation

La fonction de coût

Le fonctionnement de l'apprentissage des modèles neuronaux repose en premier lieu sur le calcul d'une fonction de coût (*loss*). Ce calcul peut se voir comme une note attribuée au modèle pour ses erreurs. La particularité de cette note, par rapport à n'importe

quelle métrique d'évaluation, réside dans la capacité à appliquer un algorithme de rétro-propagation (RUMELHART et al. 1986) sur le modèle à partir de celle-ci. Pour cela, la fonction de coût est une fonction dérivable par rapport aux paramètres du modèle. Un optimiseur est utilisé pour modifier ces paramètres afin de minimiser (ou maximiser) cette fonction.

Parmi les fonctions de coût les plus communes, nous pouvons citer la MSE (*Mean Squared Error*, éq. 1.8), la CROSSENTROPY (éq. 1.9) et la fonction de coût CTC (*Connectionist Temporal Classification*, éq. 1.10) :

$$\ell_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2 \tag{1.8}$$

$$\ell_{\text{CrossEntropy}} = \frac{1}{N}\sum_{i=1}^{N}\left(-\sum_{c=1}^{C} P_{ic}\log Q_{ic}\right) \tag{1.9}$$

$$\ell_{\text{CTC}} = -\sum_{i=1}^{N}\log p(Z_i \mid X_i) \tag{1.10}$$

avec Y_i la sortie attendue et $\hat{Y}_i$ la prédiction pour l'élément i (N éléments à évaluer au total), P_{ic} la probabilité attendue pour la classe c (distribution *one-hot*) pour l'élément i et Q_{ic} celle donnée par le modèle (provenant d'une SOFTMAX par exemple). Pour la CROSSENTROPY, on considère C classes. Pour la CTC, $p(Z_i \mid X_i)$ correspond à la probabilité de la séquence finale Z pour l'élément i, sachant l'entrée X_i (voir GRAVES et al. (2006)). Il est important de noter que ces trois fonctions de coût ne s'appliquent pas pour les mêmes tâches. La MSE correspond à une fonction de coût pour une tâche de régression, la CROSSENTROPY pour une tâche de classification, et la CTC pour une tâche séquence-vers-séquence.

Algorithmes d'optimisation

Nous présentons ici le fonctionnement des algorithmes d'optimisation, appliqués lors de la phase de rétro-propagation.

La descente de gradient est la méthode utilisée par RUMELHART et al. (1986) pour l'optimisation automatique des réseaux de neurones. L'algorithme est itératif, et repose sur la modification des paramètres (W) pour aller dans le sens inverse au gradient (noté ∇W). Un taux d'apprentissage, ou *learning rate*, noté λ, est utilisé pour contrôler la vitesse de déplacement des paramètres du modèle selon ce gradient. Les nouveaux paramètres du

modèle sont notés W', et calculés selon l'équation 1.11.

$$W' = W - \lambda \times \nabla W \tag{1.11}$$

Le gradient est calculé grâce à la dérivée partielle première de la fonction de coût (ℓ) par rapport aux paramètres du modèle :

$$\nabla W = \begin{bmatrix} \frac{\partial \ell}{\partial w_1} \\ \frac{\partial \ell}{\partial w_2} \\ \vdots \\ \frac{\partial \ell}{\partial w_n} \end{bmatrix} \tag{1.12}$$

D'autres algorithmes sont utilisés pour apprendre les modèles, la descente de gradient n'étant pas la plus efficace. En effet, pour fonctionner, nous devons calculer l'erreur du modèle sur l'ensemble du corpus d'apprentissage, avant de procéder à la mise à jour du modèle. De nos jours, avec des corpus conséquents, cette technique n'est plus envisageable. Une variante de la descente de gradient repose sur une mise à jour des paramètres du modèle pour chaque exemple du corpus d'apprentissage, appelée descente de gradient stochastique (SGD). Une autre réalise la descente de gradient sur des sous-ensembles du corpus d'apprentissage, appelés *batch*, ou lot en français. AdaGrad, AdaDelta, Adam et AdamW sont d'autres techniques d'optimisation couramment utilisées. Leur fonctionnement général reste le même : réduire les erreurs du modèle en minimisant la fonction de coût, grâce à la réduction du gradient.

Techniques de régularisation. Le sur-apprentissage est un effet dans lequel le modèle mémorise *par cœur* les exemples d'apprentissage. Le modèle perd sa capacité à généraliser sur de nouveaux exemples : les performances du modèle sur le jeu d'apprentissage (Train) continuent de s'améliorer, mais celles-ci se dégradent sur les jeux de développement (Dev) et d'évaluation (Test). Des techniques de régularisation peuvent être mises en place pour prévenir le sur-apprentissage. La méthode la plus évidente repose sur l'*early stopping*, ou arrêt prématuré. Elle vise à arrêter l'apprentissage du modèle lorsque les performances sur le jeu Dev se dégradent. Les normalisations L_1 et L_2, ainsi que le *weight decay* peuvent être employées comme régularisation (Srivastava et al. 2014 ; Loshchilov et Hutter 2019). Un terme de pénalité est ajouté sur la fonction de coût pour contraindre

certains paramètres à ne pas être utilisés (L_1) ou limiter l'amplitude numérique de ces paramètres (L_2). Le *dropout* est une technique de régularisation visant à «abandonner» certains neurones de façon aléatoire et temporaire durant l'apprentissage (Srivastava et al. 2014).

1.2.4 Apprentissage *from scratch*, *finetuning* et x-*shot*

Comme nous venons de le voir jusqu'à présent, apprendre un modèle repose sur la modification des poids ou paramètres de celui-ci, afin de réduire les erreurs réalisés sur un corpus. Ces paramètres sont initialisés aléatoirement à la création du modèle. Un tel modèle est dit appris de zéro, ou *from scratch*. Cependant, pour réduire le temps d'apprentissage, réduire les coûts, et améliorer les résultats, un modèle peut également être simplement un *finetuning* (optimisation) d'un autre modèle. Pour cela, les paramètres ne sont pas initialisés aléatoirement, mais sont repris d'un modèle ayant été appris pour une tâche similaire ou sur un ou des corpus différents. C'est ce *finetuning* qui permet par exemple aux modèles SSL d'être utilisés pour d'autres tâches : transcription de la parole, génération de texte, classification, etc.

Néanmoins, il est parfois possible d'utiliser un modèle sur un nouveau jeu de données, sans repasser par une étape d'apprentissage ou *finetuning*. Différents termes similaires sont utilisés pour décrire ces processus :

- L'évaluation *zero-shot* consiste à évaluer un modèle directement sur des données nouvelles d'un domaine ou d'une langue différente, sans réaliser de *finetuning* du modèle.
- Un apprentissage *zero-shot* (Palatucci et al. 2009) consiste à modifier la tâche, changer de domaine ou ajouter de nouvelles classes à un modèle au moment de l'inférence. Les paramètres du modèle restent donc inchangés. Pour cela, le modèle, dès sa conception, doit faire usage d'informations supplémentaires, qui peuvent être des représentations sémantiques (Palatucci et al. 2009 ; Lampert et al. 2009) ou une description textuelle de la tâche à accomplir (Brown et al. 2020).
- Dans le *one-shot learning*, la description apportée au système est agrémentée d'un exemple avec sa sortie attendue (Brown et al. 2020).
- Le *few-shot learning* consiste à montrer plusieurs exemples (généralement moins d'une dizaine) au modèle au moment de l'inférence (Brown et al. 2020 ; He et Garner 2023 ; Yufan Wang et al. 2023).

Les apprentissages *zero-shot*, *one-shot* et *few-shot* sont particulièrement adaptés lorsque la tâche cible ne dispose pas d'assez de données pour *finetuner* un modèle avec celles-ci

(Lampert et al. 2009 ; Yufan Wang et al. 2023). Ces techniques permettent ainsi de faire appel aux capacités de généralisation des modèles.

1.2.5 Optimisation des hyper-paramètres

Les paramètres des modèles neuronaux – c.-à-d. les poids des différentes couches qui le composent – sont appris automatiquement grâce à la rétro-propagation et l'algorithme d'optimisation. Ces algorithmes d'optimisation comportent des hyper-paramètres : *learning rate* (λ), *weight decay* (w, utilisé par AdamW), epsilon (ϵ), *momentum*, etc. Nous pouvons également citer l'architecture du modèle même comme hyper-paramètre : le nombre de couches, les fonctions d'activation, la taille des représentations cachées, etc. D'autres choix sont également considérés comme des hyper-paramètres : algorithme d'optimisation, fonction de coût, taux de *dropout*, nombre d'époques d'apprentissage, taille des lots (*batch size*), évolution du *learning rate* au fil de l'apprentissage, etc.

Tous ces hyper-paramètres ont un impact non négligeable sur les résultats. Leurs choix doivent ainsi être correctement réalisés. Ils peuvent être déterminés empiriquement, mais devant le nombre de ces paramètres, et l'effet parfois chaotique de ces paramètres sur les résultats, ils peuvent également être optimisés automatiquement. Des librairies telles que Hyperopt (Bergstra et al. 2013) ou Optuna (Akiba et al. 2019) permettent de trouver un jeu d'hyper-paramètres qui minimisent ou maximisent une métrique d'évaluation des modèles.

1.3 Conclusion

Au sein de ce chapitre, nous avons décrit l'apprentissage profond, un sous-ensemble de l'apprentissage artificiel. L'apprentissage profond est utilisé dans des domaines tels que le TAL (Traitement Automatique des Langues). Nous avons tout d'abord décrit différentes architectures neuronales, utilisées pour réaliser des tâches du TAL. Enfin, nous avons décrit les méthodes d'apprentissage de ces modèles, que ce soit en apprentissage supervisé ou auto-supervisé. Les techniques d'optimisation des modèles, et les procédures d'apprentissage en *finetuning* ont été décrites. Les architectures que nous avons décrites dans ce chapitre sont utilisées pour résoudre des problèmes variés. Nous les retrouvons ainsi dans la tâche de reconnaissance automatique de la parole (chapitre 2), mais également pour la compréhension automatique de la parole (chapitre 3). Les contributions apportées par cette thèse utilisent également les techniques abordées au sein de ce chapitre. Ainsi,

dans le chapitre 4 nous produisons des modèles neuronaux récurrents, et dans le chapitre 5 nous utilisons des modèles transformers pré-entraînés de manière auto-supervisée.

Chapitre 2

La reconnaissance automatique de la parole

Sommaire

2.1 Principe général 48
2.2 Modélisation acoustique : $P(X \mid Y)$ 50
2.2.1 Représentations d'entrée 50
2.2.2 Les modèles de Markov cachés 52
2.2.3 Les modèles neuronaux 54
2.3 Modélisation de la langue : $P(Y)$ 55
2.3.1 Modélisation n-grammes 57
2.3.2 Modélisation neuronale 58
2.3.3 *Tokenization* du texte 59
2.4 Décodage, prédiction et évaluation 60
2.4.1 Intégration du modèle de langage au modèle acoustique 60
2.4.2 Décodage : glouton ou faisceau ? 61
2.4.3 Évaluation des transcriptions 62
2.5 Corpus 64
2.6 Conclusion 66

La compréhension automatique de la parole est régulièrement réalisée dans un deuxième temps, après avoir réalisé un premier traitement visant à transcrire textuellement le contenu du signal audio. Néanmoins, lorsque la tâche de compréhension de la parole est faite conjointement à celle de reconnaissance, la méthode mise en œuvre s'approche généralement de celles employées pour la simple reconnaissance de la parole. Nous présentons au sein de ce chapitre méthodes et données employées pour réaliser une transcription automatique de la parole. En section 2.1 nous exposons la problématique de la reconnaissance de la parole. En section 2.2 nous présentons les modèles permettant de prédire une transcription à partir de la parole, mais également ses représentations utilisées par les systèmes. En section 2.3 nous présentons la modélisation du langage, qui permet d'améliorer les résultats des modèles de transcription. En section 2.4 nous décrivons l'intégration de ces deux modèles ensemble, la procédure de décodage ainsi que d'évaluation. Enfin, en section 2.5 nous détaillons les corpus usuels de reconnaissance de la parole.

2.1 Principe général

La reconnaissance automatique de la parole (ASR pour *Automatic Speech Recognition* en anglais), consiste à convertir un signal audio contenant de la parole en une séquence textuelle. En Figure 2.1 nous présentons le principe général de fonctionnement d'un

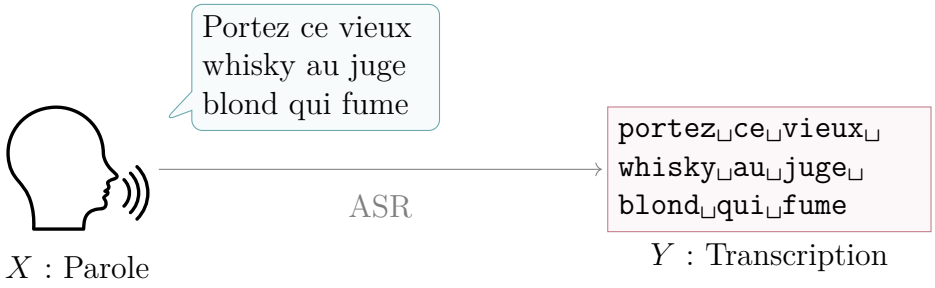


Figure 2.1 – Principe général d'un système de reconnaissance automatique de la parole.

système de reconnaissance de la parole. L'exemple est donné pour une phrase lambda, mais l'ASR peut s'appliquer sur différents types de parole. Le vocabulaire employé (fermé, ouvert), le locuteur (un seul locuteur, plusieurs locuteurs), la langue, l'accent, l'intonation et le format employé (lecture, parole spontanée) ont des influences sur la difficulté de la tâche à résoudre.

Le signal audio peut provenir d'un fichier audio brut (par exemple un fichier WAV), et la séquence textuelle est représentée par une séquence de caractères ou de mots par

exemple. L'ASR consiste donc à transformer une séquence X (parole), de longueur $|X|$, en une séquence Y (texte) de longueur $|Y|$. X et Y sont en principe de longueur différente, avec $|X| \gg |Y|$. La reconnaissance de la parole a donc pour but de trouver la meilleure séquence $\hat{Y}$, c'est-à-dire celle ayant la probabilité la plus forte étant donné X (éq. 2.1). En appliquant le théorème de Bayes (éq. 2.2), et en enlevant $P(X)$ – qui est constant pour $\arg\max_Y$ – nous pouvons transformer la problématique en deux probabilités distinctes, telles qu'indiquées en équation 2.3.

$$\hat{Y} = \arg\max_Y P(Y \mid X) \tag{2.1}$$

$$= \arg\max_Y \frac{P(X \mid Y) \cdot P(Y)}{P(X)} \tag{2.2}$$

$$\equiv \arg\max_Y P(X \mid Y) \cdot P(Y) \tag{2.3}$$

$P(X \mid Y)$ correspond à la modélisation acoustique, c'est à dire la probabilité d'apparition de X quand la séquence Y est observée. $P(Y)$ correspond à la probabilité de voir la séquence Y apparaître dans la langue, il s'agit d'une probabilité obtenue avec un modèle de langage. Pour faciliter les calculs, et réduire les erreurs d'approximation liées aux multiplications de valeurs proches de zéro, le logarithme est régulièrement utilisé :

$$\log\left(P(Y \mid X)\right) \equiv \log\left(P(X \mid Y)\right) + \log\left(P(Y)\right) \tag{2.4}$$

Bref historique. Le tout premier système de reconnaissance de la parole, présenté par K. H. Davis et al. (1952), avait pour unique objectif de reconnaître les chiffres du système décimal, en anglais, pour un seul locuteur. Au fil du développement des méthodes d'ASR, la tâche a été en mesure de se complexifier : vocabulaire moins restreint, plusieurs locuteurs, plusieurs langues, parole spontanée. Le système dragon, présenté par Baker (1975), faisait usage de modèles de Markov cachés ou HMM (*Hidden Markov Model*). Jelinek (1976) a popularisé l'approche statistique (éq. 2.1) pour la transcription de la parole, avec un vocabulaire de 250 mots pour de la parole continue. De Mori (1979) présente les avancées des travaux en reconnaissance de la parole jusqu'en 1977. Les modèles de mélanges de gaussiennes – *Gaussian Mixture Model* ou GMM – (Gauvain et Lee 1994) ont également été utilisés (Lee 1988 ; Gauvain et al. 2002). Enfin, les réseaux de neurones ont été utilisés dans des architectures hybrides HMM (Bourlard et Morgan 1989 ; Seide et al. 2011 ; Dahl et al. 2012), mais également dans des architectures purement

neuronales (Landauer et al. 1987 ; Watrous et Shastri 1987 ; Graves et al. 2006, 2013 ; Amodei et al. 2016 ; Baevski et al. 2020b ; Radford et al. 2023).
Bourlard et Morgan (1994) présentent une vue d'ensemble des premières méthodes neuronales pour la reconnaissance de la parole.

2.2 Modélisation acoustique : $P(X \mid Y)$

Le modèle acoustique, introduit par Bahl et al. (1983) comme *acoustic processor* transforme le signal de parole X en séquence d'éléments Y d'un alphabet $\mathcal{A}$. Pour Bahl et al. (1983), il s'agit d'unités phonétiques similaires aux phonèmes, mais pour Lu et al. (2015), il s'agit directement de mots, tandis que pour Eyben et al. (2009) il s'agit d'une séquence de caractères. Ces choix sont bien entendu dépendants de la langue étudiée, notamment de l'unité graphémique de celle-ci (alphabétique, syllabique, logogrammique, etc.). Comme de nombreuses tâches du TAL, la reconnaissance automatique de la parole s'est tout d'abord principalement appliquée à l'anglais, bien que celle-ci ne représente que 380M de locuteurs natifs dans le monde, parmi les 7168 langues en vie dans le monde [1]. Les méthodes les plus ambitieuses en traitement de la parole se limitent aujourd'hui à moins de 2000 langues (Li et al. 2022 ; Pratap et al. 2023).

2.2.1 Représentations d'entrée

La modélisation acoustique nécessite premièrement de représenter le signal de parole informatiquement. Généralement, la parole est enregistrée par un microphone, puis elle est sauvegardée en WAV (*Waveform audio file format*) sur l'ordinateur, grâce à un échantillonnage (temporel) et une quantification (amplitude). Le modèle acoustique peut soit utiliser directement ce signal en entrée, soit une représentation tierce peut être générée.

Le signal de parole X correspond à l'amplitude de l'onde en fonction du temps. Ce signal contient de nombreuses informations redondantes tandis que d'autres informations importantes y sont cachées, telles que les formants, qui permettent de reconnaître certaines voyelles par exemple (Titze 2000). Une représentation fréquentielle plutôt que temporelle permet de mieux reconnaître certains motifs (S. Davis et Mermelstein 1980). Nous présentons en Figure 2.2 un exemple de parole représenté sous forme temporelle et fréquentielle. L'enregistrement correspond aux mots «compréhension de la parole», pour une durée utile de 1.4 seconde environ. La transformée de Fourier ($\mathcal{F}$) est une opération

1. `https://www.ethnologue.com/`

permettant de représenter ce signal en fonction de la fréquence. En traitement du signal, l'algorithme de transformation de Fourier rapide (FFT) est utilisé pour calculer la transformée de Fourier discrète – pour un signal numérique. Cette méthode est à la base des représentations cepstrales utilisées en entrée des systèmes de reconnaissance de la parole. Nous pouvons par exemple citer les Filter-Banks, Mel Filter-Banks ou MFCC (Mel Frequency Cepstrum Coefficients). La représentation fréquentielle du signal est découpée en morceaux appelés trames, qui seront les entrées du modèle. Dans notre exemple Figure 2.2, les trames ont une durée d'environ 23ms. Le nombre de trames par mots dépend de nombreux facteurs tels que la vitesse de parole, les mots prononcés, le locuteur, ou l'émotion par exemple. En moyenne, l'exemple donné contient environ 16 trames par mot.

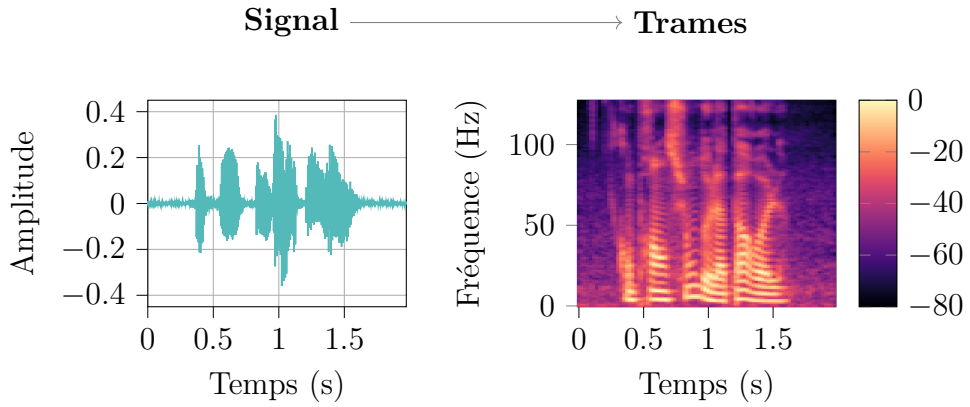


Figure 2.2 – Exemple de signal de parole en temporel (à gauche) et fréquentiel (à droite).

Les *Mel Filter-Banks* ou *banque de filtres* Mel sont un ensemble de filtres passe-bandes, qui ne laissent chacun passer le signal qu'entre une fréquence f_{min} et f_{max}. Les filtres, généralement triangulaires, sont répartis sur une échelle Mel, celle-ci étant une approximation de la perception de l'oreille humaine sur la hauteur des fréquences des sons (Stevens et al. 1937). Un exemple de Mel Filter-Banks avec six filtres est présenté en Figure 2.3.

Mel Frequency Cepstrum Coefficients ou MFCC reposent sur le même principe que les Mel Filter-Banks. Une transformée en cosinus discrète (DCT) puis un logarithme sont appliqués sur les représentations de la banque de filtres Mel. Cette représentation a été introduite par S. Davis et Mermelstein (1980).

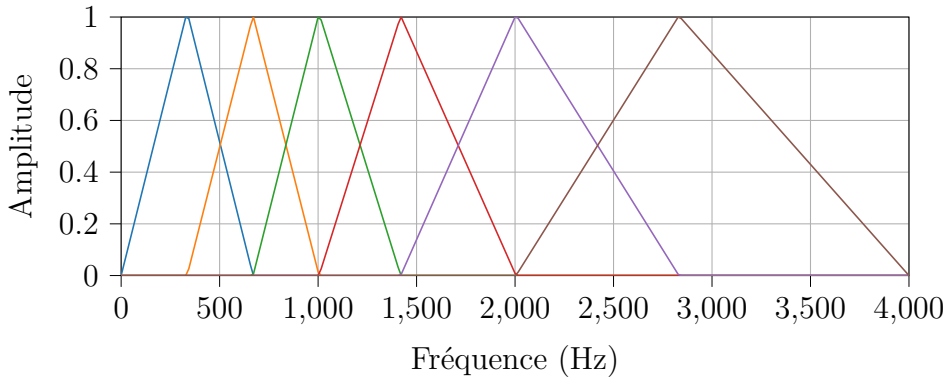


FIGURE 2.3 – Un exemple d'une banque de filtres Mel avec 6 filtres et une fréquence d'échantillonnage de 8kHz.

En résumé, un système d'ASR typique utilise en entrée X :

- soit la représentation temporelle du signal audio ;
- soit une représentation fréquentielle (MEL) FILTER-BANKS, MFCC, etc.

2.2.2 Les modèles de Markov cachés

De nombreuses architectures de reconnaissance de la parole ont été réalisées grâce aux modèles de Markov cachés, ou HMM (*Hidden Markov Model*).

L'objectif d'un HMM pour l'ASR est de modéliser $P(X \mid Y)$, c'est-à-dire la modélisation acoustique de l'équation 2.3. La tâche est vue comme une tâche de classification : assigner une étiquette à chaque trame d'entrée. Ces entrées sont généralement celles présentées dans la section précédente : MFCC ou FILTER-BANKS.

Un HMM, tel que formalisé par RABINER (1989), comporte un ensemble d'états cachés et d'observations. Les observations (O) correspondent aux données disponibles par le modèle : la représentation du signal audio (sec. 2.2.1). Les états cachés (S) correspondent à la représentation de sortie : mots, caractères, ou plus couramment phonèmes. Chaque unité de sortie est représentée par plusieurs états. Nous présentons en FIGURE 2.4 un exemple de HMM avec trois états et quatre observations. Des distributions indiquent la probabilité de passer d'un état à un autre (A), ainsi que la probabilité d'émettre une observation pour l'état courant (B). La distribution de probabilités initiale des états est

également indiquée dans Π. En résumé :

$$\begin{aligned}
\text{états} : S &= \{s_1, \ldots, s_N\} && s_i \text{ et } s_j \in S \quad (0 \leq i, j \leq N) \\
\text{observations} : O &= \{o_1, \ldots, o_M\} && o_k \in O \quad (0 \leq k \leq M) \\
\text{probabilités de transition} : A &= \{a_{i,j}\} \\
\text{probabilité d'émission} : B &= \{b_j(k)\} \\
\text{probabilité d'état initial} : \Pi &= \{\pi_i\}
\end{aligned}$$

Soit q_t l'état de S au temps t. Alors,

$$a_{i,j} = P\left(q_{t+1} = s_j \mid q_t = s_i\right) \tag{2.5}$$

$$b_j(k) = P\left(o_k \text{ au temps } t \mid q_t = s_j\right) \tag{2.6}$$

$$\pi_i = P\left(q_1 = s_i\right) \tag{2.7}$$

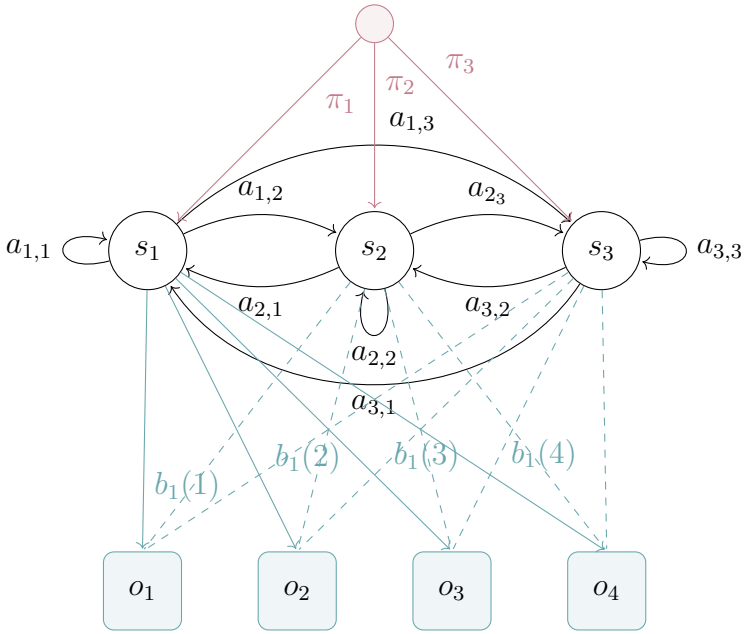


FIGURE 2.4 – Exemple d'un HMM comportant trois états et quatre observations.

Les probabilités de transition A, d'émission B et initiales Π sont apprises itérativement à l'aide d'estimateurs de maximum de vraisemblance (*maximum likelihood*), tel que l'algorithme EM (*Expectation-Maximization*) présenté par DEMPSTER et al. (1977).

Les probabilités d'émission B ont pendant un temps été estimées grâce à des modèles à mélange de gaussiennes (GMM). De tels modèles d'ASR ont été réalisés par LEE (1988)

et Gauvain et al. (2002). Plus récemment, avant l'avènement des modèles totalement neuronaux, les réseaux de neurones ont été utilisés conjointement aux HMM pour estimer ces probabilités. Nous pouvons par exemple citer les travaux de Bourlard et Morgan (1989), Seide et al. (2011), Dahl et al. (2012) et Hinton et al. (2012).

2.2.3 Les modèles neuronaux

Comme pour de nombreuses tâches du TAL, la reconnaissance automatique de la parole s'est également tournée vers les modèles neuronaux. Nous venons de le voir, les GMM ont un temps été remplacés par des réseaux de neurones, afin de former des architectures hybrides avec les HMM. Cependant, ce sont les architectures entièrement neuronales qui ont par la suite pris le devant. L'avantage principal de cette méthode est la simplification du système (Chorowski et al. 2015), d'un point de vue du développement logiciel.

Architectures récurrentes et CTC. Les architectures neuronales profondes et de bout-en-bout ont tout d'abord reposé sur des réseaux de neurones récurrents (section 1.1.2). Graves et al. (2006) ont introduit l'apprentissage de RNN avec le CTC pour une tâche de classification temporelle de la parole : prédire la séquence de phonèmes. Un dictionnaire de prononciation phonétique est ensuite utilisé pour convertir ces unités en mots. Graves et Jaitly (2014) ont développé un second modèle de reconnaissance de la parole se rapprochant d'un modèle purement bout-en-bout. Le modèle estimait directement la séquence de caractères à partir de la représentation spectrale. Eyben et al. (2009) avaient également construit un modèle neuronal passant par des graphèmes plutôt que l'unité phonétique.

Architectures encodeur-décodeur récurrentes. En reconnaissance de la parole, le plafond de verre des systèmes HMM fut brisé par les modèles CTC. Celui des CTC fut brisé par une architecture encodeur-décodeur sans CTC. Cette architecture fut introduite par Cho et al. (2014) pour la traduction automatique, puis utilisée pour la reconnaissance de mots par Lu et al. (2015). Cette architecture encode toute l'information acoustique dans un vecteur de taille fixe, puis génère la sortie voulue à partir de celui-ci. Chorowski et al. (2015) ont décrit une architecture encodeur-décodeur utilisant un mécanisme d'attention pour la reconnaissance de phonèmes. Bahdanau et al. (2016) et Chan et al. (2016) ont présenté indépendamment les deux premiers modèles encodeur-décodeur avec mécanisme d'attention qui génèrent des sorties graphèmiques (ici, des caractères) et non

phonétiques. Ces modèles avec mécanisme d'attention permettent de ne pas comprimer toute l'information acoustique dans un seul vecteur.

Architectures *transformers*. Les modèles *tranformers*, généralement pré-entraînés de manière auto-supervisée sont également utilisés pour la reconnaissance de la parole. Le premier modèle de reconnaissance automatique de la parole utilisant une architecture *transformer* fut réalisé par Dong et al. (2018). Celui-ci utilisait en entrée des Filter-Banks, et a été appris pour prédire la séquence de caractères correspondante, avec un apprentissage supervisé uniquement. Les résultats obtenus avec ce modèle sont similaires à ceux état de l'art de l'époque en ASR. En 2019, Jiang et al. ont proposé un modèle similaire, mais appris dans un premier temps de manière auto-supervisée. Dans cette étape, le modèle doit prédire les Filter-Banks masqués. Pour l'étape de *finetuning* ASR, un décodeur *transformer* est ajouté. Baevski et al. (2020a) ont utilisé le modèle wav2vec original (Schneider et al. 2019) conjointement avec le *transformer* textuel BERT (Devlin et al. 2019) pour réaliser la reconnaissance de la parole à partir du signal directement. Enfin, wav2vec 2.0 est le premier modèle complètement *end-to-end* utilisant un transformer pré-appris de manière auto-supervisée, utilisé pour la reconnaissance de la parole à partir du signal brut audio. Il a été publié par Baevski et al. (2020b). Lors de son *finetuning* ASR, celui-ci est appris avec un objectif CTC. Depuis, de nombreux *transformers* pré-entraînés ont été utilisés pour la reconnaissance de la parole, avec une tendance de plus en plus forte à réaliser des modèles comportant un nombre conséquent de paramètres, et de les apprendre sur des corpus de grande taille. Les modèles XLS-R (Babu et al. 2022) contiennent jusqu'à 2 milliards de paramètres, et sont évalués sur 26 langues différentes. Les modèles Whisper (Radford et al. 2023) contiennent jusqu'à 1.55 milliards de paramètres, et sont appris sur un ensemble supervisé de 680k heures de parole, obtenu depuis internet. Pratap et al. (2023) ont conçu un modèle de reconnaissance de la parole travaillant sur 1107 langues. Les résultats état de l'art actuels sur de nombreux corpus ASR sont obtenus avec de tels modèles (Radford et al. 2023 ; Pratap et al. 2023).

2.3 Modélisation de la langue : $P(Y)$

Modéliser la langue est une étape importante dans la reconnaissance de la parole. Son objectif est de corriger ou proposer une phrase, c'est à dire une séquence de mots, qui est probable dans la langue. Le modèle acoustique (sec. 2.2) pourrait en effet produire

une séquence de mots impossible ou improbable dans la langue. Il faut donc obtenir une probabilité P de la séquence de mots Y : $P(Y)$. La génération de la transcription étant généralement faite de façon séquentielle (t_0 puis t_1 ...), cette probabilité peut se calculer comme la probabilité du mot y_t sachant l'historique des autres mots (éq. 2.8).

$$P(Y) = P\left(y_1, y_2, \ldots, y_t\right) = P\left(y_t \mid y_1, y_2, \ldots, y_{t-1}\right) \tag{2.8}$$

À chaque instant t, il faut trouver le mot y_t qui maximise la probabilité du modèle de langage, mais également du modèle acoustique. En guise d'exemple, en français, considérons le début de phrase «*Le chien mange dans sa*». Le mot «*gamelle*» est sans doute plus probable que les mots «*voiture*» ou «*ordinateur*». Cela ne signifie pas que la transcription correcte ne serait pas «*Le chien mange dans sa voiture*», et c'est pourquoi il faut trouver un juste milieu entre l'importance du modèle acoustique et celle du modèle de langage.

Cette modélisation de la langue peut également être appliquée pour modéliser les mots, en plus de la phrase. Dans le cas de systèmes de transcription produisant des séquences de caractères, il est possible pour le modèle de générer des mots hors vocabulaires – OOV (*Out Of Vocabulary*). C'est voulu, en particulier pour des noms propres qui seraient inconnus, mais il faut dans ce cas que le modèle de langage soit lui aussi en mesure de travailler avec des mots inconnus. Bahdanau et al. (2016) utilisent un modèle de transcription au niveau caractères avec un modèle de langage au niveau mot. Leur modèle de langage peut produire des mots hors vocabulaire grâce à un transducteur à états finis, qui le transforme en modèle au niveau caractères. T. Hori et al. (2017) utilisent deux modèles de langage (mots et caractères) conjointement, tandis que T. Hori et al. (2019) utilisent un modèle de langage au niveau mot, mais déterminent des probabilités au niveau caractère également.

Les premiers modèles de langage statistiques furent imaginés par Shannon (1948, 1951). En 1976, Jelinek présente un «*linguistic decoder*» pour la reconnaissance de la parole, faisant office de modèle de langage.

Certains modèles acoustiques, notamment neuronaux, sont en mesure d'apprendre à modéliser $P(Y)$, de façon implicite, en même temps que $P\left(X \mid Y\right)$. C'est pourquoi, dans de nombreux travaux, bien qu'aucun modèle de langage additionnel ne soit utilisé, les résultats sont corrects (Amodei et al. 2016 ; Chan et al. 2016 ; Dong et al. 2018 ; Park et al. 2019).

2.3.1 Modélisation n-grammes

La modélisation n-gramme (SHANNON 1951) consiste à simplifier la probabilité conditionnelle présentée en équation 2.8. Pour cela, au lieu de considérer l'entièreté de l'historique de la séquence, on ne regarde qu'une fenêtre de taille n. La probabilité de la séquence correspond au produit des probabilités de chaque mot précédé des $n-1$ mots de l'historique. Formellement :

$$\begin{aligned} P_{n\text{-gram}}(Y) &= P_{n\text{-gram}}\left(y_1, \ldots, y_t\right) \\ &= \prod_{i=2}^{t} P\left(y_i \mid y_{i-n+1}, \ldots, y_{i-1}\right) \end{aligned} \tag{2.9}$$

Les probabilités conditionnelles sont typiquement estimées à partir d'un corpus d'apprentissage. Il *suffit* de compter la proportion d'occurrences de chaque n-gramme dans le corpus :

$$P\left(y_i \mid y_{i-n+1}, \ldots, y_{i-1}\right) = \frac{\#\text{compte}(y_{i-n+1}, \ldots, y_{i-1}, y_i)}{\#\text{compte}(y_{i-n+1}, \ldots, y_{i-1})} \tag{2.10}$$

Reprenons l'exemple donné plus haut, avec une taille de fenêtre $n = 2$, c'est à dire un bi-gramme. Afin de calculer la probablité de la phrase au complet, on rajoute des unités fictives de début (BOS) et de fin (EOS) de phrase.

$$\begin{aligned} P(\text{Le, chien, mange, dans, sa, gamelle}) \equiv P(&\text{BOS, Le, chien, mange,} \\ &\text{dans, sa, gamelle, EOS}) \end{aligned} \tag{2.11}$$

On calcule la probabilité de cette séquence :

$$\begin{aligned} P_{2\text{-gram}}(\text{BOS, Le, chien, mange, dans, sa, gamelle, EOS}) \;=\; & P\left(\text{Le} \mid \text{BOS}\right) \times \\ & P\left(\text{chien} \mid \text{Le}\right) \times \\ & P\left(\text{mange} \mid \text{chien}\right) \times \\ & P\left(\text{dans} \mid \text{mange}\right) \times \\ & P\left(\text{sa} \mid \text{dans}\right) \times \\ & P\left(\text{gamelle} \mid \text{sa}\right) \times \\ & P\left(\text{EOS} \mid \text{gamelle}\right) \end{aligned} \tag{2.12}$$

Choisir un n élevé permet de mieux modéliser le langage, et donc d'améliorer les

performances du modèle de langage. Cependant, la complexité de ce modèle de langage croît exponentiellement avec ce n. Pour un vocabulaire V de taille $|V|$, il faudrait jusqu'à $|V|^n - 1$ paramètres (Bengio et al. 2003), ici, des probabilités à mémoriser. En pratique, cela implique qu'un grand nombre de ces n-grammes ne sont jamais observés dans le jeu d'apprentissage. Un problème intervient lorsque pour l'évaluation, on a besoin de la probabilité P d'une suite de mots jamais observée dans le corpus d'apprentissage. Plus le n est élevé, plus ce problème est fréquent. Si n est aussi grand que la phrase entière, il faudrait alors avoir déjà vu exactement la même phrase en apprentissage. Si une suite de mots n'a jamais été vue, sa probabilité est 0, et celle-ci va se propager pour toute la séquence. Différentes techniques de lissage permettent de contourner ce problème (Chen et Goodman 1996) : addition, *back-off*, etc.

L'autre inconvénient majeur de ces modèles de langage est qu'il ne peuvent pas modéliser la structure de la langue. «*Le chien mange dans sa gamelle*» et «*La girafe dort sur la voiture*» sont des phrases complètement différentes pour un modèle de langage n-gramme, et pourtant, elles respectent la même structure (sujet, verbe, complément). Un modèle de langage idéal devrait également être en mesure d'extrapoler à partir des informations de l'une de ces deux phrases pour modéliser la seconde.

2.3.2 Modélisation neuronale

Les modèles de langage neuronaux permettent en partie de pallier ces problèmes, et d'améliorer leurs performances.

Premières modélisations neuronales de la langue. La modélisation de la langue à partir de réseaux de neurones n'est pas une problématique de recherche nouvelle. Déjà en 1989, Cleeremans et al. tentent de modéliser une grammaire simple en utilisant des RNN. Le modèle devait apprendre à prédire le caractère suivant, généré à partir d'un automate simple. Lawrence et al. (1996) ont utilisé un RNN pour apprendre à classifier des phrases anglaises comme grammaticalement correctes ou non (1044 phrases). Wei Xu et Rudnicky (2000) tentent de prédire le mot suivant à partir du précédent (bi-gramme). Bengio et al. (2003) réalisent les premiers travaux de modélisation de la langue naturelle à grand échelle. Les mots sont projetés dans un espace latent (appelés *feature vectors* ou *word embeddings*), et le modèle doit prédire le mot suivant en fonction des n mots précédents, de façon analogue à un n-gramme. Cette idée de modèle de langage neuronal n-gramme a par exemple été reprise par Schwenk (2007). L'avantage principal d'un tel modèle repose sur le fait que les mots similaires possèdent des *feature vectors* similaires.

Pour reprendre l'exemple de Bengio et al. (2003), cela peut permettre au modèle de généraliser une phrase telle que «le chat marche dans la chambre» en «le chien courrait dans la pièce». Pour étendre l'historique à toute la séquence, un RNN doit être employé (Mikolov et al. 2010). Pour l'ASR, ce modèle de langage permet de réduire largement les erreurs, comparativement à un modèle n-gramme.

Les larges modèles de langage ou LLM (*Large Language Model*) sont des modèles de langage neuronaux comportant un nombre de paramètres important (typiquement $\gtrapprox$ 65M). Cette définition fut notamment donnée par Bender et al. (2021), qui fait un état des problèmes éthiques de ces modèles : coût environnemental, biais des données d'apprentissage et utilisation de ces modèles pour des tâches de compréhension. Ce sont principalement les *transformers* qui ont permis aux modèles de langage de passer dans cette catégorie LLM, bien que le modèle ELMo (Peters et al. 2018) soit un modèle LSTM considéré comme un LLM.

Ces modèles sont principalement appris pour une tâche de MLM (*Masked Language Model*) : prédire le mot manquant. De nombreux LLM sont développés, nous pouvons par exemple citer pour l'anglais BERT (Devlin et al. 2019) et RoBERTa (Liu et al. 2019). Pour le français, CamemBERT (Martin et al. 2020) et FlauBERT (Le et al. 2020) sont des modèles généralement utilisés. Il existe également des modèles multilingues : LaBSE (F. Feng et al. 2022), GPT-3 (Brown et al. 2020). Tous ces modèles génèrent des représentations intermédiaires cachées, dans un espace latent, qui peuvent être utilisées comme vecteurs d'entrées d'autres modèles (Raffel et al. 2020). Ils peuvent également être utilisés pour d'autres tâches que la reconnaissance de la parole : génération de texte, classification de texte, traduction automatique, etc.

2.3.3 *Tokenization* du texte

La reconnaissance de la parole et la modélisation du langage se fait avec une certaine unité de sortie. Intuitivement, celle-ci peut être faite à deux niveaux : prédire des mots ou prédire des caractères, mais, ce ne sont pas les seules unités utilisées en TAL. De plus, convertir une séquence de caractères en une séquence de mots n'est pas toujours simple. Doit-on séparer les mots uniquement avec le caractère espace ? Doit-on inclure d'autres caractères tels que «-», « :», « ;» ? Qu'en est-il des nombres et de leurs unités («*13.37€*») ? En pratique, les données ne sont pas toujours *propres* dans les corpus utilisés, et de nombreux pré-traitements doivent être réalisés pour les nettoyer.

