# Reconnaissance de parole pour le français et intégration dans un système de compréhension du langage parlé

Florian Boyer

# THÈSE

pour obtenir le grade de

## DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

Spécialité: Informatique

---

Présentée et soutenue le 20 octobre 2021 par

**Florian Boyer**

# Reconnaissance de parole pour le français et intégration dans un système de compréhension du langage parlé

---

École doctorale : Mathématique et Informatique
Équipe d'accueil : Image et Son (LaBRI, Bordeaux)

Directeur de thèse: **Dr. Jean-Luc ROUAS** (LaBRI, Bordeaux)
Co-directeur de thèse: **Antoine SOTTIAU** (Airudit, Bordeaux)

**COMPOSITION DU JURY**:

Rapporteur: **Pr. Anthony LARCHER** (LIUM, Le Mans)
Rapporteur: **DR. Gérard CHOLLET** (SAMOVAR, Paris)
Examinateur: **Pr. Laurent BESACIER** (Naver Labs, Grenoble)
Examinateur: **Pr. Aurélie BUGEAU** (LaBRI, Bordeaux)

airudit LaBRI cnrs université de BORDEAUX

# Résumé

Dans une société où un nombre considérable de systèmes complexes et d'objets connectés voient le jour, le besoin de simplifier les interactions Homme-Machine est devenu une problématique importante aussi bien pour la communauté scientifique que les entreprises. Dans ce cadre, la parole étant considérée comme un moyen de communication évident, la définition d'un système de Reconnaissance Automatique de Parole (RAP) répondant à différents critères de performance, de robustesse et de rapidité s'avère cruciale. Celui-ci doit aussi admettre un caractère évolutif et pouvoir s'adapter facilement en considération de l'évolution d'une langue, d'un contexte métier précis ou encore de la mise en relation avec d'autres briques logicielles de la chaîne de communication (p. ex., les modules de détection des intentions utilisateurs ou de compréhension d'un énoncé textuel). Dans ce contexte, l'objectif principal de cette thèse est le développement d'un système de reconnaissance automatique de parole pour le français et sa mise en relation avec un système de compréhension du langage naturel (CLN) proposé par la société Airudit, porteuse de cette thèse avec le Laboratoire Bordelais de Recherche en Informatique (LaBRI).

Afin de répondre au premier objectif, nous proposons une étude comparative des principales approches de RAP existantes de nos jours pour le français. Une attention particulière est donnée ici sur le type d'approche (RAP traditionnelle ou RAP bout-en-bout), la définition de l'architecture optimale ainsi que le type d'unités en sortie (caractères, sous-mots ou mots). Cette étude est ensuite étendue par une comparaison des erreurs formulées par les différents systèmes produits en vue d'une interprétation par un système de compréhension du langage naturel. Parallèlement, je présente mes contributions dans le cadre du projet ESPnet qui met à disposition de la communauté scientifique des outils pour le traitement de la parole. Toujours dans une optique de construction d'un système de RAP *optimal* pour le français, l'accent a été mis sur la proposition de techniques d'entraînement de d'inférence pour l'un des systèmes les plus performants durant notre première étude : le RNN-Transducer.

Concernant le second objectif visant à étudier et améliorer la mise en relation avec un système de compréhension, nous proposons de plus une nouvelle approche ayant donné lieu à un brevet. Celle-ci vise à restituer des mécanismes observés de la communication naturelle (p. ex., la connaissance a priori du contexte, l'inférence du sens malgré le manque d'informations ou la présence d'erreurs) et se propose aussi de résoudre des problèmes inhérents de l'association d'un système de RAP traditionnel avec un système de CLN, créés et optimisés de manière indépendante.

**Mots-clefs:** Reconnaissance Automatique de Parole, Compréhension du Langage Parlé, Apprentissage profond, Reconnaissance de Parole bout-en-bout.

# Abstract

In a world where a considerable number of complex systems and smart objects are emerging, the need to simplify human-machine interactions has become an important issue for both the scientific community and the industrial field. In this context, speech is considered an obvious means of communication. Thus, the definition of an Automatic Speech Recognition (ASR) system answering different criteria of performance, robustness, and speed is becoming crucial. This system must also admit an evolutionary trait and be able to adapt easily in consideration of the evolution of a language, of a precise context, or even of the connection with other programs making up the communication chain (e.g.: user-intent detection, language understanding). In this context, the main objective of this thesis is the development of an automatic speech recognition (ASR) system for French and its linking with a natural language understanding (NLU) system proposed by the company Airudit, which is the advisor of this thesis with the Laboratoire Bordelais de Recherche en Informatique (LaBRI).

In order to address the first objective, we propose a comparative study of the main ASR approaches existing nowadays, applied to French. A focus is given on the type of approach (traditional ASR or end-to-end ASR), the definition of the optimal architecture, as well as the type of output units (characters, sub-words, or words). This study is then extended by a comparison of the errors formulated by the different systems with an emphasis on the interpretation by a natural language understanding system. At the same time, I present my contributions to the ESPnet project which provides the scientific community with tools for speech processing. With the perspective of building an optimal ASR system for French, a particular attention has been given to the proposal of training and inference techniques for one of the most performing systems during our first study: the RNN-Transducer.

Concerning the second objective aiming at studying and improving the relationship with a comprehension system, we also propose a new approach that has been patented. This approach aims at restoring some observed mechanisms of the natural communication (e.g.: prior context knowledge, meaning extraction from incomplete or erroneous information) and also proposes to solve problems inherent to the association of a traditional ASR system with an NLU system, created and optimized independently.

**Keywords:** Automatic Speech Recognition, Spoken Language Understanding, Deep Learning, End-to-End Speech Recognition.

# Remerciements

En premier lieu je voudrais remercier M. Gérard Chollet et M. Anthony Larcher d'avoir accepté de rapporter cette thèse ainsi que les différents membres du jury, Mme. Aurélie Bugeau et M. Laurent Besacier, d'avoir accepté d'être présents aujourd'hui malgré les contraintes sanitaires.