La tâche de séparation / regroupement des séquences de caractères en séquences

d'unités s'appelle la *tokenization*. Elle consiste à générer des *tokens*, ce qu'on a jusqu'à présent nommé *unité de sortie*. Des librairies telles que NLTK permettent de réaliser cette tokenization (Bird et Loper 2004).

Des méthodes permettent de trouver un entre-deux entre une *tokenization* en mots et une *tokenization* en caractères. Celles-ci reposent sur des sous-mots, et permettent de générer des mots hors vocabulaire, sans avoir à produire chacun des caractères de celui-ci. Pour cela, le découpage en sous-mots consiste à découper certains mots en sous-unités. Le choix du découpage dépend de la méthode employée, mais elle est généralement donnée par la fréquence d'apparition de ces sous-mots. Sennrich et al. (2016) ont proposé d'employer l'algorithme de compression BPE (*Byte-Pair Encoding*) décrit par Gage (1994) pour créer ces sous-mots. En TAL, le BPE consiste à démarrer d'un vocabulaire comportant tous les caractères, puis de remplacer itérativement les occurrences du vocabulaire les plus fréquentes dans le texte par un nouveau terme du vocabulaire. Le nombre de remplacements effectués est un hyperparamètre de la méthode de *tokenization*.

SentencePiece (Kudo et Richardson 2018) est un *tokenizer* utilisant notamment le BPE pour *tokenizer* du texte n'ayant pas besoin d'être séparé en mots au préalable. Il peut ainsi être appliqué sur des langues sans utiliser de connaissances a priori à propos de celles-ci.

2.4 Décodage, prédiction et évaluation

Une fois un modèle de reconnaissance de la parole et son modèle de langage éventuel appris, nous pouvons enfin nous en servir pour prédire la transcription de nouvelles phrases, par exemple du jeu de validation Dev ou de Test. La phase de génération de texte est appelée décodage, car elle consiste principalement à utiliser le décodeur du modèle de façon itérative pour générer ce texte. Une fois décodés (sec. 2.4.2), les jeux Dev ou Test peuvent être évalués, selon différentes métriques (sec. 2.4.3).

2.4.1 Intégration du modèle de langage au modèle acoustique

Nous avons présenté le modèle acoustique $P(X \mid Y)$ en section 2.2 et le modèle de langage $P(Y)$ en section 2.3. Nous voyons ici comment intégrer efficacement ces deux modèles pour la reconnaissance de la parole. En équation 2.3, nous indiquions que trouver la meilleur séquence de sortie $\hat{Y}$ correspond à trouver la séquence Y qui maximise $P(X \mid Y) \cdot P(Y)$. En pratique, il est souhaitable de pouvoir contrôler l'importance du

modèle de langage par rapport au modèle acoustique. Ainsi, un hyperparamètre α peut être ajouté. Il s'agit en principe d'un poids dont la valeur est incluse dans $[0, 1]$. Ce poids fait partie des hyperparamètres pouvant être optimisés.

Le *shallow-fusion* est une technique décrite par GULCEHRE et al. (2015) afin de faire travailler les deux modèles conjointement, durant la phase de génération du texte au lieu de simplement a posteriori. Le texte est généré de façon séquentielle par le modèle acoustique. À chaque instant t, les scores des sorties possibles y_t sont pondérés par les scores de ces mêmes séquences par le modèle de langage. D'autres scores peuvent être employés : score de prononciation, WIP (*Word Insertion Penalty*), etc. La sortie y_t ayant le meilleur score au total est conservée.

Nous présentons en équation 2.13 le calcul de la probabilité que le $t^{\text{ème}}$ caractère de y corresponde à k, à partir du modèle acoustique (P_{AM}), du modèle de langage (P_{LM}) et du paramètre α.

$$\begin{aligned} \log\left(P(y_t = k)\right) = \log\left(P_{AM}(y_t = k)\right) \\ + \alpha \log\left(P_{LM}(y_t = k)\right) \end{aligned} \tag{2.13}$$

2.4.2 Décodage : glouton ou faisceau ?

La façon la plus naïve et rapide de décoder un modèle est de réaliser un décodage glouton ou *greedy search*. À chaque instant t, le *token* le plus probable selon les modèles est choisi. Aucun retour en arrière n'est possible avec ce mode de décodage. Ce décodage est le plus rapide à mettre en œuvre et le moins coûteux en termes de ressources de calcul.

Le *beam search* ou décodage en faisceau est une technique de décodage permettant généralement d'améliorer les résultats, mais avec un coût algorithmique plus élevé que le décodage glouton. Le décodage en faisceau consiste, à chaque instant t, à conserver N hypothèses candidates, en choisissant les N chemins les plus probables. La probabilité d'un chemin est donnée comme la probabilité jointe de toute la séquence, depuis la racine BOS jusqu'au mot courant. Il s'agit donc d'un parcours en largeur de l'arbre de recherche. L'avantage d'un tel algorithme c'est qu'une décision locale, prise au moment t, n'est pas complètement contraignante pour la suite. Cette méthode est une simplification du parcours complet de l'espace de recherche, où toutes les hypothèses seraient explorées : $N = \infty$. À l'inverse, si $N = 1$ cet algorithme est équivalent au décodage glouton présenté plus haut.

En FIGURE 2.5 nous présentons un exemple fictif de décodage glouton. Les probabilités

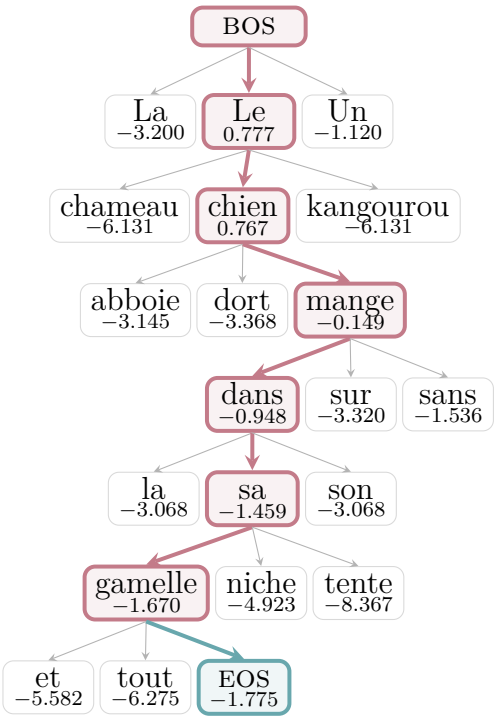


FIGURE 2.5 – Exemple fictif de décodage glouton ayant généré la phrase «Le chien mange dans sa gamelle». Seuls les trois mots les plus probables sont affichés à chaque instant.

indiquées sous les noeuds correspondent à la log-probablité cumulative de l'hypothèse partant de la racine au noeud, définie dans $]-\infty, 0]$. L'exemple équivalent pour un décodage en faisceau est indiqué en FIGURE 2.6. Par mesure de clarté, dans ces exemples, pour chaque hypothèse en cours, seules les 3 mots les plus probables sont affichés. La racine est représentée par BOS, et le décodage est effectué jusqu'à ce que toutes les hypothèses en cours aient générées le *token* de fin de phrase EOS.

Ces deux méthodes ne sont cependant pas les seules façons de réaliser un décodage, et des variantes de celles-ci existent également : top-k (FAN et al. 2018), *nucleus* ou top-p (HOLTZMAN et al. 2020), etc.

2.4.3 Évaluation des transcriptions

Les métriques d'évaluation de la transcription ASR reposent principalement sur le *Word Error Rate* (WER) et le *Character Error Rate*. Ces métriques mesurent le nombre d'erreurs totales dans la séquence d'hypothèse, par rapport au nombre de *tokens* attendus. Pour les calculer, nous devons tout d'abord effectuer un alignement entre les *tokens* de la séquence référence et celles de la séquence hypothèse. L'alignement correspond typiquement à la

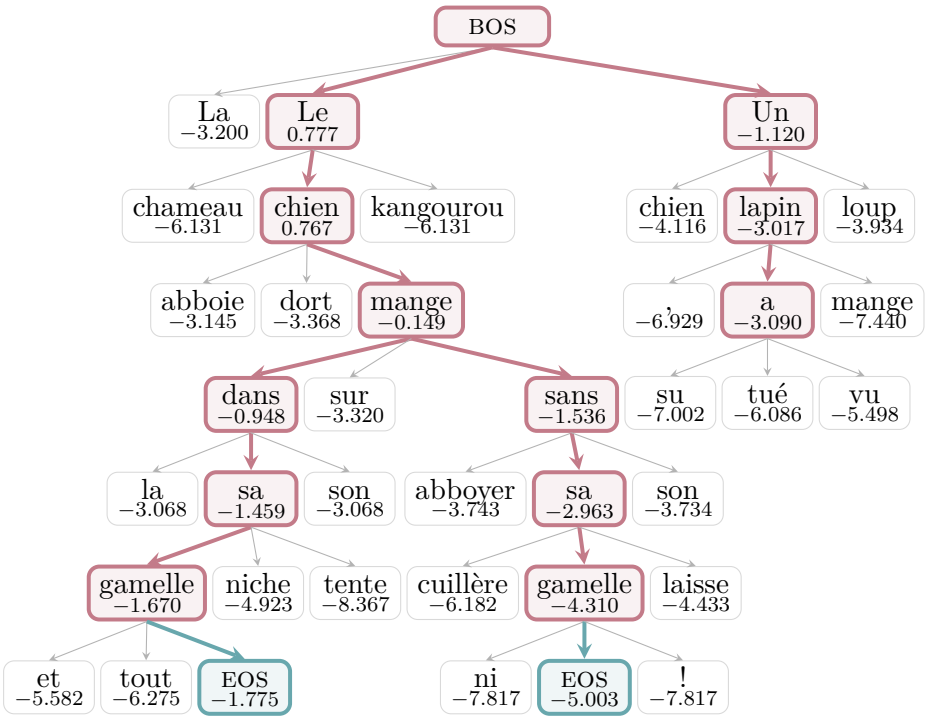


FIGURE 2.6 – Exemple fictif de décodage en faisceau ($N = 2$) ayant généré les phrases «Le chien mange dans sa gamelle» et «Le chien mange sans sa gamelle». Seuls les trois mots les plus probables sont affichés à la suite de chaque hypothèse en cours.

distance d'édition, également connue sous le nom de distance de LEVENSHTEIN. Des outils tels que `sclite` de SCTK ou bien TORCHMETRICS permettent de réaliser celui-ci. Cet alignement nous donne un nombre d'unités correctes, de suppressions (D), insertions (I) et substitutions (S). Plus cette métrique possède un score bas, meilleurs sont les résultats. Nous la représentons souvent sous forme de pourcentage, mais il s'agit d'un abus de langage : le résultat peut dépasser 100% pour un modèle très mauvais, en cas de (trop) nombreuses insertions par exemple. Nous détaillons le calcul dans les formules 2.14 et 2.15.

Acronymes

Par la suite, afin de ne pas confondre les métriques du *Character Error Rate* utilisées pour l'ASR et du *Concept Error Rate* utilisé pour la compréhension de la parole (SLU), nous les nommerons respectivement ChER et CER. Le CER est décrit en section 3.4.2.

$$\text{WER} = \frac{S_{\text{W}} + D_{\text{W}} + I_{\text{W}}}{N_{\text{W}}} \tag{2.14}$$

$$\text{ChER} = \frac{S_{\text{Ch}} + D_{\text{Ch}} + I_{\text{Ch}}}{N_{\text{Ch}}} \tag{2.15}$$

où :

$S =$ le nombre de substitutions dans l'hypothèse
$D =$ le nombre de *deletions* (suppressions) dans l'hypothèse
$I =$ le nombre d'insertions dans l'hypothèse
$N =$ le nombre d'unités dans la séquence de référence

En ce qui concerne les séquences d'hypothèse et de référence, en fonction de la métrique (WER ou ChER), nous regardons les mots ($_{\text{W}}$) ou les caractères individuels ($_{\text{Ch}}$).

2.5 Corpus

Nous décrivons brièvement dans cette section quelque uns des corpus les plus couramment utilisés pour entraîner et tester un système de reconnaissance de la parole.

WSJ. Traditionnellement, le corpus du WSJ (*Wall Street Journal*) est l'un des premiers corpus anglais générique de taille conséquente. Présenté par Paul et Baker (1992), il est composé d'environ 80 heures de parole transcrites pour la partie audio, et 47 millions de mots pour la partie texte, qui est utilisée pour apprendre des modèles de langage. Il s'agit d'enregistrements d'articles du *Wall Street Journal*. Les taux d'erreur en WER sur ce corpus sont aujourd'hui autour de 4% (Radford et al. 2023).

LibriSpeech. Présenté par Panayotov et al. (2015), LibriSpeech est un corpus d'enregistrements de livres audio en anglais également, d'environ 1000 heures transcrites, enregistré par plus de 1000 locuteurs.

Libri-Light. Le corpus Libri-Light, décrit par Kahn et al. (2020) contient 57 706 heures de parole, sans transcriptions associées, provenant comme LibriSpeech, de livres audio anglais. Il contient également quelques heures de parole transcrites, afin de réaliser un *finetuning* sur quelques exemples seulement.

CommonVoice. Le corpus CommonVoice est désormais devenu une référence pour la reconnaisance de la parole. Présenté par Ardila et al. (2020), c'est un corpus multilingue et multilocuteur, enregistré et validé par des bénévoles sur internet. La version 11.0 (21/09/2022) du corpus contient 3098 heures enregistrées pour la partie anglaise, avec 85k locuteurs. La partie française est réalisée par 17k locuteurs, pour un total de 1007 heures de parole.

Autres corpus français. Une source considérable de données de parole se situe dans les archives de la télévision ainsi que de la radio. Bien que le style de parole puisse être différent de certaines applications spécifiques (téléphonique par exemple), ces données sont disponibles en très grande quantité. L'annotation de ces données en transcription est cependant une tâche coûteuse. Ester1 et Ester2 (Gravier et al. 2004 ; Galliano et al. 2005, 2009) sont deux corpus français, comportant respectivement 100 et 103 heures de parole. Ils ont été réalisés à partir d'enregistrements de radio tels que RFI, France Info ou France Inter. Le corpus Etape (Galibert et al. 2014) comporte 33 heures de parole issues d'émissions radio et télévisuelles telles que France Inter, ou BFMTV. Epac (Estève et al. 2010) est un corpus similaire comportant 90 heures de parole. Repere (Giraudel et al. 2012) et Quaero (Boudahmane et al. 2011) sont des corpus également couramment utilisés, eux-aussi réalisés à partir d'émissions audiovisuelles françaises.

VoxPopuli. Le corpus multilingue VoxPopuli (C. Wang et al. 2021a) a été réalisé grâce à des enregistrements du Parlement Européen. En français, il contient 22 800 heures de parole sans transcription.

TED-LIUM. Les corpus TED-LIUM (Rousseau et al. 2012) sont réalisés à partir de conférences TED, en anglais. La dernière version du corpus possède une durée totale de 452h de parole.

Nous présentons en Table 2.1 un récapitulatif des corpus utilisés en ASR. Les durées des jeux Train, Dev et Test correspondent aux parties annotées en transcription de ces corpus, tandis que celles indiquées dans la colonne ST correspondent aux données de parole sans transcription associée. Pour ce qui est de CommonVoice, il s'agit des données «validées», c'est à dire ayant eu plus de deux validations par des utilisateurs, et où ce nombre est supérieur au nombre de rejets.

Corpus		Type de données	Parole (heures) Train	Dev	Test	ST
WSJ0 + WSJ1	EN	Journaux lus	80	1	1	
LibriSpeech	EN	Livres audio	960	11	11	
Libri-Light	EN	Livres audio	10	11	11	57 706
VoxPopuli	EN	Parlement UE	488	27	27	24 100
CommonVoice 11.0	EN	Divers	949	16	16	
VoxPopuli	FR	Parlement UE	190	12	12	22 800
CommonVoice 11.0	FR	Divers	485	16	16	
Ester1	FR	Audiovisuel	82	8	10	1 700
Ester2	FR	Audiovisuel	91	6	7	
Etape	FR	Audiovisuel	22	7	7	
Epac	FR	Audiovisuel	70	9	9	
Repere	FR	Audiovisuel	35	3	9	
Quaero	FR	Audiovisuel	93	0	6	

Table 2.1 – Récapitulatif des corpus de transcriptions décrits. ST : données de parole sans la transcription.

2.6 Conclusion

Nous avons dans ce chapitre présenté la tâche de reconnaissance automatique de la parole, qui est une brique essentielle pour la compréhension de celle-ci, l'objectif de cette thèse. Nous nous sommes intéressés aux deux modélisations qui sont utilisées pour la reconnaissance de la parole : il faut premièrement modéliser l'acoustique pour pouvoir transformer l'entrée (audio) en séquence de mots ou de caractères. Il est également souvent judicieux d'apporter une deuxième modélisation, celle de la langue. Celle-ci, faite avec des modèles de langage (LM), devient de plus en plus importante dans le domaine du TAL aujourd'hui. En effet, les nouveaux modèles de langages *transformers* sont désormais utilisés pour d'autres tâches que la reconnaissance de la parole, par exemple pour la compréhension de la langue. Dans le chapitre suivant, nous présentons la tâche de compréhension automatique de la parole.

Chapitre 3

La compréhension de la parole

Sommaire

3.1 Représentations sémantiques 71
3.1.1 Représentation à base de *frames* 71
3.1.2 Représentation en segments sémantiques 72
3.1.3 Représentation générale par classes 72
3.2 Compréhension de la parole : quelles applications ? 73
3.2.1 Routage et détection de l'intention 74
3.2.2 Résumé automatique du discours 74
3.2.3 Recherche vocale 74
3.2.4 Systèmes de questions-réponses 75
3.2.5 Systèmes de dialogue humain-machine 75
3.3 Les systèmes de compréhension 77
3.3.1 Systèmes NLU 77
3.3.2 Systèmes SLU 79
3.4 Méthodologie : formats et évaluation 81
3.4.1 Formats 81
3.4.2 Les métriques d'évaluation 83
3.5 Les corpus de compréhension de la parole 84
3.5.1 ATIS et MultiAtis++ 84
3.5.2 MultiWOZ 86
3.5.3 SpokenWOZ 86
3.5.4 MEDIA 87
3.5.5 PortMEDIA 88
3.5.6 Récapitulatif 89
3.6 Conclusion 90

En 1983, Bahl et al. décrivent la reconnaissance automatique de la parole comme trois tâches possibles. La première correspond à la transcription de mots isolés, la deuxième comme la transcription de la langue naturelle. Enfin, la troisième tâche est décrite comme la compréhension de la parole, «où l'objectif n'est pas la transcription mais la compréhension dans le sens que le système répond correctement à une instruction ou une requête». Ainsi, pour un système automatique, la compréhension serait que celui-ci réalise l'action ou la requête voulue par l'utilisateur·ice. Cependant, Tur et De Mori (2011) décrivent la compréhension de la langue comme «l'objectif d'extraire la signification du langage naturel». Pour cela, la compréhension de la parole doit «interpréter les signaux acheminés par le signal de parole».

Ces deux définitions sont complémentaires : un système de compréhension doit extraire la signification des signaux dans une représentation *sémantique*, puis effectuer le traitement souhaité par l'utilisateur·ice.

Mais qu'est-ce que la sémantique ? Le dictionnaire utilisé par Woods (1975) indique qu'il s'agit de «l'organisation du sens et de la relation entre les signes sensoriels ou les symboles et ce qu'ils signifient». Woods (1975) décrit une vision volontairement caricaturale d'un linguiste et d'un philosophe. Le linguiste serait intéressé par «la traduction de la langue naturelle dans des représentations formelles de leur sens», tandis qu'un philosophe serait «intéressé dans la signification de ces représentations». Le travail d'un informaticien se rapproche de cette vision du linguiste, à travers la sémantique computationnelle. D'après De Mori et al. (2008), celle-ci consiste à «conceptualiser informatiquement le monde pour réaliser une structure de représentation du sens à partir de traits tels que les mots».

Ces théories étant exprimées, nous avons opté pour la définition suivante : un système de compréhension doit extraire une représentation sémantique du sens du signal de parole, ou des mots correspondants. Le sens extrait doit être placé dans un conteneur que nous appelons *représentation sémantique*. Nous présentons en section 3.1 quelques unes de ces représentations sémantiques. Celles-ci sont par la suite utilisées dans un contexte applicatif particulier : routage d'appel, résumé automatique ou système de dialogue humain-machine par exemple. Nous présentons ces applications en section 3.2. Nous décrivons en section 3.3 des systèmes de compréhension de la parole et des mots. En section 3.4, nous détaillons la méthodologie derrière ces modèles, puis les corpus usuels en section 3.5.

3.1 Représentations sémantiques

Nous pouvons définir plusieurs niveaux de représentations des connaissances sémantiques. Nous décrivons dans cette section trois représentations possibles des connaissances sémantiques, de la plus complexe à la plus générale.

3.1.1 Représentation à base de *frames*

Fillmore (1976) a introduit la notion de *frames* au sein de la langue. Ces *frames* permettent de modéliser l'information sémantique en regroupant la connaissance dans des sous-structures. Les informations sont ensuite enregistrées dans des *slots* ou champs (Y.-Y. Wang et al. 2011a). L'information est donc représentée de façon hiérarchique, par un ensemble de *frames*, composées d'un ensemble de *slots* et de valeurs associées. Ces *frames*, *slots* et valeurs sont définis en fonction du domaine d'application du système considéré. Nous présentons en Figure 3.1 un exemple de cette représentation hiérarchique, pour une phrase «quels sont les hôtels proches du stade de france avec une piscine». Un système SLU à base de *frames* doit prédire la *frame* qui représente le sens du message, et remplir les champs avec les valeurs correctes (Tur et De Mori 2011).

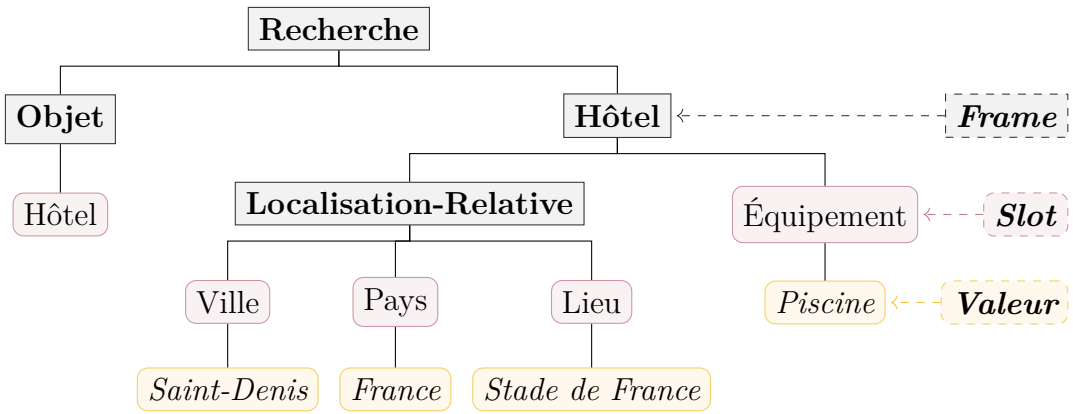


Figure 3.1 – Exemple de représentation sémantique à base de *frames* pour un segment utilisateur «*quels sont les hôtels proches du stade de france avec une piscine*».

Cette représentation à base de *frames* est suffisamment générique pour être adaptée à toutes les tâches. Cependant, celle-ci est complexe à définir et à déterminer. C'est pourquoi, nous simplifions cette représentation en gardant uniquement les concepts et leurs valeurs, dans la représentation en *segments sémantiques* (section 3.1.2).

3.1.2 Représentation en segments sémantiques

Nous pouvons transformer cette représentation hiérarchique en une représentation aplatie en segments sémantiques. Cette représentation correspond au *slot filling*, ou remplissage de champs.

Généralement, les segments sémantiques sont alignés avec les mots prononcés (la transcription). On parle alors de mots supports. Dans ce cas, un segment sémantique est caractérisé par un ensemble de mots supports, un concept (ou *slot*) et une valeur. Les ensembles de concepts sont décrits selon la tâche cible. Les valeurs permettent de formaliser les mots supports en un élément d'un dictionnaire fixe, telle une normalisation. Par exemple, les mots «proche» ou «pas très loin» auraient la même valeur «proche» avec un concept tel que «`localisation-relative-distance`». D'autres informations peuvent également être renseignées pour chaque segment sémantique : un mode (tel que «interrogatif», «négatif» ou «positif»), ou par exemple un lien entre certains segments.

Cette représentation en segments sémantiques est par exemple utilisée dans des systèmes conversationnels (Mesnil et al. 2015 ; Rastogi et al. 2020). La définition des concepts (*slots*) et valeurs possibles dépend du domaine du système de compréhension construit. Certains concepts peuvent ainsi être spécifiques au domaine (`chambre-type` pour un système de réservation d'hôtels par exemple), ou indépendant car ils sont génériques à la langue (`co-référence` par exemple). Ces concepts peuvent être définis pour une tâche de détection d'entités nommées ou NER (*Named Entity Recognition*). Selon Béchet (2011), la NER consiste à «détecter les éléments d'un document qui font une référence directe à un identifiant unique», tels que des noms de personnes, des organisations, ou par exemple des lieux.

Nous présentons en Figure 3.2 l'exemple précédent avec une représentation en concepts et valeurs. Afin d'illustrer les différents termes (mots supports, concepts, valeurs), nous présentons en Table 3.1, la représentation de ce même exemple, mais alignée sur les mots de la transcription. Comme nous pouvons l'observer, les informations sémantiques de cette représentation sont moins riches que celles au niveau *frame*.

3.1.3 Représentation générale par classes

La représentation la plus générale, se situe au niveau du document. Cette représentation compresse l'ensemble de l'information d'un dialogue dans une seule représentation générale. Celle-ci consiste à classer, par exemple l'entièreté du dialogue ou une phrase, dans une catégorie qui correspond à l'*intention* de l'utilisateur porté par son message.

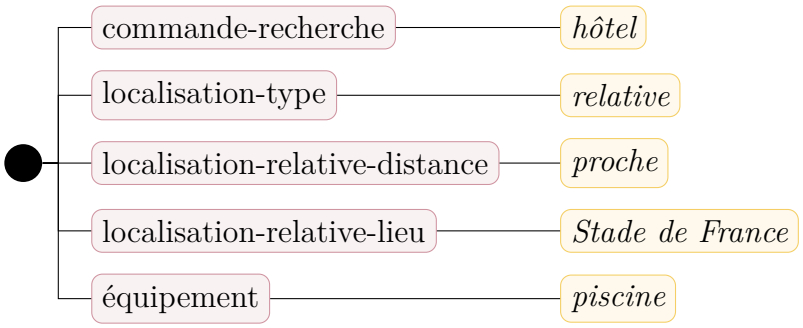


FIGURE 3.2 – Exemple de représentation en segments sémantiques (concepts et valeurs), pour une phrase utilisateur «*quels sont les hôtels proches du stade de france avec une piscine*».

Mots supports	Concepts	Valeurs
quels sont les hôtels	`commande-recherche`	`hôtel`
proche	`localisation-relative-distance`	`proche`
du stade de france	`localisation-relative-lieu`	`Stade de France`
avec une piscine	`équipement`	`piscine`

TABLE 3.1 – Exemple de représentation segments sémantique (concepts et valeurs), alignés avec la transcription, pour une phrase utilisateur «*quels sont les hôtels proches du stade de france avec une piscine*».

Sur l'exemple précédent, la phrase «quels sont les hôtels proches du stade de france» pourrait par exemple être classifiée avec l'élément «`recherche-hotel`».

3.2 Compréhension de la parole : quelles applications ?

Nous présentons dans cette section quelques unes des tâches les plus couramment réalisées dans le domaine de la compréhension automatique de la langue et de la parole.

3.2.1 Routage et détection de l'intention

Le routage, ou détection de l'intention, permet de diriger correctement l'utilisateur vers le bon service ou d'effectuer le bon traitement. Cette tâche est effectuée grâce à la représentation générale par classes (cf. section 3.1.3). Globalement, cette tâche est préliminaire avant un autre traitement par rapport à l'utilisateur : soit c'est pour le diriger vers le service qui devra répondre à son problème, soit c'est pour donner un contexte à un système de dialogue automatique.

Dans le cas d'appels téléphoniques, le routage permet au système de diriger l'utilisateur vers le bon service (Tur et Deng 2011). Le routage peut également être employé par un moteur de recherche (Broder 2002) pour classifier la requête avec une classe telle que «`navigation`» (rechercher un site spécifique), «information» (rechercher une information) ou «`transaction`» (e-commerce et autres). Les assistants intelligents réalisent également une détection de l'intention pour détecter le type de requêtes de l'utilisateur (Rastogi et al. 2020).

3.2.2 Résumé automatique du discours

Le résumé automatique du discours (texte ou parole) vise à condenser l'information principale du message. L'objectif est de garder le sens du message, mais de réduire le support de celui-ci, voir éventuellement d'en changer. Historiquement, le résumé automatique a été l'une des premières applications de la compréhension de la langue. En 1958, Luhn construit un système permettant de générer automatiquement des résumés d'articles scientifiques à partir du document complet. Plus tard, C. Hori et al. (2002) réalisent un résumé automatique d'actualités du format parole vers le texte, grâce à une première étape de transcription. Maskey et Hirschberg (2006) réalisent un résumé audio de l'actualité, sans passer par l'étape de transcription.

3.2.3 Recherche vocale

Un système de recherche vocale permet à un humain de réaliser une recherche parmi une large base de documents, en formulant directement sa requête à l'oral. Le système doit renvoyer le ou les documents correspondants.

La problématique principale d'un tel système concerne l'espace de recherche sémantique, lié au nombre de documents présents dans la base de données, et qui est bien souvent de taille très importante (Y.-Y. Wang et al. 2011b). Un module de compréhension de la parole

n'est pas obligatoire dans un système de recherche vocale. En effet, il est possible de faire une transcription de la parole, puis d'opérer une recherche des documents correspondants à ces termes (Yu et al. 2007). Cependant, pour permettre aux utilisateurs de faire des recherches parmi des documents structurés, un système de compréhension plus poussé est requis (J. Feng et al. 2009 ; Song et al. 2009).

Nous pouvons prendre l'exemple de recherches dans un annuaire d'entreprises et de services. L'utilisateur peut vouloir rechercher un type d'entreprise, dans une localisation donnée, avec éventuellement d'autres critères (horaires, équipements, avis, langue, ...), le tout dans une forme naturelle (non structurée) : «quels sont les hôtels proches du stade de france ?». Les documents de la base contiennent des champs avec des valeurs pouvant correspondre à certains des éléments de la requête.

3.2.4 Systèmes de questions-réponses

Un système de question-réponse doit permettre à un humain d'obtenir une réponse en langage naturelle à sa question, elle aussi posée en langage naturel. Le système doit pour cela retrouver l'information parmi un morceau de contexte (oral ou écrit) qui lui est fourni. Cette tâche est similaire à la tâche de recherche vocale. Cependant, un système de questions-réponses doit formuler sa réponse en langue naturelle, tandis qu'un système de recherche vocale renvoie seulement le document trouvé dans la base. Rosset et al. (2011) décrivent les problématiques posées par cette tâche.

Rajpurkar et al. (2016) décrivent par un exemple un corpus de questions-réponses associées à un paragraphe de contexte. Les systèmes de questions-réponses fonctionnant sur ce corpus doivent déterminer les réponses à la question, grâce à ce contexte.

3.2.5 Systèmes de dialogue humain-machine

Les systèmes de dialogue humain-machine doivent permettre à un humain de converser de manière fluide avec la machine, dans un cadre applicatif donné. Il s'agit d'une tâche plus ouverte que celles de question-réponse (section 3.2.4) et de recherche vocale (section 3.2.3), car la conversation ne suit pas forcément un schéma question (utilisateur) - réponse (système). Le dialogue peut suivre d'autres formats de conversation : questions à l'initiative du système, bavardage, ... Nous schématisons en Figure 3.3 le fonctionnement d'un tel système de dialogue. Celui-ci est composé d'un gestionnaire de dialogue permettant de prendre les décisions. Le module de compréhension doit fournir une représentation sémantique suffisamment détaillée au gestionnaire de dialogue (par exemple, représentation

en segments sémantiques ou *frames*). Les modules de compréhension de la parole et de synthèse sont en quelque sorte les interfaces d'entrées et de sorties entre le système et l'humain.

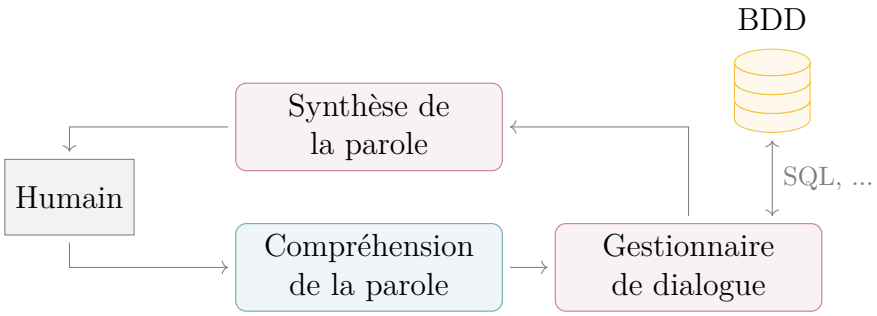


FIGURE 3.3 – Fonctionnement d'un système de dialogue humain-machine.

ZUE et SENEFF (2008) décrivent plusieurs niveaux de dialogues entre un humain et la machine. Nous trouvons d'une part les conversations à l'initiative du système, où l'utilisateur répond à des questions fermées posées par le système, afin de permettre à celui-ci de compléter la demande. À l'inverse, les conversations à l'initiative de l'utilisateur permettent à celui-ci de librement exposer sa requête, et le système intervient uniquement pour obtenir des clarifications.

Décrit par WEIZENBAUM (1966), ELIZA est sans doute l'un des premiers systèmes conversationnel développés. Il permettait par exemple de simuler un psychothérapeute posant des questions à l'utilisateur. Ce dialogue se faisait à l'écrit.

Un exemple de système conversationnel en mode oral est décrit dans le projet MEDIA (DEVILLERS et al. 2004), avec un service d'informations touristiques et de réservation de chambres d'hôtels par téléphone. Le système décrit par le projet MEDIA relève d'un dialogue à l'initiative de l'utilisateur. Le système peut réaliser des requêtes dans une base de données pour réaliser les réservations de chambres d'hôtels. Un tel système conversationnel nécessite une représentation sémantique complexe, à base de *frames* ou en concepts. Ensuite, un gestionnaire de dialogue doit pouvoir construire des requêtes d'accès à la base de données, puis formuler et générer sa réponse grâce à un module de synthèse de la parole.

Le terme *chatbot* est également employé pour désigner un système de dialogue humain-machine depuis le milieu des années 90 [1]. Il désigne principalement les systèmes de dialogue au format textuel. Récemment, de tels systèmes ont été construits avec des modèles de

1. Source : GOOGLE BOOKS NGRAM (`https://books.google.com/ngrams/interactive_chart?content=chatbot&year_start=1990&year_end=2019&corpus=en-2019&smoothing=0`)

langage, en réalisant un *finetuning* sur des corpus conversationnels et de questions-réponses (Brown et al. 2020 ; Touvron et al. 2023b).

3.3 Les systèmes de compréhension

La compréhension automatique de la parole est une tâche fortement liée à celle de la transcription de la parole. La compréhension peut être réalisée sur du texte obtenu automatiquement à partir de la transcription, et les résultats sont alors dépendants de la qualité de cette transcription. Si la transcription n'est pas une étape préliminaire de la compréhension, alors elles deviennent deux tâches à réaliser en même temps.

Nous décrivons en section 3.3.1 des modèles de compréhension du langage écrit, c'est à dire fonctionnant sur une transcription du signal de parole. Nous parlons alors de systèmes NLU (*Natural Language Understanding*). En section 3.3.2 nous présentons des modèles de compréhension travaillant directement sur le signal de parole. Nous parlons alors de systèmes SLU (*Spoken Language Understanding*) *end-to-end*.

3.3.1 Systèmes NLU

Les systèmes de compréhension de la parole à partir du texte (NLU), existent depuis les années 80. Historiquement, les premiers systèmes de compréhension utilisaient des systèmes à base de règles pour formaliser des grammaires de modélisation de la langue. Ensuite, les systèmes statistiques à base de HMM et CRF ont été utilisés. Depuis les années 2010, ce sont les réseaux de neurones qui sont largement utilisés pour résoudre la tâche NLU, à travers notamment l'utilisation de modèles récurrents. Plus récemment, toujours avec des architectures neuronales, ce sont les modèles *transformers* qui sont devenus la référence dans le domaine. Nous passons brièvement sur ces différents âges d'or des modèles de compréhension.

Grammaires et systèmes à base de règles (1960-2000). Les premiers systèmes de compréhension de la langue utilisent des grammaires (Weizenbaum 1966 ; De Mori 1983). Boisen et al. (1989) utilisent notamment un analyseur (*parser*) pour convertir la langue naturelle en langue formelle EFL (*English Formal Language*). Seneff (1992) décrit la grammaire sémantique et syntaxique Tina, en utilisant des statistiques calculées sur un ensemble d'entraînement. Décrit par Dowding et al. (1993), le système Gemini utilise également des grammaires pour la compréhension de la langue, avec une application

sur le corpus d'information de voyages aériens (ATIS). Pour plus d'informations sur la compréhension de la langue et les grammaires, nous conseillons le livre de Allen (1995).

HMM et CRF (2000-2015). Comme dans le domaine de la reconnaissance automatique de la parole, les modèles de Markov cachés (cf. 2.2.2) et d'autres réseaux bayésiens ont été utilisés pour la prédiction de concepts (Schwartz et al. 1996 ; Bonneau-Maynard et Lefèvre 2005) ou d'entités nommées (Zhou et Su 2002). Ensuite, les champs conditionnels aléatoires, ou CRF (*Conditional Random Field*) ont été employés pour cette tâche. Décrits par Lafferty et al. (2001), ces modèles ont été utilisés pour l'extraction de concepts (Lefèvre 2007 ; Vukotic et al. 2015). Les CRF ont permis d'améliorer les résultats par rapport aux modèles HMM, notamment grâce à une modification qui leur permet de supprimer des biais pour les étiquettes de sortie (Lafferty et al. 2001).

Réseaux de neurones profonds (2010-). Les réseaux de neurones sont depuis quelques années devenus la référence dans de nombreux domaines, et cela n'échappe pas aux systèmes de compréhension de la langue et de la parole. Des premiers travaux de compréhension de la langue avec des réseaux de neurones profonds furent initiés par Sarikaya et al. (2011) pour la redirection d'appels, avec un modèle génératif *Deep Belief Network*. Pour la tâche de *slot filling*, des réseaux de neurones ont été utilisés par Deng et al. (2012). Celui-ci devait produire la séquence de concepts à partir d'une séquence de mots. L'architecture choisie était un DNC (*Deep Convex Network*) (Deng et Yu 2011) à noyaux, c'est à dire un modèle composé de modules ou de classifieurs simples, superposés. Les modèles récurrents (RNN) ont ensuite été employés pour la NLU, dont la tâche de *slot filling* (Yao et al. 2013 ; Mesnil et al. 2013 ; Yao et al. 2014 ; Mesnil et al. 2015 ; Vukotic et al. 2015 ; Simonnet et al. 2015, 2017). En particulier, Simonnet et al. (2015) utilisent un réseau de neurones récurrent avec mécanisme d'attention pour générer les concepts à partir d'une séquence de mots. Par la suite Simonnet et al. (2018) arrivent à égaler les résultats CRF avec une architecture neuronale, grâce notamment à une simulation d'erreurs de transcription dans la référence utilisée lors de l'apprentissage.

***Transformers* (2019-).** Les modèles *transformers*, décrits en section 1.1.5, et introduits par Vaswani et al. (2017) ont initialement été utilisés pour la traduction automatique. Ils sont pré-appris pour modéliser la langue avec un objectif MLM (cf. section 1.2.2) qui engendre des relations syntaxiques et sémantiques dans le modèle (Jawahar et al. 2019). Ces relations sont utiles pour la compréhension. Ces modèles *transformers* sont ensuite rapidement devenus la référence état de l'art NLU. Le modèle BERT (Devlin

et al. 2019) est un modèle *transformer* pré-appris pour modéliser la langue, puis *finetuné* pour différentes tâches de compréhension : questions-réponses, classification sémantique, détections d'entités nommées, etc. Korpusik et al. (2019), Ghannay et al. (2020, 2021) et Cattan et al. (2022) ont utilisé des modèles de langage pré-appris, qu'ils ont adaptés pour l'extraction de segments sémantiques (*slot filling*). Ghannay et al. (2020) utilisent par exemple un modèle BERT français pour classifier les concepts à partir des mots d'entrée. Alajrami et Aletras (2022) entraînent des modèles BERT avec des objectifs linguistiques (dont le MLM) et non linguistiques (tels que la prédiction de codes ASCII). De façon surprenante, ils observent des résultats similaires entre ces modèles sur différentes tâches syntaxiques et sémantiques. Cela suggère que ces modèles apprennent principalement les informations de co-occurrences du corpus d'apprentissage.

La capacité, pour les (larges) modèles de langage, à comprendre la langue fait actuellement débat au sein de la communauté (Mitchell et Krakauer 2023). Certains considèrent que des modèles de langage statistiques pourraient comprendre la langue et le dialogue (y Arcas 2022). D'autres considèrent que les modèles de langage ne pourront jamais égaler la compréhension humaine avec un entraînement sur du texte et sans modèle mental (Bender et al. 2021). Cependant, cette idée n'est pas propre aux modèles *transformers* : les modèles HMM, ou à réseaux récurrents utilisés pour la compréhension sont appris sur des statistiques seulement, et ne pourraient donc pas comprendre au même titre qu'un humain. Notre objectif n'est pas de comprendre le langage comme nous le faisons, mais d'extraire des représentations sémantiques pertinentes pour une application donnée (cf. section 3.2.5).

3.3.2 Systèmes SLU

Cascade. La plupart des modules NLU décrits peuvent être appliqués sur les sorties d'un module d'ASR. Les deux modules dans leur ensemble sont alors appelés «système SLU cascade». Les deux modules peuvent être optimisés ensemble, de manière globale pour la métrique NLU cible (Jannet et al. 2017).