Je voudrais remercier tout particulièrement M. Jean-Luc Rouas qui m'a dirigé tout au long de cette thèse. Il a toujours été disponible et à l'écoute de mes (très) nombreuses questions, et s'est toujours intéressé à l'avancée de mes travaux. Les échanges que nous avons eus ainsi que les conseils qu'il m'a donné sont pour beaucoup dans la réussite de cette thèse. En outre, ses nombreuses relectures et corrections du manuscrit ont été d'une grande aide. Pour toutes ces raisons, merci.

Je remercie aussi M. Philippe Lebas et M. Antoine Sottiau de m'avoir accueilli dans l'entreprise Airudit durant cette thèse, de m'avoir fait confiance et de m'avoir soutenu professionnellement et personnellement tout au long de ma thèse. Il m'ont offert une liberté pour mes recherches et m'ont permis de m'investir aussi bien dans des projets industriels que des projets académiques avec des personnes extérieures. En outre, leur compréhension des contraintes, temporelles et humaines, que pose la recherche sur l'industrialisation de solutions dans l'immédiat ont été appréciables.

Je remercie mes collègues d'Airudit et notamment Mathilde de m'avoir aidé dans la rédaction du chapitre 9, qui présente le brevet que nous avons déposé, ainsi que pour ses différentes relecture du manuscrit, mais aussi Anthony, Angélique, Thibault, ..., pour les nombreux échanges que nous avons eu ainsi que leur aide apportée sur les parties traitant des ontologies.

Je remercie également le LaBRI de m'avoir accueilli et notamment mes collègues de bureau du LaBRI au cours de ces trois années, Pierre et Vincent tout particulièrement ainsi que les nombreuses personnes avec qui j'ai pu échanger et qui m'ont aidé à différentes occasions. Entre autres Mme. Aurélie Bugeau, M. Lionel Clément, M. Boris Mansencal...

Je tiens aussi à remercier M. Shinji Watanabe de m'avoir donné l'opportunité de rejoindre le projet ESPnet, ainsi que les différents membres du projet avec qui j'ai pu travailler et échanger, M. Kamo Nayoki, M. Yusuke Shinohara, M. Takaaki Hori et tant d'autres. Les différents travaux auxquels j'ai pu participer ainsi que les échanges que j'ai pu avoir dans le cadre de ce projet m'ont permis de m'épanouir pleinement dans le cadre académique et m'ont donné la possibilité de prendre part à différentes contributions scientifiques.

Enfin, je remercie mon père et mon chat pour leur soutien indéfectible.

# Table des matières

# Introduction générale

## Contexte scientifique : brève introduction

Depuis une centaine d'années maintenant avec l'émergence de travaux permettant l'automatisation des tâches pilotées par une machine, la possibilité de construire une machine capable de converser avec un humain à l'oral et en langage naturel a fasciné les chercheurs. Dans ce contexte, et en corrélation avec l'émergence de l'informatique et des disciplines liées, une branche de recherche dédiée au développement de méthodes et de techniques pour le traitement de la parole s'est peu à peu démocratisée. Malgré des travaux notables dans les années 50 avec un premier système reconnaissant des chiffres prononcés par un locuteur donné en se basant sur la localisation des formants pour chaque prononciation [Davis et al., 1952], puis au début des années 60 avec l'introduction du premier modèle source-filtre pour décrire la production orale par l'utilisation de différentes sources sonores et filtres [Fant, 1970], il aura toutefois fallu attendre la fin des années 60 et les avancées majeures en modélisation statistique avant de considérer les systèmes de Reconnaissance Automatique de Parole (RAP) comme technologie pivot pour la communication homme-machine [Pierce, 1969].

Ainsi, sur la base de leurs travaux sur la théorie des fonctions probabilistes des chaînes de Markov à la fin des années 60 et au début des années 70, L. E. Braum et ses collègues proposent dans une série de publications plusieurs contributions qui vont permettre d'avancer considérablement le développement des systèmes de RAP. Par l'introduction, tout d'abord, d'un modèle statistique appelé Modèle de Markov Caché (ou *Hidden Markov Model* en anglais, abrégé ici HMM) qui permettra de caractériser les propriétés temporelles de la parole par le biais d'un diagramme d'états de Markov et de combiner différentes sources de connaissances [Baum and Petrie, 1966, Baum and Eagon, 1967], et d'un algorithme d'estimation de ses paramètres cachés appelé algorithme Baum-Welch [Baum et al., 1970]. Puis, par des travaux sur l'application des HMMs pour la reconnaissance automatique de parole par différentes équipes de recherche [Baker, 1973, Baker, 1975, Jelinek, 1976] qui introduisent en suivant l'utilisation des densités de mélanges dans les architectures des HMMs à la place des densités de probabilité discrètes [Biing-Hwang, 1984], ainsi qu'un ensemble de procédures et techniques pour la ré-estimation de la méthode forward-backward ou encore l'entraînement des paramètres du HMM [Bahl et al., 1983, Biing-Hwang, 1985, Rabiner, 1991].

L'ensemble de ces travaux, conjointement avec l'apparition de systèmes démontrant des améliorations considérables en termes de performances et de taille de vocabulaire, installeront les HMMs comme technologie déterminante à la fin des années 80 et au début des années 90 pour la construction de systèmes de RAP. Dès lors, et jusqu'à encore aujourd'hui, les systèmes utilisant des HMMs sont légions et s'appliquent à un large panel

de tâches : de la transcription d'un énoncé (reconnaissance automatique de la parole) à la restitution d'une réponse à l'oral (synthèse), en passant par la compréhension du langage parlé ou encore l'identification du locuteur. Les technologies issues des travaux de ces différents domaines se retrouvent aussi maintenant dans de nombreuses applications (traduction automatique, assistants vocaux, système de dialogue, apprentissage au dialogue, etc.) et de nombreux secteurs comme l'éducation, le militaire, l'hospitalier ou encore le secteur bancaire. En outre, ces technologies touchent maintenant un nombre conséquent de personnes de par la démocratisation des solutions mobiles et assistants vocaux.