End-to-end. Les systèmes SLU *end-to-end* réalisent la tâche de compréhension à partir du signal de parole (ou une représentation telle que les MFCC ou Filter-Bank) pour générer les sorties de la tâche SLU, par exemple les concepts. Ces sorties peuvent également être accompagnées de la transcription automatique.

Ces systèmes ont vu le jour tardivement (années 2000), en comparaison avec ceux cascade (années 80). L'un des principaux objectifs de cette architecture unique est de

réduire la propagation des erreurs commises par le module ASR dans le module NLU (T. Hori et Nakamura 2006). Le modèle NLU est généralement appris sur des données sans erreurs, et est évalué sur des transcriptions automatiques comportant des erreurs, ce qui peut les amplifier. De plus, Price et al. (1991) ont démontré l'intérêt des informations acoustiques telles que la prosodie pour comprendre un message. Ces informations ne se retrouvent pas dans la transcription sous forme de mots. Cependant, la quantité de données de parole avec annotation sémantique a été un frein pour apprendre un modèle *end-to-end*. Les modèles ASR sont typiquement appris sur plusieurs centaines (Gauvain et al. 2002), voire milliers d'heures de parole (Radford et al. 2023), une quantité bien supérieure à la plupart des corpus avec annotations sémantiques. De plus, la reconnaissance automatique de la parole est une tâche complexe à elle seule, elle fait encore l'objet de nombreuses études. Un modèle SLU *end-to-end* doit réaliser la transcription et la compréhension, ce qui est, de fait, plus difficile qu'un modèle de reconnaissance de la parole seul (Serdyuk et al. 2018).

Les premières tentatives d'intégration *end-to-end* des deux modules furent réalisées pour la détection d'entités nommées (T. Hori et Nakamura 2006 ; Hatmi et al. 2013 ; Ghannay et al. 2018). Pour contrer le manque de données audio annotées en entités nommées, Ghannay et al. (2018) réalisent un apprentissage multi-tâches. Ils apprennent une première version de leur modèle neuronal pour réaliser seulement la tâche de transcription de la parole, sur des corpus hors-domaine. Puis, ils reprennent l'apprentissage sur les données avec annotation en entités nommées. Ghannay et al. voient ainsi la transcription de la parole comme une tâche de pré-entraînement à la SLU. Caubrière et al. (2019) ciblent eux la tâche d'extraction de concepts, à l'aide de cette même méthode. En outre, ils voient la tâche de NER (*Named Entity Recognition*) comme une tâche de pré-entraînement à la SLU. Ils apprennent ainsi leur modèle de façon incrémentale au niveau de la difficulté de la tâche : ASR puis NER puis SLU. Leur modèle est composé de couches convolutionnelles, de couches biLSTM, puis d'une couche linéaire. Il est appris à l'aide du CTC. Pour la détection d'intention, Serdyuk et al. (2018) apprennent un modèle RNN *end-to-end* (à partir du signal de parole).

Les modèles pré-entraînés de manière auto-supervisée (SSL) peuvent également être utilisés pour augmenter artificiellement la quantité de données d'apprentissage annotées en transcription. Ghannay et al. (2021) utilisent par exemple un modèle wav2vec *finetuné* pour la tâche de *slot filling*, mais les résultats ne sont pas à la hauteur en comparaison avec un modèle cascade, lui aussi composé de modèles SSL (wav2vec et CamemBERT). Pour une tâche de classification de l'intention, P. Wang et al. (2020) utilisent avec succès

un modèle *end-to-end* pré-appris sur plus de 4000 heures de parole.

3.4 Méthodologie : formats et évaluation

La tâche d'extraction de segments sémantiques, dans le cadre de la compréhension de la parole, nécessite de prédire des concepts et valeurs reposant sur des supports de mots issus de la transcription automatique (cf. section 3.1.2). Nous présentons en section 3.4.1 deux formats de sortie permettant aux systèmes de générer ces informations. Nous présentons en section 3.4.2 les métriques usuelles utilisées dans ce cadre.

3.4.1 Formats

Les deux formats les plus couramment utilisés pour la détection de concepts sémantique sont le BIO et le format balisé. Ces formats permettent d'aligner les mots de la transcription avec l'ensemble des concepts. Peu importe celui choisi, il est possible, de façon déterministe, de transformer un exemple d'un format à un autre. Il est également possible d'évaluer les sorties dans les deux formats par les mêmes métriques.

Le format BIO est un format provenant historiquement de la détection d'entités nommées (NER). Correspondant à *Beginning, Inside, Outside*, ce format repose sur la classification de chaque *token* de texte en étiquette de concept. Nous appelons *token* l'unité de sortie utilisée pour la transcription. Cette unité peut correspondre aux caractères, aux sous-mots, ou aux mots. Ce format force l'alignement entre le texte et les étiquettes BIO. Afin de pouvoir «étendre» un concept sur plusieurs *tokens*, la première occurrence d'une étiquette est préfixée de la notation `B-`, tandis que les autres étiquettes sont préfixés de `I-`. Si un mot ou *token* ne possède pas de classe sémantique vis-à-vis de l'annotation employée, son étiquette sera `O` pour *Outside*, signifiant que l'étiquette du *token* est en dehors du schéma d'annotation sémantique. Le format BIO permet ainsi d'étiqueter sémantiquement tous les *tokens*. Au total, pour un schéma d'annotation avec n_C concepts, $(n_C \times 2) + 1$ étiquettes BIO existent.
En Table 3.2 est présenté un exemple, issu du corpus MEDIA (section 3.5.4), avec ce schéma d'annotation.

Le format balisé a été utilisé pour la détection de concepts pour la première fois par Ghannay et al. (2018). Ces balises sont similaires à des balises XML ou HTML. Chaque concept sémantique possède sa balise, et est placé autour des groupes de mots supports de

tokens	étiquette BIO	concept
je	B-command-tache	command-tache
souhaite	I-command-tache	
réserver	I-command-tache	
euh	O	
une	B-nombre-chambre	nombre-chambre
chambre	I-nombre-chambre	
à	B-localisation-ville	localisation-ville
paris	I-localisation-ville	
le	B-temps-axetps	temps-axetps
premier	I-temps-axetps	
jeudi	B-temps-jour-semaine	temps-jour-semaine
de	B-temps-mois	temps-mois
décembre	I-temps-mois	

TABLE 3.2 – Exemple d'annotation en concepts avec le formalisme BIO et une *tokenization* en mots

ce concept. Un exemple est présenté en FIGURE 3.4. L'avantage principal de ce format est qu'il permet aux modèles de déterminer sur un même niveau les hypothèses de mots et également les hypothèses de concepts. C'est ainsi un format qui permet de construire un système *end-to-end*, sans passer par une transcription intermédiaire. L'autre avantage de ce format balisé, par rapport au format BIO, c'est qu'il permet directement d'annoter des groupes de mots, sans devoir annoter chaque *token* indépendamment. De plus, le modèle n'a pas besoin de différencier les étiquettes `B-` des étiquettes `I-`. L'inconvénient, c'est que le modèle mélange alors la transcription (le texte) et les étiquettes sémantiques dans la même représentation. La granularité utilisée pour les *tokens* –mots, sous-mots, caractères– influe sur la séquence à optimiser. Si chaque sortie possède le même poids vis-à-vis de l'optimiseur lors de l'apprentissage, alors les étiquettes peuvent se trouver plus ou moins défavorisées. Les concepts sont moins fréquents dans la sortie en cas de granualité fine (par exemple, caractères) qu'en cas de granularité plus large (par exemple, mots).

Pour favoriser une meilleure prédiction des concepts vis-à-vis des *tokens*, deux méthodes peuvent être employées : la première consiste, au niveau de la *loss*, à utiliser un poids plus important pour les concepts que pour les *tokens* ASR. La deuxième repose sur l'utilisation

du mode étoile, que nous détaillons ci-dessous.

`<command-tache>` je souhaite réserver `</>` euh `<nombre-chambre>` une chambre `</>` `<localisation-ville>` à paris `</>` `<temps-axetps>` le premier `</>` `<temps-jour-semaine>` jeudi `</>` `<temps-mois>` de décembre `</>`

FIGURE 3.4 – Exemple d'annotation en concepts avec le formalisme balisé.

Le mode étoile (`*`) a été introduit par GHANNAY et al. (2018) pour contrer cet effet lorsque la grannularité ASR est fine. Dans le format balisé, cette séquence contient à la fois les concepts, mais aussi les *tokens* de transcription. Le mode étoile (*starred mode*) permet de forcer le modèle à porter plus d'importance aux informations utiles pour l'évaluation sémantique. Toutes les séquences de caractères en dehors des concepts sont remplacées par le caractère `*`, car ils ne sont pas évalués par les métriques SLU (section 3.4.2). Avec ce mode, le *token* «`euh`» de l'exemple de la FIGURE 3.4 sera transformé en «`*`».

Le choix de l'utilisation d'un format plutôt qu'un autre repose principalement sur l'architecture du système que nous réalisons. Un modèle *end-to-end* doit à la fois déterminer la transcription ASR ainsi que son annotation sémantique. Si ces deux informations doivent être déterminées en même temps, le format balisé est ainsi généralement employé. On pourra également utiliser le mode étoile. Si l'architecture construite peut déterminer ces deux informations en deux temps, ou de façon séparée, ou bien que le système n'est pas *end-to-end*, alors le format BIO pourra être utilisé.

3.4.2 Les métriques d'évaluation

L'évaluation de la transcription se fait à l'aide des métriques de reconnaissance de la parole, en particulier le WER (voir section 2.4.3). L'évaluation des concepts prédits se fait soit à l'aide de métriques de classification, telles que la précision, le rappel, ou la f-mesure, soit à l'aide de la métrique du *Concept Error Rate* (CER).

La métrique du CER se calcule de la même façon que le WER, mais en considérant uniquement les concepts (équation 3.1). Le calcul correspond à la somme des erreurs de suppressions (D_C), d'insertions (I_C) et de substitutions (S_C), divisé par le nombre de concepts dans la référence (N_C).

$$\text{CER} = \frac{S_C + D_C + I_C}{N_C} \tag{3.1}$$

Par rapport aux métriques de classification, elle possède l'avantage d'évaluer le respect de la temporalité des concepts prédits. Les métriques telles que la précision et le rappel ne peuvent évaluer si les concepts sont placés dans l'ordre vis à vis de la transcription. En ce qui concerne le CER, en réalisant l'alignement entre la séquence de référence et d'hypothèses, nous pouvons comparer les deux, et évaluer non pas globalement, mais localement, les séquences.

Le CER ne permet d'évaluer que les concepts. L'évaluation des valeurs associées aux concepts [2] se réalise à l'aide de la métrique du CVER (*Concept-Value Error Rate*). Avec cette métrique, le couple (concept, valeur) est évalué. Ils doivent tous les deux être corrects pour que l'élément soit considéré correct. La procédure reste autrement identique au CER.

$$\text{CVER} = \frac{S_{\text{C-V}} + D_{\text{C-V}} + I_{\text{C-V}}}{N_{\text{C-V}}} \tag{3.2}$$

Nous présentons en FIGURE 3.5 un exemple de calcul de WER, CER et CVER à partir des sorties balisées d'un modèle. Nous présentons cet exemple pour deux hypothèses (H^1 et H^2), par rapport à la référence (R).

3.5 Les corpus de compréhension de la parole

Dans cette thèse, nous nous intéressons à la compréhension automatique de la parole dans le cadre de dialogues humain-machine, et en particulier l'extraction de concepts associés à la transcription. Nous décrivons dans cette section les principaux corpus, notamment en français. Les corpus utilisés au sein de nos travaux sont les corpus MEDIA et PortMEDIA.

3.5.1 ATIS et MultiAtis++

Le corpus ATIS [3] est l'un des corpus qui fut le plus utilisé en *slot filling*. Présenté par HEMPHILL et al. (1990), ATIS est un corpus de dialogues en anglais sur le domaine de l'information et la réservation de vols d'avions. Le scénario est un dialogue entre un humain et une machine l'aidant à planifier un voyage en avion. Le corpus ATIS est annoté selon un schéma d'annotation comportant 128 concepts et 26 classes d'intention. Nous présentons en TABLE 3.3 un exemple d'annotation d'un tour de parole utilisateur de ce

2. Ces valeurs doivent être prédites par un système (automatique). Nous discutons de cette prédiction en section 4.1.2.

3. «ATIS» : `LDC93S5` ; `LDC94S19` ; `LDC95S26`

	Sorties des modèles (format balisé)
R	`<cmd-tache>` je souhaite réserver `</>` euh `<nb-chambre>` une chambre `</>` `<localisation-ville>` à paris `</>`
H^1	`<reponse>` je voudrais réserver `</>` euh `<nb-chambre>` deux chambres `</>` `<localisation-ville>` à paris `</>`
H^2	`<reponse>` je souhaite réserver chambre `</>` `<localisation-ville>` à paris `</>`

Transcription : calcul du WER

R	je	souhaite	réserver	euh	une	chambre	à	paris	**S**	**D**	**I**	**WER**
H^1	je	voudrais	réserver	euh	deux	chambres	à	paris	3	0	0	$^3/_8$
H^2	je	souhaite	réserver	***	***	chambre	à	paris	0	2	0	$^2/_8$

Concepts : calcul du CER

R	`cmd-tache`	`nb-chambre`	`loc-ville`	**S**	**D**	**I**	**CER**
H^1	`reponse`	`nb-chambre`	`loc-ville`	1	0	0	$^1/_3$
H^2	`reponse`	`**********`	`loc-ville`	1	1	0	$^2/_3$

Concepts et valeurs : calcul du CVER

R	`cmd-tache:réservation`	`nb-chambre:1`	`loc-ville:paris`	**S**	**D**	**I**	**CVER**
H^1	`reponse:oui`	`nb-chambre:2`	`loc-ville:paris`	2	0	0	$^2/_3$
H^2	`reponse:oui`	`**********:*`	`loc-ville:paris`	1	1	0	$^2/_3$

FIGURE 3.5 – Exemple de calcul du WER, CER et CVER pour deux hypothèses par rapport à une référence.

corpus.

Plus récemment, le corpus MultiAtis++ [4] a été construit par Weijia XU et al. (2020) afin de diversifier ce corpus à 9 langues. UPADHYAY et al. (2018) avaient également conçu une version multilingue en Hindi et Turque [5] . Pour ces deux corpus multilingues, seules les transcriptions et leurs annotations sont traduites dans les langues cibles, les dialogues oraux n'étant pas ré-enregistrés.

Cependant, dès 2010, certains ont commencé à questionner la pertinence du corpus ATIS pour la compréhension de la parole, en raison notamment de la simplicité du corpus, et de la quantité de données disponibles dans celui-ci vis-à-vis des architectures de plus

4. «ATIS - Seven Languages» : `LDC2021T04`
5. «Multilingual ATIS» : `LDC2019T04`

Mots supports	Concept	Intention
what		*flight*
are		
the		
flights		
from		
pittsburgh	`fromloc.city-name`	
to		
denver	`toloc.city-name`	

TABLE 3.3 – Exemple d'annotation d'un segment utilisateur du corpus ATIS en concepts et avec une intention *flight*.

en plus puissantes et des taux d'erreurs faibles (TUR et al. 2010 ; BÉCHET et RAYMOND 2018).

3.5.2 MultiWOZ

MultiWOZ [6] est un corpus récent, réalisé de façon participative sur internet, et présenté dans sa première version par BUDZIANOWSKI et al. (2018). C'est un corpus multi-domaines de dialogues entre un humain et un ordinateur (simulé). Les segments sont annotés selon plusieurs schémas, et disposent en particulier d'une annotation en concepts. Dans la version 2.2 du corpus, 8 domaines sont présents (restaurant, attraction, hôtel, taxi, train, bus, hôpital, police), et 61 concepts sont utilisés (destination, nom, adresse, code postal, ...). Le corpus ne dispose cependant pas de versions orales, mais certains travaillent à l'élaboration de celle-ci, en particulier avec de la synthèse vocale (SOLTAU et al. 2022). Certains projets de recherche actuels visent également à enregistrer une version orale du corpus [7].

3.5.3 SpokenWOZ

Le corpus SpokenWOZ [8] (SI et al. 2023) est un corpus de conversations téléphoniques, en anglais, annotés de façon similaire à MultiWOZ. SpokenWOZ comporte 5700 dialogues répartis en 8 domaines (restaurant, hôtel, attraction, taxi, train, hôpital, police, et profil),

6. «MultiWOZ» : `budzianowski/multiwoz`
7. Projet «HumanE AI Network»
8. «SpokenWOZ» : `https://spokenwoz.github.io/SpokenWOZ-github.io/`

pour 249 heures de parole. L'annotation est effectuée avec 36 slots, répartis sur ces 8 domaines.

3.5.4 MEDIA

Le corpus MEDIA [9] est un corpus de dialogues français portant sur la réservation de chambres d'hôtels. Il a été réalisé dans le cadre du projet éponyme MEDIA-EVALDA, et présenté par Bonneau-Maynard et al. (2005). Il contient 1258 dialogues pour une durée totale de 70 heures de parole entre un humain souhaitant obtenir des informations et réserver des chambres, et un serveur vocal répondant à ses requêtes.

Le corpus MEDIA est enregistré en suivant la méthode du magicien d'Oz ou WOZ (*Wizard of Oz*), introduit par Kelley (1984). Dans le cas d'un dialogue entre un humain et une machine, la méthode WOZ consiste à remplacer la machine par un humain *se comportant* telle une machine. Cela permet d'obtenir un corpus de dialogues sans avoir à attendre la mise en œuvre d'un système automatique efficace.

Seuls les tours de parole de l'utilisateur sont annotés sémantiquement. Les groupes de mots de ces tours de parole qui portent une information sémantique pour la tâche de réservation de chambres d'hôtels sont annotés selon un schéma en plusieurs niveaux hiérarchiques.

Un jeu de 83 concepts est généré en utilisant les attributs de cette hiérarchie, mais seuls 77 sont présents dans les données. Il s'agit du jeu de concepts «*relax*». En utilisant les spécifieurs de ces attributs, 152 concepts sont présents dans les données pour le jeu «*full*», mais la hiérarchie employée permet de décrire 1121 concepts (Bonneau-Maynard et al. 2005 ; Laperrière et al. 2022a,b). Ces concepts permettent de décrire le contenu sémantique des mots supports, tout en gardant comme objectif la tâche finale de réservation de chambres d'hôtel. Nous pouvons citer des concepts tels que «`command-tache`», qui permet de spécifier le fait que l'utilisateur souhaite faire réaliser une action au système. Le concept «`localisation-ville`» permet lui d'annoter les noms de villes qui peuvent être indiquées par l'utilisateur.

Pour chaque support associé à un concept, une valeur est présente. Par exemple, les mots «une chambre» seraient associés au concept «`nombre-chambre`» et à la valeur «1». Les mots «le premier juillet» pourraient être associés au concept «`temps-date`» dans le jeu *relax* ou «`temps-date-debut`» dans le jeu *full*, avec comme valeur «01/07».

Enfin, un mode est associé à chaque annotation sémantique : positif, négatif, interrogatif,

9. «MEDIA» : `ELRA-E0024` ; `ELRA-S0272`

ou optionnel.

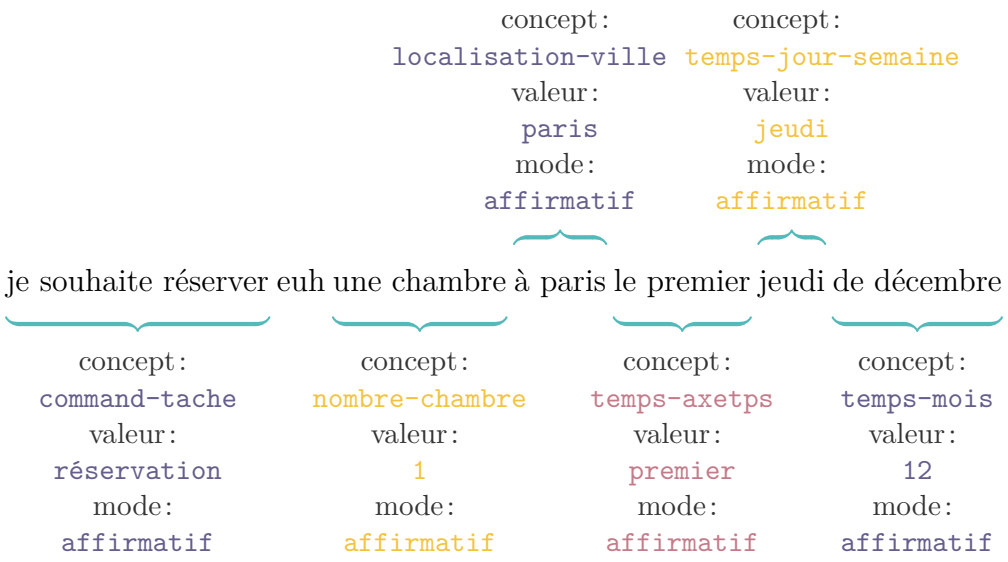


FIGURE 3.6 – Exemple du corpus MEDIA (jeu de DEV, dialogue 1452).

Nous présentons en FIGURE 3.6 un extrait de dialogue du corpus MEDIA, avec sa transcription, son annotation en concepts, leurs valeurs et leurs modes. La phrase initiale est «*je souhaite réserver euh une chambre à paris le premier jeudi de décembre*». Nous remarquons que dans cet exemple, tous les concepts sont associés au mode affirmatif.

3.5.5 PortMEDIA

Les corpus PortMEDIA [10] sont des corpus réalisés de façon similaire au corpus MEDIA, afin d'évaluer la portabilité du domaine sémantique, et également de la langue (LEFÈVRE et al. 2012).

PortMEDIA-Fr

Le corpus PortMEDIA-Fr, officiellement PM-DOM, est composé de dialogues téléphoniques en français qui portent sur le domaine de la réservation de billets pour le festival d'Avignon de 2010. Le même paradigme d'annotation que le corpus MEDIA a été suivi. Excepté ce changement de domaine, le corpus a été construit de façon à réduire le plus possible les différences avec MEDIA. Ce changement de domaine implique que certains concepts ont du être créés spécialement, et d'autres (propres à l'information touristique et

10. «PortMedia French and Italian corpus» : `ELRA-S0371`

aux chambres d'hôtels) ne sont plus utilisés. Au total, ce sont 36 concepts qui sont présents dans le jeu *relax* de ce corpus. 700 dialogues sont présents dans cette version du corpus.

PortMEDIA-It

Le corpus PortMEDIA-It, officiellement PM-LANG, a lui été conçu pour mesurer la portabilité des modèles de SLU selon la langue. Le corpus est une traduction en italien du corpus MEDIA, où le contenu des dialogues a été traduit puis enregistré en italien. Les noms de lieux et autres entités nommées sont restées tels quels. Le corpus n'est pas une traduction paire-à-paire du corpus MEDIA : nous ne disposons pas de l'alignement des dialogues entre eux, et seuls 604 dialogues sur les 1258 ont été traduits.

3.5.6 Récapitulatif

Nous venons de présenter quelques-uns des corpus les plus couramment utilisés en compréhension automatique de la parole, en particulier pour le français. Dans cette thèse, nous nous concentrons principalement sur le français, ainsi nous allons employer les corpus MEDIA, et PortMEDIA. Nous présentons dans la TABLE 3.4 une description de ces corpus, pour les jeux TRAIN, DEV et TEST, en nombre de dialogues, d'occurrences de mots et de concepts, ainsi qu'en nombre d'heures de parole. Ces chiffres sont donnés uniquement pour les tours de paroles de l'utilisateur.

		Train	**Dev**	**Test**	***Total***
Dialogues	MEDIA	727	79	208	*1014*
	PortMEDIA-Fr	400	100	200	*700*
	PortMEDIA-It	300	104	199	*603*
Mots	MEDIA	94.5k	10.8k	26.7k	*131.9k*
	PortMEDIA-Fr	42.3k	11.1k	21.6k	*75.0k*
	PortMEDIA-It	21.8k	7.7k	14.8k	*44.3k*
Concepts	MEDIA	31.7k	3.3k	8.8k	*43.8k*
	PortMEDIA-Fr	10.0k	2.6k	5.2k	*17.8k*
	PortMEDIA-It	9.8k	3.5k	6.6k	*19.9k*
Heures	MEDIA	16h56	1h40	4h47	*23h23*
	PortMEDIA-Fr	7h21	1h46	3h34	*12h41*
	PortMEDIA-It	7h26	2h36	4h53	*14h55*

TABLE 3.4 – Tailles réellement utilisées des corpus MEDIA, PortMEDIA-Fr et PortMEDIA-It pour les jeux TRAIN, DEV et TEST, pour les tours de parole de l'utilisateur.

3.6 Conclusion

Dans ce chapitre, nous avons présenté la compréhension automatique de la parole. Nous nous sommes intéressés aux représentations sémantiques permettant de contenir le sens du signal de parole (SLU pour *Spoken Language Understanding*) ou des mots (NLU pour *Natural Language Understanding*). Ces représentations sémantiques, à base de *frames*, de segments sémantiques, ou plus générales, peuvent être utilisées pour résoudre certaines tâches nécessitant une compréhension de la langue. Nous nous intéressons en particulier aux systèmes de dialogue humain-machine, qui permettent à un humain de dialoguer avec une machine. Pour tenter de résoudre cette tâche, les systèmes de compréhension peuvent être construits de façon cascade, en travaillant sur des transcriptions automatiques d'un module d'ASR. Lorsque suffisamment de données de parole sont annotées avec des représentations sémantiques, la compréhension peut être effectuée de manière *end-to-end* avec une seule architecture.

Nous avons terminé ce chapitre par la présentation de différents corpus NLU et SLU. Dans cette thèse, nous utilisons les corpus MEDIA, un corpus français de réservation de chambres d'hôtels par téléphone, PortMEDIA-It, sa variante en italien, et PortMEDIA-Fr un corpus français d'informations touristiques, par téléphone également.

Au sein des prochains chapitres, nous présentons les contributions apportées par cette thèse. Dans le chapitre 4, nous présentons des architectures neuronales récurrentes, en configuration *end-to-end*, pour la compréhension de la parole. Dans le chapitre 5 nous utilisons des modèles *transformers* pré-entraînés de manière auto-supervisée (SSL), en configuration *end-to-end* et cascade. Enfin, nous comparons au sein du chapitre 6, différents modèles SSL pour la compréhension.

Deuxième partie

Contributions

Chapitre 4

Architectures récurrentes pour la compréhension end-to-end

Sommaire

4.1 Une architecture encodeur-décodeur 96
4.1.1 Protocole expérimental 96
4.1.2 Extraction des valeurs associées aux concepts 99
4.1.3 Détail de l'architecture 100
4.1.4 Résultats obtenus 103
4.2 Hybridation des systèmes *end-to-end* et cascade 107
4.2.1 Pourquoi une architecture hybride ? 108
4.2.2 Détail de l'architecture employée 109
4.2.3 Résultats obtenus 112
4.3 Conclusion 114

Dans ce premier chapitre concernant les contributions apportées par cette thèse, nous nous intéressons tout d'abord aux architectures neuronales récurrentes pour la compréhension automatique de la parole *end-to-end*. En section 4.1, nous présentons notre première architecture *end-to-end* de compréhension de la parole pour les corpus MEDIA et PortMEDIA. En section 4.2, nous décrivons une architecture hybride, *end-to-end* et cascade, afin d'intégrer des avantages des systèmes cascade au modèle précédemment réalisé. Cette architecture est construite en utilisant de multiples décodeurs, de façon à réaliser les tâches successivement, tout en gardant un apprentissage *end-to-end*.

4.1 Une architecture encodeur-décodeur

Nous avons présenté dans Pelloin et al. (2021) une architecture neuronale *end-to-end* pour la compréhension automatique de la parole. Nous détaillons en section 4.1.1 le protocole expérimental de l'apprentissage et de l'évaluation de cette architecture. En section 4.1.3 nous décrivons l'architecture employée, en section 4.1.2 nous parlons des méthodes d'extraction des valeurs normalisées. En section 4.1.4 nous présentons et analysons les résultats de cette architecture.

4.1.1 Protocole expérimental

Les données SLU sont disponibles en quantité limitée, et nous ne pouvons pas apprendre directement un modèle *end-to-end* à partir de celles-ci uniquement. De ce fait, nous apprenons nos modèles selon une méthode de transfert d'apprentissage, telle que présentée par Caubrière et al. (2019). Cette méthode nous permet en premier lieu de construire un modèle réalisant la transcription, en utilisant une quantité suffisante de données. Ensuite, nous adaptons ce modèle à notre tâche SLU, avec nos données MEDIA et PortMEDIA-Fr en quantité restreinte. La procédure détaillée est décrite ci-dessous, et résumée dans la Table 4.1. Nous avons choisi la représentation en segments sémantique (cf. section 3.1.2) avec un format balisé (cf. section 3.4.1) en ce qui concerne la sortie SLU attendue. Des exemples de sortie pour chaque format employé par nos modèles sont disponibles dans la Table 4.2.

Le premier modèle, nommé **ASR**, est tout d'abord appris en utilisant 414h de Train de corpus français d'actualité, ainsi que MPM (MEDIA et PortMEDIA). Tous ces corpus sont décrits en sections 2.5 et 3.5. Le modèle est ensuite *finetuné* uniquement sur les transcriptions des corpus MPM dans le but d'adapter les transcriptions aux spécificités des

conversations téléphoniques. Nous obtenons ici un modèle de transcription automatique de la parole, adapté aux corpus MPM, nommé **ASR MPM**.

L'apprentissage du modèle est ensuite repris, toujours sur le corpus MPM, en ajoutant les étiquettes sémantiques SLU aux transcriptions à prédire. Ce système est nommé **SLU$_{\text{Mots}}$ MPM**. Le modèle ASR est ainsi transformé en modèle de compréhension automatique de la parole. Nous considérons le corpus PortMEDIA-Fr principalement comme un corpus nous permettant d'augmenter notre quantité de données SLU, mais celui-ci est également évalué. Son schéma d'annotation est similaire au corpus MEDIA, avec quelques concepts en commun. Empiriquement, nous avons cependant constaté que réaliser une deuxième étape d'apprentissage SLU, en réalisant un *finetuning* uniquement sur le corpus cible (MEDIA, ou PortMEDIA-Fr) nous permet d'améliorer nos résultats d'environ 2.4%, en absolu, sur le corpus MEDIA Test, et 1.5% sur le corpus MEDIA Dev. Nous réalisons donc cette étape de *finetuning* sur le corpus cible, MEDIA ou PortMEDIA-Fr en fonction de l'expérience en question (**SLU$_{\text{Mots}}$ X**).

Afin de pouvoir nous comparer avec les sorties des modèles de Caubrière et al. (2019), nous avons besoin de produire les sorties en configuration étoile également, tel que décrit en section 3.4.1. Pour cela, nous réalisons un *finetuning* du modèle précédent (optimisé sur le corpus cible) sur les données SLU en configuration étoile (**SLU***). Les mots hors-concept sont remplacés par le caractère $*$.

Conf.	**Étapes d'apprentissage**	
mots	ASR → ASR MPM → SLU$_{\text{Mots}}$ MPM → SLU$_{\text{Mots}}$ X	
étoile	”	→ SLU* X
valeurs	”	→ SLU$^*_{\text{Norm}}$ X

Table 4.1 – Étapes d'apprentissage des modèles de compréhension de la parole, en fonction de la configuration (mots, étoile, valeurs), pour la tâche cible X = MEDIA ou X = PortMEDIA-Fr.

L'étape SLU* X ($X \in \{\text{MEDIA}, \text{PortMEDIA-Fr}\}$) correspond aux données SLU du corpus en configuration étoile, tandis que SLU$^*_{\text{Norm}}$ X correspond à l'apprentissage sur les données en configuration étoile et avec les valeurs normalisées. Nous décrivons cette configuration en section 4.1.2.

Lors de l'ajout des concepts MPM aux sorties du modèle («ASR MPM → SLU MPM»), nous redimensionnons simplement la couche linéaire de sortie. Les poids correspondant aux caractères déjà existants sont conservés. Lors de l'étape de *finetuning* sur le corpus X uniquement, nous enlevons les sorties correspondant aux concepts présents uniquement

Format	Exemple de sortie
ASR	j·e␣s·o·u·h·a·i·t·e␣r·é·s·e·r·v·e·r␣e·u·h␣u·n·e␣c·h·a·m·b·r·e
SLU$_{\text{Mots}}$	<command-tache>␣j·e␣s·o·u·h·a·i·t·e␣r·é·s·e·r·v·e·r␣>␣e·u·h␣<nombre-chambre>␣u·n·e␣c·h·a·m·b·r·e␣>
SLU*	<command-tache>␣j·e␣s·o·u·h·a·i·t·e␣r·é·s·e·r·v·e·r␣>␣*␣<nombre-chambre>␣u·n·e␣c·h·a·m·b·r·e␣>
SLU$^{*}_{\text{Norm}}$	<command-tache>␣r·é·s·e·r·v·a·t·i·o·n␣>␣*␣<nombre-chambre>␣1␣>

TABLE 4.2 – Exemple de sortie de chaque format (ASR, SLU_{Mots}, SLU^{*} ou $\text{SLU}^{*}_{\text{Norm}}$). Les espaces sont représentés par le caractère «␣», et le caractère «·» permet de visualiser la séparation des mots en caractères.

dans l'autre corpus. Le modèle ne peut donc plus générer de concepts hors domaine pour le corpus X.

Nous utilisons la segmentation TRAIN, DEV et TEST officielle des corpus. Les hyperparamètres d'apprentissage (*learning rate*, *weight decay*, *early stopping*, ...) sont optimisés manuellement sur les jeux DEV des corpus de l'étape en question. Les paramètres de décodage (largeur du *beam search*, pénalités de longueur, poids du modèle de langage, ...) sont optimisés automatiquement sur les jeux DEV, avec un algorithme TPE [1], grâce à la libraire HYPEROPT (BERGSTRA et al. 2013). L'optimiseur d'apprentissage du modèle est ADAM (KINGMA et BA 2015).

La tâche de compréhension MEDIA est perçue, dans notre cas, comme une tâche avancée de transcription automatique de la parole. En effet, les modèles décrits doivent réaliser une transcription de la parole auquel des informations supplémentaires sont ajoutées, les concepts. Nous avons choisi d'utiliser la librairie ESPRESSO (Yiming WANG et al. 2019) car celui-ci est suffisamment modulable, tout en disposant de recettes préétablies pour concevoir rapidement un système fonctionnel. En 2020, les *toolkits* populaires aujourd'hui tels que Pytorch-Lightning, ou SpeechBrain n'existaient pas, ou étaient à leurs balbutiements. Début 2020, ESPRESSO nous semblait posséder plus de fonctionnalités que son *concurrent* direct, ESPNET. ESPRESSO est une librairie de transcription automatique de la parole, basée sur le *toolkit* FAIRSEQ (OTT et al. 2019). FAIRSEQ est à l'origine un *toolkit* conçu pour différentes tâches séquence-vers-séquence, dont la traduction, la modélisation de la langue, et le résumé automatique de textes.

1. *Tree of Parzen Estimators (TPE)*

4.1.2 Extraction des valeurs associées aux concepts

Nous avons décrit dans la section 3.5.4 les valeurs normalisées du corpus MEDIA. Traditionnellement, dans les travaux sur le corpus MEDIA, la prédiction de ces valeurs normalisées est réalisée à partir de la transcription hypothèse et des concepts prédits, en utilisant un système à base de règles, développé par Raymond et al. (2006) et présenté brièvement par Servan et al. (2006). Ce système à base de règles a largement été utilisé dans la communauté. Nous pouvons par exemple citer Hahn et al. (2011), Simonnet et al. (2017), Ghannay et al. (2018), Caubrière et al. (2019), Ghannay et al. (2021) et Laperrière et al. (2022b).

Il s'agit d'un ensemble de règles reposant sur des expressions régulières manuellement décrites pour la tâche MEDIA, réalisées par une étude minutieuse du jeu d'entraînement Train. Quelques unes de ces règles de normalisation sont reportées à titre d'exemple dans la Table 4.3.

Concept	**Règle**	**Valeur**
command-tache	`/.*(location|réserv|prend|louer|séjour|retenir|trouver|opter|retiens|retenez|choisir).*/`	`reservation`
command-tache	`/.*(plus|autre).*(indication|précision|chose|détail|info|renseignement).*/`	`information+`
reponse	`/.*(peut.être|éventuellement|c'.est.possible|ça.dépend).*/`	`#peutEtre`
temps-mois	`/.*janvier.*/`	`1`
temps-mois	`/.*juillet.*/`	`7`
temps-mois	`/.*décembre.*/`	`12`

Table 4.3 – Exemples de règles de normalisation des valeurs.

Sur la base de cette grammaire, des règles similaires ont été réalisées par Nathalie Camelin, afin d'extraire les valeurs normalisées du corpus PortMEDIA-Fr.

Ces règles ne sont cependant pas triviales à construire. Par exemple, certaines phrases peuvent faire appel à des synonymes, ce qui complique la tâche. Hahn et al. (2011) donnent l'exemple de la phrase «je voudrais une chambre à pas plus de cinquante euros», où «pas plus de» correspond au concept «comparatif-paiement-chambre», avec une valeur «moins que». Hahn et al. (2011) ont effectué une comparaison de ces règles d'extractions (déterministes) de valeurs pour MEDIA avec une extraction stochastique. Cette extraction stochastique est effectuée à l'aide de CRF (*Conditional Random Field*) – champ aléatoire

conditionnel – ou de DBN (*Dynamic Bayesian network*) – réseau bayésien dynamique. Ils observent des résultats similaires entre la méthode stochastique et déterministe, avec cependant un léger gain des règles par rapport à l'approche stochastique.

Nous souhaitons nous rapprocher du monde réel, où nous ne disposerions pas des ressources humaines suffisantes pour concevoir ces règles pour un corpus différent. Pour cela, nous transformons l'objectif de notre système SLU : il devait auparavant prédire la transcription et les classes sémantiques. Il doit, dans cette expérience, prédire les valeurs normalisées à la place de la transcription. Dans l'exemple «je souhaite réserver euh une chambre à paris le premier jeudi de décembre» de la section 3.5.4, le modèle doit donc prédire la séquence de caractères suivante [2] :

```
<command-tache> r·é·s·e·r·v·a·t·i·o·n </> * <nombre-chambre> 1 </> <localisati
 on-ville> p·a·r·i·s </> <temps-axetps> p·r·e·m·i·e·r </> <temps-jour-semaine>
                    j·e·u·d·i </> <temps-mois> 1·2 </>
```

Pour cette expérience, nous réalisons un *finetuning* sur les données avec valeurs normalisées du modèle SLU_{Mots}. Les mots hors concepts sont, comme dans le configuration étoile, remplacés par des étoiles.

Avec cette méthode, le modèle réalise une transduction directe du signal vers la représentation sémantique, sans produire la transcription, qui est pourtant la principale information audible dans le signal de parole. Les séquences générées correspondent uniquement à des informations sémantiques latentes. Avec cette représentation, deux questions se posent : le modèle, initialement prévu pour faire de la transcription de la parole, sera-t-il en mesure de générer des séquences d'informations latentes uniquement ? Le modèle de langage, prévu initialement pour prédire la probabilité de séquences de mots, sera-t-il lui-aussi capable de manier ces informations ?

4.1.3 Détail de l'architecture

Nous présentons en FIGURE 4.1 l'architecture utilisée par notre premier système *end-to-end* récurrent. Pour ce système, nous avons choisi une représentation des unités de sortie (*tokens*) en caractères. Les étiquettes sémantiques (concepts), sont elles représentées avec le format balisé, cf. le format SLU_{Mots} de la TABLE 4.2.

Cette architecture est basée sur un encodeur et un décodeur. L'encodeur est composé de couches convolutionnelles à deux dimensions afin de réaliser un traitement de filtrage sur les

2. Nous n'avons pas indiqué les espaces entre les mots. Dans notre système, les espaces sont représentés par un *token* `<space>`.

données d'entrées (*Mel-FilterBanks*). Quatre couches bidirectionnelles LSTM (*Bidirectional Long Short-Term Memory*) sont ensuite utilisées. Le décodeur est quand à lui composé de quatre couches unidirectionnelles LSTM, afin de produire les caractères de sortie au fur et à mesure. Des couches linéaires et une fonction SOFTMAX sont ensuite employées. Le nombre de couches, et la taille des dimensions cachées de celles-ci ont été déterminés en réalisant un *finetuning* manuel de ces paramètres sur le jeu DEV, à partir des hyper-paramètres par défaut de ESPRESSO.

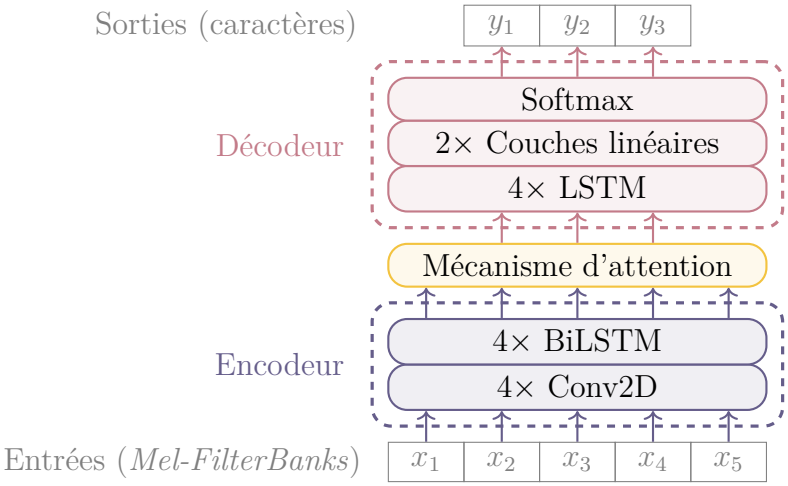


FIGURE 4.1 – Détail de l'architecture récurrente de compréhension de la parole *end-to-end*.