D'autres avancées viendront contribuer à l'amélioration des systèmes basés sur les HMMs les années suivantes, par exemple avec l'introduction des modèles back-off [Katz, 1987] et de méthodes d'entraînement discriminantes [Bahl et al., 1986]. Il faudra toutefois attendre les avancées majeures en apprentissage machine et notamment l'avènement de l'apprentissage profond pour voir l'arrivée d'architectures alternatives, pouvant s'abstraire de l'utilisation des HMMs ou même de l'utilisation de modélisation linguistique.

Parallèlement, au milieu des années 2000, on peut voir l'apparition de systèmes de RAP matures et *généralisés* (c'est-à-dire robustes en terme de locuteurs et d'environnements) à destination des particuliers et ce, pour diverses langues comme l'anglais, le français, l'allemand ou encore le mandarin. Accéléré et démocratisé par l'usage massif des solutions mobiles et des objets connectés, il faudra toutefois attendre jusqu'au début des années 2010 pour voir le secteur industriel, et notamment français, s'intéresser à l'usage de la RAP pour différents secteurs et services.

## Contexte industriel

Dans une société où la mobilité est devenue la norme, la simplification des interactions entre l'humain et les systèmes informatiques reste un challenge technologique fort. Aujourd'hui les employés sont focalisés sur leurs missions et croulent sous les pressions de toutes sortes, et dans de nombreux cas, l'utilisation d'un écran ou d'interfaces humain/machine de manière tactile est devenue une contrainte importante. Les exemples que nous avons pu rencontrer chez nos partenaires étant nombreux :

- Un pilote d'avion de chasse qui effectue deux métiers, pilote et combattant, et qui en pleine mission, doit en même temps manipuler une tablette, des documents papiers, des organes de commande.
- Un opérateur sur une chaîne de montage ou de maintenance ayant les mains prises par de gros gants de protection et devant quitter son poste de travail pour interroger le stock ou effectuer la saisie d'un contrôle qualité sans interrompre la chaîne logistique.
- Un employé d'entreprise qui doit comprendre instantanément une situation de terrain pour agir en équipe alors que les données sont toutes dispersées dans le système d'information.

La parole dans l'entreprise est un formidable moyen de fluidifier les relations humain-machine, et d'apporter du confort là où les interfaces informatiques classiques ne sont plus adéquates. Là où des informations peuvent devenir complexes à restituer ou communiquer, la voix s'impose naturellement.

Toutefois, à une époque où les offres d'assistants vocaux se multiplient, il subsiste un problème majeur de compréhension et de *contextualisation* des demandes pour la prise

en compte des requêtes complexes. La richesse du langage naturel vient du fait que, contrairement aux langages formels, le sens d'une phrase peut venir autant de ce qui est énoncé, que du contexte non-dit, voire implicite. Les approches archaïques par mots clefs ont montré qu'elles n'étaient pas pertinentes pour interpréter des phrases qui peuvent être complexes ou embarquant un vocabulaire métier spécifique. Les approches statistiques semblent elles aussi peu efficaces, et nécessitent des milliers de formulations et de données pour commencer à être pertinentes.

## Objectifs et portée des travaux

Ce manuscrit synthétise les travaux menés entre 2017 et 2020 au sein de l'entreprise Airudit (anciennement EA4T) et du Laboratoire Bordelais de Recherche en Informatique (LaBRI, Université de Bordeaux, INP, CNRS, INRIA UMR5800), sous la direction de Antoine SOTTIAU (Airudit) et Jean-Luc ROUAS (LaBRI, CNRS). Ces travaux s'inscrivent dans le cadre d'un contrat CIFRE entre l'université de Bordeaux et Airudit, situé à la La Cité de la Photonique à Pessac. L'objectif principal de cette thèse a été la réalisation d'un système de RAP pouvant être utilisé dans un contexte de communication homme-machine et déployable sur des interfaces avec de faibles ressources (p. ex., sur des téléphones mobiles ou des Raspberry Pi). Cet objectif comprend donc deux problématiques générales à traiter : 1) l'étude du rôle et de la place d'un système de reconnaissance automatique de parole dans le cadre d'une communication homme-machine, représenté sous la forme d'un système de Compréhension du Langage Parlé (abrégé CLP), et 2) l'étude sur la robustesse du système de RAP en considération de la présence d'environnements, de scénarios et de locuteurs français variés. À cela, et en corrélation avec les travaux effectués dans le cadre de projets industriels de l'entreprise et de projets de recherche au sein du LaBRI, je propose de traiter les sous-problématiques suivantes afin de proposer un système et un ensemble de techniques permettant la résolution des objectifs généraux:

- Examiner l'état actuel de la recherche en RAP de manière générale et adaptée pour le français.
- Proposer un système de RAP temps-réel en français adapté à différents scénarios industriels.
- Proposer de nouvelles approches pour le problème des mots hors vocabulaire en RAP, en considération de l'évolution du langage et d'une association avec un système de CLP.
- Parallèlement, évaluer l'impact des mots hors vocabulaire et mots erronés, produits par un système de RAP. Un focus sera donné sur l'étude d'une unité minimale et de mesures en considération du problème de reconnaissance automatique de parole et du problème de compréhension du langage parlé dans le cadre d'une communication orale Homme-machine.
- Étudier et proposer des méthodes pour la conceptualisation du dialogue s'appuyant sur l'utilisation de connaissances a priori, représentées sous la forme d'ontologies.

## Contributions

Au cours de cette thèse, un nombre de contributions originales ont été apportées dans le domaine de la reconnaissance de parole et de la compréhension du langage parlé. Celles-ci

peuvent être de nature académique ou industrielle. Ces contributions sont résumées dans la liste suivante et seront développées dans leur parties respectives :

- Évaluation de l'ensemble des méthodes et architectures actuellement utilisées en RAP pour le français.
- Étude sur l'unité minimale à modéliser pour le problème de RAP en français, de manière isolée et en considération du problème de CLP.
- Formalisation des erreurs produites par les systèmes de RAP *bout-en-bout* en français et évaluation de l'impact des erreurs pour le problème de compréhension du langage.
- Contributions à différents projets et boites à outils libre pour la RAP. Ces contributions portant sur de nombreux aspects tel que : l'entraînement, l'inférence en mode hors-ligne et en ligne (ou *streaming*) ou encore l'ajout de fonctionnalités et de recettes pour la construction de systèmes de RAP pour différentes langues.
- Proposition d'une méthode s'appuyant sur l'utilisation d'informations provenant d'ontologies pour la contextualisation de l'étape de décodage d'un système de RAP.

## Structure du document

Afin de répondre aux problématiques présentées précédemment et dans un souci de faciliter la compréhension du lecteur, ce document est structuré en 4 parties, chacune composée de plusieurs chapitres. La première partie introduit le lecteur aux différents domaines traités et donne les pré-requis nécessaires pour aborder les parties suivantes, la seconde et troisième partie décrivent respectivement les contributions scientifiques et industrielles apportées durant la thèse et la dernière partie conclut sur les problématiques traitées et les axes de travail futurs.

## Description des chapitres

Les chapitres 1 à 3, représentant la première partie de ce manuscrit, sont dédiés à l'introduction de l'ensemble des notions nécessaires pour aborder les contributions présentées dans les parties suivantes. Respectivement, ces chapitres se rapportent à l'introduction générale du domaine de la reconnaissance de parole, la description des composants d'un système de RAP traditionnel basé sur les HMMs et les transducteurs à états finis (*weighted Finite-State Transducer* en anglais) jusqu'à la démocratisation des réseaux de neurones dits *feed-forward*, et enfin, la description des approches de RAP bout-en-bout (ou *End-to-End* en anglais, abrégé E2E), et architectures neuronales liées, apparues ces dernières années.

Pour initier la seconde partie concernant les contributions scientifiques, une description complète des bases de données et des corpus de la parole utilisés, ainsi que ses sous-ensembles, est tout d'abord donné dans le chapitre 4.

Dans le chapitre 5, je présente la boite à outils ESPnet et les contributions apportées à celle-ci. Ce chapitre est séparé en trois sous-parties servant respectivement à 1) présenter la boite à outils et les fonctionnalités principales existantes, 2) décrire les thématiques et contributions personnelles et 3) introduire les publications et travaux majeurs de différents acteurs du secteur académique ou industriel reposant sur mes contributions.

Dans le chapitre 6, je me focalise sur l'évolution des systèmes en considération de l'évolution du langage. Principalement, je m'intéresse à deux sous-problèmes particuliers : 1) le cas des mots hors-vocabulaire en RAP, et 2) la définition et la sélection d'une unité minimale, orthographique ou phonétique, pour le problème de la RAP. Pour cela, une revue des différentes méthodes de reconnaissance de parole existantes à ce jour est proposée pour la langue française spontanée. Un focus particulier est donné sur les méthodes bout-en-bout et l'utilisation d'unités orthographiques en lieu et place de l'unité phonétique usuelle.

Sur les bases de ces travaux et de ces résultats, j'étends mon étude dans le dernier chapitre de cette partie, le chapitre 7, en me focalisant sur les erreurs formulées par les différentes méthodes et leur impact en considération d'une association avec le problème de compréhension du langage parlé. Pour cela, différentes analyses et mesures sur la production des systèmes évalués sont proposées.

Je débute la dernière partie portant sur les contributions industrielles en consacrant la première partie du chapitre 8 aux travaux relatifs à des projets industriels effectués avec Airudit. Des applications industrielles en rapport direct ou indirect avec des travaux de recherche précédemment introduits seront données et détaillées dans la limite des accords de non-divulgation signés pour différents projets. Les projets et thématiques discutés étant, entre autre, le *Man Machine Teaming* sur le thème "Assistant Virtuel & Smart Cockpit", le projet de la DGA sur le thème de la "Formation militaire avec assistant virtuel et augmenté" et différents projets en lien avec la domotique et les objets connectés.

Dans la deuxième partie de ce même chapitre, je me concentre ensuite sur la notion de connaissance a priori et les mécanismes de restitution du sens dans le dialogue. Au travers d'un cas de figure mettant en scène un locuteur "non-expert" et un locuteur "expert" (dont le vocabulaire métier est connu mais dont le vocabulaire courant peut être différent du premier locuteur), une étude sur les mécanismes d'inférence du sens dans le dialogue est formulée. De cette étude, une méthode avec plusieurs modes de fonctionnement est proposée pour incorporer des informations conceptuelles et contextuelles dans le système de RAP par le biais de différentes sources et systèmes experts pouvant exister dans un système de CLP complet. Cette méthode a fait l'objet d'un dépôt de brevet national qui a été accepté.

Je conclue ce manuscrit en résumant les travaux effectués durant la thèse et les résolutions apportées pour les problématiques mises en évidence dans cette introduction. Enfin, je présente différents travaux et projets initiés au moment de l'écriture de ce manuscrit qui s'inscrivent dans la continuité des travaux scientifiques et industriels présentés ici.

# Chapter 1

# Problème de la reconnaissance automatique de parole

Ce chapitre fournit les informations générales relatives au problème de la Reconnaissance Automatique de Parole (RAP), de la description, la production et la perception de la parole à la représentation structurelle d'un système de RAP. En outre, la représentation mathématique du problème, la description succincte des composants d'un système de RAP, ainsi que les méthodes d'évaluation de ce système sont données. Pour conclure ce chapitre, je fournis une liste chronologique des contributions notables dans le domaine de la RAP entre la fin des années 80 et le milieu des années 2010.

## 1.1 Production de la parole

La production de la parole désigne le processus permettant de transformer une représentation lexico-grammaticale, initiée par le cerveau, en parole. Ce processus encapsule l'ensemble des actions permettant de produire la parole : la sélection des mots à produire, l'organisation des formes grammaticales pertinentes et l'articulation des sons par le système moteur via un ensemble d'organes de la parole schématisé par la figure 1.1. À cela, il est nécessaire de définir une classification des sons suivant leur mode de production et un programme moteur adéquat avant d'initier une production physique des sons.