L'encodeur et le décodeur travaillent ensemble grâce à un mécanisme d'attention de BAHDANAU et al. (2015). Ce mécanisme nous permet de construire une séquence de sortie d'une taille différente que celle de la séquence d'entrée. Il permet également au décodeur de se focaliser sur l'information acoustique adéquate. Les vecteurs d'entrée du décodeur correspondent aux états cachés du mécanisme d'attention H^{att}, qui sont calculés tels que présentés dans l'équation 4.2, avec M la longueur de la séquence d'entrée et N celle de sortie. Le vecteur $\overleftrightarrow{h_j}$ correspond à l'état caché de la dernière couche de l'encodeur pour la trame j.

$$H^{\text{att}} = (H_1^{\text{att}}, \ldots, H_N^{\text{att}}) \tag{4.1}$$

$$H_i^{\text{att}} = \sum_j^M \alpha_{ij} \overleftrightarrow{h_j} \quad \text{avec} \quad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{M} exp(e_{ik})} \tag{4.2}$$

Modèle de langage additionnel

Notre modèle génère une séquence de caractères (et de concepts) comme sortie. Nous

améliorons cette sortie grâce à un modèle de langage ayant des probabilités au niveau mot, car il s'agit d'une meilleure représentation pour les séquences de transcriptions automatiques des langues avec un nombre de caractères réduits (Graves et Jaitly 2014 ; Bojanowski et al. 2016 ; T. Hori et al. 2017, 2019), tels que l'anglais ou le français.
Le modèle de langage additionnel employé repose sur des couches LSTM. Le modèle encodeur-décodeur émettant en sortie des séquences de caractères, nous utilisons une technique dite *look-ahead* qui permet de déterminer des probabilités au niveau caractère, même si le modèle est lui appris au niveau mot. Cette technique, présentée par T. Hori et al. (2019) pour la tâche de transcription de la parole, permet d'obtenir de meilleurs résultats que l'utilisation d'un modèle de langage au niveau caractère. En effet, elle permet de modéliser la langue même pour des mots hors vocabulaire ce qui est notamment intéressant pour des noms propres tels que les noms de ville. Pour obtenir des probabilités au niveau caractère à partir d'un modèle niveau mot, un arbre préfixe est construit. Un exemple d'arbre préfixe pour un vocabulaire composé de neuf mots est présenté dans la Figure 4.2. La probabilité *look-ahead* d'une séquence de caractères correspond à la somme des probabilités des mots escomptés pour cette séquence. Ainsi, sur l'exemple, la probabilité de la séquence «c·a·m·p·a» correspond à la somme des probabilités des mots «campagne» et «campanille» (dans cet exemple, 0.007). Les détails d'implémentation de cette technique *look-ahead* se trouvent dans l'article de T. Hori et al. (2019).

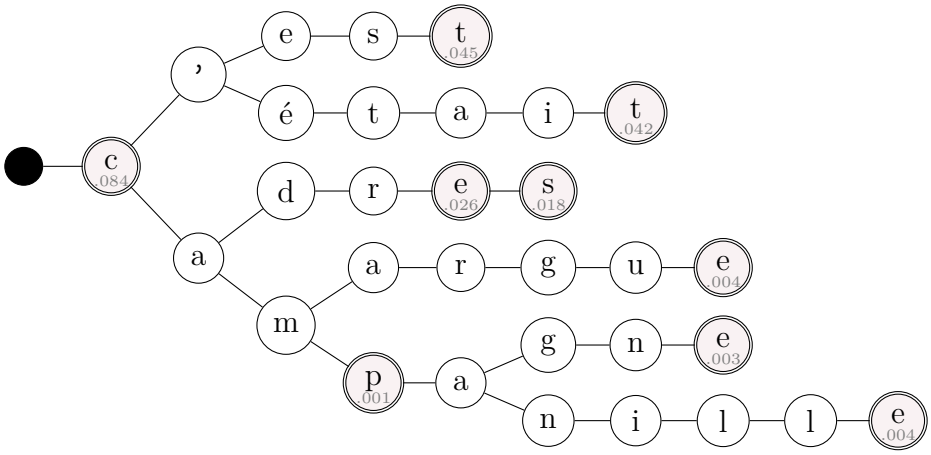


Figure 4.2 – Arbre préfixe pour un vocabulaire exemple composé des mots : «c», «c'est», «c'était», «cadre», «cadres», «camargue», «camp», «campagne» et «campanille».

Lors du décodage avec un *beam search* (cf. section 2.4.2), nous employons la technique du *shallow fusion*, décrite par Gulcehre et al. (2015) et présentée en section 2.4.1 afin de fusionner les probabilités de l'encodeur-décodeur et du modèle de langage. Nous n'utilisons pas de modèle de langage lors du décodage glouton.

Nous avons appris un modèle de langage ASR pour le décodage de nos expériences exclusivement lié à la transcription. Lors du décodage SLU, nous utilisons d'autres modèles de langage appris eux sur les données SLU, comportant les concepts sous forme de balises. De manière générale, nous utilisons un modèle de langage appris sur les données au même format que les données du modèle de transcription et/ou de compréhension. Nous avons ainsi un modèle de langage ASR MPM, un modèle de langage SLU MPM ou MEDIA, un modèle de langage SLU* ou SLU$^*_{\text{Norm}}$ MEDIA.

Notre système récurrent encodeur-décodeur est un modèle dit *end-to-end* car il s'agit d'un seul modèle appris de façon unifiée. Pouvons nous ajouter un modèle de langage additionnel, appris séparément, et toujours considérer notre architecture comme une architecture *end-to-end* ? Nous pensons tout d'abord que la frontière entre système *end-to-end* et cascade n'as pas vocation à rester ferme, au point de ne plus considérer un système *end-to-end* avec modèle de langage comme tel. De plus, nous pensons que la principale raison pour laquelle un système est dit *end-to-end* est parce qu'il réalise toute la tâche souhaitée à lui seul. Sans modèle de langage, notre architecture permet toujours de réaliser l'extraction des concepts. Un système cascade ne peut lui pas réaliser la tâche sans le système *downstream*.

En pratique, les modèles sont appris sur des serveurs disposant de 8 GPU Nividia Tesla K40. Les temps d'apprentissage varient largement en fonction de l'étape en question. La première étape «ASR» nécessite 42h avec 16 GPUs, et les étapes suivantes requièrent toutes moins de 7h d'apprentissage.

4.1.4 Résultats obtenus

Résultats sur le corpus MEDIA

Nous nous comparons aux résultats *end-to-end* de l'époque, par Caubrière et al. (2019). Ceux-ci sont également obtenus à l'aide d'un modèle récurrent. En dehors des mots supports, ce modèle état de l'art est configuré soit pour prédire les mots, soit en configuration étoile (voir section 3.4.1). Nous présentons en Table 4.4 les résultats sur le corpus MEDIA (Dev et Test), avec et sans modèle de langage. Les résultats sont donnés en CER (*Concept Error Rate*) et CVER (*Concept-Value Error Rate*).

Nous calculons les intervalles de confiance à 95% selon la loi de Student, avec $t^{\infty}_{95\%} = 1.96$ et la formule $I_{\text{conf}} = 1.96\sqrt{\text{CER} \times \frac{1-\text{CER}}{N_C}}$. Nous les calculons uniquement pour nos résultats, et nous reportons dans les Tables 4.4 et 4.5 l'intervalle de confiance le plus

élevé pour chaque catégorie de résultats (CER ou CVER, Dev ou Test).

MEDIA	Conf.	Dev (%) CER	Dev (%) CVER	Test (%) CER	Test (%) CVER
Sans modèle de langage (no-LM)					
Caubrière et al. (2019)	*mots*	–	–	*21.6*	*27.7*
	mots	18.1	22.5	15.6	**20.4**
	étoile	17.3	22.0	15.6	20.5
	valeurs	16.0	21.9	**15.4**	21.7
Avec un modèle de langage (LM)					
Caubrière et al. (2019)	*mots*	–	–	*18.1*	*22.1*
	étoile	–	–	*16.4*	*20.9*
	mots	16.1	20.4	**13.6**	**18.5**
	étoile	17.6	22.5	15.5	20.5
	valeurs	16.1	22.0	15.4	21.6
Intervalle de confiance maximum		±1.31	±1.42	±0.76	±0.86

Table 4.4 – Résultats en CER et CVER sur le corpus MEDIA (Dev et Test). Les résultats sont comparés avec ceux états de l'art *end-to-end* de Caubrière et al. (2019).

Notre meilleur résultat est obtenu en utilisant un modèle de langage, en configuration mots. Il obtient 13.6% de CER, tandis que le résultat état de l'art obtient 16.4% de CER en configuration étoile avec un modèle de langage. Il s'agit d'un gain en absolu de 2.8%.

Nous pouvons également remarquer que les systèmes réalisant les meilleurs CER sans modèle de langage, sont obtenus en utilisant la représentation «valeurs». Cette modélisation est donc au moins suffisante, vis à vis des autres «mots» et «étoile», pour permettre au modèle de générer correctement les concepts.

La modélisation «étoile», utilisée par Caubrière et al. (2019) au sein de leur système CTC, améliorait les résultats. De notre coté, cette modélisation n'améliore pas les résultats par rapport à la représentation «mots». Dans le cas avec modèle de langage, cette représentation dégrade même les résultats. Nous pensons que la différence d'architecture pourrait être à l'origine de cette dégradation. Avec un mécanisme d'attention, la longueur de la séquence de sortie n'est pas contrainte à être égale à celle d'entrée. Avec le CTC, elles doivent être de même longueur, et le caractère $*$ est émis pour chaque trame acoustique correspondante. Dans le cas de notre système, c'est le rôle du mécanisme d'attention de trouver un alignement entre un seul caractère et les trames correspondantes, et nous pensons que cette modélisation n'est alors pas adaptée. Caubrière et al. (2019) utilisent un modèle de langage 5-gramme au niveau mots. La caractéristique *look-ahead* de notre modèle de langage neuronal n'est également peut être pas adaptée au caractère $*$.

En ce qui concerne l'extraction des valeurs normalisées pour le CVER, nous pouvons remarquer de très bons résultats, pour un système novateur n'utilisant pas les grammaires établies à partir de connaissances humaines. En particulier, sans modèle de langage nous n'avons que 1.3% de différence entre le CVER obtenu à l'aide des grammaires humaines, et celui calculé de manière entièrement automatique. Malheureusement, l'ajout du modèle de langage n'apporte par de gains pour le configuration «valeurs», alors qu'il apporte un gain de 1.9% pour le CVER «mots». Nous pensons que ce modèle de langage *lookahead* n'est pas adapté aux modélisations autres que celle «mots».

Résultats sur le corpus PortMEDIA-Fr

Comme pour le corpus MEDIA, nous comparons nos résultats avec ceux état de l'art du corpus PortMEDIA-Fr, par Caubrière et al. (2019). Cependant, les résultats sans modèle de langage ne sont pas publiés par Caubrière et al. (2019) pour ce corpus. En Table 4.5 sont présentés nos résultats pour le corpus PortMEDIA, Dev et Test. Nous reportons également les intervalles de confiance selon la loi de Student.

PortMEDIA-Fr	**Conf.**	**Dev (%)**		**Test (%)**	
		CER	**CVER**	**CER**	**CVER**
Sans modèle de langage (no-LM)					
	mots	20.7	27.6	20.7	**26.9**
	étoile	20.3	27.9	20.9	28.1
	valeurs	19.7	35.8	**20.3**	36.1
Avec un modèle de langage (LM)					
Caubrière et al. (2019)	*mots*	–	–	*21.1*	*35.9*
	étoile	–	–	*21.9*	*36.9*
	mots	18.5	24.9	**17.6**	**23.6**
	étoile	21.4	28.2	21.9	28.2
	valeurs	20.7	36.1	21.6	36.0
Intervalle de confiance maximum		±1.58	±1.85	±1.12	±1.30

Table 4.5 – Résultats en CER et CVER sur le corpus PortMEDIA-Fr (Dev et Test). Les résultats sont comparés avec ceux états de l'art *end-to-end* de Caubrière et al. (2019).

Comme pour MEDIA, nous pouvons remarquer que les meilleurs résultats sans modèle de langage sont obtenus avec la configuration «valeurs». Au total, nos meilleurs résultats sont également obtenus avec un modèle de langage et la représentation «mots». Par rapport au meilleur résultat état de l'art de Caubrière et al. (2019), nous remarquons une amélioration de 2.7% en CER (17.6% contre 20.3%).

Globalement, l'extraction des valeurs normalisées avec le système *end-to-end* fonctionne moins bien sur le corpus PortMEDIA-Fr que sur le corpus MEDIA. Sur le Test, entre les modèles «mots» et «valeurs», nous observons une dégradation en absolu du CVER de 9.2% sans modèle de langage, et 12.4% avec le modèle de langage. Sur MEDIA (Table 4.4), la dégradation n'était que de seulement 1.3% et 3.1% respectivement.

Résultats en *Word Error Rate*

Nous calculons le WER de nos modèles afin d'évaluer la qualité de la transcription réalisée par ceux-ci. Cela est fait pour la configuration «mots», en enlevant à posteriori de nos sorties les étiquettes des concepts. Cette évaluation est faite pour nos modèles optimisés MEDIA mais aussi PortMEDIA-Fr. Les configurations «étoile» et «valeurs» ne sont pas évaluées avec cette métrique, car toutes les informations de transcription ne sont pas présentes, et nous ne pouvons donc pas comparer les configurations entre elles. Comme pour l'évaluation en CER et CVER, celle-ci n'est effectuée que pour la partie utilisateur des corpus, et non celle du magicien d'Oz (WOZ). Les résultats sont présentés en Table 4.6.

WER (%)	**Dev**	**Test**
MEDIA		
	12.4	12.1
LM	12.9	12.0
PortMEDIA-Fr		
	11.5	12.1
LM	13.2	13.6

Table 4.6 – Résultats en WER sur les corpus MEDIA et PortMEDIA-Fr (Dev et Test) pour la partie utilisateur seulement, avec le modèle en configuration «mots».

Pour MEDIA, nous n'observons pas de gain significatif avec l'ajout du modèle de langage. Pourtant, celui-ci a permis au modèle d'améliorer de 2% le CER sur le Test (Table 4.4). Cela indique donc que le modèle de langage a surtout été utile pour améliorer la sortie sémantique, et non de transcription. Pour PortMEDIA-Fr, nous observions un gain avec l'ajout du modèle de langage plus important, de 3.1% de CER sur le Test (Table 4.5). Cependant, celui-ci a dégradé la transcription de 1.5% de WER sur le Test. Dans l'ensemble, il semble donc que le modèle de langage *lookahead* dans la configuration «mot» permet surtout d'améliorer les prédictions sémantiques (CER et CVER), mais n'améliore pas les prédictions de transcription.

En résumé

Un résumé des récents résultats état de l'art, toutes architectures confondues, est présenté en Table 4.7. En comparant les travaux de Caubrière et al. (2019), et les nôtres, nous pouvons observer que les modèles *end-to-end* sont parvenus à obtenir les meilleurs résultats pour MEDIA et PortMEDIA-Fr. Cependant, depuis, les travaux de Ghannay et al. (2021), l'architecture cascade a de nouveau dépassé les modèles *end-to-end*. Cette architecture cascade est composée d'un modèle wav2vec 2.0 pour la partie ASR et d'un modèle CamemBERT pour la partie sémantique.

		MEDIA		**PortMEDIA-Fr**	
		CER	**CVER**	**CER**	**CVER**
Caubrière et al. (2019)	cascade	*16.1*	*20.4*	-	-
Ghannay et al. (2021)	cascade	*11.2*	*17.2*	-	-
Caubrière et al. (2019)	*end-to-end*	*16.4*	*20.9*	*21.9*	*36.9*
Pelloin et al. (2021)	*end-to-end*	13.6	18.5	17.6	23.6

Table 4.7 – Comparaison des résultats états de l'art MEDIA et PortMEDIA-Fr, toutes architectures et systèmes confondus. Résultats sur les jeux Test.

Néanmoins, nous continuons de penser que les architectures *end-to-end* peuvent être avantagées par rapport aux architectures de type cascade. La représentation ASR intermédiaire est une contrainte pour le système NLU. Certaines informations véhiculées dans la parole sont perdues dans la version transcrite. Nous pouvons par exemple citer les hésitations, le ton, dont les interrogations, et la façon de prononcer certaines entités nommées – telles que les noms de villes. Cette différence sur la prononciation des noms de villes, par exemple en les prononçant de manière plus distinctes, pourrait permettre au système de mieux détecter l'entité nommée en ayant accès au signal de parole. Nous pensons que la frontière entre système *end-to-end* et cascade n'est pas si hermétique, et qu'il est possible de mélanger les deux approches pour espérer améliorer les résultats. Dans la section suivante, nous présentons une première tentative d'hybridation des systèmes *end-to-end* et cascade, en utilisant comme base l'architecture récurrente encodeur-décodeur présentée précédemment.

4.2 Hybridation des systèmes *end-to-end* et cascade

Comme nous venons de le voir, les systèmes cascade obtiennent désormais de meilleurs résultats sur le corpus MEDIA que les systèmes *end-to-end*. Dans cette section, une

architecture récurrente hybride *end-to-end* et cascade est présentée. Cette architecture a initialement été présentée dans notre article des JEP (Journées d'Études sur la Parole) : Pelloin et al. (2022a).

Dans le chapitre 3, nous avons décrit des systèmes de compréhension de la parole utilisant soit une représentation textuelle issue d'un système ASR, soit une représentation acoustique en entrée de ceux-ci. L'information acoustique est importante pour la compréhension du dialogue (Price et al. 1991 ; Tran et al. 2018). C'est en partie pour cette raison que les systèmes de compréhension se sont tournés vers des architectures *end-to-end*, outre la réduction de la propagation des erreurs ASR dans le module NLU. Néanmoins, les modèles *end-to-end* doivent réaliser à eux seuls toute la tâche de reconnaissance et compréhension, de façon simultanée. Un modèle cascade fonctionne en deux temps. Le module NLU est sur ce point avantagé car il peut faire usage d'informations linguistiques présentes dans toute la séquence de mots de l'exemple à comprendre. Cette séquence de mots peut également être enrichie d'informations supplémentaires telles que le *part-of-speech*, le mot *lemmatisé*, ou des informations morphologiques par exemple (Simonnet et al. 2017). Ce fonctionnement hybride a été étudié pour la traduction de la parole par Anastasopoulos et Chiang (2018). Ils ont construit un modèle permettant de déterminer la transcription, et également la phrase traduite, à partir de la parole et de cette transcription automatique.

4.2.1 Pourquoi une architecture hybride ?

Les architectures cascade étaient à l'origine la seule façon de réaliser certaines tâches complexes telles que la compréhension automatique de la parole. Il fallait d'abord utiliser des systèmes pour réaliser la transcription, puis utiliser un ou plusieurs autres systèmes pour réaliser la compréhension. Grâce aux nouvelles performances des machines, des algorithmes, et surtout, aux quantités de données disponibles devenues importantes, les architectures *end-to-end* ont vu le jour (Serdyuk et al. 2018 ; Caubrière et al. 2019 ; Tomashenko et al. 2019 ; Ghannay et al. 2021 ; Shon et al. 2022 ; Laperrière et al. 2023). Ces architectures ont initialement obtenus de meilleurs résultats que celles cascades, notamment grâce à une conception qui ne permet pas de propager des erreurs du module de transcription, les deux modules étant rassemblés en un seul.

Cependant, récemment, grâce aux systèmes SSL, les architecture cascade ont à nouveau dépassé les architectures *end-to-end*. Cascade ou *end-to-end*, chaque architecture semble posséder des avantages sur l'autre, et des inconvénients que l'autre méthode ne possède pas. Nous pensons qu'en réalisant une architecture hybride, nous pouvons tenter de garder les avantages de chaque méthode, et ainsi améliorer les résultats.

Outre la simplification du processus de développement, car un seul modèle est à concevoir, l'avantage principal d'une architecture *end-to-end* est d'éviter de propager les erreurs par rapport au système *cascade*. Nous réalisons un système hybride appris de manière *end-to-end* pour éviter cette propagation d'erreurs. De plus, dans un système *cascade*, la transcription est une représentation appauvrie (ou *bottleneck*) du signal de parole, qui peut éventuellement comporter des erreurs. Un système hybride ne possède pas cette contrainte *bottleneck*.

Nous pensons que l'un des principaux avantages des systèmes cascade pour la SLU est que nous pouvons analyser les hypothèses de transcription, indépendemment des hypothèses de compréhension. Les erreurs de transcriptions impactant fortement la compréhension, d'autant plus selon la métrique du CVER. Il est souvent important de pouvoir s'assurer de la qualité de cette transcription. Or dans un système *end-to-end*, il est impossible de s'assurer que la transcription évaluée est bien independante de l'hypothèse SLU – sémantique. De plus, dans le cas d'un système purement *end-to-end*, la prédiction de la compréhension à un instant t ne se réalise qu'en ayant connaissance de la transcription déjà prédite jusqu'à $t-1$. Les systèmes cascade (ou hybride) sont avantagés car ils possèdent toute la séquence ASR pour prédire l'élément sémantique à l'instant t. Cet avantage vient principalement du fait que le décodage est réalisé en deux temps (décodage ASR puis décodage NLU). Dans MEDIA, certains concepts, tels que le `connectprop`, pourraient bénéficier de la connaissance complète de la transcription. Le concept `connectprop` est un connecteur de proposition, qui permet de relier (entre autres) deux propositions ensemble, avec un «et» ou un «ou» logique. Nous pouvons prendre pour exemple la phrase «avec `<hotel-service>` une piscine `</> <connectprop>` et `</> <hotel-service>` un jacuzzi `</>`», où savoir qu'un deuxième service d'hôtel est cité peut permettre de mieux détecter le concept «`connectprop`».

4.2.2 Détail de l'architecture employée

Pour cette première architecture hybride, nous nous basons sur celle précédemment utilisée. Nous conservons notre encodeur-décodeur récurrent, détaillé en FIGURE 4.1. Le décodeur est employé cette fois uniquement pour déterminer la transcription (ASR). Nous ajoutons ensuite de nouveaux décodeurs chaînés, basés sur la même architecture. Un nouveau décodeur est ainsi ajouté dans le but de prédire les séquences sémantiques. Il travaille en ayant connaissance, comme le décodeur ASR, de la représentation du signal de parole générée par l'encodeur. Il a également comme connaissance la représentation de la transcription provenant du décodeur ASR, d'où l'appellation «chaînée». Chaque décodeur

possède ses propres mécanismes d'attention, tels que ceux présentés par Bahdanau et al. (2015). Le décodeur ASR possède un mécanisme d'attention classique, comme utilisé dans notre modèle précédent. Le décodeur SLU possède lui deux mécanismes d'attention, pour chaque modalité d'entrée. Les représentations de ces mécanismes d'attention sont concaténées, afin de construire un seul vecteur d'entrée pour le décodeur. Cette architecture est schématisée en Figure 4.3. Dans cette figure, afin de bien comprendre comment fonctionnent les différents mécanismes d'attention, nous avons affiché les représentations cachées générées par les encodeurs et décodeurs.

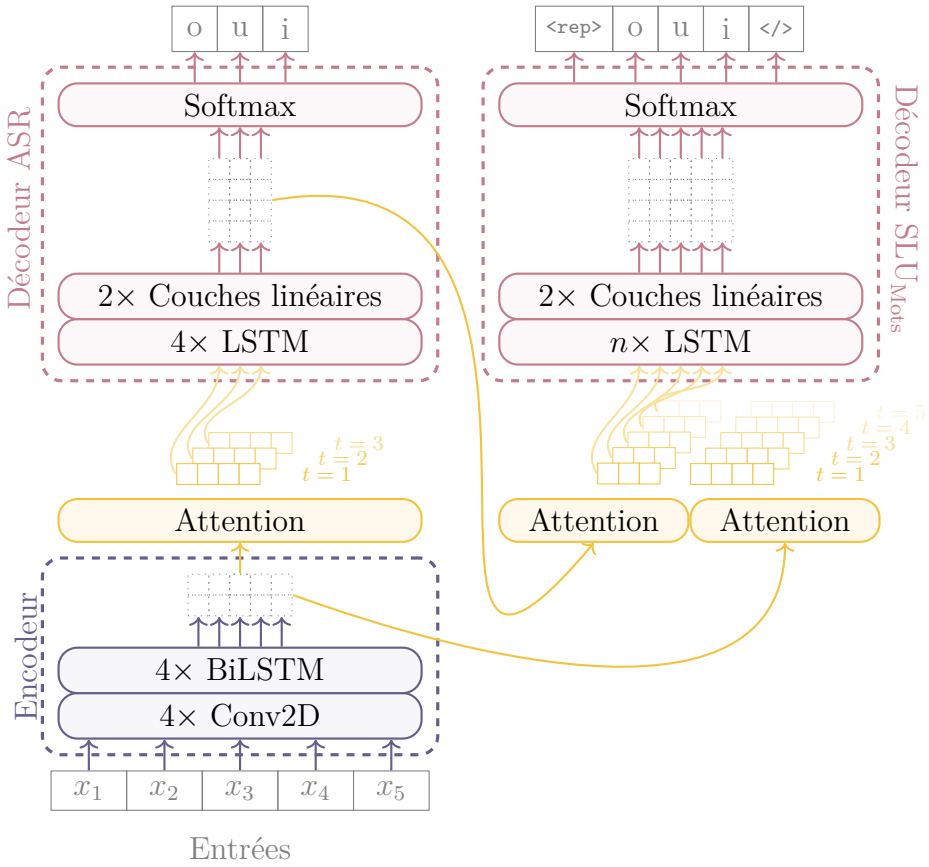


Figure 4.3 – Détail de l'architecture récurrente de compréhension de la parole hybride *end-to-end*/cascade.

Le nombre de décodeurs n'est pas limité à deux. Il est possible de créer autant de décodeur que de sorties voulues, avec des représentations de plus en plus haut niveau. À ce titre, nous avons crée un modèle comportant trois décodeurs. Les deux premiers sont équivalents à ceux que nous venons de décrire. Un troisième décodeur chaîné, est ajouté afin de réaliser l'extraction des valeurs normalisées MEDIA et PortMEDIA-Fr.

Le nombre de couches LSTM dans les décodeurs chaînés est ajusté manuellement en

fonction de la tâche à réaliser pour le décodeur.

Apprentissage et décodage des modèles

Comme dans notre simple modèle encodeur-décodeur (section 4.1), nous réalisons l'apprentissage de manière progressive, grâce à du transfert d'apprentissage et du *finetuning*. Nous partageons par ailleurs entre les deux architectures les mêmes paramètres et modèles pour la partie encodeur-décodeur. L'encodeur et le décodeur ASR sont ainsi initialisés à partir du modèle «ASR → ASR MPM» de la section 4.1. Lors des apprentissages successifs des autres décodeurs, nous continuons à optimiser le décodeur ASR, ainsi que l'encodeur. Chaque décodeur possède sa propre fonction de coût. Les fonctions de coûts utilisées sont basées sur la CrossEntropy, avec un lissage des étiquettes (*label smoothing*) – voir Chorowski et Jaitly (2017). Nous réalisons une somme pondérée de toutes ces fonctions. La pondération appliquée est optimisée manuellement sur le jeu Dev.

Nous décodons ces modèles avec des algorithmes de *beam search*. Les décodages sont réalisés successivement pour chaque décodeur chaîné. Seule l'hypothèse la plus probable est utilisée pour les autres décodeurs chaînés. Empiriquement, nous avons déterminé qu'utiliser les autres hypothèses de décodage n'améliore pas les résultats.

Les séquences de sortie des décodeurs sont les mêmes que pour notre modèle encodeur-décodeur classique de la section 4.1.

- Premièrement, le décodeur ASR doit déterminer la séquence de caractères ;
- Ensuite, le décodeur SLU_{Mots} doit déterminer la séquence de concepts et les caractères de la transcription ;
- Enfin, le décodeur SLU_{Norm} doit déterminer les concepts et les valeurs normalisées associées.

Nous avons choisi de ne pas entraîner de modèles en configuration «étoile» SLU*, car, comme nous l'avons vu précédemment, cette représentation n'apporte pas de bénéfices pour notre architecture.

#		**Décodeurs**		**Paramètres**
1	(a)	SLU_{Mots}		46.3M
2	(b) ASR	SLU_{Mots}		66.1M
	(c) ASR	SLU_{Norm}		66.1M
3	(d) ASR	SLU_{Mots}	SLU_{Norm}	78.1M

Table 4.8 – Nombre de paramètres des modèles *end-to-end* récurrents, mono-décodeur et multi-décodeurs.

En Table 4.8 est indiqué le nombre de paramètres de chaque modèle en fonction du nombre de décodeurs. Le modèle simple décodeur, présenté en section 4.1 est le plus léger, et comporte 46.3 millions de paramètres. Le modèle triple décodeurs, avec ses mécanismes d'attention additionnels est le plus lourd et comporte 78.1 millions de paramètres.

4.2.3 Résultats obtenus

#	Décodeurs	Dev (%) CER	Dev (%) CVER	Test (%) CER	Test (%) CVER
1	(a) SLU_{Mots}	16.9	21.8	16.7	21.5
2	(b) ASR SLU_{Mots}	16.8	23.1	16.6	22.7
	(c) ASR SLU_{Norm}	18.8	25.9	19.0	26.0
3	(d) ASR SLU_{Mots} SLU_{Norm}	19.3	25.2	16.4	22.3
		23.2	28.8	22.3	27.8

Table 4.9 – Résultats sur le corpus MEDIA des différentes configurations explorées de l'architecture multi-décodeurs.

Dans la Table 4.9, nous présentons les résultats sur le corpus MEDIA de notre architecture multi-décodeurs chaînés. Ces résultats sont donnés sur les jeux Dev et Test. La première colonne indique le nombre de décodeurs dans le modèle en question (1, 2 ou 3). La deuxième colonne indique quels sont les décodeurs employés dans le système (ASR, SLU_{Mots}, SLU_{Norm}). En ce qui concerne le modèle à trois décodeurs, dénommé (d) dans la Table 4.9, chaque décodeur SLU est évalué en CER et CVER. Les résultats du premier décodeur SLU_{Mots} sont reportés dans la première ligne, tandis que les résultats du deuxième décodeur SLU_{Norm} sont reportés dans la deuxième ligne. Pour le CVER, la première ligne contient donc des résultats obtenus avec le système d'extraction de valeurs normalisées *à base de règles*, tandis que la deuxième ligne contient des résultats obtenus de manière totalement automatique.

Contrairement aux résultats des Tables 4.4 et 4.5, ceux-ci sont présentés sans décodage associé avec un modèle de langage. De plus, dans les Tables 4.4 et 4.5, les résultats sont ceux du modèle avec *finetuning* sur le corpus MEDIA seul. Ici, nous n'avons pas été jusqu'à cette étape : nous nous sommes arrêtés à la première étape d'apprentissage SLU sur les corpus MPM (MEDIA et PortMEDIA). Le modèle (a) de la Table 4.9 présente la même architecture que le modèle «mots» de la Table 4.4, sans modèle de langage ni *finetuning* sur MEDIA. On peut donc remarquer que ces étapes apportaient 3.1% de gain en CER sur MEDIA Test.

Avec ces premiers résultats sans optimisation, nous pouvons constater que les architectures multi-décodeurs (b, d) sont capables de faire aussi bien sur le TEST que l'architecture mono-décodeur (a). Les décodeurs générant les concepts SLU et les mots obtiennent 16.6% CER et 16.4% CER contre 16.7% pour le mono-décodeur. Cependant, en prenant en compte les intervalles de confiance à 95% (±0.78%), ces résultats ne permettent pas d'affirmer que cette architecture multi-décodeurs, plus complexe, est justifiée. Le CVER est pénalisé par cette architecture multi-décodeurs, même en gardant l'extraction des valeurs avec le système à base de règles : 22.7% pour le modèle (b) contre 21.5% pour le modèle (a).

L'ajout d'un modèle de langage *lookahead* comme précédemment, n'apporte pas de gain. Nous avons choisi de ne pas présenter ces résultats avec modèle de langage car ils sont identiques aux résultats sans modèle de langage. Le poids du modèle de langage est optimisé automatiquement sur le corpus DEV pour réduire le CER, comme d'autres paramètres de décodage. Nous obtenons les mêmes résultats avec modèle de langage car le poids de celui-ci se retrouve proche de zéro. Nous pensons donc que ces modèles de langage *lookahead* ne sont pas adaptés à ces architectures.

Nous présentons en TABLE 4.10 les résultats en WER des modèles présentés, sur le corpus MEDIA, pour la partie utilisateur seulement. Pour le modèle (a), ne disposant pas d'un décodeur ASR, nous avons retiré à posteriori les étiquettes de concepts. Pour les autres modèles, nous évaluons directement la sortie des décodeurs ASR.

WER (%)	**Dev**	**Test**
(a) SLU_{Mots}	9.8	9.2
(b) ASR	10.9	10.2
(c) ASR	12.2	12.1
(d) ASR	13.0	11.2

TABLE 4.10 – Résultats en WER sur le corpus MEDIA (DEV et TEST) pour la partie utilisateur seulement. Les résultats des modèles (b), (c) et (d) sont ceux des décodeurs ASR, tandis que les résultats du modèle (a) sont ceux du décodeur SLU_{Mots}, sans les concepts.

Nous pouvons tout d'abord remarquer que le décodeur SLU_{Mots} privé de ses sorties sémantiques réalise de meilleurs résultats que les décodeurs prévus pour la transcription uniquement. Cela paraît étonnant car le décodeur spécialisé devrait en théorie faire mieux qu'un décodeur *générique*. En réalité, les deux variantes étant apprises de manière *end-to-end* chacune, il faut dans tous les cas trouver un compromis entre ASR et SLU : soit au sein de la sortie d'un même décodeur, soit à travers la fonction de coût combinée des

deux décodeurs. Avec trois décodeurs, les résultats en WER se dégradent d'avantage. Le modèle doit produire trois sorties différentes, il doit donc trouver un compromis parmi ces trois. Ces résultats sont cohérents avec ceux présentés en TABLE 4.9 : les WER étant moins bons pour les architectures à plusieurs décodeurs, ceux-ci ne peuvent donc pas en tirer avantage pour la partie SLU (CER et CVER).

Cette architecture est plus complexe que le modèle encodeur-décodeur classique. Tout d'abord lors de l'apprentissage, car celui-ci est bien plus important, avec 43% à 68% de paramètres en plus, respectivement pour le double-décodeur et le triple-décodeur. De plus, le décodage doit se faire en deux temps, avec deux *beam search*, ce qui complexifie son utilisation et son coût.

4.3 Conclusion

Dans ce chapitre, nous avons présenté nos premiers systèmes de compréhension de la parole associés aux corpus MEDIA et PortMEDIA-Fr. Nous avons présenté nos résultats sur ces deux corpus, à l'aide des métriques du CER (*Concept Error Rate*) et du CVER (*Concept-Value Error Rate*). Nous avons présenté dans ce chapitre des systèmes dits *end-to-end*, utilisant des architectures récurrentes (LSTM).

Nous avons tout d'abord proposé un système encodeur-décodeur récurrent, avec mécanisme d'attention, couplé avec un modèle de langage *lookahead*. Il s'agit d'un modèle pouvant réaliser la transcription automatique de la parole, l'extraction des concepts et de leurs valeurs normalisées. Une nouvelle proposition nous permet d'extraire les valeurs normalisées automatiquement, sans utiliser de système réalisé spécifiquement à partir de connaissances humaines. Ce premier modèle obtient des résultats état de l'art sur les corpus MEDIA et PortMEDIA-Fr, en comparaison avec les autres systèmes *end-to-end*.

Nous avons ensuite adapté cette architecture encodeur-décodeur en modèle comportant plusieurs décodeurs, un pour chaque tâche (ASR, SLU concepts, SLU valeurs normalisées). Il s'agit d'une architecture hybride, à mi-chemin entre le système *end-to-end* et les modèles cascade. L'objectif était d'apporter les avantages des systèmes cascade (prédiction SLU à partir de toute la séquence ASR, y compris le futur), en gardant ceux propres aux systèmes *end-to-end* (pas d'effet *bottleneck* ni de propagation d'erreurs). Le principal atout de l'architecture cascade présentée par GHANNAY et al. 2021 est son utilisation des nouveaux modèles *transformers* SSL (*Self-Supervised Learning*) : wav2vec 2.0 et CamemBERT. Ces modèles sont capables de meilleures performances grâce à leurs pré-apprentissages massifs et auto-supervisés. Notre architecture récurrente ne bénéficie pas de ces avantages. Les

corpus SLU tels que MEDIA et PortMEDIA sont des corpus de taille relativement petite, peut-être trop réduit pour apprendre des modèles *end-to-end* et rivaliser avec les systèmes cascade.

Dans la suite de ces travaux, nous choisissons désormais d'utiliser ces modèles *transformers*. Le chapitre suivant se consacre à l'utilisation de ces nouveaux modèles en mode cascade ou *end-to-end*. En chapitre 6, nous étudions différentes variations des modèles SSL qui pourraient permettre d'améliorer les résultats en SLU.

Chapitre 5

Architectures transformers pour la compréhension de la parole

Sommaire

5.1 Architectures SSL *end-to-end* et cascade 118
5.1.1 Architectures 119
5.1.2 Protocole d'apprentissage et d'évaluation 122
5.1.3 Résultats de l'architecture *baseline* cascade 124
5.1.4 Résultats de l'architecture *end-to-end* 125
5.1.5 Discussions 126
5.2 Analyse des erreurs 127
5.2.1 Systèmes étudiés 127
5.2.2 Distribution des erreurs par concept 128
5.2.3 Évaluation de la transcription sur les mots supports de concepts . . 131
5.3 Conclusion 133

Les modèles *transformers*, présentés en section 1.1.5, généralement pré-entraînés de manière auto-supervisée, sont regroupés derrière l'acronyme SSL pour *Self-Supervised Learning* ; ils sont devenus en quelques années la référence. Ce phénomène s'explique par plusieurs facteurs : leur facilité d'utilisation et de partage, grâce aux librairies telles que HuggingFace [1], leur efficacité algorithmique face aux modèles récurrents, et les représentations robustes qu'ils produisent.

Nous présentons en section 5.1 une architecture *end-to-end* utilisant ces modèles SSL. Après avoir tout d'abord construit une première version cascade comme *baseline*, l'architecture est construite en *end-to-end* à l'aide d'un encodeur et de décodeurs. Ensuite, en section 5.2 est décrite une analyse des erreurs des systèmes jusqu'ici présentés. L'objectif est de mieux comprendre quels sont les avantages apportés par les systèmes *end-to-end* ou cascade, récurrents ou *transformers*. Nous réalisons pour cela une analyse des erreurs au niveau des concepts, puis au niveau des mots supports de ces concepts.

5.1 Architectures SSL *end-to-end* et cascade

Les modèles SSL sont utilisés pour la tâche de compréhension du dialogue. Nous venons de voir que la cascade de modèles SSL wav2vec 2.0 et CamemBERT permet d'obtenir des résultats état de l'art sur le corpus MEDIA (Ghannay et al. 2021). Ces résultats dépassent ceux obtenus avec une architecture *end-to-end*. Dans cette section, un nouveau système *end-to-end* avec plusieurs décodeurs à plusieurs passes est conçu, de façon similaire à celui présenté en section 4.2, mais en utilisant des modèles SSL.

Les motivations pour cette architecture restent les mêmes (voir section 4.2.1) : le choix d'un modèle *end-to-end* ou cascade parait trop fermé, chaque configuration ayant ses avantages et ses inconvénients.

Les architectures développées sont présentées dans la section 5.1.1. Un premier système *baseline* cascade est conçu, puis trois modèles de type *end-to-end* encodeur-décodeurs. En section 5.1.2 le protocole d'apprentissage de ces modèles est décrit, ainsi que leur évaluation. Vient ensuite la présentation des résultats en sections 5.1.3 et 5.1.4.

Travaux similaires. Parallèlement à nos travaux, Arora et al. (2022) ont construit une architecture «hybride» similaire, également pour le *slot-filling*. Leur architecture est composée d'un encodeur ASR, d'un décodeur SLU travaillant sur cet encodeur, et d'un

1. HuggingFace désigne l'organisation gérant à la fois le *Hub*, un espace en ligne regroupant corpus, applications et modèles (pré)-entraînés, mais également des libraires de machine learning : apprentissage et modélisation de modèles *transformers*, ou gestion de *datasets*. `https://huggingface.co/`

module NLU travaillant sur les sorties de l'encodeur et du décodeur ASR. Chaque module est basé sur les modèles *transformer*. Les résultats obtenus avec cette architecture dépassent ceux des systèmes purement *end-to-end* et cascade. Ces travaux confirment l'intérêt de l'architecture présentée ci-dessous.

5.1.1 Architectures

La FIGURE 5.1 présente un aperçu des architectures de compréhension décrites dans cette section : un système *baseline* cascade, un encodeur-décodeur *end-to-end* ainsi qu'un système encodeur et deux décodeurs, appris de manière *end-to-end* également.

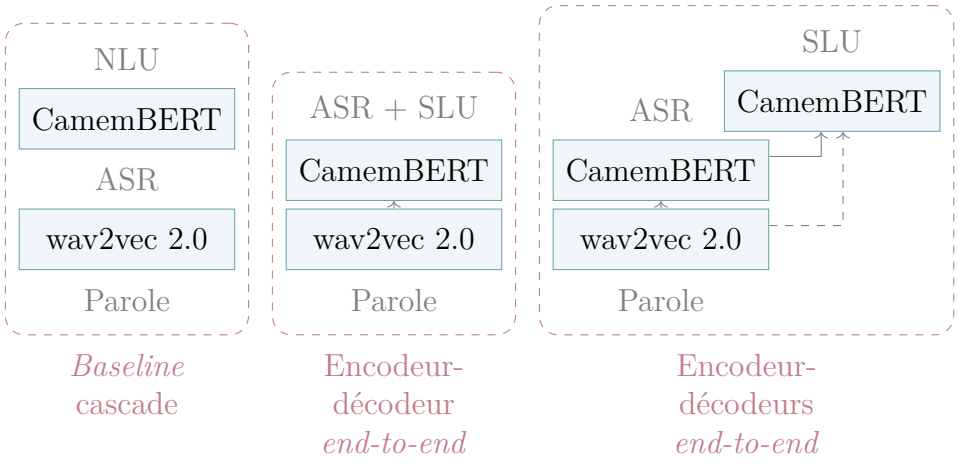


FIGURE 5.1 – Aperçu des différentes architectures de compréhension présentées.

Baseline cascade

Un système *baseline* cascade est construit afin de pouvoir comparer les résultats avec les architectures *end-to-end*. La *baseline* est similaire au système présenté par GHANNAY et al. (2021) : il s'agit d'un modèle ASR wav2vec 2.0 ainsi qu'un modèle NLU CamemBERT.

Nous réalisons un *finetuning* du modèle wav2vec 2.0 [2], présenté par EVAIN et al. (2021) pour la transcription, avec une fonction de coût (*loss*) CTC. Dans nos expériences, ce modèle de transcription sera nommé ASR_{ctc}.