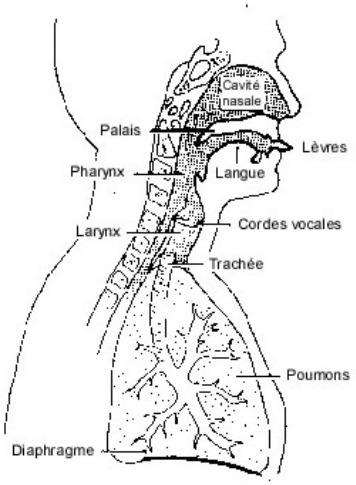



Figure 1.1: Schéma des parties du corps impliquées dans le processus de production de la parole. *Cours de l'Université de technologie de Compiègne, 1998.*

Cette production physique commence par un souffle d'air expulsé par compression des poumons. Ce souffle d'air est ensuite modulé par le larynx via les plis vocaux et plis vestibulaires puis différents articulateurs (c.-à-d., le voile du palais, la cavité buccale, les lèvres et la langue) modifiant la forme du conduit vocal. Cela a pour effet de créer de nombreuses fréquences de résonance et, par conséquent, différents sons de la parole.

## 1.2 Perception de la parole

La perception de la parole est le processus par lequel les humains sont capables d'interpréter et de comprendre les sons utilisés dans un langage. Dans un souci d'amélioration des performances du système de RAP et le décodage acoustico-phonétique, des aspects physiques de la perception de parole sont utilisés, basés sur des connaissances relatives au système auditif humain.

### 1.2.1 Système auditif humain

Le système auditif humain est divisé, de manière anatomique et fonctionnelle, en trois grandes zones : l'oreille externe composée d'un pavillon et d'un conduit auditif (ou méat), l'oreille moyenne composée d'une chaîne de trois osselets (le marteau, l'enclume et l'étrier), et enfin l'oreille interne composée d'un système vestibulaire et de l'organe de l'audition (la cochlée) abritant les cellules sensorielles. Une représentation schématisée du système est donnée par la Figure 1.2.

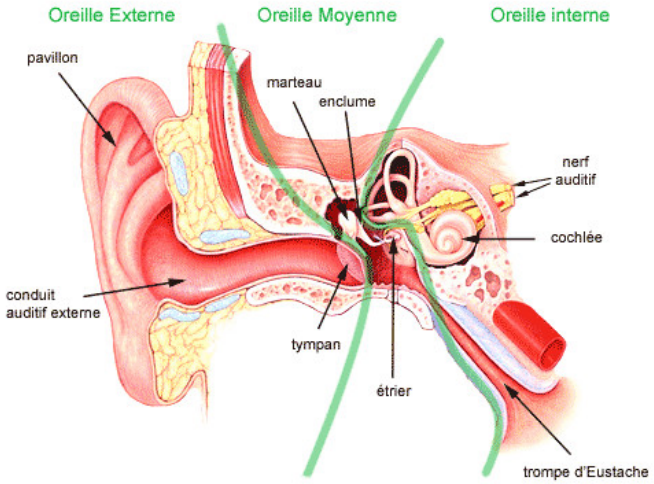



Figure 1.2: Schéma des parties du système auditif impliquées dans le processus de perception de parole. Extrait de *Cours de linguistique théorique et descriptive*. Récupéré sur studylib[1]

Étant donné un son en entrée de l'oreille extérieure, celui-ci est d'abord modifié par le pavillon afin de filtrer les sons reçus, de privilégier les sons dans la gamme de fréquences de la parole humaine et d'ajouter les informations de direction. Ceux-ci sont dirigés en suivant vers le tympan, par le biais du conduit auditif, qui est mis en vibration. Les mouvements du tympan sont ensuite transmis à l'oreille interne via les osselets. Pour finir, le cochlée amplifie les vibrations reçues, analyse et dirige les différentes vibrations vers les fibres nerveuses en considération de leur fréquence, et enfin, les transforme en messages pouvant être interprétés par le cerveau, c'est-à-dire des influx nerveux.

## 1.3 Représentation mathématique du problème

Le problème de la reconnaissance automatique de parole, ou plus spécifiquement de la transcription, est défini comme suit. Étant donné une séquence d'observations acoustiques $O = (o_1, ..., o_N)$ issue d'un signal de parole, un module de reconnaissance de parole a pour objectif de trouver la séquence de mots $\hat{W} = (w_1, ..., w_M)$ la plus probable sachant l'observation $O$, tel que :

$$\hat{W} = \arg\max_{w} P(W|O) \tag{1.1}$$

En pratique, il est toutefois difficile de modéliser directement $P(W|O)$. Ainsi, afin de pallier ce problème, une décomposition de la probabilité en plusieurs composantes est traditionnellement effectuée en appliquant la règle de Bayes [Baum et al., 1970, Rabiner and Juang, 1986] de la manière suivante :

$$\begin{aligned}\hat{W} &= \arg\max_{w} \frac{P(O|W)P(W)}{P(O)} \\ &= \arg\max_{w} P(O|W)P(W)\end{aligned}$$

[1] `https://studylibfr.com/catalog/Sciences/M%C3%A9decine/Audiologie`

Ici, $P(O|W)$ et $P(W)$ désignent respectivement la probabilité conditionnelle calculée à partir des observation acoustiques $O$ et d'un modèle acoustique, et la probabilité donnée par un modèle de langage. Traditionnellement, le modèle acoustique produisant des unités élémentaires intermédiaires à la place de mots, des transformations ou modèles supplémentaires peuvent intervenir afin de transformer la séquence d'unités élémentaires en séquence de mots.

Dans ce cadre de la RAP traditionnelle, l'unité élémentaire préférée est habituellement le phonème et désigne la plus petite unité distinctive que l'on puisse isoler du flux de parole. L'ensemble des phonèmes distinctifs $Q$ pour une langue permet de former l'ensemble des prononciations possibles pour les mots $W$ de cette même langue. Dans la section 2.2, je discuterai plus en détails de cette représentation. A la suite, et notamment à partir du chapitre 3, d'autres unités intermédiaires de nature orthographique seront aussi abordés.

## 1.4 Découpage fonctionnel

Étant donné un signal de la parole en entrée, le système de RAP restitue une représentation orthographique en suivant le découpage fonctionnel suivant :

- Le signal de la parole est tout d'abord transformé en une série de vecteurs de caractéristiques acoustiques, chaque vecteur représentant une durée $M$ ou un nombre $N$ de trames (habituellement 20-30ms de signal avec un pas de 10ms).
- L'apprentissage automatique réalise une association entre ces vecteurs acoustiques et les éléments lexicaux, en passant par une représentation intermédiaire (habituellement le phonème). Cette association fait appel à une modélisation statistique couplant deux modélisations: les Modèles de Markov cachés (ou *Hidden Markov Models* en anglais, abrégé HMMs) pour décrire la variabilité temporelle, et les modèles de mélanges gaussiens (ou *Gaussian Mixture Models* en anglais, abrégé GMMs), ou plus récemment des réseaux de neurones profonds, pour définir les probabilités d'émission à chaque état.
- Les modèles élémentaires sont ensuite concaténés et un processus de décodage permet de reconstituer la séquence de mots le plus probable. Le processus de décodage effectuant ici une correspondance de motif temporelle, le plus souvent réalisé via un algorithme de déformation temporelle dynamique (ou *dynamic time warping*) [Sakoe and Chiba, 1978, Juang, 1984].

Le processus complet ainsi que les éléments cités seront détaillés dans le chapitre 2.

## 1.5 Métriques usuelles pour l'évaluation d'un système de RAP

Traditionnellement, les systèmes de RAP sont évalués objectivement sur deux composantes : la précision de la reconnaissance et la vitesse de la reconnaissance. Les métriques communément utilisées étant le taux d'erreur de mots (*Word Error Rate* en anglais, abrégé WER) pour la première composante et le facteur temps-réel (*Real-Time Factor* en anglais, abrégé RTF) pour la seconde.

Le taux d'erreurs de mots, cas particulier de la distance de Levenshtein appliquée au niveau des mots, est obtenu par le calcul suivant :

$$\text{TEM} = \frac{S + D + I}{N} \tag{1.2}$$

Où $S$ désigne le nombre de mots substitués, $D$ le nombre de mots supprimés et $I$ le nombre de mots insérés, par rapport à la séquence de mots de référence. $N$ correspond au nombre de mots dans la séquence de référence. En pratique, la comparaison entre la séquence de mots produites et la séquence de mots de référence est effectuée avec l'algorithme *edit distance* permettant de calculer le coût minimum de mise en correspondance des séquences [Wagner and Fischer, 1976].

Réciproquement, le taux de précision de mots (TPM) peut être obtenu de la manière suivante :

$$\text{TPM} = 1 - \text{TEM} \tag{1.3}$$

Le facteur temps-réel peut quant à lui être obtenu en comparant le temps de reconnaissance $T$ par rapport à la durée initiale de l'échantillon d'entrée $I$. Soit, de manière plus formelle :

$$\text{FTR} = \frac{T}{I} \tag{1.4}$$

## 1.6 Reconnaissance automatique de parole entre 1970 et 2016

La reconnaissance automatique de parole a connue de très nombreuses évolutions depuis le premier système proposé par AT&T Bell au début des années 50 [Juang and Rabiner, 2005]. En près de 70 ans, nous sommes passé d'un système pouvant reconnaître des chiffres prononcés par un locuteur unique et connu, à des systèmes de RAP indépendants du locuteur et pouvant reconnaître un vocabulaire quasi illimité et différents évènements sonores. Dans cette section, je présente les avancés majeures du domaine entre le début des années 70 et le début de ma thèse, au milieu des années 2010.

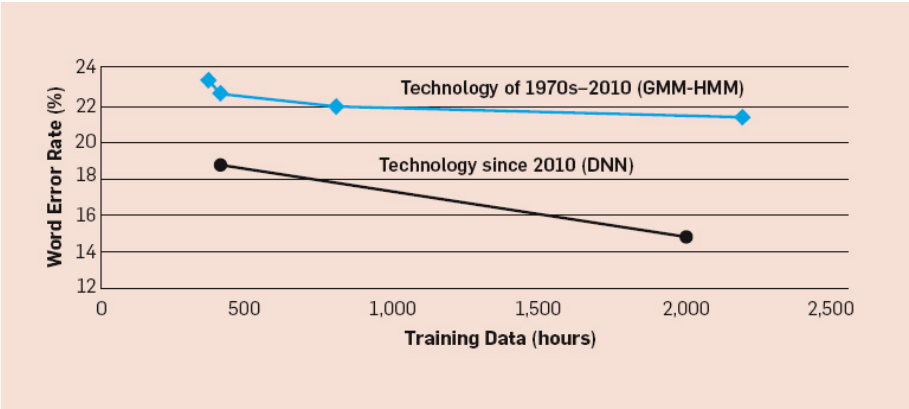



Figure 1.3: Évolution du taux d'erreurs de mots entre 1970 et le milieu des années 2010. [Huang et al., 2014]

**A partir du milieu des années 60**, L.E Braum et al. présentent dans une série de publications [Baum and Petrie, 1966, Baum and Eagon, 1967, Baum and Sell, 1968, Baum et al., 1970] un modèle statistique appelé Modèle de Markov caché qui deviendra quelques années plus tard une pierre angulaire de la modélisation acoustique en RAP.

**Durant les années 70**, aux États-Unis, la Defense Advanced Research Projects Agency (abrégé DARPA) lance un projet pour la compréhension de la parole qui débouche sur de nombreux systèmes et technologies de pointe. Parallèlement, A&T Bell Laboratories mènent des expériences dans le but de mettre au point un système de reconnaissance de mots (l'ensemble des chiffres plus les termes "oui" et "non") indépendant du locuteur [Juang and Rabiner, 2005]. De ces expériences résultent un ensemble d'algorithmes permettant de déterminer le nombre de modèles distincts nécessaires pour représenter l'ensemble des variations pour les différents mots à travers une large population d'utilisateurs.