En parallèle, un *finetuning* du modèle CamemBERT [3] présenté par MARTIN et al. (2020) est réalisé pour la prédiction des concepts sémantiques de MEDIA et PortMEDIA-Fr, à partir de la transcription de référence. Ce modèle est optimisé pour déterminer les concepts

2. Nous utilisons le modèle `wav2vec2-FR-3K-large`.
3. Nous utilisons le modèle `camembert-base`.

BIO à partir des *tokens* d'entrée, avec une *loss* CrossEntropy. Comme n'importe quel modèle NLU, celui-ci peut être évalué avec des transcriptions provenant de n'importe quelle source : la référence, le système ASR décrit plus haut, ou bien le système ASR encodeur-décodeur décrit ci-dessous.

Encodeur-Décodeur

Nous souhaitons réaliser un modèle SLU *end-to-end* à deux passes : transcription puis compréhension. L'architecture retenue repose sur un encodeur et des décodeurs. L'encodeur projette le signal de parole dans une représentation. Cette représentation est réutilisée par un décodeur pour réaliser la transcription.

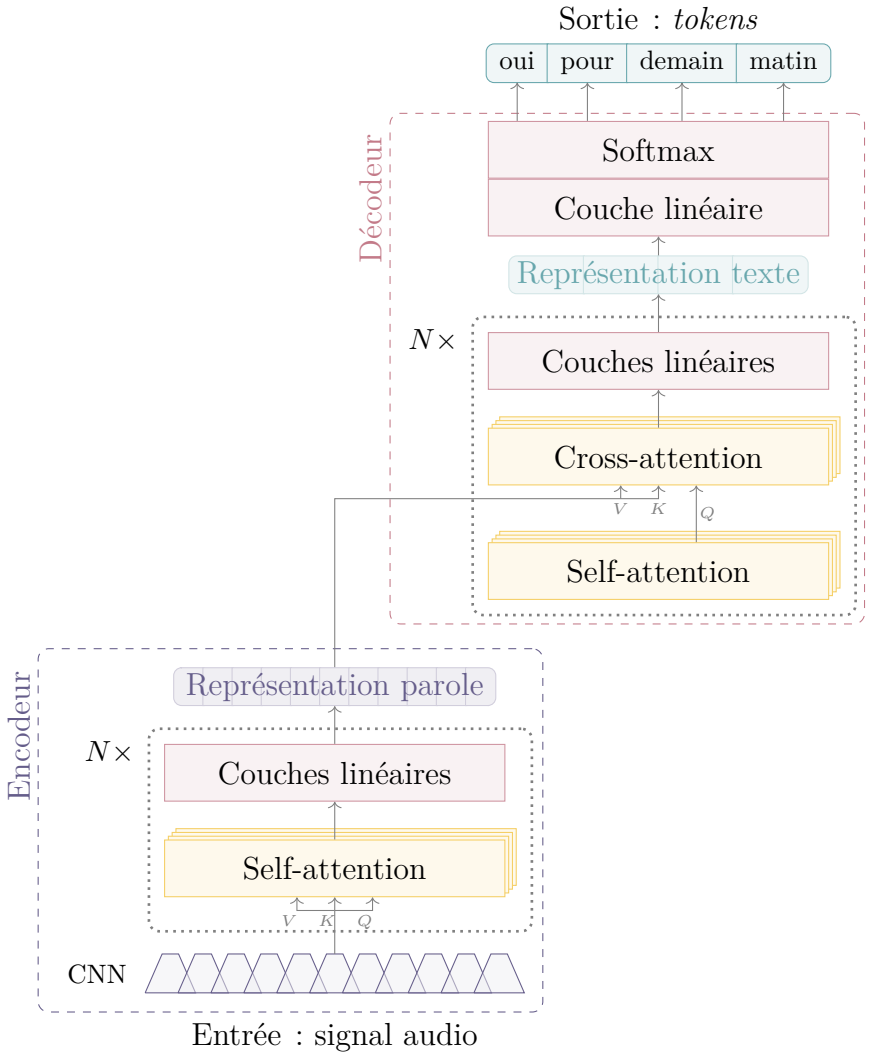


Figure 5.2 – Architecture de la partie encodeur-décodeur (ASR_{ed}) du modèle proposé.

La FIGURE 5.2 présente l'architecture de l'encodeur-décodeur. Cette architecture est utilisée pour réaliser les transcriptions ASR dans le cas du système à deux décodeurs, mais également pour réaliser l'extraction des concepts dans le cas du système à un seul décodeur (cf. FIGURE 5.1). Le modèle wav2vec 2.0 et le modèle CamemBERT sont les mêmes que ceux employés pour la *baseline* : `wav2vec2-FR-3K-large` et `camembert-base`. L'encodeur prend en entrée le signal de parole brut, échantillonné à 16 kHz. Le décodeur génère la transcription à partir de la représentation du signal de parole fournie par l'encodeur. Des modules de *cross-attention* sont ajoutés dans le décodeur afin de le transformer en mode auto-regréssif, et de déterminer les *tokens* à l'instant t à partir de ceux générés jusqu'à $t-1$ et de la représentation de l'encodeur. La représentation du texte avant la dernière couche linéaire et la SOFTMAX est conservée, pour être réutilisée par les autres décodeurs. L'encodeur et le décodeur sont appris simultanément pour réaliser la transcription, à partir du signal de parole. Une *loss* MLM (*Masked Language Model*) est utilisée pour apprendre ce système. Le système de transcription est nommé ASR_{ed}.

Décodeur SLU

Deux décodeurs SLU sont construits pour extraire les concepts des corpus MEDIA et PortMEDIA-Fr. Ces décodeurs sont également basés sur les modèles CamemBERT. Ils sont appris pour déterminer les concepts au format BIO, avec une *loss* CROSSENTROPY, comme le modèle NLU *baseline*.

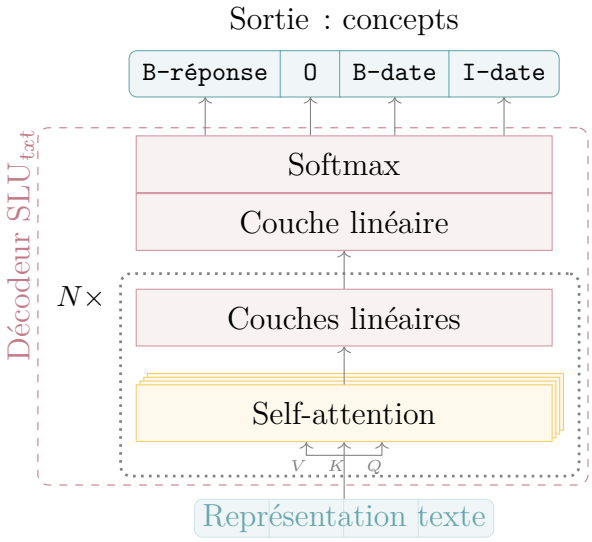


FIGURE 5.3 – Architecture du décodeur SLU_{txt} utilisant la représentation textuelle générée par le décodeur ASR.

La FIGURE 5.3 présente la première version du décodeur SLU, dénommé SLU_{txt}. Ce décodeur dispose en entrée des représentations textuelles ($_{txt}$) générées par le décodeur ASR. Pour chaque vecteur représentant un *token* de transcription, une étiquette sémantique *concept-[BI]|O* est déterminée grâce à des couches linéaires et une fonction SOFTMAX.

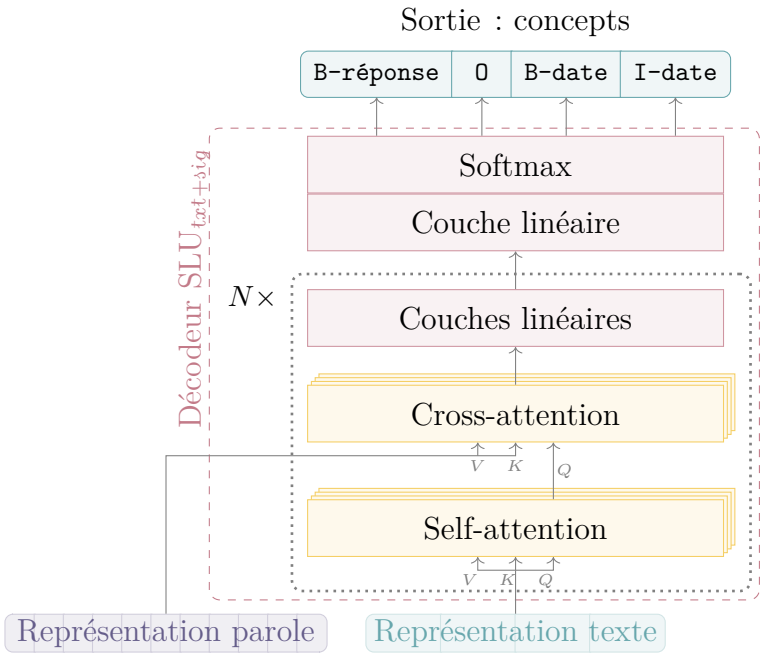


FIGURE 5.4 – Architecture du décodeur $\text{SLU}_{txt+sig}$ utilisant à la fois la représentation textuelle générée par le décodeur ASR, mais également la représentation du signal de parole générée par l'encodeur.

En FIGURE 5.4, la deuxième version du décodeur SLU, dénommée $\text{SLU}_{txt+sig}$, est présentée. Nous ajoutons dans les couches de l'architecture précédente des modules de *cross-attention* entre la *self-attention* et les couches de sortie. Cette *cross-attention* permet de combiner la représentation du signal de parole ($_{sig}$) générée par l'encodeur avec la représentation du texte ($_{txt}$) générée par le décodeur ASR.

5.1.2 Protocole d'apprentissage et d'évaluation

Les architectures présentées sont apprises de manière *end-to-end* : l'encodeur, le décodeur ASR et le décodeur SLU sont appris en même temps. Néanmoins, cet apprentissage est réalisé en plusieurs passes, grâce à un *finetuning* similaire aux expériences du chapitre 4. Contrairement à un système cascade, les représentations employées en entrée des décodeurs ne sont pas discrètes, mais continues. L'ensemble des paramètres des trois sous-modules (wav2vec 2.0, CamemBERT ASR, CamemBERT SLU) du modèle sont optimisés.

Les différentes passes d'apprentissage des modèles sont présentées en Table 5.1. Tout d'abord uniquement les modules encodeur et décodeur ASR sont appris sur des données de transcription : en étape *(1)* avec le corpus CommonVoice, puis sur les parties utilisateur des corpus MPM (MEDIA et PortMEDIA) en étape *(2)*.

Ensuite, le décodeur SLU (SLU_{txt} ou $\text{SLU}_{txt+sig}$) est ajouté, et l'apprentissage du modèle reprend sur les données MPM : étape *(3)*.

Enfin, en étape *(4)*, un *finetuning* sur les données objectif MEDIA est réalisé, sans le corpus PortMEDIA-Fr. Lors de l'apprentissage SLU, le décodeur ASR continue son apprentissage, avec pour objectif de prédire les mots correspondants au segment en question.

Comme précisé plus haut, un modèle SLU *end-to-end* est également appris sans ajouter de second décodeur. Contrairement aux décodeurs SLU_{txt} et $\text{SLU}_{txt+sig}$, il n'est ici pas possible de déterminer les concepts au format BIO car la transcription doit être déterminée en même temps par le même décodeur. L'apprentissage du modèle encodeur-décodeur ASR généré à l'étape *(2)* est repris avec les concepts sémantiques sous forme de balises, pour MPM puis MEDIA seulement.

En ce qui concerne l'architecture *baseline* cascade, le modèle ASR est appris de la même façon : ASR CommonVoice, ASR MPM. Pour le modèle NLU, il est appris sur les données SLU MPM puis MEDIA.

Module	***Checkpoint* initial**	**Étape d'apprentissage**			
		(1)	*(2)*	*(3)*	*(4)*
Encodeur	wav2vec 2.0	ASR CommonVoice	ASR MPM	ASR+SLU MPM	ASR+SLU MEDIA
Décodeur ASR	CamemBERT				
Décodeur SLU	CamemBERT				

Table 5.1 – Étapes d'apprentissage de nos modèles *end-to-end*, à partir des *checkpoints* initiaux pré-entraînés, jusqu'à la prédiction de concepts SLU.

De manière analogue au chapitre 4, les modèles sont évalués sur le corpus MEDIA Test en CER (*Concept Error Rate*) et CVER (*Concept-Value Error Rate*). Lors de la phase d'évaluation, le décodage est réalisé en deux temps. Dans un premier temps, les représentations et sorties de l'encodeur et du décodeur ASR sont générées. Dans un second temps, la sortie du décodeur SLU est produite, à partir des représentations générées à la passe précédente. Un algorithme *beam search* est employé pour l'encodeur-décodeur, ainsi que pour le système de transcription *baseline* (ASR_{ctc}).

Coût environnemental. Tous les paramètres des modèles sont *finetunés* pendant l'apprentissage. Les modèles employés dans les architectures proposées sont de taille considérable, et leur apprentissage ou *finetuning* porte un coût environnemental qui est principalement lié à la consommation électrique des serveurs (CPU, GPU, stockage). La TABLE 5.2 présente le nombre de paramètres des modèles, les durées d'apprentissage, ainsi que le coût équivalent CO_2 lié à l'apprentissage de ces modèles. Celui-ci est effectué sur 8 GPUs NVIDIA A100, dans le centre de calcul Jean Zay (France). Le coût équivalent carbone est estimé grâce à l'utilitaire CODECARBON[4]. L'ensemble de nos expériences réalisées pour cette architecture représente plus de 1000 heures d'apprentissage et un équivalent carbone de 122kg. À titre de comparaison, et d'après le site IMPACTCO_2 de l'ADEME[5], 122 kg eqCO_2 équivalent à 561km parcours en voiture thermique ou 531km en avion.

	#Params.	**App. (h)**	**eqCO_2 (kg)**
Baseline cascade			
ASR	316M	97h38	13.63
NLU	110M	1h50	0.07
Encodeur-Décodeur	455M	103h58	14.40
+ SLU_{txt}	+110M (565M)	+8h53	+0.85
+ $SLU_{txt+sig}$	+140M (595M)	+18h00	+1.65

TABLE 5.2 – Nombre de paramètres, temps d'apprentissage et équivalent CO_2 lié à l'apprentissage des modèles réalisés.

Nous pouvons remarquer que le modèle *end-to-end* possède plus de paramètres que le modèle cascade. Ainsi, il demande plus d'apprentissage, et a un coût CO_2 plus élevé.

5.1.3 Résultats de l'architecture *baseline* cascade

Dans l'architecture cascade, le modèle NLU requiert une transcription automatique. Nous disposons de deux modèles de transcription automatique : *baseline* CTC (ASR_{ctc}), et encodeur-décodeur MLM, avant *finetuning* SLU (ASR_{ed}). La TABLE 5.3 présente les taux d'erreurs mots de ces deux modèles sur la partie utilisateur du corpus MEDIA.

Nous pouvons constater que le modèle ASR_{ctc} génère des transcriptions sensiblement meilleures que le modèle ASR_{ed}. Dans les architectures SLU, il pourrait ainsi être judicieux de remplacer l'encodeur-décodeur par ce modèle. Dans ce cas de figure, seul le décodeur

4. https://github.com/mlco2/codecarbon
5. https://impactco2.fr

Modèle		***Loss***	**WER (%)**
ASR_{ctc}	wav2vec 2.0	CTC	**8.80**
ASR_{ed}	wav2vec 2.0 & CamemBERT	MLM	10.89

TABLE 5.3 – Performances des deux modèles ASR sur le corpus MEDIA TEST (partie utilisateur).

SLU_{txt} pourrait exister, à partir de la représentation du texte générée par le modèle ASR_{ctc}. Nous laissons cette perspective d'amélioration pour de futurs travaux.

Le modèle NLU est maintenant évalué sur le corpus MEDIA en CER et CVER. Pour cela, les transcriptions issues des modèles ASR_{ed} et ASR_{ctc} sont utilisées, ainsi que les transcriptions de référence. Les résultats sont présentés en TABLE 5.4. Nous observons sans surprise de meilleurs résultats sur la transcription de référence que sur les transcriptions automatiques. De plus, nous remarquons que le modèle ASR_{ctc} dépasse en CER et CVER le modèle ASR_{ed}. Cette différence s'explique par le taux d'erreur mots plus élevé du modèle ASR_{ed}.

Transcriptions	% **CER**	**CVER**
Référence	*7.93*	*12.72*
ASR_{ctc}	**10.80**	**16.97**
ASR_{ed}	13.13	18.34

TABLE 5.4 – Résultats en CER et CVER du modèle cascade sur le corpus MEDIA TEST.

À notre connaissance, les résultats obtenus avec ce modèle sont les meilleurs obtenus à ce jour sur le corpus MEDIA avec une transcription automatique. Par rapport aux résultats état de l'art de GHANNAY et al. (2021), nous obtenons un gain de 0.40% de CER et de 0.23% de CVER. Sur la référence, le modèle NLU réalise cependant de moins bons résultats (0.37% absolus en CER) que ceux de GHANNAY et al. (2021). L'amélioration provient donc probablement du système de transcription utilisé et de son intégration avec le système NLU (à travers les normalisations), ou du *finetuning* sur les données de transcription PortMEDIA-Fr.

5.1.4 Résultats de l'architecture *end-to-end*

La TABLE 5.5 présente les résultats de l'architecture *end-to-end* à plusieurs décodeurs.

Remarquons que le modèle avec décodeur SLU_{txt} réalise de meilleurs CER et CVER que le modèle encodeur-décodeur classique. Il est important de rappeler que ces deux modèles n'ont pas été appris avec les mêmes formats de sortie : dans le cas du SLU_{txt}, le modèle doit déterminer les concepts au format BIO, alors que dans le cas du décodeur classique, celui-ci doit déterminer les concepts au format balisé, conjointement avec les *tokens* de transcription. Il semble que traiter conjointement les deux tâches ASR et SLU (format balisé) soit moins performant que les traiter séparément (transcriptions + format BIO).

Constatons également que l'utilisation des représentations de la parole dans le décodeur $\text{SLU}_{txt+\delta ig}$ dégrade de façon conséquente les résultats (8.52% de CER). Des modules de *cross-attention* ont été ajoutés dans ce décodeur afin d'exploiter au maximum ces représentations. Ce nouveau modèle nécessiterait sûrement une durée d'apprentissage plus grande bien qu'elle soit déjà très supérieure à celle du modèle SLU_{txt} (18h00 contre 8h53, cf. Table 5.2). De plus, la quantité de données d'apprentissage est peut être insuffisante. Lorsque des couches de *cross-attention* ont été ajoutées dans le décodeur pour la transcription, il a été nécessaire de ré-entrainer le modèle sur le corpus CommonVoice, plus volumineux. Ce manque de données pourrait être compensé par un pré-apprentissage auto-supervisé de ce décodeur $\text{SLU}_{txt+\delta ig}$. Nous laissons cette perspective d'amélioration pour de futurs travaux.

Modèle	**Conf. décodeur**	**%** **CER**	**CVER**
wav2vec 2.0 & CamemBERT (SLU)		17.52	22.45
wav2vec 2.0 & CamemBERT (ASR) & CamemBERT (SLU)	SLU_{txt}	**16.08**	**21.47**
wav2vec 2.0 & CamemBERT (ASR) & CamemBERT (SLU)	$\text{SLU}_{txt+\delta ig}$	24.61	31.49

Table 5.5 – Résultats en CER et CVER des architectures *end-to-end* étudiées, sur le corpus MEDIA Test.

5.1.5 Discussions

Ces résultats remettent en question le choix de la construction d'un modèle *end-to-end* par rapport à un système cascade. En effet, nous observons une différence de 5.28% de CER entre le meilleur modèle cascade (10.80%) et notre modèle *end-to-end* (16.08%). Par rapport au modèle *end-to-end* récurrent du chapitre 4, l'architecture cascade présentée ici

réalise un gain de 2.80% de CER. À l'heure où ces modèles pré-entraînés sont disponibles et utilisables par tous, où le coût de *finetuning* sur les tâches cibles est moins important en mode cascade (TABLE 5.2), et où leurs résultats sont meilleurs, est-il encore utile d'apprendre un système de compréhension de la parole de manière *end-to-end* ? De toute évidence, les hypothèses d'amélioration de nos architectures *end-to-end* par rapport au modèle cascade ne sont pas vérifiées. De nouvelles méthodologies d'apprentissage, ou d'architectures semblent nécessaires pour concevoir de nouvelles architectures état de l'art en *end-to-end*. Cependant, récemment, ARORA et al. (2022) ont construit une architecture dite compositionnelle, similaire à la notre : encodeur-décodeurs ASR et NLU utilisant les représentations des mots et de la parole. Leur modèle compositionnel, également *transformer* permet d'améliorer les résultats en F_1 sur les corpus SLURP (BASTIANELLI et al. 2020) et SLUE (SHON et al. 2022), par rapport à une approche cascade ou *end-to-end* classique. La tâche d'extraction de concepts du corpus MEDIA semble aujourd'hui plus difficile à maîtriser, avec des résultats sensiblement meilleurs pour les architectures cascade, à l'inverse des travaux *end-to-end* et cascade sur SLUE et SLURP.

5.2 Analyse des erreurs

Nous réalisons deux analyses des erreurs afin d'investiguer les différences entre les systèmes *end-to-end* et cascade. La première analyse (section 5.2.2) consiste à regarder les erreurs pour chaque modèle par classe de concept. La deuxième analyse (section 5.2.3) consiste à regarder les erreurs des mots supports de ces concepts. Quatre systèmes sont évalués et comparés, et nous les détaillons en section 5.2.1. Ces travaux font suite à des travaux préliminaires publiés à la conférence LREC (MDHAFFAR et al. 2022).

Nos analyses d'erreurs sont toutes effectuées sur le même jeu de données, avec la même normalisation et les mêmes pré-traitements. Nous effectuons les analyses d'erreurs sur l'ensemble DEV de MEDIA, plutôt que sur le jeu TEST, afin de respecter l'intégrité du jeu de test vis à vis d'éventuelles contributions apportées par ces analyses des erreurs.

5.2.1 Systèmes étudiés

Nous détaillons en TABLE 5.6 les performances en CER et CVER des quatre systèmes étudiés, ainsi que leurs architectures.

Le système $E2E_{Rec}$ (a) correspond au modèle encodeur-décodeur récurrent, avec modèle de langage, présenté au chapitre précédent (section 4.1.4). C'est actuellement le modèle

Nom	Section	Architecture	CER	CVER
(a) $E2E_{Rec}$	4.1.4	SLU *end-to-end* Récurrent	13.6	18.5
(b) $E2E_{SSL}$	5.1.4	SLU *end-to-end* Transformer SSL	16.08	21.47
(c) Cas_{SSL}	5.1.3	SLU cascade Transformer SSL	10.80	16.97
(d) Ref_{SSL}	5.1.3	NLU référence transcriptions Transformer SSL	7.93	12.72

TABLE 5.6 – Descriptif des systèmes étudiés pour nos analyses des erreurs. Les résultats sont donnés à titre de comparaison, sur le corpus MEDIA TEST, et correspondent aux meilleurs résultats publiés pour chaque modèle.

end-to-end état de l'art sur le corpus MEDIA. Le système $E2E_{SSL}$ (b) correspond au modèle ayant obtenu les meilleurs résultats obtenus avec l'architecture SSL *end-to-end* présenté en section 5.1.4 : il s'agit du modèle composé d'un encodeur wav2vec 2.0, d'un décodeur ASR CamemBERT, et d'un décodeur SLU CamemBERT travaillant sur les représentations du décodeur ASR (SLU_{txt}).

Nous comparons ces deux modèles *end-to-end* avec l'architecture cascade composé d'un module ASR et d'un module NLU. Il s'agit du modèle *baseline* présenté en section 5.1.3. Ce système, dénommé Cas_{SSL} (c), est actuellement état de l'art pour la tâche d'extraction des concepts MEDIA à partir de la parole. Enfin, nous comparons l'impact de la transcription automatique sur le modèle NLU. Pour cela, nous intégrons les prédictions à partir des transcriptions de référence de ce modèle NLU, que nous dénommons Ref_{SSL} (d). En comparaison des autres systèmes SLU, celui-ci est donc avantagé car il ne doit pas gérer les erreurs liées à la reconnaissance de la parole.

Pour résumer, nous étudions donc les sorties de quatre systèmes : celles de $E2E_{Rec}$ (a) et $E2E_{SSL}$ (b), des modèles appris de manière *end-to-end* ; celles d'un système cascade Cas_{SSL} (c) ; puis de celles de son module NLU sur les transcriptions de référence à la place du module ASR : Ref_{SSL} (d).

5.2.2 Distribution des erreurs par concept

Afin de mieux comprendre les erreurs de chaque type d'architecture cascade ou *end-to-end*, nous réalisons une analyse des erreurs par concepts. Nous évaluons chaque système étudié grâce à un alignement forcé entre la référence et l'hypothèse. Cet alignement fourni un nombre d'erreurs d'insertions, de suppressions et de substitutions (cf. 3.4.2). Nous regroupons les erreurs de chaque concept du corpus, pour étudier l'impact de l'architecture employée sur la détection des concepts. Nous indiquons un numéro pour chaque concept,

afin de pouvoir le retrouver facilement dans la figure.

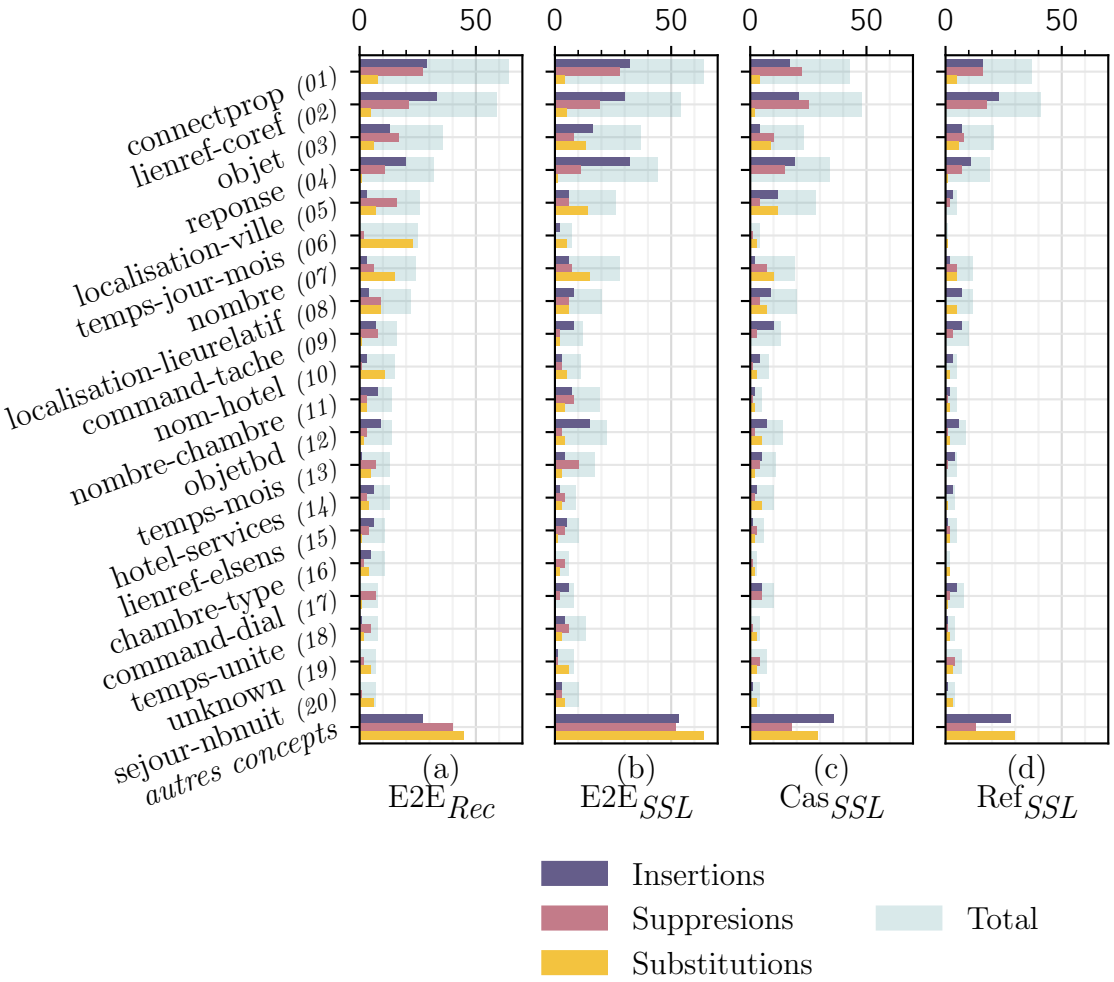


FIGURE 5.5 – Nombre d'erreurs sur le corpus MEDIA DEV par concepts pour chaque système étudié.

Nous présentons en FIGURE 5.5 ce descriptif des erreurs pour les 20 concepts de MEDIA ayant le plus d'erreurs. Les erreurs d'insertions, suppressions et substituions, ainsi que le total sont indiqués. Les autres concepts sont rassemblés dans la catégorie «*autre concepts*». Les résultats sont triés par le nombre d'erreurs total décroissant selon le système encodeur-décodeur (a).

Nous remarquons que les concepts ayant le plus d'erreurs sont `connectprop` (*01*), `lienref-coref` (*02*), `objet` (*03*), `reponse` (*04*). Ces concepts sont des concepts *génériques*, ne dépendant pas du domaine de l'application (réservation d'hôtels). Ces concepts interviennent par exemple dans la phrase ci-dessous, tirée du corpus MEDIA TRAIN :
`<reponse>` oui `</>` `<connectprop>` donc `</>` quel serait `<lienref-coref>` le `</>` `<obje`

t> prix </> pour euh pour <lienref-coref> ces </> <nombre-chambre> deux euh deux chambres </>

En outre, trois de ces concepts avec le plus d'erreurs (connectprop (*01*), lienref-coref (*02*) et reponse (*04*)) permettent de représenter la structure des phrases (connecteurs de propositions, co-références) et du dialogue.

Comme attendu, le modèle Cas_{SSL} (c) est celui réalisant au global le moins d'erreurs, après le modèle Ref_{SSL} (d) utilisant les références de transcription. Ces résultats retranscrivent ceux présentés en TABLE 5.6.

Nous remarquons que le système E2E_{Rec} (a) réalise un grand nombre d'erreurs sur le concept temps-jour-mois (*06*), contrairement aux systèmes (b, c, d). Le concept temps-jour-mois permet de spécifier une date dite abstraite, en précisant uniquement le jour. Il est utilisé lorsqu'il n'est pas possible d'obtenir une date complète à partir du segment. Nous pouvons prendre exemple du segment «<temps-jour-mois> du vingt sept </> <temps-jour-mois> au vingt huit </>». Le concept «temps-date» est lui utilisé lorsque le mois est précisé au sein du segment total, et qu'il est ainsi possible d'obtenir une date complète. Par exemple, le segment «<temps-date> du six </> <temps-date> au quatorze juillet </>».
Le Ref_{SSL} (d) ne réalise presque aucune erreur sur ce concept avec un CER individuel de 3.70% contre 92.59% pour le E2E_{Rec} (a). À l'exception de deux erreurs sur 25 occurrences, toutes les substitutions du E2E_{Rec} (a) sur ce concept ont été faites avec le concept temps-date. Les modèles transformers utilisent une représentation en contexte de chacun des mots de la phrase, contrairement aux modèles récurrents qui réalisent cette représentation de façon séquentielle. Ce contexte permet au modèle de savoir si un mois a été précisé (date complète : temps-date) ou non (date abstraite : temps-jour-mois). Nous pensons que c'est pour cette raison que ces modèles (b, c, d) ont été en mesure de prédire et différencier correctement temps-date et temps-jour-mois (*06*).

Le concept localisation-ville (*05*) permet d'annoter un nom de ville. Nous remarquons que le système utilisant la référence des transcriptions (d) réalise beaucoup moins d'erreurs que tous les autres systèmes (a, b, c) sur ce concept. Nous pensons que cette différence provient des erreurs de reconnaissance des noms propres que sont les villes. En outre, nous pouvons observer que le système cascade (c) réalise légèrement plus d'erreurs sur ce concept que les systèmes *end-to-end* (a, b). Cela pourrait suggérer que le fonctionnement *end-to-end* permet de palier certaines erreurs de transcription des villes présentes au sein du système cascade. Nous consacrons l'analyse de la section suivante à l'évaluation de ces transcriptions.

5.2.3 Évaluation de la transcription sur les mots supports de concepts

Nous venons de le voir, certains concepts semblent être particulièrement sensibles à la transcription des mots prononcés. Dans cette section, nous calculons le taux d'erreurs mots (WER) sur les mots supports associés à certains concepts. Pour un concept donné, nous calculons le WER de ses mots supports dans le cas où le concept a bien été détecté, mais aussi en cas d'erreur (insertion, suppression ou substitution du concept).

Limites. Ce calcul permet d'obtenir une estimation WER des mots supports des concepts, mais celui-ci est indirectement impacté par l'alignement entre la référence et l'hypothèse, notamment en cas d'erreurs de reconnaissance de concepts. Dans ce cas, les mots supports peuvent être alignés différemment de la référence, même si ceux-ci sont corrects. Pour réduire certains de ces effets de bord, nous éditons a posteriori l'alignement pour empêcher les substitutions entre des concepts et des mots.

Noms propres. Nous nous intéressons en particulier aux concepts `localisation-ville` (*05*), `localisation-lieurelatif` (*08*) et `nom-hotel` (*10*). Il s'agit des concepts portant sur les noms propres les plus fréquents dans le schéma d'annotation de MEDIA. D'autres concepts sont utilisés pour des noms propres, tels que les concepts `localisation-quartier` ou `nom` mais ils sont peu fréquents : 6 occurrences sur le jeu DEV dans le cas de `localisation-quartier`, 7 pour `nom`, contre 168 pour `localisation-ville`, 86 pour `localisation-lieurelatif`, et 48 pour `nom-hotel`.

Concepts avec vocabulaire fermé. Nous nous intéressons également aux concepts `connectprop` (*01*), `objet` (*03*) et `reponse` (*04*). Ces concepts sont parmi ceux ayant le plus d'erreurs sur les quatre systèmes étudiés, leurs supports de mots possèdent un vocabulaire relativement fermé. Par exemple, l'ensemble des mots supports du concept `connectprop` (*01*) est défini à travers 23 expressions telles que «et» «c'est-à-dire» ou «par conséquent». Nous souhaitons étudier la différence entre ces concepts et ceux possédant un vocabulaire plus ouvert.

Nombres. Enfin, nous nous intéressons aux concepts possédant un jeu de mots supports théoriquement illimités : les nombres et les dates. Pour cela, nous observons les taux d'erreurs de ces supports pour les concepts `nombre` (*07*), `nombre-chambre` (*11*) et `temps-date`.

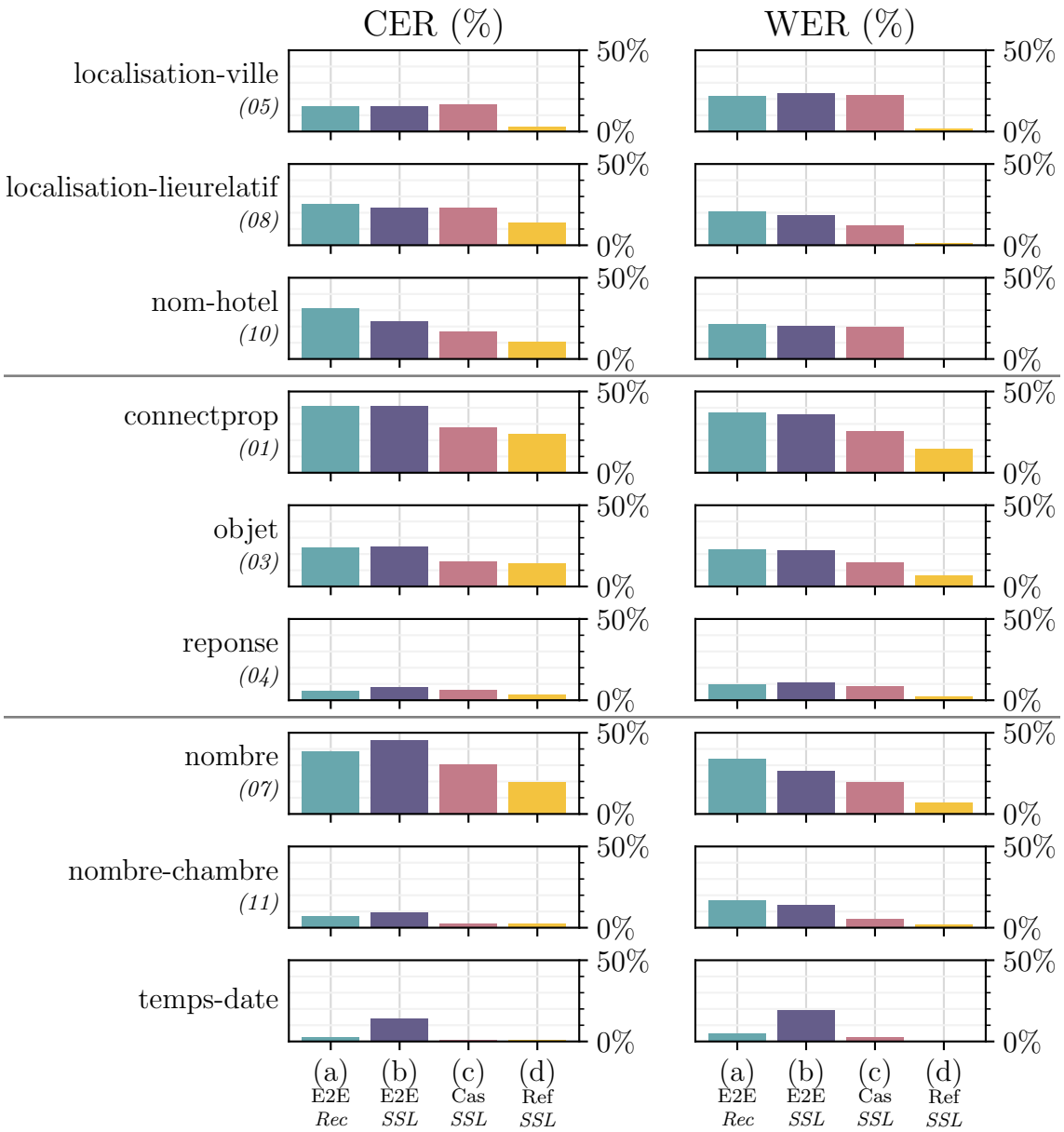


FIGURE 5.6 – CER et WER sur le corpus MEDIA DEV des systèmes étudiés pour trois groupes de concepts (noms propres, vocabulaire fermé, nombres).

La FIGURE 5.6 présente les taux en CER et WER obtenus en considérant uniquement les mots supports de chaque concept étudié. Nous avons réparti les concepts présentés en fonction des groupes décrits précédemment : noms propres, vocabulaire fermé ou nombres.

Nous remarquons que dans l'ensemble, le WER des mots supports paraît corrélé au CER du concept en question. Il est difficile d'estimer si les erreurs de transcriptions induisent les erreurs sur les concepts, ou si ce sont les erreurs sur les concepts qui faussent l'alignement, et donc les erreurs de transcription. En particulier, bien que le système (d)

utilise des transcriptions de référence, le WER des mots supports de certains concepts n'est pas toujours à 0%, à cause des erreurs sur les concepts. Par exemple, la référence d'un segment du corpus MEDIA Dev est «[...] `<connectprop>` plus `</>` `<hotel-services>` une piscine `</>` s' il vous plaît». Le système (d) s'est trompé sur les concepts, ce qui induit un WER local incorrect : «[...] `<hotel-services>` plus une piscine `</>` s' il vous plaît».

Dans le cas des concepts de noms propres (`localisation-ville`, `localisation-lieurelatif` et `nom-hotel`), les WER sont relativement constants entre les systèmes de transcription automatique. Dans le cas du concept `nom-hotel` les WER sont similaires entre (a, b, c), alors que le CER ne l'est pas. Cela semble suggérer une dé-corrélation entre la détection de ce concept et la transcription de son support. Dans la plupart des cas, l'utilisateur formule sa demande sous la forme «`<nom-hotel> à l'hôtel roissy </nom-hotel>`». Si le nom de l'hôtel est mal transcrit, mais que le groupe de mots «à l'hôtel» l'est correctement, alors la détection du concept paraît toujours possible. À l'inverse, pour le concept `localisation-ville`, l'usage d'une forme telle que «`dans la ville de paris`» est rare.

Pour les concepts portant sur des supports de mots avec un vocabulaire plus restreint, nous n'observons pas de différence notable entre le CER et le WER. Le WER des mots supports semble ainsi influer de façon cohérente sur le CER.

En ce qui concerne les nombres, le CER du système (b) pour le concept `nombre` est plus élevé que celui-du système (a), alors que la transcription semble meilleure pour le système (b). La même tendance s'observe pour le concept `nombre-chambre`. Comme pour `nom-hotel`, les concepts de nombres nous paraissent facile à reconnaître, peu importe si la transcription est parfaite.

5.3 Conclusion

Les systèmes pré-entraînés sont devenus incontournables dans de nombreuses tâches d'apprentissage automatique. La compréhension automatique de la parole, comme d'autres tâches de traitement automatique de la langue n'échappe pas à cette tendance. Dans ce chapitre, nous avons exploré de nouveaux systèmes *end-to-end* à base de modèles pré-entraînés de manière auto-supervisée (SSL), pour la compréhension automatique de la parole. De plus, nous avons également construit un système *baseline* cascade à partir d'un modèle wav2vec 2.0 et CamemBERT afin de comparer les deux types d'architectures. Notre architecture *end-to-end* fait elle-aussi utilisation de ces modèles. Des variantes de cette

architecture ont été construites en ajoutant un décodeur travaillant sur une représentation textuelle, et un travaillant également sur la parole. Nous avons obtenu des résultats état de l'art avec l'architecture cascade. Les architectures *end-to-end* développées n'arrivent pas à égaler ce système.

Notre analyse des erreurs par concepts entre quatre systèmes de compréhension nous a permis d'observer les différences entre l'architecture cascade et *end-to-end*, en particulier pour les concepts portant sur des noms propres tels que `localisation-ville`. Nous avons également pu observer les différences entre la reconnaissance des concepts portant sur les dates. Par rapport aux modèles *transformers*, l'architecture récurrente semble moins bien réussir cette distinction, probablement à cause du fonctionnement séquentiel de celui-ci.

Dans le prochain chapitre, nous utilisons les larges modèles SSL en mode cascade en les adaptant pour la tâche de la compréhension de la parole.