**Dans les années 80**, un changement majeur de méthodologie vient chambouler le domaine de la RAP. Ainsi, les acteurs du domaine passent peu à peu d'une approche intuitive basée sur des modèles distincts à un cadre de modélisation statistique plus rigoureux. Ce cadre s'appuie sur différentes avancées et approches jusqu'alors peu utilisées pour la RAP. Les plus notables étant : la combinaison des coefficients cepstraux avec leur coefficients polynomiaux du premier et second ordre proposé par S. Furui [Furui, 1986], l'utilisation des HMMs pour modéliser la variabilité temporelle du problème de RAP via un processus stochastique [Rabiner, 1991], et enfin l'apparition des N-grammes, définissant la probabilité d'occurrence d'un séquence ordonnée de N mots, ainsi que modèles *back-off* [Katz, 1987] permettant au modèle de langage d'utiliser des n-grammes de longueurs multiples.

**Les années 90** voient l'apparition de différentes solutions commerciales pouvant reconnaître un vocabulaire large en anglais (comparable au vocabulaire moyen que peut connaître un locuteur lambda). Dans cet élan, le premier système indépendant du locuteur, admettant un large vocabulaire et pouvant effectuer une reconnaissance de la parole en continue voit le jour : le système Sphinx II proposé par X. Huang et al. [Huang et al., 1993], qui obtenu les meilleurs résultats durant la campagne d'évaluation de la DARPA en 1992 et qui marque une étape importante dans le domaine de la reconnaissance automatique de parole.

**Entre 1996 et 2002**, M. Mohri démontre l'utilité des transducteurs à états finis pondérés (ou *weighted finite-state transducers* en anglais, abrégé wFST) comme représentation naturelle des différents composants d'un système de RAP [Mohri et al., 1996, Mohri et al., 2000, Mohri et al., 2002]. Les transducteurs peuvent être utilisés pour représenter les HMMs, les dictionnaires de prononciations ou encore les modèles de langage. Ainsi, en s'appuyant sur les propriétés et opérations pour wFST, les différentes représentations peuvent être combinées de manière flexible et efficace. En outre, les algorithmes de détermination et de minimisation pondérée permettent d'optimiser les besoins en temps et en espace, tandis que l'algorithme de pondération permet de distribuer les poids le long des chemins d'un transducteur de façon optimale pour la reconnaissance de la parole.

**La fin des années 2000 et le début des années 2010** voient l'apparition de nouvelles architectures pour la modélisation acoustique, comme le réseau à croyance profonde (*Deep Belief Network* en anglais) [Hinton et al., 2006, Mohamed et al., 2009] reposant sur un empilement de machines de Boltzmann restreintes [Salakhutdinov and Hinton, 2009], mais aussi l'avènement de l'apprentissage profond et des réseaux de neurones dans de nombreux domaines et notamment celui de la RAP. En 2012, [Hinton et al., 2012a] démocratisent plus largement l'utilisation des réseaux de neurones profonds (ou *Deep Neural Network*

en anglais, abrégé DNN) dans les systèmes de RAP, et mettent en évidence leur efficacité pour traiter le problème de modélisation acoustique comparés aux représentations précédentes. Il est ainsi démontré que les réseaux de neurones profonds sont particulièrement intéressants en remplacement ou en association avec les HMMs, détrônant du fait les GMMs à haut niveau. Ces derniers resteront essentiels durant de nombreuses années afin de générer les alignements nécessaires pour l'entraînement des réseaux de neurones profonds.

**En parallèle de ces avancées**, des travaux sont initiés sur l'usage des réseaux de neurones récurrents (ou *Recurrent Neural Networks* en anglais, abrégé RNN) pour la modélisation du langage [Mikolov et al., 2011, Mikolov and Zweig, 2012] mais aussi la modélisation acoustique [Graves et al., 2006, Graves, 2012a, Graves et al., 2013]. Si par nature ces réseaux de neurones apparaissent intéressants pour modéliser des problèmes temporels, ils demandent toutefois des données d'entraînement pré-segmentées et des prétraitements pour transformer les sorties en séquences d'étiquettes.

**En 2006**, A. Graves [Graves et al., 2006] propose la Connectionnist Temporal Classification (abrégé CTC) une nouvelle méthode d'entraînement des réseaux de neurones récurrents pour étiqueter directement des séquences non-segmentées. Remplaçant le triplet traditionnel modèle acoustique, dictionnaire phonétique et modèle de langage par un réseau faisant directement le lien entre une séquence de trames et une séquence de phonèmes.

**Au début des années 2010**, Povey et al. présentent une des contributions qui a permis l'accélération des travaux sur les réseaux de neurones et plus généralement les travaux appliqués pour la RAP. Le papier introduit et décrit la conception de Kaldi [Povey et al., 2011], une boite à outils pour la recherche en RAP basée sur l'utilisation des wFST. Cette boite à outils propose des scripts et outils pour construire des systèmes de RAP complets et une multitude de recettes sont proposées pour créer des systèmes de RAP compétitifs pour la majorité des corpus distribués. Les outils proposés couvrent toutes les parties d'un système de RAP traditionnel : de la modélisation du contexte phonétique à la modélisation linguistique, en passant par la modélisation acoustique, les transformations de l'espace de caractéristiques ou encore l'adaptation au contexte. Au moment de la soumission du papier, seuls les GMMs et subspace-GMMs (abrégé sGMM) étaient supportés pour la modélisation acoustique. À ce jour, une grande partie des avancées existantes pour la RAP traditionnelle est mise à disposition des chercheurs dans la boite à outils.

**En 2012 et 2013**, A. Graves étudie les cellules dites *long-short term memory* (abrégé LSTM) [Hochreiter and Schmidhuber, 1997] pour remplacer les cellules des RNNs [Graves, 2012a, Graves, 2013]. Il démontre en outre par ses applications à la reconnaissance de chiffres manuscrits et à la modélisation du langage, la capacité des cellules LSTM à modéliser un contexte plus important que les modèles statistiques ou probabilistes précédents.