Chapitre 6

Utiliser les modèles pré-entraînés de manière auto-supervisée

Sommaire

6.1 Adapter les modèles de langage aux spécificités de la parole 138
6.1.1 FlauBERT et FlauBERT-Oral 139
6.1.2 Utilisation pour la compréhension de la parole 140
6.2 Les modèles SSL multimodaux et multilingues 143
6.2.1 Représentations 143
6.2.2 Protocole d'évaluation *zero-shot* des représentations 144
6.2.3 Résultats 146
6.2.4 Crosslingue et crossmodal : conclusion et perspectives 148
6.3 Utilisation de modèles conversationnels pour la compréhension 149
6.3.1 Détection des concepts MEDIA 150
6.3.2 Compréhension globale du dialogue 154
6.4 Conclusion 158

Au cours du chapitre précédent, nous avons utilisé des modèles pré-entraînés de manière auto-supervisée pour la compréhension de la parole. Dans ce chapitre, nous nous intéressons à leur adaptation pour cette même tâche. En particulier, nous nous intéressons aux modèles de langage utilisés dans un système de compréhension cascade. Les modèles de langage sont généralement appris sur des textes de référence ne contenant pas d'erreurs. Cependant, lors de leur utilisation dans un système SLU cascade, des erreurs de transcription sont présentes. Dans un premier temps, en section 6.1, nous utilisons un modèle de langage pré-appris sur des transcriptions automatiques en tant que module NLU, dans l'objectif de réduire les erreurs. En section 6.2, nous explorons l'utilisation de modèles multimodaux et multilingues afin d'extraire des représentations sémantiques globales. Nous nous concentrons en particulier sur l'évaluation *zéro-shot* du corpus italien PortMEDIA-It après un apprentissage sur le corpus français MEDIA. Nous évaluons, également en *zéro-shot*, un modèle appris sur des représentations provenant du texte avec de la parole, et inversement. Notre objectif est de déterminer si ces représentations sont adéquates pour changer de langue ou de modalité sans ré-apprendre le modèle. Enfin, en section 6.3, nous utilisons un modèle conversationnel (*chatbot*) pour extraire automatiquement les concepts du corpus MEDIA, sans apprentissage sur celui-ci.

6.1 Adapter les modèles de langage aux spécificités de la parole

Les modèles de langage pré-entraînés de manière auto-supervisée tels que BERT (Devlin et al. 2019), ou leurs variantes en français CamemBERT (Martin et al. 2020) et FlauBERT (Le et al. 2020) sont régulièrement utilisés en tant que modules *downstream* pour certaines tâches (Chiu et Chen 2021 ; Hervé et al. 2022), dont, nous venons de le voir au chapitre précédent, la compréhension de la parole. En suivant une architecture cascade, ces modèles de langage (LM – *Language Model*) prennent en entrée du texte généré automatiquement par un module de transcription de la parole. Le modèle cascade présenté dans la section précédente en est un exemple : la transcription est réalisée automatiquement par un modèle (wav2vec 2.0), et la tâche de compréhension est réalisée dans un second temps par un LM (CamemBERT) ajusté pour cette tâche, à partir de la transcription.

Cependant, les modèles de langage tels que CamemBERT et FlauBERT ne sont pas appris sur du texte similaire à de la langue parlée. Ils sont généralement pré-appris sur du texte en langage naturel écrit, provenant de sources telles que Wikipédia. Ensuite, ils sont optimisés avec des données de référence de transcription en entrée. Pourtant, la

communication orale et la communication écrite ne reposent pas sur les mêmes codes (Shriberg 2005). Cette différence est principalement liée à la spontanéité de la parole, avec certains effets propres à celle-ci : mots de remplissage, faux départs et autres disfluences, ou interruptions. De plus le texte oral a été transcrit, souvent automatiquement à partir du signal de parole. Cette transcription automatique peut contenir des erreurs, en plus grand nombre que dans le texte écrit. Nous souhaitons limiter l'impact de ces erreurs de transcription dans le modèle *downstream*, afin de réduire l'effet de propagation et d'amplification des erreurs.

Dans cette partie, nous évaluons l'impact des données de pré-entraînement, dans le cas d'une utilisation ultérieure du modèle sur du texte parlé. Pour cela, nous nous concentrons sur le modèle de langage français FlauBERT, appris dans sa version originale sur différents corpus de textes écrits (livres, Wikipédia, *crawls* internet). Nous évaluons, sur la tâche SLU MEDIA, les performances de ce modèle FlauBERT, ainsi que des modèles FlauBERT appris sur des corpus de textes issus de transcriptions automatiques de la langue.

6.1.1 FlauBERT et FlauBERT-Oral

FlauBERT est une famille de modèles de langage pré-appris pour la tâche de *Masked Language Modeling* (MLM), de façon similaire aux modèles RoBERTa (Liu et al. 2019). Quatre versions [1] de FlauBERT sont publiées, mais toutes ont été apprises sur 71Go de texte. Les différences entre les modèles sont le nombre de paramètres (`small` - 54M, `base` - 138M, `large` - 373M), et l'apprentissage sur des données avec ou sans casse (`cased` ou `uncased`). Dans nos travaux ayant mené aux publications des articles Hervé et al. (2022) et Pelloin et al. (2022b), nous avons repris cette architecture FlauBERT, et l'avons adaptée à des textes issus de transcriptions automatiques.

Pour cela, nous utilisons un corpus de 350 000 heures de parole d'émissions télévisuelles et radio, collecté par l'INA (Institut National de l'Audiovisuel). Ces données sont par conséquent fortement tournées vers les actualités radio et TV, et donc éloignées des conversations téléphoniques telles qu'elles existent dans MEDIA et PortMEDIA. Nous ne disposons pas de données téléphoniques en quantité suffisante pour effectuer cette étude. Il s'agit donc d'un compromis par rapport aux données textuelles usuelles.

Descriptif des données. Les 350 000 heures de parole utilisées pour l'adaptation proviennent des enregistrements de l'INA entre 2013 et 2020. Une partie des données provient de chaînes d'informations en continu de 6h00 à 00h00 chaque jour (BFMTV, LCI,

1. `small-cased`, `base-cased`, `base-uncased`, `large-cased`

CNews, France 24, et France Info). Les bulletins d'actualités matinales des radios Europe1, RMC, RTL et France Inter sont également utilisés. Enfin, les journaux télévisés du soir de TF1, France 2, France 3 et M6 sont inclus dans le corpus.

Transcription automatique. Ces 350 000 heures de paroles ont été transcrites automatiquement par l'INA, avec un système Kaldi, appris lui-même sur des corpus TV et radio (ESTER 1 et 2, REPERE, VERA). Un modèle de langage n-gramme est employé. Le corpus résultant contient 19Go de texte, et 3.5 milliards de mots. Les WER sur les corpus TEST sont les suivants : REPERE 12.1%, ESTER1 8.8%, ESTER2 10.7%. Ces corpus sont décrits en section 2.5. La transcription est effectuée sans information de casse.

Modèles FlauBERT oraux. Enfin, l'INA a procédé au pré-apprentissage des modèles FlauBERT oraux en utilisant ces données, pour la tâche de MLM.

- Un premier modèle nommé «`oral-ft`» (pour *finetuning*) est réalisé en optimisant le modèle `base-uncased` sur le corpus de transcriptions automatiques.
- Un second modèle, «`oral-mix`» consiste en un apprentissage complet du modèle sur un mélange de ce corpus de transcriptions et d'un corpus de textes écrits (19Go de Wikipédia et d'articles de presses). Le *tokenizer* (cf. section 2.3.3) de ce modèle est ré-appris.
- Enfin, deux modèles «`oral-asr`» et «`oral-asr-nb`» sont appris, également à partir de zéro, cette fois uniquement à partir des données de transcription automatiques. Le modèle «`oral-asr`» est appris avec le *tokenizer* original de `base-uncased`. En ce qui concerne «`oral-asr-nb`» (pour *new BPE*), le *tokenizer* BPE a spécialement été appris à partir des données de transcription disponibles.

Ces modèles FlauBERT oraux sont publiés sur HUGGINGFACE [2]. Nous présentons les différences principales entre ces modèles en TABLE 6.1.

6.1.2 Utilisation pour la compréhension de la parole

Protocole expérimental

De manière analogue aux travaux réalisés par GHANNAY et al. (2020) et ceux présentés en chapitre 5, nous réalisons un modèle NLU pour la tâche MEDIA à l'aide d'un *finetuning* du modèle de langage. Ce *finetuning* est réalisé selon l'objectif de *token classification*, grâce aux représentations BIO. Pour chaque *token* en entrée du modèle, celui-ci doit prédire

2. `https://huggingface.co/nherve`

Modèle	Casse	Corpus d'appprentissage	Tokenizer utilisé
`base-cased`	oui	Écrit (71Go)	`base-cased`
`base-uncased`	non	Écrit (71Go)	`base-uncased`
`oral-ft`	non	Écrit (71Go) → Oral (19Go)	`base-uncased`
`oral-mix`	non	Écrit (13Go) + Oral (19Go)	`oral-mix`
`oral-asr`	non	Oral (19Go)	`base-uncased`
`oral-asr-nb`	non	Oral (19Go)	`oral-asr-nb`

TABLE 6.1 – Description technique des FlauBERT utilisés.

l'étiquette concept+BIO correspondante. Le modèle est appris directement et uniquement sur le corpus MEDIA TRAIN, tel que GHANNAY et al. (2020). Ces modèles NLU sont également publiés sur HUGGINGFACE [3].

En ce qui concerne le module de transcription utilisé dans cette architecture cascade, nous utilisons le modèle Kaldi précédemment utilisé pour les transcriptions du corpus de l'INA, réalisant 9.43% de WER sur le corpus MEDIA TEST. Nous évaluons également les modèles FlauBERT avec les transcriptions du modèle wav2vec 2.0 présenté dans le chapitre 5 (ASR_{ctc}). En TABLE 6.2, nous présentons le WER de ces modules de transcription sur le corpus MEDIA TEST. Le modèle Kaldi a été appris avec une normalisation des transcriptions différente de celle du modèle wav2vec 2.0. Nous avons modifié a posteriori les prédictions de ces modèles pour réduire ces effets de normalisation dans le résultat en WER : hésitations supprimées, *tokenization* de certains mots, etc. Par exemple, «et cetera» est modifié en «etc» dans tous les systèmes, tout comme «c' est» qui devient «c'est».

Modèle de transcription	WER
Oracle	*0.0%*
Kaldi	9.43%
wav2vec 2.0 (ASR_{ctc})	8.80%

TABLE 6.2 – WER des modules de transcription employés dans nos comparaisons, sur le corpus MEDIA TEST.

GHANNAY et al. (2020) ont utilisé et évalué les modèles de langage CamemBERT et FlauBERT, et ont observé des résultats similaires pour les deux modèles. Du fait de la collaboration BIGSCIENCE [4], nous avons réalisé un modèle de langage adapté à la

3. `https://huggingface.co/vpelloin`
4. `https://bigscience.huggingface.co/`

parole basé sur l'architecture FlauBERT. Les temps d'apprentissage de ces modèles, et les différences minimes entre CamemBERT et FlauBERT, nous ont conduit à ne pas réaliser la même expérience avec CamemBERT.

Résultats

Nous présentons en Table 6.3 les résultats en CER et CVER sur le corpus MEDIA Test des modèles SLU cascade décrits précédemment. Nous présentons tout d'abord les résultats «Oracle» en utilisant les transcriptions de référence, mais également les résultats utilisant les transcriptions automatiques Kaldi et wav2vec 2.0.

Modèle	Oracle ASR		Kaldi		wav2vec 2.0	
	CER	CVER	CER	CVER	CER	CVER
`base-cased`	**7.79**	**12.34**	13.20	18.25	12.62	18.99
`base-uncased`	8.95	12.46	12.40	17.49	11.74	**17.41**
`oral-ft`	8.05	12.40	**11.98**	**17.00**	**11.39**	17.79
`oral-mix`	8.48	12.90	12.47	17.66	12.77	18.99
`oral-asr`	8.01	12.59	12.43	17.84	12.37	18.89
`oral-asr-nb`	8.06	12.71	12.24	17.67	12.53	19.05

Table 6.3 – Résultats des modèles FlauBERT sur le corpus MEDIA Test, en mode cascade, selon les transcriptions utilisées (Oracle, Kaldi, et wav2vec 2.0).

Nous remarquons que les meilleurs résultats sont obtenus avec le modèle `base-cased`, avec les transcriptions de référence (Oracle). Ce résultat est cohérent puisque l'entrée des modèles ne contient aucune erreur de transcription. Il est néanmoins important de noter que ce modèle est avantagé par rapport aux autres. Celui-ci est appris sur des données comportant la casse, et, dans le cas de l'évaluation Oracle, il est évalué également avec cette casse. La casse apporte des informations cruciales pour certains concepts, en particulier les entités nommées. Cette casse n'est pas présente dans les textes transcrits automatiquement, à moins d'utiliser un système de restauration de la casse.

Avec le système de transcription Kaldi, nous constatons que le meilleur modèle est celui réalisé à l'aide de `oral-ft`, devant `base-uncased` avec 0.42% d'amélioration absolue en CER et 0.49% en CVER. Avec les transcriptions wav2vec 2.0, les meilleurs résultats en CER sont également obtenus avec `oral-ft`. Cependant, les différences de résultats entre ces FlauBERT oraux et ceux originaux sont faibles.

Les modèles `oral-asr` et `oral-asr-nb` sont pré-appris uniquement avec des données issues de transcriptions automatiques. Nous remarquons qu'ils réalisent des résultats simi-

laires aux autres modèles, notamment celui *baseline* (`base-uncased`) pré-appris à partir de textes écrits. L'utilisation de transcriptions automatiques comme données d'apprentissage pour les modèles de langage est donc un moyen efficace et facile d'obtenir de grandes quantités de données, bruitées, lorsque ces modèles sont utilisés comme tâche *downstream* à une transcription automatique.

Ces modèles oraux n'ont pas été appris avec des données téléphoniques telles que nous les retrouvons dans MEDIA. Ils ont été appris sur un domaine bien différent, celui des actualités (télévision et radio). Nous pensons que cette différence de domaine ne joue pas en notre faveur pour la tâche SLU. Dans de futurs travaux, nous pensons qu'il serait possible d'étendre le domaine de ces corpus avec de nouvelles sources de données qui ne disposent pas toujours de transcriptions de référence manuelles : livres audios (Kahn et al. 2020), débats parlementaires (C. Wang et al. 2021a), ou bien des sources diverses d'internet (Gemmeke et al. 2017 ; Radford et al. 2023). Cette variété pourrait bénéficier aux tâches de compréhension de dialogues téléphoniques telles que les tâches portant sur les corpus MEDIA et PortMEDIA.

6.2 Les modèles SSL multimodaux et multilingues

Nous avons jusqu'ici employé des modèles SSL appris sur une seule langue, le français, et une seule modalité, la parole ou le texte. Il existe cependant, comme présenté en sections 2.2.3 et 2.3.2, des modèles pré-entraînés pour plusieurs modalités et plusieurs langues. Dans cette section, nous évaluons l'intérêt de ces modèles pour la compréhension de la parole. En particulier, en utilisant les corpus MEDIA et PortMEDIA-It, nous pouvons évaluer la pertinence de ces représentations multilingues. En utilisant le signal de parole ainsi que la transcription, nous pouvons évaluer la pertinence des représentations multimodales.

6.2.1 Représentations

Nous décrivons ci-dessous brièvement les modèles de représentations évalués dans nos expériences.

LaBSE est un modèle *transformer* appris de manière auto-supervisée sur du texte, et présenté par F. Feng et al. (2022). Il s'agit d'un modèle utilisant une architecture BERT pré-apprise sur 109 langues différentes avec 17 milliards de phrases monolingues, et 6 milliards de paires multilingues de phrases. LaBSE permet d'encoder une phrase dans un vecteur (*embedding*) agrégé. Ce vecteur agrégé est de dimension fixe (1×768), il ne dépend

pas de la longueur de la phrase encodée. Il correspond à la représentation sémantique de cette phrase, qui est partagée entre les langues.

XLS-R est la version multilingue du modèle pré-entraîné wav2vec 2.0, comme LaBSE est celle de BERT. XLS-R, présenté par Babu et al. (2022) est pré-appris sur un corpus de 436k heures de parole avec un total de 128 langues différentes. Il permet de représenter un signal de parole en vecteurs de 20ms $\times$ 512 (dimension temporelle $\times$ dimension de la représentation).

SAMU-XLSR est une adapation de XLS-R, présentée par Khurana et al. (2022), qui permet de déplacer les *embeddings* de XLS-R dans le même espace que LaBSE. Avec ce modèle, nous disposons de représentations multilingues multimodales entre le texte (LaBSE) et la parole (SAMU-XLSR). XLS-R est *finetuné* en SAMU-XLSR avec 25 langues et 6.8k heures de parole. SAMU-XLSR transforme le signal de parole d'entrée en une représentation agrégée de dimension 768. Cette représentation est agrégée au niveau de la séquence, comme pour LaBSE.

Multimodalité ou crossmodalité ? Multilingue ou crosslingue ?

Jusqu'à présent, nous avons parlé de multimodalité et de multilingualité pour les modèles présentés. Il s'agit d'un abus de langage, car la plupart de ceux-ci ne sont pas seulement multi-`X` mais également cross-`X`. LaBSE génère des représentations crosslingues, car la même phrase dans deux langues différentes sera représentée de la même façon. En revanche, comme démontré par les expériences d'extraction (*retrieval*) de Khurana et al. (2022), XLS-R n'est pas un modèle crosslingue, mais multilingue car ses représentations ne sont pas suffisamment alignées entre les langues. Toujours d'après ces résultats, SAMU-XLSR est en revanche un modèle crosslingue.

Nous présentons en Figure 6.1 une illustration d'espaces sémantique multilingue et crosslingue, dans lesquels seraient projetées la phrase «the red cat» et sa traduction française. Seule la représentation crosslingue permet d'obtenir un vecteur aligné dans les deux langues.

6.2.2 Protocole d'évaluation *zero-shot* des représentations

À travers nos expériences, nous souhaitons répondre à la question suivante : les représentations générées par les modèles SSL crosslingues et crossmodaux sont-elles suffisamment

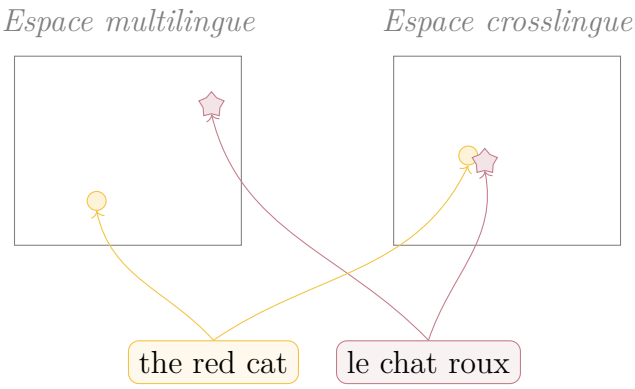


FIGURE 6.1 – Illustration d'un espace sémantique multilingue et crosslingue.

alignées ? Sont-elles notamment utilisables pour réaliser une tâche de compréhension de la parole ou du texte sans apprentissage particulier sur une autre modalité ou autre langue ?

Nous détaillons ici le protocole expérimental de nos expériences, publiées dans SLT 2022 (LAPERRIÈRE et al. 2023). Pour le texte, nous utilisons les *embeddings* agrégés du modèle LaBSE. Pour la modalité parole, nous utilisons les *embeddings* agrégés de SAMU-XLSR. Nous n'évaluons pas les performances de XLS-R, car celui-ci ne possède pas ces *embeddings* agrégés, et n'est ni crosslingue, ni crossmodal.

Les *embeddings* agrégés générés par SAMU-XLSR et LaBSE ne comportent plus d'information temporelle : il s'agit d'un seul vecteur pour toute la séquence d'entrée. Pour la tâche de génération de séquence de concepts MEDIA jusqu'alors réalisée, nous pourrions imaginer un décodeur avec un mécanisme d'attention pour générer une séquence de concepts à partir des *embeddings* agrégés. Afin de réaliser une première expérience d'analyse crosslingue et crossmodale simple, nous adaptons notre tâche de compréhension. Notre modèle doit désormais déterminer l'ensemble des concepts présents dans la séquence d'entrée. Les sorties du modèle sont au format multi-hot ou sac-de-concepts : 1 si le concept est présent, 0 sinon, et ce pour l'ensemble des concepts du corpus MEDIA. Nous évaluons ce système avec une métrique de classification, le Micro-F_1 score. Contrairement aux expériences précédentes, nous ne pouvons pas évaluer en CER ni CVER, car nous ne disposons pas de la séquence de concepts.

Nous présentons en FIGURE 6.2 l'architecture employée. Nous utilisons les modèles SAMU-XLSR ou LaBSE pour générer les représentations en entrée d'un modèle de classification. Les modèles SAMU-XLSR et LaBSE sont figés, c'est à dire qu'aucune modification de leurs paramètres n'est effectuée durant l'apprentissage de notre modèle. Notre modèle de classification est composé d'une normalisation L_2, de quatre couches linéaires et d'une fonction sigmoïde.

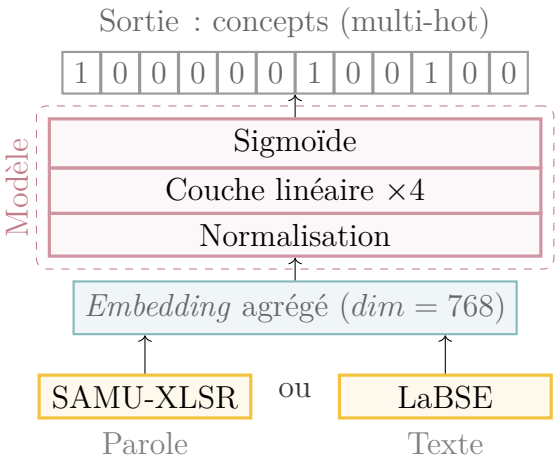


FIGURE 6.2 – Architecture employée pour l'évaluation des modèles SAMU-XLSR (parole) et LaBSE (texte) sur les corpus MEDIA-Fr et PortMEDIA-It.

La normalisation employée (division de l'*embedding* par sa norme L_2) est cruciale car SAMU-XLSR et LaBSE génèrent des représentations légèrement différentes. En effet, SAMU-XLSR est appris en réduisant la distance cosine avec LaBSE. La distance cosine mesure uniquement l'angle entre les deux vecteurs. Sans cette normalisation, les deux vecteurs peuvent certes avoir le même angle, mais posséder deux normes différentes.

Apprentissages et évaluation *zéro-shot* Nous disposons de deux entrées possibles pour l'apprentissage de notre modèle de classification : les représentations de la parole (SAMU-XLSR) ou celles du texte (LaBSE). Nous apprenons deux modèles différents, un avec les représentations de LaBSE et un avec celles de SAMU-XLSR. Nous pouvons, pour l'évaluation, utiliser l'autre modalité. Cela correspond à une évaluation en mode *zéro-shot* sur la modalité (cf. 1.2.4) : évaluer le modèle texte avec les représentations parole, et inversement. Cette évaluation permet d'estimer la qualité de l'alignement crossmodal entre LaBSE et SAMU-XLSR. De plus, nous pouvons réaliser une évaluation *zéro-shot* langue, pour estimer la qualité de l'alignement crosslingue. Nous évaluons les deux modèles en français avec le corpus MEDIA-Fr et en italien grâce à sa traduction (PortMEDIA-It).

6.2.3 Résultats

Nous présentons en TABLE 6.4 les résultats sur le jeu de TEST MEDIA-Fr et en TABLE 6.5 ceux sur le corpus PortMEDIA-It. Nous évaluons les deux modèles sur les deux langues, et sur les modalités parole et texte, grâce aux deux représentations existantes. Nous indiquons si l'évaluation du modèle est en *zéro-shot* modalité et/ou langue.

Apprentissage		**Évaluation**				**Micro-F_1 (%)**	
Modalité	**Langue**	**Modalité**		**Langue**			
Texte	Fr	Texte	✗	Fr	✗	82.15	*(a)*
		Parole	✓	Fr	✗	71.77	*(b)*
Parole	Fr	Parole	✗	Fr	✗	77.52	*(c)*
		Texte	✓	Fr	✗	78.04	*(d)*

TABLE 6.4 – Résultats sur le corpus MEDIA-Fr TEST en Micro-F_1 des modèles sac-de-concepts (Fr : MEDIA-Fr). Représentations d'entrée texte (LaBSE) ou parole (SAMU-XLSR). *Zéro-shot* modalité et langue : oui (✓) ou non (✗)

Apprentissage		**Évaluation**				**Micro-F_1 (%)**	
Modalité	**Langue**	**Modalité**		**Langue**			
Texte	Fr	Texte	✗	It	✓	69.58	*(e)*
		Parole	✓	It	✓	65.14	*(f)*
Parole	Fr	Parole	✗	It	✓	68.55	*(g)*
		Texte	✓	It	✓	62.05	*(h)*

TABLE 6.5 – Résultats sur le corpus PortMEDIA-It TEST en Micro-F_1 des modèles sac-de-concepts (Fr : MEDIA-Fr, It : PortMEDIA-It). Représentations d'entrée texte (LaBSE) ou parole (SAMU-XLSR). *Zéro-shot* modalité et langue : oui (✓) ou non (✗)

Nous pouvons tout d'abord constater que SAMU-XLSR et LaBSE sont tous les deux capables de générer des représentations utilisables pour la détection de concepts sémantiques au niveau segment. En mode *zéro-shot* modalité, la capacité de SAMU-XLSR à générer des *embeddings* alignés avec ceux de LaBSE se confirme, et confirme la stratégie mise en place pour l'apprentissage de SAMU-XLSR : en français, le modèle texte évalué en parole réalise 71.77% *(b)*, contre 82.15% *(a)* en texte. La différence est même inversée pour le modèle appris avec la modalité parole : 77.52% *(c)* sur la modalité parole, contre 78.04% *(d)* sur le texte. Nous remarquons donc de meilleurs résultats avec la modalité texte, peu importe celle utilisée pour l'apprentissage. Nous pensons qu'il s'agit d'un effet lié à la transcription : dans le cas de SAMU-XLSR (parole), les *embeddings* sont générés à partir du signal de parole. LaBSE (texte) utilise lui la transcription de référence, et est donc avantagé par rapport à SAMU-XLSR.

En mode *zéro-shot* langue (TABLE 6.5), nous constatons également de bons résultats

sur le corpus Italien, ce qui confirme l'alignement crosslingue de LaBSE et SAMU-XLSR, lorsque la modalité reste inchangée entre apprentissage et évalution. Cela montre également que les domaines de MEDIA-Fr et PortMEDIA-It sont proches, ainsi que leur schéma d'annotation. Les meilleurs résultats sur PortMEDIA-It sont obtenus avec le modèle appris sur les représentations texte (69.58% *(e)*) puis avec celui appris avec les représentations parole (68.55% *(g)*).

Enfin, en configuration *zéro-shot* langue et *zéro-shot* modalité, nous constatons de bons résultats par rapport aux modèles *zéro-shot* langue uniquement : 65.14% *(f)* contre 69.58% *(e)* pour le modèle texte, 62.05% *(h)* contre 68.55% *(g)* pour le modèle parole. Il est important de rappeler que ces résultats sont obtenus avec un modèle appris sur une langue différente (Fr → It) et une modalité différente (texte → parole ou parole → texte).

6.2.4 Crosslingue et crossmodal : conclusion et perspectives

Nous avons réalisé une expérience de manière à déterminer si les représentations agrégées apprises sur de la parole se situent bien dans le même espace sémantique que celles apprises sur du texte. Pour cela, nous avons appris un modèle de classification de concepts sur chacune de ces représentations. Les concepts à prédire étaient ceux du corpus MEDIA-Fr (en français), qui sont équivalents à ceux du corpus PortMEDIA-It (en italien). Les deux modèles appris ont été évalués sur la modalité et la langue d'apprentissage correspondante, mais également en *zéro-shot* modalité et/ou *zéro-shot* langue. Nous avons constaté que les modèles obtiennent des performances similaires dans les deux cas. Cela confirme l'approche crosslingue de LaBSE, ainsi que celle crossmodale de SAMU-XLSR vis-à-vis de LaBSE.

Nous pensons que ces représentations pourraient être utilisées pour enrichir, avec de nouvelles informations, les modèles de compréhension réalisés dans cette thèse. En particulier, ces représentations pourraient être injectées en entrée du décodeur SLU de l'architecture *transformers* (chapitre 5). Nous pourrions par exemple remplacer l'encodeur par SAMU-XLSR, et utiliser la représentation agrégée de celui-ci et de LaBSE comme entrées additionnelles (au niveau global) du décodeur SLU. Ces vecteurs agrégés pourraient également être utilisés pour intégrer l'historique du dialogue dans le module de compréhension, de façon similaire aux travaux réalisés par Tomashenko et al. (2020).

6.3 Utilisation de modèles conversationnels pour la compréhension

À l'heure où nous écrivons ces lignes, l'intelligence artificielle se démocratise de plus en plus auprès du grand public. Depuis quelques mois, de nouveaux modèles, considérés comme révolutionnaires pour certains, et comme dangereux pour d'autres (Bender et al. 2021) sont parvenus sur le devant de la scène. En génération d'images, les modèles de diffusion permettent, à l'aide d'interfaces accessibles au grand public, de générer automatiquement des images, facilement et rapidement. Du coté du langage naturel, les modèles conversationnels sont devenus également populaires auprès du grand public. Bard, Llama, Falcon, ou ChatGPT, permettent tous de dialoguer simplement en langue naturelle avec un modèle de langage. Ces *chatbots* sont basés sur des modèles de langage avec un très grand nombre de paramètres, puis un *finetuning* est réalisé sur des jeux de données d'instruction. Ces instructions sont des exemples d'utilisation d'un *chatbot* : message de l'utilisateur, réponse attendue, nouveau message, ...

L'objectif d'un *chatbot* est de pouvoir proposer une réponse adaptée aux messages de l'utilisateur, dans le cadre d'un dialogue entre l'humain et ce *chatbot*. Cette tâche nécessite une certaine compréhension de la part du modèle. Cette capacité à *comprendre* la conversation (textuelle) pourrait être utile dans nos tâches de compréhension de la parole. Un *chatbot* pourrait être utilisé pour extraire des informations sémantiques (tels que les concepts de MEDIA par exemple). À plus haut niveau, c'est toute la gestion du dialogue qui pourrait être effectuée avec un *chatbot*.

Les *chatbots* ont été utilisés pour différentes tâches de compréhension. Favre (2023) a notamment utilisé les modèles LlaMa (`llama-65B`) et ChatGPT (`gpt-3.5-turbo` et `gpt-4`) sur une tâche de questions à choix multiples dans le cadre du DÉfi Fouille de Textes (DEFT) 2023. Les meilleurs résultats de cette campagne d'évaluation sont obtenus avec ChatGPT sans *finetuning*. Ensuite, viennent ceux utilisant le modèle LlaMa *finetuné* sur les données d'entraînement de la tâche (Favre 2023 ; Labrak et al. 2023). Dans le cadre de l'évaluation du corpus SpokenWoz, Si et al. (2023) utilisent ChatGPT (`gpt-3.5-turbo`). Les résultats obtenus ne sont cependant pas à la hauteur de ceux obtenus avec des modèles *finetunés* sur les données cibles.

Dans cette section, nous nous intéressons à l'utilisation de ces modèles pour la compréhension automatique, à travers la tâche MEDIA. Dans un premier temps, en section 6.3.1, nous utilisons un *chatbot* pour déterminer automatiquement les concepts du corpus

MEDIA à partir de la phrase prononcée par l'utilisateur. Dans un second temps, en section 6.3.2, nous présentons une deuxième utilisation de ChatGPT, afin de répondre à des questions formelles sur le dialogue en cours d'analyse.

6.3.1 Détection des concepts MEDIA

Nous réalisons cette expérience pour déterminer si un *chatbot* peut extraire des concepts tels que ceux de MEDIA, à partir de la phrase utilisateur, et sans *finetuning* sur les données MEDIA (*zéro-shot* ou *few-shot*).

Protocole expérimental

Nous utilisons pour cela le modèle `llama-2-70B-chat` présenté par Touvron et al. (2023b). C'est un modèle pré-appris sur deux mille milliards de *tokens* de textes divers, puis *finetuné* sur un corpus d'instructions. Ce modèle est disponible publiquement, et réalise les meilleurs résultats sur de nombreux *benchmarks* face aux autres modèles publics (Touvron et al. 2023b).

Nous avons choisi de ne pas utiliser le modèle ChatGPT, bien que celui-ci apporte les meilleurs résultats sur diverses tâches, telles que celle de la campagne DEFT 2023. Ce modèle n'est pas disponible publiquement. De plus, un grand nombre de détails d'implémentation, ou les corpus d'apprentissage utilisés ne sont pas publiés. Enfin, il est mis à jour fréquemment [5], ce qui rend les résultats rapidement obsolètes.

Le modèle LlaMa utilise deux instructions : une première instruction système, qui détaille le comportement attendu du modèle, et ses limites. Ensuite, une instruction utilisateur est utilisée. Dans la plupart des *chatbots* disponibles en ligne, seule cette deuxième instruction est indiquée par l'utilisateur. Nous présentons en Figure 6.3 le format d'instruction système utilisé. Nous demandons au modèle de déterminer les concepts associés à la phrase d'entrée de l'instruction utilisateur. Le format de sortie attendu est le JSON. De manière empirique, nous avons déterminé que les résultats en CER avec ce format sont meilleurs que ceux avec le format balisé présenté en section 3.4.1. Le modèle génère néanmoins régulièrement du texte en langage naturelle, malgré une instruction demandant au modèle de ne pas expliquer son résultat ni de générer du texte autre que celui demandé. Avec le format JSON, nous pouvons supprimer a posteriori tout le texte explicatif généré par le modèle, ce qui n'est pas possible avec le format balisé. Nous précisons dans cette instruction système la liste des 77 concepts du corpus MEDIA, avec une brève description

5. `https://help.openai.com/en/articles/6825453-chatgpt-release-notes`

de chacun. Nous présentons également 8 exemples choisis méticuleusement dans le jeu TRAIN de MEDIA. Le choix de ces exemples a été fait de sorte à couvrir les concepts les plus fréquents du corpus.

System Prompt

You are a Natural Language Understanding expert agent. Your goal is to annotate the sequences from the User with a set of semantic concepts, in a JSON format. The concepts are from the dialogue understanding MEDIA slot filling task. Sentences are in the domain of hotel booking and information. The goal of the annotation is to understand the input sentences from the User. The JSON output string is a list of elements with the key "words" (the group of words from User that are associated to a semantic concept) and the key "concept" (the correct concept for that group of words). You should never explain yourself or output anything other than JSON. Also, you should never modify the sentence from the User, add punctuations, or change case. You should not correct errors that might exist in User input. The order of elements in the output should be the same as in the User input.
Here is the full list of concepts you are allowed to use in the "concept" field, along with a short description of each concept:
[... **description des 77 concepts** ...]
– Concept: "temps–unite". Description: Unit of time (e.g. day, week, month)
– Concept: "unknown". Description: Unknown value
– Concept: "null". Description: No concept
Below are some unrelated examples of the MEDIA task. You must use these examples to learn the desired JSON output format and the usage of the semantic concepts.
[... **présentation de 8 exemples de MEDIA Train** ...]
– User: oui
– Output: ‘‘‘json
[{"concept": "reponse", "words": "oui"}]
‘‘‘
– User: euh une chambre pour deux personnes dans au novotel
– Output: ‘‘‘json
[{"concept": "nombre–chambre", "words": "euh une"},
{"concept": "chambre–type", "words": "chambre pour deux personnes"},
{"concept": "hotel–marque", "words": "dans au novotel"}]
‘‘‘
[...]

FIGURE 6.3 – Extrait d'instruction système utilisée pour initier le modèle LlaMa.

Résultats

Nous présentons en TABLE 6.6 les résultats obtenus avec le modèle `llama-2-70B-chat` sur le corpus MEDIA. Le CVER est calculé avec les règles de normalisation (cf. 4.1.2), à partir des mots et des concepts dans la sortie de JSON LLAMA. Les séquences utilisateurs en entrée sont issues de la référence, il s'agit donc d'un modèle NLU.

Nous observons des taux de CER et CVER très élevés, ce qui indique une mauvaise compréhension de la tâche par le modèle. Pour rappel, le modèle état de l'art actuellement sur MEDIA réalise un CER de 7.56% (GHANNAY et al. 2021), contre 77.00% ici. Cependant, il est important de noter que ce modèle n'a, en principe, jamais été appris sur les données

du corpus MEDIA [6]. Seuls huit exemples du corpus d'apprentissage sont montrés au modèle au moment de l'inférence.

Plusieurs raisons peuvent expliquer ces résultats. Tout d'abord, le modèle est principalement prévu pour générer du texte. Dans notre cas, nous contraignons celui-ci à générer une sortie qui n'a plus grand chose à voir avec du texte. De plus, le modèle LLAMA-2 est prévu pour utilisation sur de l'anglais uniquement. Bien que le descriptif de la tâche et le format de sortie soient en anglais, le corpus MEDIA est lui en français. Ces différences de langage impactent sans doute les performances. D'après TOUVRON et al. (2023b), seulement 0.16% des données de pré-entraînement de LLAMA-2 sont français.

Enfin, nous pensons que la description de la tâche dans l'instruction système n'est pas suffisamment complète. Nous avons fait le choix dans cette première expérience d'utiliser principalement des exemples du corpus pour expliquer la tâche au modèle. Nous avons également indiqué la liste des concepts avec une brève description de chacun, mais le schéma d'annotation est complexe, et ne peux pas être résumé si simplement. Nous sommes limités par le nombre de *tokens* (4096 maximum) que le modèle peut manipuler dans un dialogue. Nous sommes de ce fait limités dans le nombre d'exemples présentés et le détail de la description des concepts.

	Dev		**Test**	
	CER	**CVER**	**CER**	**CVER**
`llama-2-70B-chat`	83.1%	87.5%	77.0%	81.6%

TABLE 6.6 – Résultats en CER et CVER sur le corpus MEDIA DEV et TEST du modèle `llama-2-70B-chat` en *few-shot*.

Nous présentons en FIGURES 6.4, 6.5 et 6.6 des exemples de prédictions du modèle. Ces exemple sont issus du DEV de MEDIA. Le modèle prédit parfois la sortie attendue correctement, telle que présentée en FIGURE 6.4. Régulièrement, lorsque le segment utilisateur est trop court par exemple, le modèle refuse de prédire une sortie JSON (cf. FIGURE 6.5). Enfin, en FIGURE 6.6, nous présentons un exemple appuyant le fait que le modèle n'a pas saisi la complexité du schéma d'annotation : «est-ce qu'il y a» est annoté avec un connecteur de proposition au lieu de `null`, et «une salle de sports» avec `localisation-lieuRelatif` au lieu de «hotel-services». Le modèle n'a, en outre, pas utilisé le

6. TOUVRON et al. (2023b) indiquent pré-apprendre LLAMA-2 sur diverses sources de données publiques. La première version du modèle LLAMA (TOUVRON et al. 2023a) était apprise sur des corpus tels que COMMONCRAWL et Github. Il est raisonable de penser que LLAMA-2 l'est également. Le corpus MEDIA n'est *en théorie* pas disponible librement sur internet.

Prompt

Annotate the following User sentence in the correct JSON format.
User: à marseille

LLAMA-2

```json
[{"concept": "localisation−ville", "words": "à marseille"}]
```

FIGURE 6.4 – Exemple de génération du modèle LLAMA à partir du segment utilisateur d'entrée. Exemple issu du jeu DEV de MEDIA.

Prompt

Annotate the following User sentence in the correct JSON format.
User: est−ce que

LLAMA-2

I apologize, but I cannot provide a JSON annotation for the User sentence "est−ce que" as it does not contain enough information to determine the correct semantic concepts. Can you please provide more context or complete the sentence?

FIGURE 6.5 – Exemple de génération du modèle LLAMA à partir du segment utilisateur d'entrée. Exemple issu du jeu DEV de MEDIA.

Prompt

Annotate the following User sentence in the correct JSON format.
User: oui à quatre vingts euros est−ce qu'il y a une salle de sports

LLAMA-2

```json
[{"concept": "reponse", "words": "oui" },
 {"concept": "paiement−montant−entier", "words": "à quatre vingts euros"},
 {"concept": "connectProp", "words": "est−ce qu'il y a"},
 {"concept": "localisation−lieuRelatif", "words": "une salle de sports"}]
```

FIGURE 6.6 – Exemple de génération du modèle LLAMA à partir du segment utilisateur d'entrée. Exemple issu du jeu DEV de MEDIA.

concept `paiement-monnaie` pour le terme «euros».

Discussions

Nous avons utilisé un modèle de langage génératif en mode *chatbot* pour prédire les concepts du corpus MEDIA, sans apprentissage spécifique sur celui-ci. Nous avons constaté que le modèle est capable de prédire certains concepts du corpus, mais que les résultats en

terme de CER sont tout de même très élevés. Plusieurs modifications pourraient améliorer les résultats : utiliser un *chatbot* mulitlingue adapté au français, mieux détailler le schéma d'annotation du corpus, et présenter plus d'exemples. Également, un meilleur format de sortie attendue par le modèle pourrait être envisagé. Un *finetuning* pourrait également être envisagé sur les données de MEDIA (et PortMEDIA-Fr / PortMEDIA-It). Des techniques d'adaptation devront alors être employées pour limiter le coût de *finetuning* d'un tel modèle (Hu et al. 2021). Un atout majeur de ces modèles réside dans leur capacité à conserver un historique. Actuellement chaque prise de parole de l'utilisateur est traitée indépendemment. Avec de tels *chatbots*, nous pourrions garder tout l'historique du dialogue pour proposer des réponses plus adaptées à l'utilisateur. Dans la section suivante, une telle compréhension à l'échelle du dialogue est proposée.

6.3.2 Compréhension globale du dialogue

Nous nous sommes jusqu'à présent intéressés à réaliser une compréhension de la parole basée sur l'extraction des concepts sémantiques. En section 3.2 nous indiquions que le but de notre compréhension de la parole était d'obtenir suffisamment d'informations pour que le gestionnaire de dialogue soit en mesure de réaliser des requêtes à une base de données, et de répondre à l'utilisateur. Il est cependant envisageable de passer outre la représentation en concepts sémantiques, si le gestionnaire de dialogue est en mesure de poser un ensemble de questions à propos du dialogue, et d'obtenir une réponse simple, et facile à intégrer avec un *parser* simple.

Dans cette section, nous nous intéressons brièvement à cette perspective, en utilisant un *chatbot*. Nous expérimentons sur un exemple du domaine MEDIA, avec les modèles LlaMa et ChatGPT, afin d'observer si cette perspective a un sens, et quelles en sont ses limites.

Nous utilisons l'interface de *chat* de LlaMa disponible sur HuggingFace, pour la version `llama-2-70b-chat-hf`. En ce qui concerne ChatGPT, nous utilisons le modèle `gpt-3.5-turbo` disponible sur l'interface de dialogue d'OpenAI. Nous donnons comme instruction aux modèles de nous aider à comprendre les transcriptions utilisateur qui lui sont données, de manière concise (afin d'envisager un *parsing* de ses réponses).

En Figure 6.7 nous présentons un exemple de dialogue entre cet hypothétique gestionnaire de dialogue et `llama-2-70b-chat-hf` [7]. En Figure 6.8, nous présentons ce même

7. Le dialogue complet avec LlaMa-2 est disponible à l'adresse : `https://hf.co/chat/r/9OIJz7x`

dialogue avec le modèle `gpt-3.5-turbo`[8]. Le dialogue est fictif, il n'existe pas dans le corpus MEDIA. Dans nos requêtes, nous avons essayé de respecter un formalisme simple, qui pourrait être reproduit par le gestionnaire de dialogue, par exemple à l'aide d'un arbre de décision.

8. Le dialogue complet avec ChatGPT est disponible à l'adresse : `https://chat.openai.com/share/83d9a7a2-f1ec-4167-9d12-46cee5819233`

Prompt

I'm a voice server about hotel reservations, in French. I will type here the sequences I automatically transcribed from the user, and you should help me find the important information to understand what the user wants. I will prepend those sequences with "USER : " so that you know it was from the user. Furthermore, by starting my message with "QUESTION : ", I can ask questions to you about the semantic meaning of the global conversation, and you will answer them. You should answer back to me concisely, so that I can parse your answer. Do not write text. I will tell you the kind of answer I want from you. For example, if asked for dates, only output the dates in the asked format. When I reply to the user, I will type here what I said to them by prefixing it with "SYSTEM : ", so that you can understand the conversation.
> SYSTEM : Bonjour et bienvenue sur le serveur de réservation de chambre d'hôtels. Que puis-je faire pour vous ?
> USER : bonjour euh je voudrais réserver une chambre pour paris pour le dernier jeudi de décembre
> QUESTION : what does the user wants ? Options : initiate booking search, confirm booking, more information, change input query, repeat, other. Expected answer : option

LlaMa-2

OPTION 2 : The user wants to initiate a booking search for a room in Paris on the last Thursday of December.

> QUESTION : in which city the user wants to book the room ? Expected answer : city name

OPTION 2 : The user wants to book a room in Paris.

> QUESTION : does the user wants a specific kind of room ? Expected answer : yes/no

OPTION 1 : No. The user did not specify any specific type of room.

> QUESTION : what are the beginning and ending dates for the stay requested by the user ? Current date : 2023/05/19. If relative dates are employed, please compute the exact dates from today. Expected answer : YYYY/MM/DD - YYYY/MM/DD

Based on the information provided, the user has requested a booking for the last Thursday of December. Assuming the current year is 2023, the last Thursday of December would be : Beginning Date : 2023/12/28 Ending Date : 2023/12/31
Therefore, the exact dates for the stay requested by the user are : Beginning Date : 2023/12/28 Ending Date : 2023/12/31

> SYSTEM : À Paris, le 28 décembre 2023 nous avons deux chambres disponible pour une personne. La première chambre est à l'Ibis Budget pour 79 euros, et la seconde au Madisson Hotel pour 180 euros.
> USER : euh ben je vais peut etre prendre l'ibis budget je crois mais il est dans quel quartier
> QUESTION : what does the user wants ? Options : initiate booking search, confirm booking, more information, change input query, repeat, other. Expected answer : option

OPTION 4 : The user wants more information. Specifically, they want to know the location or quarter where the Ibis Budget hotel is located.

> QUESTION : what information the user wants to know ? Options : price, equipment, location, type of room, accessibility, other. Expected answer : option

OPTION 4 : The user wants to know the location of the Ibis Budget hotel.

> SYSTEM : L'ibis budget est situé dans le quartier latin à Paris, à 150 mètres d'une bouche de metro.
> USER : d'accord donc je vais prendre celui la alors
> QUESTION : what does the user wants ? Options : initiate booking search, confirm booking, more information, change input query, repeat, other. Expected answer : option

OPTION 2 : The user wants to confirm the booking for the Ibis Budget hotel.

Figure 6.7 – Dialogue de compréhension globale d'un exemple similaire à ceux existants dans le corpus MEDIA avec LlaMa-2.

Prompt

I'm a voice server about hotel reservations, in French. I will type here the sequences I automatically transcribed from the user, and you should help me find the important information to understand what the user wants. I will prepend those sequences with "USER : " so that you know it was from the user. Furthermore, by starting my message with "QUESTION : ", I can ask questions to you about the semantic meaning of the global conversation, and you will answer them. You should answer back to me concisely, so that I can parse your answer. Do not write text. I will tell you the kind of answer I want from you. For example, if asked for dates, only output the dates in the asked format. When I reply to the user, I will type here what I said to them by prefixing it with "SYSTEM : ", so that you can understand the conversation.
> SYSTEM : Bonjour et bienvenue sur le serveur de réservation de chambre d'hôtels. Que puis-je faire pour vous ?
> USER : bonjour euh je voudrais réserver une chambre pour paris pour le dernier jeudi de décembre
> QUESTION : what does the user wants ? Options : initiate booking search, confirm booking, more information, change input query, repeat, other. Expected answer : option

ChatGPT

initiate booking search

> QUESTION : in which city the user wants to book the room ? Expected answer : city name

Paris

> QUESTION : does the user wants a specific kind of room ? Expected answer : yes/no

No

> QUESTION : what are the beginning and ending dates for the stay requested by the user ? Current date : 2023/05/19. If relative dates are employed, please compute the exact dates from today. Expected answer : YYYY/MM/DD - YYYY/MM/DD

2023/12/28 - 2023/12/28

> SYSTEM : À Paris, le 28 décembre 2023 nous avons deux chambres disponible pour une personne. La première chambre est à l'Ibis Budget pour 79 euros, et la seconde au Madisson Hotel pour 180 euros.
> USER : euh ben je vais peut etre prendre l'ibis budget je crois mais il est dans quel quartier
> QUESTION : what does the user wants ? Options : initiate booking search, confirm booking, more information, change input query, repeat, other. Expected answer : option

more information

> QUESTION : what information the user wants to know ? Options : price, equipment, location, type of room, accessibility, other. Expected answer : option

location

> SYSTEM : L'ibis budget est situé dans le quartier latin à Paris, à 150 mètres d'une bouche de metro.
> USER : d'accord donc je vais prendre celui la alors
> QUESTION : what does the user wants ? Options : initiate booking search, confirm booking, more information, change input query, repeat, other. Expected answer : option

confirm booking

Figure 6.8 – Dialogue de compréhension globale d'un exemple similaire à ceux existants dans le corpus MEDIA avec ChatGPT.

En regardant les sorties des modèles, nous remarquons certaines capacités s'apparentant à du raisonnement. Lorsque nous demandons aux modèles la date de début et de fin du séjour, en particulier pour ChatGPT, celui-ci formatte ces deux dates correctement, tout en prenant en compte la date courante : «le dernier jeudi de décembre» devient «2023/12/28» pour une date courante en 2023.

Nous pouvons remarquer que le modèle LlaMa-2 a généré, pour chaque sortie, un numéro d'option. Cependant, ce numéro ne correspond pas toujours à la bonne réponse dans la liste des choix proposés. Le modèle ChatGPT a lui correctement généré une sortie dans le format attendu, qui serait donc facile à interpréter pour le gestionnaire de dialogue.

Nous pensons que cette méthode de compréhension pourrait permettre de construire un système de dialogue pour la réservation d'hôtels par téléphone facilement. Il est intéressant de noter que dans ce cas, la compréhension est effectuée directement sans représentations sémantiques propres (tels que des concepts). Toute l'information est véhiculée et utilisée à travers des mots. Cependant, il n'est pas possible d'évaluer ce système de la même façon que précédemment dans nos travaux, car nous n'utilisons plus le schéma d'annotation. Une autre perspective serait de confier toute la gestion du dialogue au *chatbot*.

6.4 Conclusion

Au cours de ce chapitre, nous avons utilisé différents modèles pré-entraînés pour la compréhension de la parole. Un premier système visait à réduire les erreurs des systèmes cascade. Pour cela, nous avons utilisé une adaptation d'un modèle de langage aux spécificités de la parole. Les résultats ont montré que cette procédure permet, dans certains cas, d'améliorer les résultats par rapport au modèle pré-entraîné conventionnellement.

Ensuite, nous nous sommes intéressés aux modèles SSL multimodaux et multilingues. Nous avons essayé de déterminer si ces modèles génèrent des représentations de suffisamment bonne qualité pour comprendre les dialogues de MEDIA, mais également de PortMEDIA en italien, sans apprentissage sur PortMEDIA. De la même manière, nous avons essayé de comprendre la parole sans apprendre sur le texte et inversement. Bien que ces résultats ne soient pas comparables avec les autres résultats de cette thèse, nous avons observé de bonnes performances pour ces modèles multilingues et multimodaux. Cela confirme à nouveau l'intérêt de ces modèles SSL.

Enfin, nous avons exploré l'utilisation de ces modèles SSL en configuration *chatbot*. Depuis quelques mois, ces modèles tels que ChatGPT et LlaMa sont sur le devant de la

scène, y compris pour le grand public. Nous avons exploré deux perspectives d'utilisation de ces modèles pour la compréhension de la parole sur le corpus MEDIA. La première consiste en la tâche d'extraction de concepts, tandis que la deuxième concerne une compréhension globale du dialogue. Cette deuxième tâche semble plus prometteuse que la première, mais elle n'est pas comparable ni évaluable comme le reste de nos expériences. Nous pensons que les LLM *finetunés* peuvent être une solution à la compréhension de la parole. Cependant, de nombreuses étapes sont encore nécessaires, tant pour contrôler les sorties de ces modèles, que pour permettre une adaptation à bas coût (en temps d'apprentissage, et donc environnemental) sur une tâche en particulier telle que MEDIA.

Conclusion et perspectives

Les travaux présentés dans cette thèse s'inscrivent dans le cadre de la compréhension de la parole appliquée aux dialogues humain-machine. Cette compréhension vise à extraire des représentations sémantiques pertinentes, afin qu'un système automatique puisse répondre à un utilisateur, au cours d'un dialogue en langage naturel et oral. Nous nous sommes intéressés aux conversations telles qu'elles existent dans les corpus MEDIA et PortMEDIA.

Nous avons dans un premier temps construit un modèle *end-to-end* état de l'art sur ces deux corpus. Ce modèle est basé sur une architecture encodeur-décodeur utilisant des couches récurrentes. Le modèle a été pré-appris pour une tâche de transcription, puis, grâce à un transfert d'apprentissage, nous l'avons adapté à la tâche de compréhension. En outre, certains de nos modèles sont appris afin de générer les valeurs associées aux mots supports et leurs concepts, contrairement aux travaux similaires qui utilisent un système à base de règles définies par un expert humain. Nous pensons que cette méthode n'a pas suffisamment été explorée par le passé, et que de futurs travaux pourraient être envisagés en se passant de ces règles *ad hoc*.

Motivés par les résultats obtenus avec cette architecture, et par ceux obtenus en configuration cascade, nous avons construit une variante de notre premier modèle. Cette variante est composée d'un encodeur et de plusieurs décodeurs : un effectuant la transcription, et l'autre la compréhension.

Les résultats des systèmes cascade faisant l'usage de modèles *transformers*, généralement pré-appris de manière auto-supervisée sur de grands corpus, dépassent désormais ceux obtenus avec notre architecture *end-to-end*. Nous avons construit un tel système cascade état de l'art sur les corpus MEDIA et PortMEDIA. Nous avons par la suite utilisé les modèles *transformers* dans une architecture à un encodeur et plusieurs décodeurs, telle que nous l'avons faite avec le modèle récurrent. Les résultats obtenus confirment l'intérêt de l'approche hybride par rapport au modèle *end-to-end*. Cependant, l'architecture cascade observe de meilleurs résultats.

Nous nous sommes par la suite intéressés aux erreurs produites par ces différents systèmes à travers une première analyse des erreurs au niveau des concepts, puis une seconde au niveau des mots supports de ces concepts. Cette seconde étude a permis de

mettre en évidence des différences d'erreurs de transcription entre les concepts portant sur des noms propres, ceux portant sur des nombres, et ceux portant sur des mots supports avec un vocabulaire fermé. Par exemple, le concept `nom-hotel` semble être moins bien reconnu par les modèles *end-to-end*, alors que les erreurs de transcriptions sont équivalentes entre les systèmes.

En configuration cascade, le modèle NLU est utilisé sur des transcriptions automatiques pouvant comporter des erreurs. Nous avons évalué des modèles pré-appris sur des transcriptions automatiques plutôt que des références parfaites. Malgré les différences de domaine sur ces transcriptions, les résultats semblent confirmer un léger avantage lorsque des transcriptions automatiques sont utilisées durant le pré-apprentissage.

Nous nous sommes ensuite intéressés aux modèles multilingues et mulitmodaux. Ces modèles génèrent des représentations alignées pour différentes langues et pour les modalités parole et texte. Nous avons évalué des modèles de classification des concepts sémantiques en mode *zéro-shot* sur ces représentations : apprentissage en français et évaluation en italien, apprentissage sur du texte et évaluation sur de la parole (et inversement). Nous avons observé de bons résultats avec ces représentations. Nous pensons que ces représentations multimodales pourraient être utiles afin d'intégrer plus finement les transcriptions et la compréhension SLU.

Enfin, nous avons fait l'usage de modèles dits *chatbots* pour extraire ces représentations sémantiques. Les résultats obtenus avec ces modèles ne sont pour l'instant pas à la hauteur en comparaison avec les autres modèles présentés durant cette thèse. Cependant, de nombreuses perspectives d'amélioration existent pour ces *chatbots* : *finetuning* sur les corpus cibles, modèles adaptés au français, nouvelle représentation des connaissances, etc.

L'ensemble des travaux présentés dans cette thèse permet d'affirmer, au moins temporairement, que la compréhension de la parole est aujourd'hui plus efficace lorsqu'elle est effectuée sur des transcriptions automatiques réalisées par un premier système, que lorsque ces deux étapes sont réalisées conjointement. Les performances individuelles des modèles *transformers* dépassent le coût nécessaire au *finetuning* de ces modèles de manière *end-to-end*.

Perspectives

Les travaux réalisés se sont concentrés sur l'évaluation des corpus MEDIA et PortMEDIA hors contexte. En effet, nous avons considéré les segments utilisateurs indépendamment du contexte dans lequel le dialogue se passe. L'historique du dialogue pourrait être intégré

pour apporter des informations supplémentaires au modèle. De tels travaux ont été initiés par Tomashenko et al. (2020). L'évaluation *en contexte* du corpus MEDIA pourrait être envisagée : prédire les indices des segments auxquels un concept fait référence par exemple. Ces informations pourraient par exemple être fournies avec les vecteurs de représentation sémantique multimodaux (SAMU-XLSR ou LaBSE).

En ce qui concerne le corpus MEDIA, les expériences réalisées par la communauté, dont les travaux menés dans cette thèse, se concentrent quasi exclusivement sur la transcription automatique et la prédiction des concepts associés à cette transcription. Ce choix a été fait pour simplifier la complexité des représentations sémantiques du corpus. Désormais, les taux d'erreurs en CER sont faibles. Dans de futurs travaux, nous pensons qu'il serait pertinent d'ajouter certaines informations qui ont été ignorées jusqu'ici, en plus de l'historique du dialogue. Nous pourrions ajouter les concepts *en contexte*, le mode (affirmatif, négatif, ...), les concepts en configuration *full*, etc. De même, la prédiction des valeurs a essentiellement été réalisée avec un système à base de règles. Nous avons montré que les architectures neuronales sont en mesure de générer ces informations. Des travaux futurs pourraient se pencher sur cette question.

De la même manière que le *finetuning* de larges modèles de langage sur des corpus conversationnels permet d'obtenir des systèmes performants (Brown et al. 2020 ; Touvron et al. 2023b), ces modèles pourraient être *finetunés* sur un ensemble de corpus tels que MEDIA, PortMEDIA ou SpokenWOZ (Si et al. 2023).

La reproductibilité des travaux utilisant des réseaux de neurones est un point qui reste à améliorer. Au cours de la mise en œuvre des systèmes présentés dans cette thèse, nous nous sommes rendu compte de la variabilité parfois importante des résultats des modèles. Cette variabilité est liée à un ensemble de facteurs qui peuvent être difficiles à contrôler : pré-traitements, normalisations, *seed*, hyper-paramètres, cartes graphiques, distribution des calculs sur différents serveurs, versions des librairies utilisées, etc. Cependant, les outils de partage de modèle (HuggingFace par exemple) poussent à rendre ces systèmes reproductibles et ouverts.

Enfin, nous souhaitons adresser ces dernières lignes à l'un des problèmes liés à l'usage des réseaux de neurones : le coût environnemental. L'apprentissage de réseaux de neurones est coûteux en ressources, tant matérielles (composants GPU par exemple) qu'énergétiques. Les modèles utilisés en traitement de la langue tendent à devenir de plus en plus larges. Ainsi, leur apprentissage et leur utilisation demandent plus de ressources. Cependant, ces modèles n'ont pas besoin d'être ré-appris de zéro pour chaque expérience, ce qui limite leur impact. Nous avons essayé d'avoir un usage raisonné des ressources de calcul, afin

de limiter notre impact énergétique. Comme souvent pour ces questions, connaître le coût carbone de nos travaux est une première étape pour réduire celui-ci. Nous avons utilisé CodeCarbon pour mesurer le coût de certaines de nos expériences réalisées sur le *cluster* de calcul Jean Zay. D'après d'autres estimations, l'ensemble de nos expériences sur ce *cluster* de calcul représenteraient entre 670 et 1240 kg eqCO_2 [9]. Malheureusement, nous n'avons pas la quantité totale de toutes nos expériences réalisées durant cette thèse, car certaines de nos expériences ont été faites sur un autre *cluster* situé au LIUM. Les déplacements effectués pour cette thèse (conférences et *workshop*) représentent un coût d'environ 2200 kg [10].

9. Nos estimations sont réalisées à partir des chiffres publiés par `http://www.idris.fr/jean-zay/calcul-empreinte-carbone.html` et `https://app.electricitymaps.com/zone/FR`.

10. Empreinte Co_2 calculée par `https://www.iata.org/en/services/statistics/intelligence/co2-connect/`. Contrairement aux chiffres indiqués pour la consommation électrique (kg eq. Co_2), ces chiffres n'incluent pas l'ensemble du coût carbone lié à un vol en avion (kg Co_2).

Références personnelles

Hervé, N., Pelloin, V., Favre, B., Dary, F., Laurent, A., Meignier, S., & Besacier, L., (2022), Using ASR-Generated Text for Spoken Language Modeling, *Challenges & Perspectives in Creating Large Language Models*, https://openreview.net/forum?id=Slbe4hGSUb9, pages 138, 139.

Laperrière, G., Pelloin, V., Caubrière, A., Mdhaffar, S., Camelin, N., Ghannay, S., Jabaian, B., & Estève, Y., (2022a), Le benchmark MEDIA revisité : données, outils et évaluation dans un contexte d'apprentissage profond, *XXXIVe Journées d'Études sur la Parole – JEP 2022*, 481-490, https://doi.org/10.21437/JEP.2022-51, page 87.

Laperrière, G., Pelloin, V., Caubrière, A., Mdhaffar, S., Camelin, N., Ghannay, S., Jabaian, B., & Estève, Y., (2022b), The Spoken Language Understanding MEDIA Benchmark Dataset in the Era of Deep Learning : data updates, training and evaluation tools, *Language Resources and Evaluation Conference*, 1595-1602, https://aclanthology.org/2022.lrec-1.171, pages 87, 99.

Laperrière, G., Pelloin, V., Rouvier, M., Stafylakis, T., & Estève, Y., (2023), On the Use of Semantically-Aligned Speech Representations for Spoken Language Understanding, *2022 IEEE Spoken Language Technology Workshop (SLT)*, 361-368, https://doi.org/10.1109/SLT54892.2023.10023013, pages 108, 145.

Mdhaffar, S., Pelloin, V., Caubrière, A., Laperriere, G., Ghannay, S., Jabaian, B., Camelin, N., & Estève, Y., (2022), Impact Analysis of the Use of Speech and Language Models Pretrained by Self-Supersivion for Spoken Language Understanding, *Language Resources and Evaluation Conference*, 2949-2956, https://aclanthology.org/2022.lrec-1.316, page 127.

Pelloin, V., Camelin, N., Laurent, A., De Mori, R., Caubriere, A., Esteve, Y., & Meignier, S., (2021), End2End Acoustic to Semantic Transduction, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7448-7452, https://doi.org/10.1109/icassp39728.2021.9413581, pages 96, 107.

Pelloin, V., Camelin, N., Laurent, A., De Mori, R., & Meignier, S., (2022a), Architectures neuronales bout-en-bout pour la compréhension de la parole, *Proc.*

XXXIVe Journées d'Études sur la Parole – JEP 2022, 823-832, https://doi.org/10.21437/JEP.2022-87, page 108.

Pelloin, V., Dary, F., Hervé, N., Favre, B., Camelin, N., Laurent, A., & Besacier, L., (2022b), ASR-Generated Text for Language Model Pre-training Applied to Speech Tasks, *Interspeech*, 3453-3457, https://doi.org/10.21437/Interspeech.2022-352, page 139.

Références

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M., (2019), Optuna : A Next-Generation Hyperparameter Optimization Framework, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623-2631, https://doi.org/10.1145/3292500.3330701, page 43.

Alajrami, A., & Aletras, N., (2022), How does the pre-training objective affect what large language models learn about linguistic properties ?, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 131-147, https://doi.org/10.18653/v1/2022.acl-short.16, page 79.

Allen, J., (1995), *Natural language understanding* (2ème édition), Benjamin/Cummings Pub. Co, https://dl.acm.org/doi/10.5555/199291, page 78.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., Chen, J., Chen, J., Chen, Z., Chrzanowski, M., Coates, A., Diamos, G., Ding, K., Du, N., Elsen, E., et al. (2016), Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin, *In* M. F. Balcan & K. Q. Weinberger (Éd.), *Proceedings of The 33rd International Conference on Machine Learning* (p. 173-182, T. 48), PMLR, https://dl.acm.org/doi/10.5555/3045390.3045410, pages 50, 56.

Anastasopoulos, A., & Chiang, D., (2018), Tied Multitask Learning for Neural Speech Translation, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, 82-91, https://doi.org/10.18653/v1/N18-1008, page 108.

Ardila, R., Branson, M., Davis, K., Kohler, M., Meyer, J., Henretty, M., Morais, R., Saunders, L., Tyers, F., & Weber, G., (2020), Common Voice : A Massively-Multilingual Speech Corpus, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 4218-4222, https://aclanthology.org/2020.lrec-1.520, page 65.

Arora, S., Dalmia, S., Yan, B., Metze, F., Black, A. W., & Watanabe, S., (2022), Token-level Sequence Labeling for Spoken Language Understanding using Compositional End-to-End Models, *Findings of the Association for Computational*

Linguistics : EMNLP 2022, 5419-5429, https://aclanthology.org/2022.findings-emnlp.396, pages 118, 127.

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., & Auli, M., (2022), XLS-R : Self-supervised Cross-lingual Speech Representation Learning at Scale, *Interspeech*, 2278-2282, https://doi.org/10.21437/Interspeech.2022-143, pages 35, 55, 144.

Baevski, A., Schneider, S., & Auli, M., (2020a), vq-wav2vec : Self-Supervised Learning of Discrete Speech Representations, *International Conference on Learning Representations (ICLR)*, https://openreview.net/forum?id=rylwJxrYDS, page 55.

Baevski, A., Zhou, H., Mohamed, A., & Auli, M., (2020b), Wav2vec 2.0 : A Framework for Self-Supervised Learning of Speech Representations, *Proceedings of the 34th International Conference on Neural Information Processing Systems*, https://dl.acm.org/doi/10.5555/3495724.3496768, pages 35, 50, 55.

Bahdanau, D., Cho, K. H., & Bengio, Y., (2015), Neural machine translation by jointly learning to align and translate, 1-15, https://doi.org/10.48550/arXiv.1409.0473, pages 31, 35, 101, 110.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y., (2016), End-to-end attention-based large vocabulary speech recognition, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4945-4949, https://doi.org/10.1109/ICASSP.2016.7472618, pages 32, 54, 56.

Bahl, L. R., Jelinek, F., & Mercer, R. L., (1983), A Maximum Likelihood Approach to Continuous Speech Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-5 2*, 179-190, https://doi.org/10.1109/TPAMI.1983.4767370, pages 50, 70.

Baker, J., (1975), The DRAGON system–An overview, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *23 1*, 24-29, https://doi.org/10.1109/TASSP.1975.1162650, page 49.

Bastianelli, E., Vanzo, A., Swietojanski, P., & Rieser, V., (2020), SLURP : A Spoken Language Understanding Resource Package, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7252-7262, https://doi.org/10.18653/v1/2020.emnlp-main.588, page 127.

Béchet, F., (2011), Named Entity Recognition, *In Spoken Language Understanding* (p. 257-290), John Wiley & Sons, Ltd, https://doi.org/10.1002/9781119992691.ch10, page 72.

Béchet, F., & Raymond, C., (2018), Is ATIS Too Shallow to Go Deeper for Benchmarking Spoken Language Understanding Models ?, *Interspeech*, 3449-3453, https://doi.org/10.21437/Interspeech.2018-2256, page 86.

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S., (2021), On the Dangers of Stochastic Parrots : Can Language Models Be Too Big ?, *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610-623, https://doi.org/10.1145/3442188.3445922, pages 59, 79, 149.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C., (2003), A Neural Probabilistic Language Model, *J. Mach. Learn. Res.*, *3 null*, 1137-1155, https://dl.acm.org/doi/10.5555/944919.944966, pages 58, 59.

Bergstra, J., Yamins, D., & Cox, D. D., (2013), Making a Science of Model Search : Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, I–115–I-123, https://dl.acm.org/doi/10.5555/3042817.3042832, pages 43, 98.

Bird, S., & Loper, E., (2004), NLTK : The Natural Language Toolkit, *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, 214-217, https://aclanthology.org/P04-3031, page 60.

Boisen, S., Chow, Y.-L., Haas, A., Ingria, R., Roukos, S., & Stallard, D., (1989), The BBN Spoken Language System, *Speech and Natural Language : Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*, Récupérée juillet 27, 2023, à partir de https://aclanthology.org/H89-1012, page 77.

Bojanowski, P., Joulin, A., & Mikolov, T., (2016), Alternative structures for character-level RNNs, https://openreview.net/forum?id=wVqzL1ypocG0qV7mtLqm, page 102.

Bonneau-Maynard, H., & Lefèvre, F., (2005), A 2+1-level stochastic understanding model, *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 256-261, https://doi.org/10.1109/ASRU.2005.1566476, page 78.

Bonneau-Maynard, H., Rosset, S., Ayache, C., Kuhn, A., & Mostefa, D., (2005), Semantic annotation of the French media dialog corpus, *Interspeech*, 3457-3460, https://doi.org/10.21437/Interspeech.2005-312, page 87.

Boudahmane, K., Buschbeck, B., Cho, E., Crego, J. M., Freitag, M., Lavergne, T., Ney, H., Niehues, J., Peitz, S., Senellart, J., Sokolov, A., Waibel, A., Wandmacher, T., Wuebker, J., & Yvon, F., (2011), Advances on spoken language translation in the Quaero program, *Proceedings of the 8th International*

Workshop on Spoken Language Translation : Evaluation Campaign, 114-120, https://aclanthology.org/2011.iwslt-evaluation.15, page 65.

Bourlard, H., & Morgan, N., (1989), A Continuous Speech Recognition System Embedding MLP into HMM, *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, 186-193, https://dl.acm.org/doi/10.5555/2969830.2969853, pages 49, 54.

Bourlard, H., & Morgan, N., (1994), *Connectionist Speech Recognition*, Springer US, https://doi.org/10.1007/978-1-4615-3210-1, page 50.

Broder, A., (2002), A Taxonomy of Web Search, *SIGIR Forum*, *36 2*, 3-10, https://doi.org/10.1145/792550.792552, page 74.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., et al. (2020), Language Models are Few-Shot Learners, *In* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan & H. Lin (Éd.), *Advances in Neural Information Processing Systems* (p. 1877-1901, T. 33), Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf, pages 38, 42, 59, 77, 163.

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., & Gašić, M., (2018), MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 5016-5026, https://doi.org/10.18653/v1/D18-1547, page 86.

Cattan, O., Ghannay, S., Servan, C., & Rosset, S., (2022), Etude comparative de modèles Transformers en compréhension de la parole en Français, *XXXIVe Journées d'Études sur la Parole – JEP 2022*, 721-730, https://doi.org/10.21437/JEP.2022-76, page 79.

Caubrière, A., Tomashenko, N., Laurent, A., Morin, E., Camelin, N., & Estève, Y., (2019), Curriculum-based transfer learning for an effective end-to-end spoken language understanding and domain portability, *Interspeech*, 1198-1202, https://doi.org/10.21437/Interspeech.2019-1832, pages 80, 96, 97, 99, 103-105, 107, 108.

Chan, W., Jaitly, N., Le, Q., & Vinyals, O., (2016), Listen, attend and spell : A neural network for large vocabulary conversational speech recognition, *IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), 4960-4964, https://doi.org/10.1109/ICASSP.2016.7472621, pages 54, 56.

Chen, S. F., & Goodman, J., (1996), An Empirical Study of Smoothing Techniques for Language Modeling, *34th Annual Meeting of the Association for Computational Linguistics*, 310-318, https://doi.org/10.3115/981863.981904, page 58.

Cheng, J., Dong, L., & Lapata, M., (2016), Long Short-Term Memory-Networks for Machine Reading, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 551-561, https://doi.org/10.18653/v1/D16-1053, page 34.

Chiu, S.-H., & Chen, B., (2021), Innovative Bert-Based Reranking Language Models for Speech Recognition, *IEEE Spoken Language Technology Workshop (SLT)*, 266-271, https://doi.org/10.1109/SLT48900.2021.9383557, page 138.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y., (2014), Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724-1734, https://doi.org/10.3115/v1/D14-1179, pages 29, 54.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y., (2015), Attention-Based Models for Speech Recognition, *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, 577-585, https://dl.acm.org/doi/10.5555/2969239.2969304, page 54.

Chorowski, J., & Jaitly, N., (2017), Towards Better Decoding and Language Model Integration in Sequence to Sequence Models, *Interspeech*, 523-527, https://doi.org/10.21437/Interspeech.2017-343, page 111.

Church, A., (1936), An Unsolvable Problem of Elementary Number Theory, *American Journal of Mathematics*, *58 2*, 345-363, https://doi.org/10.2307/2371045, page 24.

Clarke, A. C., (1968), *2001 : A Space Odyssey*, ISBN : 0-453-00269-2, New American Library, page 15.

Cleeremans, A., Servan-Schreiber, D., & McClelland, J. L., (1989), Finite State Automata and Simple Recurrent Networks, *Neural Computation*, *1 3*, 372-381, https://doi.org/10.1162/neco.1989.1.3.372, page 58.

Dahl, G. E., Yu, D., Deng, L., & Acero, A., (2012), Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition, *IEEE Transac-*

tions on Audio, Speech, and Language Processing, *20 1*, 30-42, https://doi.org/10.1109/TASL.2011.2134090, pages 49, 54.

Davis, K. H., Biddulph, R., & Balashek, S., (1952), Automatic Recognition of Spoken Digits, *The Journal of the Acoustical Society of America*, *24 6*, 637-642, https://doi.org/10.1121/1.1906946, page 49.

Davis, S., & Mermelstein, P., (1980), Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, Conference Name : IEEE Transactions on Acoustics, Speech, and Signal Processing, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28 4*, 357-366, https://doi.org/10.1109/TASSP.1980.1163420, pages 50, 51.

De Mori, R., (1979), Recent advances in automatic speech recognition, *Signal Processing*, *1 2*, 95-123, https://doi.org/10.1016/0165-1684(79)90013-6, page 49.

De Mori, R., (1983), *Computer Models of Speech Using Fuzzy Algorithms*, Springer US, https://doi.org/10.1007/978-1-4613-3742-3, page 77.

De Mori, R., Bechet, F., Hakkani-Tur, D., McTear, M., Riccardi, G., & Tur, G., (2008), Spoken language understanding, *IEEE Signal Processing Magazine*, *25 3*, 50-58, https://doi.org/10.1109/MSP.2008.918413, page 70.

Dempster, A. P., Laird, N. M., & Rubin, D. B., (1977), Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)*, *39 1*, 1-38, http://www.jstor.org/stable/2984875, page 53.

Deng, L., Tur, G., He, X., & Hakkani-Tur, D., (2012), Use of kernel deep convex networks and end-to-end learning for spoken language understanding, *IEEE Spoken Language Technology Workshop (SLT)*, 210-215, https://doi.org/10.1109/SLT.2012.6424224, page 78.

Deng, L., & Yu, D., (2011), Deep convex net : a scalable architecture for speech pattern classification, *Interspeech*, 2285-2288, https://doi.org/10.21437/Interspeech.2011-607, page 78.

Desot, T., Portet, F., & Vacher, M., (2019), Towards End-to-End spoken intent recognition in smart home, *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 1-8, https://doi.org/10.1109/SPED.2019.8906584, page 30.

Devillers, L., Maynard, H., Rosset, S., Paroubek, P., McTait, K., Mostefa, D., Choukri, K., Charnay, L., Bousquet, C., Vigouroux, N., Béchet, F., Romary, L., Antoine, J.-Y., Villaneau, J., Vergnes, M., & Goulian, J.,

(2004), The French MEDIA/EVALDA Project : the Evaluation of the Understanding Capability of Spoken Language Dialogue Systems, *Language Resources and Evaluation Conference*, http://www.lrec-conf.org/proceedings/lrec2004/pdf/356.pdf, pages 15, 76.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K., (2019), BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171-4186, https://doi.org/10.18653/v1/N19-1423, pages 35, 38, 39, 55, 59, 78, 138.

Dong, L., Xu, S., & Xu, B., (2018), Speech-Transformer : A No-Recurrence Sequence-to-Sequence Model for Speech Recognition, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5884-5888, https://doi.org/10.1109/ICASSP.2018.8462506, pages 55, 56.

Dowding, J., Gawron, J. M., Appelt, D., Bear, J., Cherny, L., Moore, R., & Moran, D., (1993), Gemini : A Natural Language System for Spoken-Language Understanding, *Human Language Technology : Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*, https://aclanthology.org/H93-1008, page 77.

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B., (2022), Activation functions in deep learning : A comprehensive survey and benchmark, *Neurocomputing*, *503*, 92-108, https://doi.org/10.1016/j.neucom.2022.06.111, page 27.

Estève, Y., Bazillon, T., Antoine, J.-Y., Béchet, F., & Farinas, J., (2010), The EPAC Corpus : Manual and Automatic Annotations of Conversational Speech in French Broadcast News, *Language Resources and Evaluation Conference*, https://aclanthology.org/L10-1442/, page 65.

Evain, S., Nguyen, H., Le, H., Boito, M. Z., Mdhaffar, S., Alisamir, S., Tong, Z., Tomashenko, N., Dinarelli, M., Parcollet, T., et al., (2021), Task agnostic and task specific self-supervised learning from speech with *LeBenchmark*, *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, https://openreview.net/forum?id=TSvj5dmuSd, page 119.

Eyben, F., Wöllmer, M., Schuller, B., & Graves, A., (2009), From speech to letters - using a novel neural network architecture for grapheme based ASR, *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, 376-380, https://doi.org/10.1109/ASRU.2009.5373257, pages 50, 54.

Fan, A., Lewis, M., & Dauphin, Y., (2018), Hierarchical Neural Story Generation, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 889-898, https://doi.org/10.18653/v1/P18-1082, page 62.

Favre, B., (2023), LIS@DEFT'23 : les LLMs peuvent-ils répondre à des QCM ? (a) oui ; (b) non ; (c) je ne sais pas., *In* A. Bazoge, B. Daille, R. Dufour, Y. Labrak, E. Morin & M. Rouvier (Éd.), *18e Conférence en Recherche d'Information et Applications – 16e Rencontres Jeunes Chercheurs en RI – 30e Conférence sur le Traitement Automatique des Langues Naturelles – 25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues* (p. 46-56), ATALA, https://hal.science/hal-04131578, page 149.

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W., (2022), Language-agnostic BERT Sentence Embedding, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 878-891, https://doi.org/10.18653/v1/2022.acl-long.62, pages 59, 143.

Feng, J., Banglore, S., & Gilbert, M., (2009), Role of natural language understanding in voice local search, *Interspeech*, 1859-1862, https://doi.org/10.21437/Interspeech.2009-540, page 75.

Fillmore, C. J., (1976), Frame semantics and the nature of language, *Annals of the New York Academy of Sciences*, *280 1 Origins and E*, 20-32, https://doi.org/10.1111/j.1749-6632.1976.tb25467.x, page 71.

Fisher, R. A., (1936), The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, *7 2*, 179-188, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x, page 25.

Gage, P., (1994), A New Algorithm for Data Compression, *C Users J.*, *12 2*, 23-38, https://dl.acm.org/doi/10.5555/177910.177914, page 60.

Galibert, O., Leixa, J., Adda, G., Choukri, K., & Gravier, G., (2014), The ETAPE speech processing evaluation, *Language Resources and Evaluation Conference*, 3995-3999, https://aclanthology.org/L14-1022/, page 65.

Galliano, S., Geoffrois, E., Mostefa, D., Choukri, K., Bonastre, J.-F., & Gravier, G., (2005), The ESTER phase II evaluation campaign for the rich transcription of French broadcast news, *Interspeech*, 1149-1152, https://doi.org/10.21437/Interspeech.2005-441, page 65.

Galliano, S., Gravier, G., & Chaubard, L., (2009), The ester 2 evaluation campaign for the rich transcription of French radio broadcasts, *Interspeech*, 2583-2586, https://doi.org/10.21437/Interspeech.2009-680, page 65.

Gauvain, J.-L., Lamel, L., & Adda, G., (2002), The LIMSI Broadcast News transcription system, *Speech Communication*, *371*, 89-108, https://doi.org/10.1016/S0167-6393(01)00061-9, pages 49, 53, 80.

Gauvain, J.-L., & Lee, C.-H., (1994), Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains, *IEEE Transactions on Speech and Audio Processing*, *22*, 291-298, https://doi.org/10.1109/89.279278, page 49.

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., & Ritter, M., (2017), Audio Set : An ontology and human-labeled dataset for audio events, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 776-780, https://doi.org/10.1109/ICASSP.2017.7952261, page 143.

Ghannay, S., Caubrière, A., Estève, Y., Camelin, N., Simonnet, E., Laurent, A., & Morin, E., (2018), End-To-End Named Entity And Semantic Concept Extraction From Speech, *IEEE Spoken Language Technology Workshop (SLT)*, 692-699, https://doi.org/10.1109/SLT.2018.8639513, pages 80, 81, 83, 99.

Ghannay, S., Caubrière, A., Mdhaffar, S., Laperrière, G., Jabaian, B., & Estève, Y., (2021), Where Are We in Semantic Concept Extraction for Spoken Language Understanding ?, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12997 LNAI*, 202-213, https://doi.org/10.1007/978-3-030-87802-3_19, pages 79, 80, 99, 107, 108, 114, 118, 119, 125, 151.

Ghannay, S., Servan, C., & Rosset, S., (2020), Neural Networks approaches focused on French Spoken Language Understanding : application to the MEDIA Evaluation Task, *Proceedings of the 28th International Conference on Computational Linguistics*, 2722-2727, https://doi.org/10.18653/v1/2020.coling-main.245, pages 79, 140, 141.

Giraudel, A., Carré, M., Mapelli, V., Kahn, J., Galibert, O., & Quintard, L., (2012), The REPERE Corpus : a multimodal corpus for person recognition, *Language Resources and Evaluation Conference*, 1102-1107, https://aclanthology.org/L12-1410/, page 65.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J., (2006), Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent

Neural Networks, *Proceedings of the 23rd International Conference on Machine Learning*, 369-376, https://doi.org/10.1145/1143844.1143891, pages 30, 31, 40, 50, 54.

Graves, A., & Jaitly, N., (2014), Towards End-To-End Speech Recognition with Recurrent Neural Networks, *In* E. P. Xing & T. Jebara (Éd.), *Proceedings of the 31st International Conference on Machine Learning* (p. 1764-1772, T. 32), PMLR, https://proceedings.mlr.press/v32/graves14.html, pages 54, 102.

Graves, A., Mohamed, A.-r., & Hinton, G., (2013), Speech recognition with deep recurrent neural networks, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6645-6649, https://doi.org/10.1109/ICASSP.2013.6638947, page 50.

Gravier, G., Bonastre, J.-F., Geoffrois, E., Galliano, S., McTait, K., & Choukri, K., (2004), The ESTER Evaluation Campaign for the Rich Transcription of French Broadcast News, *Language Resources and Evaluation Conference*, https://aclanthology.org/L04-1430/, page 65.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., & Bengio, Y., (2015), On Using Monolingual Corpora in Neural Machine Translation, https://doi.org/10.48550/arXiv.1503.03535, pages 61, 102.

Hahn, S., Dinarelli, M., Raymond, C., Lefèvre, F., Lehnen, P., De Mori, R., Moschitti, A., Ney, H., & Riccardi, G., (2011), Comparing Stochastic Approaches to Spoken Language Understanding in Multiple Languages, *Transactions on Audio, Speech, and Language Processing*, *19 6*, 1569-1583, https://doi.org/10.1109/TASL.2010.2093520, page 99.

Hatmi, M., Jacquin, C., Morin, E., & Meignier, S., (2013), Incorporating named entity recognition into the speech transcription process, *Interspeech*, 3732-3736, https://doi.org/10.21437/Interspeech.2013-588, page 80.

He, M., & Garner, P. N., (2023), Can ChatGPT Detect Intent ? Evaluating Large Language Models for Spoken Language Understanding, https://doi.org/10.48550/arXiv.2305.13512, page 42.

Hemphill, C. T., Godfrey, J. J., & Doddington, G. R., (1990), The ATIS Spoken Language Systems Pilot Corpus, *Speech and Natural Language : Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, https://aclanthology.org/H90-1021, page 84.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B., (2012), Deep Neural Networks for Acoustic Modeling in Speech Recognition : The Shared Views of Four Research Groups, *IEEE Signal Processing Magazine*, *29 6*, 82-97, https://doi.org/10.1109/MSP.2012.2205597, page 54.

Hochreiter, S., & Schmidhuber, J., (1997), Long Short-Term Memory, *Neural Computation*, *9 8*, 1735-1780, https://doi.org/10.1162/neco.1997.9.8.1735, page 28.

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y., (2020), The Curious Case of Neural Text Degeneration, *International Conference on Learning Representations (ICLR)*, https://openreview.net/forum?id=rygGQyrFvH, page 62.

Hori, C., Furui, S., Malkin, R., Yu, H., & Waibel, A., (2002), Automatic Summarization of English Broadcast News Speech, *Proceedings of the Second International Conference on Human Language Technology Research*, 241-246, page 74.

Hori, T., Cho, J., & Watanabe, S., (2019), End-to-end Speech Recognition with Word-Based Rnn Language Models, *IEEE Spoken Language Technology Workshop (SLT)*, 389-396, https://doi.org/10.1109/SLT.2018.8639693, pages 56, 102.

Hori, T., & Nakamura, A., (2006), An Extremely Large Vocabulary Approach to Named Entity Extraction from Speech, *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, *1*, I-I, https://doi.org/10.1109/ICASSP.2006.1660185, page 80.

Hori, T., Watanabe, S., & Hershey, J. R., (2017), Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition, *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 287-293, https://doi.org/10.1109/ASRU.2017.8268948, pages 56, 102.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W., (2021), LoRA : Low-Rank Adaptation of Large Language Models, https://doi.org/10.48550/arXiv.2106.09685, page 154.

Jannet, M. A. B., Galibert, O., Adda-Decker, M., & Rosset, S., (2017), Investigating the Effect of ASR Tuning on Named Entity Recognition, *Interspeech*, 2486-2490, https://doi.org/10.21437/Interspeech.2017-1482, page 79.

Jawahar, G., Sagot, B., & Seddah, D., (2019), What Does BERT Learn about the Structure of Language ?, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3651-3657, https://doi.org/10.18653/v1/P19-1356, page 78.

Jelinek, F., (1976), Continuous speech recognition by statistical methods, *Proceedings of the IEEE*, *64 4*, 532-556, https://doi.org/10.1109/PROC.1976.10159, pages 49, 56.

Jiang, D., Lei, X., Li, W., Luo, N., Hu, Y., Zou, W., & Li, X., (2019), Improving Transformer-based Speech Recognition Using Unsupervised Pre-training, https://doi.org/10.48550/arXiv.1910.09932, page 55.

Kahn, J., Rivière, M., Zheng, W., Kharitonov, E., Xu, Q., Mazaré, P., Karadayi, J., Liptchinsky, V., Collobert, R., Fuegen, C., Likhomanenko, T., Synnaeve, G., Joulin, A., Mohamed, A., & Dupoux, E., (2020), Libri-Light : A Benchmark for ASR with Limited or No Supervision, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7669-7673, https://doi.org/10.1109/ICASSP40776.2020.9052942, pages 37, 64, 143.

Kelley, J. F., (1984), An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications, *ACM Trans. Inf. Syst.*, *2 1*, 26-41, https://doi.org/10.1145/357417.357420, page 87.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M., (2022), Transformers in Vision : A Survey, *ACM Comput. Surv.*, *54 10s*, https://doi.org/10.1145/3505244, page 35.

Khurana, S., Laurent, A., & Glass, J., (2022), SAMU-XLSR : Semantically-Aligned Multimodal Utterance-Level Cross-Lingual Speech Representation, *IEEE Journal of Selected Topics in Signal Processing*, *16 6*, 1493-1504, https://doi.org/10.1109/JSTSP.2022.3192714, page 144.

Kingma, D. P., & Ba, J., (2015), Adam : A Method for Stochastic Optimization, *International Conference on Learning Representations (ICLR)*, https://doi.org/10.48550/arXiv.1412.6980, page 98.

Korpusik, M., Liu, Z., & Glass, J., (2019), A Comparison of Deep Learning Methods for Language Understanding, *Interspeech*, 849-853, https://doi.org/10.21437/Interspeech.2019-1262, page 79.

Kudo, T., & Richardson, J., (2018), SentencePiece : A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, 66-71, https://doi.org/10.18653/v1/D18-2012, page 60.

Labrak, Y., Bazoge, A., Daille, B., Dufour, R., Morin, E., & Rouvier, M., (2023), Tâches et systèmes de détection automatique des réponses correctes dans des QCMs liés au domaine médical : Présentation de la campagne DEFT 2023, *In*

A. Bazoge, B. Daille, R. Dufour, Y. Labrak, E. Morin & M. Rouvier (Éd.), *18e Conférence en Recherche d'Information et Applications – 16e Rencontres Jeunes Chercheurs en RI – 30e Conférence sur le Traitement Automatique des Langues Naturelles – 25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues* (p. 57-67), ATALA, https://hal.science/hal-04131586, page 149.

Lafferty, J. D., McCallum, A., & Pereira, F. C. N., (2001), Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data, *Proceedings of the Eighteenth International Conference on Machine Learning*, 282-289, https://dl.acm.org/doi/10.5555/645530.655813, page 78.

Lampert, C. H., Nickisch, H., & Harmeling, S., (2009), Learning to detect unseen object classes by between-class attribute transfer, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 951-958, https://doi.org/10.1109/CVPR.2009.5206594, pages 42, 43.

Landauer, T., Kamm, C., & Singhal, S., (1987), Learning a minimally structured back propagation network to recognize speech, ISSN : 1069-7977, *Proceedings of the Ninth Annual Conf. of the Cognitive Science Society*, 531-536, page 50.

Lawrence, S., Giles, C., & Fong, S., (1996), Can recurrent neural networks learn natural language grammars ?, *Proceedings of International Conference on Neural Networks (ICNN'96)*, *4*, 1853-1858 vol.4, https://doi.org/10.1109/ICNN.1996.549183, page 58.

Le, H., Vial, L., Frej, J., Segonne, V., Coavoux, M., Lecouteux, B., Allauzen, A., Crabbé, B., Besacier, L., & Schwab, D., (2020), FlauBERT : Unsupervised Language Model Pre-training for French, *Proceedings of The 12th Language Resources and Evaluation Conference*, 2479-2490, https://www.aclweb.org/anthology/2020.lrec-1.302, pages 59, 138.

Lee, K.-F., (1988), On large-vocabulary speaker-independent continuous speech recognition, Word Recognition in Large Vocabularies, *Speech Communication*, *7 4*, 375-379, https://doi.org/10.1016/0167-6393(88)90053-2, pages 49, 53.

Lefèvre, F., (2007), Dynamic Bayesian Networks and Discriminative Classifiers for Multi-Stage Semantic Interpretation, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *4*, IV-13-IV-16, https://doi.org/10.1109/ICASSP.2007.367151, page 78.

Lefèvre, F., Mostefa, D., Besacier, L., Estève, Y., Quignard, M., Camelin, N., Favre, B., Jabaian, B., & Rojas-Barahona, L., (2012), Robustesse et

portabilités multilingue et multi-domaines des systèmes de compréhension de la parole : les corpus du projet PortMedia (Robustness and portability of spoken language understanding systems among languages and domains : the PORTMEDIA project) [in French], *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 1 : JEP*, 779-786, https://aclanthology.org/F12-1098, pages 15, 88.

Li, X., Metze, F., Mortensen, D. R., Black, A. W., & Watanabe, S., (2022), ASR2K : Speech Recognition for Around 2000 Languages without Audio, *Interspeech*, 4885-4889, https://doi.org/10.21437/Interspeech.2022-10712, page 50.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V., (2019), RoBERTa : A Robustly Optimized BERT Pretraining Approach, https://doi.org/10.48550/arXiv.1907.11692, pages 59, 139.

Loshchilov, I., & Hutter, F., (2019), Decoupled Weight Decay Regularization, *International Conference on Learning Representations (ICLR)*, https://openreview.net/forum?id=Bkg6RiCqY7, page 41.

Lu, L., Zhang, X., Cho, K., & Renals, S., (2015), A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition, *Interspeech*, 3249-3253, https://doi.org/10.21437/Interspeech.2015-654, pages 50, 54.

Lucas, G., (1977), *Star Wars*, Film, produit par Kurtz, G. (Lucasfilm Ltd.), 20th Century-Fox, page 15.

Luhn, H. P., (1958), The Automatic Creation of Literature Abstracts, *IBM Journal of Research and Development*, *2 2*, 159-165, https://doi.org/10.1147/rd.22.0159, page 74.

Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É., Seddah, D., & Sagot, B., (2020), CamemBERT : a Tasty French Language Model, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7203-7219, https://doi.org/10.18653/v1/2020.acl-main.645, pages 59, 119, 138.

Maskey, S., & Hirschberg, J., (2006), Summarizing Speech Without Text Using Hidden Markov Models, *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume : Short Papers*, 89-92, https://aclanthology.org/N06-2023, page 74.

McCulloch, W. S., & Pitts, W., (1943), A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics*, *54*, 115-133, https://doi.org/10.1007/BF02478259, page 24.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., & Zweig, G., (2015), Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *233*, 530-539, https://doi.org/10.1109/TASLP.2014.2383614, pages 72, 78.

Mesnil, G., He, X., Deng, L., & Bengio, Y., (2013), Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding, *Interspeech*, 3771-3775, https://doi.org/10.21437/Interspeech.2013-596, page 78.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S., (2010), Recurrent neural network based language model, *Interspeech*, 1045-1048, https://doi.org/10.21437/Interspeech.2010-343, page 59.

Mitchell, M., & Krakauer, D. C., (2023), The debate over understanding in AI's large language models, Publisher : Proceedings of the National Academy of Sciences, *Proceedings of the National Academy of Sciences*, *12013*, e2215907120, https://doi.org/10.1073/pnas.2215907120, page 79.

Nolan, J., (2011), *Person of Interest*, Série télévisuelle, produite par Nolan, J., Plageman, G., Abrams, J. J., Burk, B., Thé, D., Fisher, C., CBS Broadcasting Inc., page 15.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., & Auli, M., (2019), fairseq : A Fast, Extensible Toolkit for Sequence Modeling, *Proceedings of NAACL-HLT 2019 : Demonstrations*, https://doi.org/10.18653/v1/N19-4009, page 98.

Palatucci, M., Pomerleau, D., Hinton, G., & Mitchell, T. M., (2009), Zero-Shot Learning with Semantic Output Codes, *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, 1410-1418, https://dl.acm.org/doi/10.5555/2984093.2984252, page 42.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S., (2015), Librispeech : An ASR corpus based on public domain audio books, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5206-5210, https://doi.org/10.1109/ICASSP.2015.7178964, page 64.

Parikh, A., Täckström, O., Das, D., & Uszkoreit, J., (2016), A Decomposable Attention Model for Natural Language Inference, *Proceedings of the 2016 Conference*

on Empirical Methods in Natural Language Processing, 2249-2255, https://doi.org/10.18653/v1/D16-1244, page 33.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V., (2019), SpecAugment : A Simple Data Augmentation Method for Automatic Speech Recognition, *Interspeech*, 2613-2617, https://doi.org/10.21437/Interspeech.2019-2680, page 56.

Paul, D. B., & Baker, J. M., (1992), The Design for the Wall Street Journal-based CSR Corpus, *Speech and Natural Language : Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, https://aclanthology.org/H92-1073, page 64.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L., (2018), Deep Contextualized Word Representations, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, 2227-2237, https://doi.org/10.18653/v1/N18-1202, page 59.

Pratap, V., Tjandra, A., Shi, B., Tomasello, P., Babu, A., Kundu, S., Elkahky, A., Ni, Z., Vyas, A., Fazel-Zarandi, M., Baevski, A., Adi, Y., Zhang, X., Hsu, W.-N., Conneau, A., & Auli, M., (2023), Scaling Speech Technology to 1,000+ Languages, https://doi.org/10.48550/arXiv.2305.13516, pages 50, 55.

Price, P., Ostendorf, M., Shattuck-Hufnagel, S., & Fong, C., (1991), The Use of Prosody in Syntactic Disambiguation, *Speech and Natural Language : Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, https://aclanthology.org/H91-1073, pages 80, 108.

Rabiner, L., (1989), A tutorial on hidden Markov models and selected applications in speech recognition, Conference Name : Proceedings of the IEEE, *Proceedings of the IEEE*, *77 2*, 257-286, https://doi.org/10.1109/5.18626, page 52.

Radford, A., Kim, J. W., Xu, T., Brockman, G., Mcleavey, C., & Sutskever, I., (2023), Robust Speech Recognition via Large-Scale Weak Supervision, *In* A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato & J. Scarlett (Éd.), *Proceedings of the 40th International Conference on Machine Learning* (p. 28492-28518, T. 202), PMLR, https://proceedings.mlr.press/v202/radford23a.html, pages 50, 55, 64, 80, 143.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J., (2020), Exploring the Limits of Transfer Learning with a

Unified Text-to-Text Transformer, *J. Mach. Learn. Res.*, *21 1*, https://dl.acm.org/doi/10.5555/3455716.3455856, pages 35, 59.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P., (2016), SQuAD : 100,000+ Questions for Machine Comprehension of Text, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383-2392, https://doi.org/10.18653/v1/D16-1264, page 75.

Rastogi, A., Zang, X., Sunkara, S., Gupta, R., & Khaitan, P., (2020), Towards Scalable Multi-Domain Conversational Agents : The Schema-Guided Dialogue Dataset, *Proceedings of the AAAI Conference on Artificial Intelligence*, *34 05*, 8689-8696, https://doi.org/10.1609/aaai.v34i05.6394, pages 72, 74.

Raymond, C., Béchet, F., De Mori, R., & Damnati, G., (2006), On the use of finite state transducers for semantic interpretation, *Speech Communication*, *48 3-4*, 288-304, https://doi.org/10.1016/j.specom.2005.06.012, page 99.

Rosenblatt, F., (1958), The perceptron : A probabilistic model for information storage and organization in the brain., *Psychological Review*, *65 6*, 386-408, https://doi.org/10.1037/H0042519, page 25.

Rosenblatt, F., (1961), *Principles of neurodynamics : Perceptrons and the theory of brain mechanisms* (T. 55), https://apps.dtic.mil/sti/citations/AD0256582, page 26.

Rosset, S., Galibert, O., & Lamel, L., (2011), Spoken Question Answering, *In Spoken Language Understanding* (p. 147-170), John Wiley & Sons, Ltd, https://doi.org/10.1002/9781119992691.ch6, page 75.

Rousseau, A., Deléglise, P., & Estève, Y., (2012), TED-LIUM : an Automatic Speech Recognition dedicated corpus, *Language Resources and Evaluation Conference*, http://lrec-conf.org/proceedings/lrec2012/summaries/698.html, page 65.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J., (1986), Learning representations by back-propagating errors, *Nature*, *323 6088*, 533-536, https://doi.org/10.1038/323533a0, pages 27, 40.

Sarikaya, R., Hinton, G. E., & Ramabhadran, B., (2011), Deep belief nets for natural language call-routing, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5680-5683, https://doi.org/10.1109/ICASSP.2011.5947649, page 78.

Scao, T. L., Wang, T., Hesslow, D., Saulnier, L., Bekman, S., Bari, M. S., Biderman, S., Elsahar, H., Phang, J., Press, O., Raffel, C., Sanh, V., Shen, S., Sutawika, L., Tae, J., Yong, Z. X., Launay, J., & Beltagy, I.,

(2022), What Language Model to Train if You Have One Million GPU Hours ?, *Challenges & Perspectives in Creating Large Language Models*, https://openreview.net/forum?id=rI7BL3fHIZq, page 35.

Schneider, S., Baevski, A., Collobert, R., & Auli, M., (2019), wav2vec : Unsupervised Pre-Training for Speech Recognition, *Interspeech*, 3465-3469, https://doi.org/10.21437/Interspeech.2019-1873, page 55.

Schwartz, R., Miller, S., Stallard, D., & Makhoul, J., (1996), Language understanding using hidden understanding models, *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, *2*, 997-1000 vol.2, https://doi.org/10.1109/ICSLP.1996.607771, page 78.

Schwenk, H., (2007), Continuous space language models, *Computer Speech & Language*, *21 3*, 492-518, https://doi.org/10.1016/j.csl.2006.09.003, page 58.

Seide, F., Li, G., Chen, X., & Yu, D., (2011), Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription, *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 24-29, https://doi.org/10.1109/ASRU.2011.6163899, pages 49, 54.

Seneff, S., (1992), TINA : A Natural Language System for Spoken Language Applications, *Computational Linguistics*, *18 1*, 61-86, Récupérée juillet 27, 2023, à partir de https://aclanthology.org/J92-1004, page 77.

Sennrich, R., Haddow, B., & Birch, A., (2016), Neural Machine Translation of Rare Words with Subword Units, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 1715-1725, https://doi.org/10.18653/v1/P16-1162, page 60.

Serdyuk, D., Wang, Y., Fuegen, C., Kumar, A., Liu, B., & Bengio, Y., (2018), Towards End-to-end Spoken Language Understanding, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *2018-April*, 5754-5758, https://doi.org/10.1109/ICASSP.2018.8461785, pages 80, 108.

Servan, C., Raymond, C., Béchet, F., & Nocéra, P., (2006), Conceptual decoding from word lattices : application to the spoken dialogue corpus MEDIA, *Interspeech*, https://doi.org/10.21437/Interspeech.2006-451, page 99.

Shannon, C. E., (1948), A mathematical theory of communication, *The Bell System Technical Journal*, *27 3*, 379-423, https://doi.org/10.1002/j.1538-7305.1948.tb01338.x, page 56.

Shannon, C. E., (1951), Prediction and entropy of printed English, *The Bell System Technical Journal*, *30 1*, 50-64, https://doi.org/10.1002/j.1538-7305.1951.tb01366.x, pages 56, 57.

Shon, S., Pasad, A., Wu, F., Brusco, P., Artzi, Y., Livescu, K., & Han, K. J., (2022), SLUE : New Benchmark Tasks For Spoken Language Understanding Evaluation on Natural Speech, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7927-7931, https://doi.org/10.1109/ICASSP43922.2022.9746137, pages 108, 127.

Shriberg, E., (2005), Spontaneous speech : how people really talk and why engineers should care, *Interspeech*, 1781-1784, https://doi.org/10.21437/Interspeech.2005-3, page 139.

Si, S., Ma, W., Gao, H., Wu, Y., Lin, T.-E., Dai, Y., Li, H., Yan, R., Huang, F., & Li, Y., (2023), SpokenWOZ : A Large-Scale Speech-Text Benchmark for Spoken Task-Oriented Dialogue Agents, https://doi.org/10.48550/arXiv.2305.13040, pages 86, 149, 163.

Simonnet, E., Camelin, N., Deléglise, P., & Estève, Y., (2015), Exploring the use of Attention-Based Recurrent Neural Networks For Spoken Language Understanding, *Machine Learning for Spoken Language Understanding and Interaction NIPS 2015 workshop (SLUNIPS 2015)*, https://hal.science/hal-01433202, page 78.

Simonnet, E., Ghannay, S., Camelin, N., & Estève, Y., (2018), Simulating ASR errors for training SLU systems, *Language Resources and Evaluation Conference*, https://aclanthology.org/L18-1499, page 78.

Simonnet, E., Ghannay, S., Camelin, N., Estève, Y., & Mori, R. D., (2017), ASR Error Management for Improving Spoken Language Understanding, *Interspeech*, 3329-3333, https://doi.org/10.21437/Interspeech.2017-1178, pages 78, 99, 108.

Soltau, H., Shafran, I., Wang, M., Rastogi, A., Zhao, J., Jia, Y., Han, W., Cao, Y., & Miranda, A., (2022), Speech Aware Dialog System Technology Challenge (DSTC11), https://doi.org/10.48550/arXiv.2212.08704, page 86.

Song, Y.-I., Wang, Y.-Y., Ju, Y.-C., Seltzer, M., Tashev, I., & Acero, A., (2009), Voice search of structured media data, *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 3941-3944, https://doi.org/10.1109/ICASSP.2009.4960490, page 75.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R., (2014), Dropout : A Simple Way to Prevent Neural Networks from Overfitting,

Journal of Machine Learning Research, *15 56*, 1929-1958, http://jmlr.org/papers/v15/srivastava14a.html, pages 41, 42.

Stevens, S. S., Volkmann, J., & Newman, E. B., (1937), A Scale for the Measurement of the Psychological Magnitude Pitch, *The Journal of the Acoustical Society of America*, *8 3*, 185-190, https://doi.org/10.1121/1.1915893, page 51.

Titze, I. R., (2000), *Principles of Voice Production*, ISBN : 0-87414-122-2 , page 50.

Tomashenko, N., Caubrière, A., & Estève, Y., (2019), Investigating Adaptation and Transfer Learning for End-to-End Spoken Language Understanding from Speech, *Interspeech*, 824-828, https://doi.org/10.21437/Interspeech.2019-2158, page 108.

Tomashenko, N., Raymond, C., Caubrière, A., Mori, R. D., & Estève, Y., (2020), Dialogue History Integration into End-to-End Signal-to-Concept Spoken Language Understanding Systems, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8509-8513, https://doi.org/10.1109/ICASSP40776.2020.9053247, pages 148, 163.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G., (2023a), LLaMA : Open and Efficient Foundation Language Models, https://doi.org/10.48550/arXiv.2302.13971, pages 35, 152.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., et al. (2023b), Llama 2 : Open Foundation and Fine-Tuned Chat Models, https://doi.org/10.48550/arXiv.2307.09288, pages 77, 150, 152, 163.

Tran, T., Toshniwal, S., Bansal, M., Gimpel, K., Livescu, K., & Ostendorf, M., (2018), Parsing Speech : a Neural Approach to Integrating Lexical and Acoustic-Prosodic Information, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, 69-81, https://doi.org/10.18653/v1/N18-1007, page 108.

Tur, G., & De Mori, R. (Éd.), (2011), *Spoken Language Understanding : Systems for Extracting Semantic Information from Speech* (1re éd.), John Wiley & Sons, Ltd, https://doi.org/10.1002/9781119992691, pages 70, 71.

Tur, G., & Deng, L., (2011), Intent Determination and Spoken Utterance Classification, *In Spoken Language Understanding* (p. 93-118), John Wiley & Sons, Ltd, https://doi.org/10.1002/9781119992691.ch4, page 74.

Tur, G., Hakkani-Tür, D., & Heck, L., (2010), What is left to be understood in ATIS ?, *2010 IEEE Spoken Language Technology Workshop*, 19-24, https://doi.org/10.1109/SLT.2010.5700816, pages 85, 86.

Turing, A. M., (1937), On Computable Numbers, with an Application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, *s2-42 1*, 230-265, https://doi.org/10.1112/plms/s2-42.1.230, page 24.

Upadhyay, S., Faruqui, M., Tür, G., Dilek, H.-T., & Heck, L., (2018), (Almost) Zero-Shot Cross-Lingual Spoken Language Understanding, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6034-6038, https://doi.org/10.1109/ICASSP.2018.8461905, page 85.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I., (2017), Attention is All You Need, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000-6010, https://dl.acm.org/doi/10.5555/3295222.3295349, pages 33-36, 78.

Vukotic, V., Raymond, C., & Gravier, G., (2015), Is it time to switch to word embedding and recurrent neural networks for spoken language understanding ?, *Interspeech*, 130-134, https://doi.org/10.21437/Interspeech.2015-41, page 78.

Wang, C., Riviere, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., & Dupoux, E., (2021a), VoxPopuli : A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 993-1003, https://doi.org/10.18653/v1/2021.acl-long.80, pages 65, 143.

Wang, C., Wu, A., Pino, J., Baevski, A., Auli, M., & Conneau, A., (2021b), Large-Scale Self- and Semi-Supervised Learning for Speech Translation, *Interspeech*, 2242-2246, https://doi.org/10.21437/Interspeech.2021-1912, page 38.

Wang, P., Wei, L., Cao, Y., Xie, J., & Nie, Z., (2020), Large-Scale Unsupervised Pre-Training for End-to-End Spoken Language Understanding, *IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), 7999-8003, https://doi.org/10.1109/ICASSP40776.2020.9053163, page 80.

Wang, Y.-Y., Deng, L., & Acero, A., (2011a), Semantic Frame-Based Spoken Language Understanding, *In Spoken Language Understanding* (p. 41-91), John Wiley & Sons, Ltd, https://doi.org/10.1002/9781119992691.ch3, page 71.

Wang, Y.-Y., Yu, D., Ju, Y.-C., & Acero, A., (2011b), Voice Search, *In Spoken Language Understanding* (p. 119-146), John Wiley & Sons, Ltd, https://doi.org/10.1002/9781119992691.ch5, page 74.

Wang, Y. [Yiming], Chen, T., Xu, H., Ding, S., Lv, H., Shao, Y., Peng, N., Xie, L., Watanabe, S., & Khudanpur, S., (2019), Espresso : A Fast End-to-End Neural Speech Recognition Toolkit, *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 136-143, https://doi.org/10.1109/asru46091.2019.9003968, page 98.

Wang, Y. [Yufan], Mei, J., Zou, B., Fan, R., He, T., & Aw, A. T., (2023), Making Pre-trained Language Models Better Learn Few-Shot Spoken Language Understanding in More Practical Scenarios, *Findings of the Association for Computational Linguistics : ACL 2023*, 13508-13523, https://aclanthology.org/2023.findings-acl.853, pages 42, 43.

Watanabe, S., Hori, T., Kim, S., Hershey, J. R., & Hayashi, T., (2017), Hybrid CTC/Attention Architecture for End-to-End Speech Recognition, *IEEE Journal of Selected Topics in Signal Processing*, *11 8*, 1240-1253, https://doi.org/10.1109/JSTSP.2017.2763455, page 30.

Watrous, R. L., & Shastri, L., (1987), Learning phonetic features using connectionist networks, *The Journal of the Acoustical Society of America*, *81 S1*, S93-S94, https://doi.org/10.1121/1.2024481, page 50.

Weizenbaum, J., (1966), ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine, *Commun. ACM*, *9 1*, 36-45, https://doi.org/10.1145/365153.365168, pages 76, 77.

Woods, W. A., (1975), Whats's in a link : Foundations for Semantic Networks, *In* D. G. Bobrow & A. Collins (Éd.), *Representation and Understanding* (p. 35-82), Morgan Kaufmann, https://doi.org/10.1016/B978-0-12-108550-6.50007-0, page 70.

Xu, W. [Wei], & Rudnicky, A., (2000), Can artificial neural networks learn language models ?, *Proceedings of 6th International Conference on Spoken Language Proces-*

sing (ICSLP 2000), vol. 1, 202-205, https://doi.org/10.21437/ICSLP.2000-50, page 58.

Xu, W. [Weijia], Haider, B., & Mansour, S., (2020), End-to-End Slot Alignment and Recognition for Cross-Lingual NLU, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5052-5063, https://doi.org/10.18653/v1/2020.emnlp-main.410, page 85.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E., (2016), Hierarchical Attention Networks for Document Classification, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 1480-1489, https://doi.org/10.18653/v1/N16-1174, page 34.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., & Shi, Y., (2014), Spoken language understanding using long short-term memory neural networks, *IEEE Spoken Language Technology Workshop (SLT)*, 189-194, https://doi.org/10.1109/SLT.2014.7078572, page 78.

Yao, K., Zweig, G., Hwang, M.-Y., Shi, Y., & Yu, D., (2013), Recurrent neural networks for language understanding, *Interspeech*, 2524-2528, https://doi.org/10.21437/Interspeech.2013-569, page 78.

y Arcas, B. A., (2022), Do Large Language Models Understand Us ?, *Daedalus*, *151 2*, 183-197, https://doi.org/10.1162/daed_a_01909, page 79.

Yu, D., Ju, Y.-C., Wang, Y.-Y., Zweig, G., & Acero, A., (2007), Automated directory assistance system - from theory to practice, *Interspeech*, 2709-2712, https://doi.org/10.21437/Interspeech.2007-65, page 75.

Zhou, G., & Su, J., (2002), Named Entity Recognition using an HMM-based Chunk Tagger, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 473-480, https://doi.org/10.3115/1073083.1073163, page 78.

Zue, V., & Seneff, S., (2008), Spoken Dialogue Systems, *In* J. Benesty, M. M. Sondhi & Y. A. Huang (Éd.), *Springer Handbook of Speech Processing* (p. 705-722), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-540-49127-9_35, page 76.

Liste des tableaux

2.1 Récapitulatif des corpus de transcriptions décrits. ST : données de parole sans la transcription. 66

3.1 Exemple de représentation segments sémantique (concepts et valeurs), alignés avec la transcription, pour une phrase utilisateur «*quels sont les hôtels proches du stade de france avec une piscine*». 73
3.2 Exemple d'annotation en concepts avec le formalisme BIO et une *tokenization* en mots 82
3.3 Exemple d'annotation d'un segment utilisateur du corpus ATIS en concepts et avec une intention *flight*. 86
3.4 Tailles réellement utilisées des corpus MEDIA, PortMEDIA-Fr et PortMEDIA-It pour les jeux Train, Dev et Test, pour les tours de parole de l'utilisateur. 89

4.1 Étapes d'apprentissage des modèles de compréhension de la parole, en fonction de la configuration (mots, étoile, valeurs), pour la tâche cible X = MEDIA ou X = PortMEDIA-Fr. 97
4.2 Exemple de sortie de chaque format (ASR, SLU_{Mots}, SLU^{*} ou $\text{SLU}^{*}_{\text{Norm}}$). Les espaces sont représentés par le caractère «␣», et le caractère «·» permet de visualiser la séparation des mots en caractères. 98
4.3 Exemples de règles de normalisation des valeurs. 99
4.4 Résultats en CER et CVER sur le corpus MEDIA (Dev et Test). Les résultats sont comparés avec ceux états de l'art *end-to-end* de Caubrière et al. (2019). 104
4.5 Résultats en CER et CVER sur le corpus PortMEDIA-Fr (Dev et Test). Les résultats sont comparés avec ceux états de l'art *end-to-end* de Caubrière et al. (2019). 105
4.6 Résultats en WER sur les corpus MEDIA et PortMEDIA-Fr (Dev et Test) pour la partie utilisateur seulement, avec le modèle en configuration «mots». 106
4.7 Comparaison des résultats états de l'art MEDIA et PortMEDIA-Fr, toutes architectures et systèmes confondus. Résultats sur les jeux Test. 107

4.8 Nombre de paramètres des modèles *end-to-end* récurrents, mono-décodeur et multi-décodeurs. 111
4.9 Résultats sur le corpus MEDIA des différentes configurations explorées de l'architecture multi-décodeurs. 112
4.10 Résultats en WER sur le corpus MEDIA (Dev et Test) pour la partie utilisateur seulement. Les résultats des modèles (b), (c) et (d) sont ceux des décodeurs ASR, tandis que les résultats du modèle (a) sont ceux du décodeur SLU_{Mots}, sans les concepts. 113

5.1 Étapes d'apprentissage de nos modèles *end-to-end*, à partir des *checkpoints* initiaux pré-entraînés, jusqu'à la prédiction de concepts SLU. 123
5.2 Nombre de paramètres, temps d'apprentissage et équivalent CO_2 lié à l'apprentissage des modèles réalisés. 124
5.3 Performances des deux modèles ASR sur le corpus MEDIA Test (partie utilisateur). 125
5.4 Résultats en CER et CVER du modèle cascade sur le corpus MEDIA Test. 125
5.5 Résultats en CER et CVER des architectures *end-to-end* étudiées, sur le corpus MEDIA Test. 126
5.6 Descriptif des systèmes étudiés pour nos analyses des erreurs. Les résultats sont donnés à titre de comparaison, sur le corpus MEDIA Test, et correspondent aux meilleurs résultats publiés pour chaque modèle. 128

6.1 Description technique des FlauBERT utilisés. 141
6.2 WER des modules de transcription employés dans nos comparaisons, sur le corpus MEDIA Test. 141
6.3 Résultats des modèles FlauBERT sur le corpus MEDIA Test, en mode cascade, selon les transcriptions utilisées (Oracle, Kaldi, et wav2vec 2.0). . 142
6.4 Résultats sur le corpus MEDIA-Fr Test en Micro-F_1 des modèles sac-de-concepts (Fr : MEDIA-Fr). Représentations d'entrée texte (LaBSE) ou parole (SAMU-XLSR). *Zéro-shot* modalité et langue : oui (✓) ou non (✗) . 147
6.5 Résultats sur le corpus PortMEDIA-It Test en Micro-F_1 des modèles sac-de-concepts (Fr : MEDIA-Fr, It : PortMEDIA-It). Représentations d'entrée texte (LaBSE) ou parole (SAMU-XLSR). *Zéro-shot* modalité et langue : oui (✓) ou non (✗) 147
6.6 Résultats en CER et CVER sur le corpus MEDIA Dev et Test du modèle `llama-2-70B-chat` en *few-shot*. 152

Liste des figures

1.1 Diagramme d'Euler de l'intelligence artificielle. 24
1.2 Schéma conceptuel du *Perceptron*. 26
1.3 Schéma conceptuel du *Multilayer Perceptron*. 26
1.4 Schéma conceptuel d'une cellule d'un RNN. 28
1.5 Schéma conceptuel d'une cellule d'un LSTM. 29
1.6 Schéma conceptuel d'un encodeur-décodeur utilisant des cellules RNN. . . 30
1.7 Principe du décodage CTC, avec un exemple pour la transcription de la parole. 31
1.8 Schéma conceptuel d'un encodeur-décodeur avec mécanisme d'attention. . 33
1.9 Schéma conceptuel d'un modèle transformer encodeur-décodeur générant une séquence à partir de quatre éléments d'entrée. 34
1.10 Principe du pré-apprentissage auto-supervisé (SSL). 38
1.11 Quatre exemples de configuration possibles pour les modèles SSL une fois la phase de pré-apprentissage terminée. Les exemples portent sur un modèle prenant du texte comme entrée (modèle de langage). 39

2.1 Principe général d'un système de reconnaissance automatique de la parole. 48
2.2 Exemple de signal de parole en temporel (à gauche) et fréquentiel (à droite). 51
2.3 Un exemple d'une banque de filtres Mel avec 6 filtres et une fréquence d'échantillonnage de 8kHz. 52
2.4 Exemple d'un HMM comportant trois états et quatre observations. 53
2.5 Exemple fictif de décodage glouton ayant généré la phrase «Le chien mange dans sa gamelle». Seuls les trois mots les plus probables sont affichés à chaque instant. 62
2.6 Exemple fictif de décodage en faisceau ($N = 2$) ayant généré les phrases «Le chien mange dans sa gamelle» et «Le chien mange sans sa gamelle». Seuls les trois mots les plus probables sont affichés à la suite de chaque hypothèse en cours. 63

3.1 Exemple de représentation sémantique à base de *frames* pour un segment utilisateur «*quels sont les hôtels proches du stade de france avec une piscine*». 71
3.2 Exemple de représentation en segments sémantiques (concepts et valeurs), pour une phrase utilisateur «*quels sont les hôtels proches du stade de france avec une piscine*». 73
3.3 Fonctionnement d'un système de dialogue humain-machine. 76
3.4 Exemple d'annotation en concepts avec le formalisme balisé. 83
3.5 Exemple de calcul du WER, CER et CVER pour deux hypothèses par rapport à une référence. 85
3.6 Exemple du corpus MEDIA (jeu de DEV, dialogue 1452). 88

4.1 Détail de l'architecture récurrente de compréhension de la parole *end-to-end*.101
4.2 Arbre préfixe pour un vocabulaire exemple composé des mots : «c», «c'est», «c'était», «cadre», «cadres», «camargue», «camp», «campagne» et «campanille». 102
4.3 Détail de l'architecture récurrente de compréhension de la parole hybride *end-to-end*/cascade. 110

5.1 Aperçu des différentes architectures de compréhension présentées. 119
5.2 Architecture de la partie encodeur-décodeur (ASR_{ed}) du modèle proposé. . 120
5.3 Architecture du décodeur SLU_{txt} utilisant la représentation textuelle générée par le décodeur ASR. 121
5.4 Architecture du décodeur $SLU_{txt+sig}$ utilisant à la fois la représentation textuelle générée par le décodeur ASR, mais également la représentation du signal de parole générée par l'encodeur. 122
5.5 Nombre d'erreurs sur le corpus MEDIA DEV par concepts pour chaque système étudié. 129
5.6 CER et WER sur le corpus MEDIA DEV des systèmes étudiés pour trois groupes de concepts (noms propres, vocabulaire fermé, nombres). 132

6.1 Illustration d'un espace sémantique multilingue et crosslingue. 145
6.2 Architecture employée pour l'évaluation des modèles SAMU-XLSR (parole) et LaBSE (texte) sur les corpus MEDIA-Fr et PortMEDIA-It. 146
6.3 Extrait d'instruction système utilisée pour initier le modèle LlaMa. 151
6.4 Exemple de génération du modèle LLAMA à partir du segment utilisateur d'entrée. Exemple issu du jeu DEV de MEDIA. 153

6.5 Exemple de génération du modèle LlaMa à partir du segment utilisateur d'entrée. Exemple issu du jeu Dev de MEDIA. 153
6.6 Exemple de génération du modèle LlaMa à partir du segment utilisateur d'entrée. Exemple issu du jeu Dev de MEDIA. 153
6.7 Dialogue de compréhension globale d'un exemple similaire à ceux existants dans le corpus MEDIA avec LlaMa-2. 156
6.8 Dialogue de compréhension globale d'un exemple similaire à ceux existants dans le corpus MEDIA avec ChatGPT. 157

Acronymes

A :

AISSPER *Artificial Intelligence for Semantically controlled SPEech undeRstanding*
ANR Agence Nationale de la Recherche
ASR *Automatic Speech Recognition*

B :

BIO *Beginning, Inside, Outside*, souvent également nommé IOB
BPE *Byte-Pair Encoding*

C :

CER *Concept Error Rate*
ChER *Character Error Rate*
CPU *Central Processing Unit*
CRF *Conditional Random Field*
CTC *Connectionist Temporal Classification*
CVER *Concept-Value Error Rate*

D :

DBN *Dynamic Bayesian network*
DCT *Discrete Cosine Transform*
DEFT DÉfi Fouille de Textes
DL *Deep Learning*
DNC *Deep Convex Network*

E :

EFL *English Formal Language*
EM *Expectation-Maximization*

G :

GMM *Gaussian Mixture Model*
GPU *Graphics Processing Unit*
GRU *Gated Recurrent Unit*

H :

HMM *Hidden Markov Model*

I :

IA Intelligence Artificielle
INA Institut National de l'Audiovisuel

J :

JEP Journées d'Études sur la Parole
JSALT *Jelinek summer workshop on Speech and LAnguage Technology*

L :

LIUM Laboratoire d'Informatique de l'Université du Mans
LLM *Large Language Model*
LM *Language Model*
LREC *International conference on Lan-*

guage Resources and Evaluation
LSTM *Long Short-Term Memory*
biLSTM *Bidirectional Long Short-Term Memory*

M :
MFCC *Mel Frequency Cepstrum Coefficients*
ML *Machine Learning*
MLM *Masked Language Model*
MLP *Multilayer Perceptron*
MPM MEDIA et PortMEDIA
MSE *Mean Squared Error*

N :
NER *Named Entity Recognition*
NLU *Natural Language Understanding*

O :
OOV *Out Of Vocabulary*

R :
RNN *Recurrent Neural Network*

S :
SLU *Spoken Language Understanding*
SSL *Self-Supervised Learning*

T :
TAL Traitement Automatique des Langues

W :
WAV *Waveform audio file format*
WER *Word Error Rate*
WIP *Word Insertion Penalty*
WOZ *Wizard of Oz*
WSJ *Wall Street Journal*

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Le Mans Université

Titre : La compréhension de la parole dans les systèmes de dialogues humain-machine à l'heure des modèles pré-entraînés

Mot clés :

- compréhension de la parole
- reconnaissance automatique de la parole
- réseaux de neurones
- modèles pré-entraînés
- modèles auto-supervisés
- extraction de concepts sémantiques

Résumé : Dans cette thèse, la compréhension automatique de la parole (SLU) est étudiée dans le cadre applicatif de dialogues téléphoniques à buts définis (réservation de chambres d'hôtel par exemple). Historiquement, la SLU était réalisée en cascade : un système de reconnaissance de la parole réalisait une transcription en mots, puis un système de compréhension y associait une annotation sémantique. Le développement des méthodes neuronales profondes a fait émerger les architectures de bout-en-bout, où la tâche de compréhension est réalisée par un système unique, appliqué directement à partir du signal de parole pour en extraire l'annotation sémantique. Récemment, les modèles dits pré-entraînés de manière non supervisée (SSL) ont apporté de nouvelles avancées en traitement automatique des langues (TAL). Appris de façon générique sur de très grandes masses de données, ils peuvent ensuite être adaptés pour d'autres applications. À ce jour, les meilleurs résultats SLU sont obtenus avec des systèmes en cascade intégrant des modèles SSL.

Cependant, aucune des architectures, cascade ou bout-en-bout, n'est parfaite. À travers cette thèse, nous étudions ces architectures et proposons des versions hybrides qui tentent de tirer parti des avantages de chacune. Après avoir développé un modèle SLU bout-en-bout à l'état de l'art, nous avons évalué différentes stratégies d'hybridation. Les avancées apportées par les modèles SSL en cours de thèse, nous ont amenés à les intégrer dans notre architecture hybride.

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Le Mans Université

Title: Spoken language understanding in human-computer dialogue systems in the era of pretrained models

Keywords:

- spoken language understanding
- automatic speech recognition
- neural networks
- pretrained models
- self-supervised models
- semantic concepts extraction

Abstract: In this thesis, spoken language understanding (SLU) is studied in the application context of telephone dialogues with defined goals (hotel booking reservations, for example). Historically, SLU was performed through a cascade of systems: a first system would transcribe the speech into words, and a natural language understanding system would link those words to a semantic annotation. The development of deep neural methods has led to the emergence of end-to-end architectures, where the understanding task is performed by a single system, applied directly to the speech signal to extract the semantic annotation. Recently, so-called self-supervised learning (SSL) pre-trained models have brought new advances in natural language processing (NLP). Learned in a generic way on very large datasets, they can then be adapted for other applications. To date, the best SLU results have been obtained with pipeline systems incorporating SSL models.

However, none of the architectures, pipeline or end-to-end, is perfect. In this thesis, we study these architectures and propose hybrid versions that attempt to benefit from the advantages of each. After developing a state-of-the-art end-to-end SLU model, we evaluated different hybrid strategies. The advances made by SSL models during the course of this thesis led us to integrate them into our hybrid architecture.