**La même année, en 2013**, A. Graves revisite le problème de la transduction de séquence en RAP en proposant une extension de son précédent système pour la reconnaissance de phonèmes [Graves et al., 2013]. Il introduit ainsi un modèle appelé RNN-transducer permettant de remédier à la principale limite de la CTC : le fait que la fonction objective ne peut pas modéliser les inter-dépendances car une indépendance conditionnelle est supposée entre les prédictions à différents pas de temps. Pour résoudre ce problème, et en se basant sur ses travaux en modélisation linguistique, un réseau de neurones récurrent distinct est ajouté en complément, prédisant chaque étiquette étant données les précédentes, de manière analogue à un modèle de langage. Par l'ajout de ce réseau prenant en compte

les sorties des encodeurs et des décodeurs, le système peut modéliser conjointement les inter-dépendances entre les entrées et les sorties mais également les inter-dépendances à l'intérieur de la séquence d'étiquettes de sortie.

**En 2014** une nouvelle approche émerge pour la modélisation acoustique dans le problème de la RAP, utilisant les récemment introduites *machines de traduction neuronales* pour le problème de traduction automatique. Ces modèles sont décomposés tel qu'un premier réseau de neurones récurrents, appelé encodeur, encode une phrase source en un vecteur de taille fixe à partir duquel un second réseau de neurones récurrents, appelé décodeur, génère une traduction. Cette nouvelle approche formulée par Bahdanau [Bahdanau et al., 2014] se base sur le postulat que l'utilisation de vecteurs de taille fixe limiterait les performances pour les problèmes séquentiels ou temporels. Ainsi il propose l'introduction d'un mécanisme d'attention permettant au modèle de rechercher automatiquement les parties d'une phrase source qui sont nécessaires pour prédire les parties de la phrase cible.

**En 2015 et 2016,** Bahdanau et Chorowsky proposent une application des travaux de Bahdanau au problème de reconnaissance automatique de parole. L'architecture Encodeur-Décodeur et le mécanisme d'attention sont adaptés et introduits dans différents articles [Chorowski et al., 2015, Bahdanau et al., 2016]. Contrairement à la CTC, l'approche basée sur le mécanisme d'attention ne suppose pas une indépendance conditionnelle entre les prédictions à différents moments et ne marginalise pas au travers de tous les alignements. Ainsi, la distribution postérieure peut être directement calculée en choisissant un alignement souple entre chaque pas de sortie et chaque pas d'entrée.

**En 2016**, D. Povey introduit dans [Povey et al., 2016] une des méthodes les plus utilisées à ce jour pour les systèmes traditionnels reposant sur les HMMs. Basé sur des mécanismes intronisés par la CTC de A. Graves, cette méthode permet d'effectuer un apprentissage discriminatoire des séquences des modèles acoustiques utilisant des réseaux de neurones, sans avoir besoin d'un pré-apprentissage par entropie croisée au niveau de la trame. Dans cet article, la version sans treillis de l'approche par Information Mutuelle Maximale (ou *Maximum Mutual Information* en anglais, abrégé MMI) est utilisée : le lattice-free MMI. Des améliorations ont été apportées au fil du temps, notamment pour s'adapter à des architectures plus performantes ou incorporer de nouveaux mécanismes tels que l'attention [Manohar et al., 2018, Povey et al., 2018].

# Chapter 2

# Systèmes de RAP traditionnels

Dans cette section, je présente les constituants principaux d'un système de reconnaissance automatique de parole dit "traditionnel" tel que illustré par la figure 2.1. Ceux-ci sont au nombre de quatre et seront détaillés dans leur section respective, à savoir : le module de calcul des paramètres acoustiques dans la section 2.1, les modèles acoustiques dans la section 2.2 et pour finir les modèles linguistiques englobant le dictionnaire (ou modèle) phonétique et le modèle de langage dans la section 2.3. Ces éléments seront ensuite mis en correspondance dans la section 2.4 où le processus de décodage sera présenté.

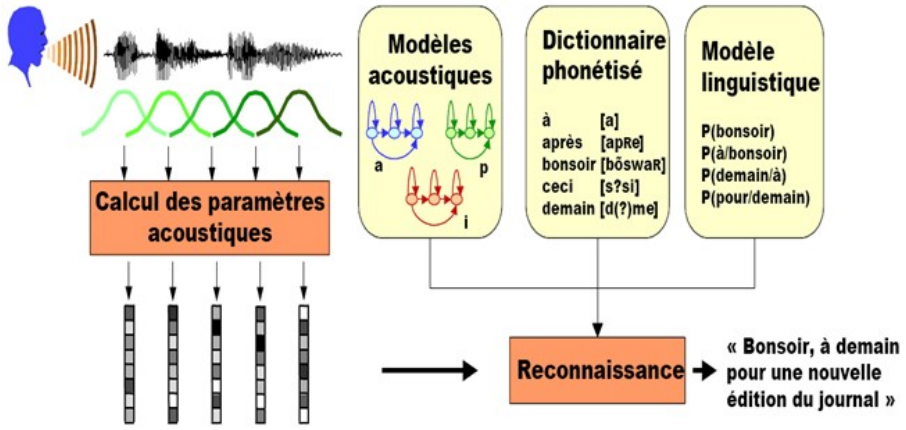



Figure 2.1: Descriptif des composants d'un système de RAP traditionel et du processus de reconnaissance associé.

Enfin, je termine ce chapitre par une introduction de différents éléments et techniques utilisés dans le cadre de la RAP moderne dans la section 2.5, avec notamment l'introduction des réseaux de neurones pour la modélisation acoustique et les transducteurs à états finis pondérés pour le processus de décodage.

## 2.1 Extraction des paramètres acoustiques

La première étape à effectuer est le calcul de l'ensemble des paramètres acoustiques $O$. Ces paramètres représentent les caractéristiques permettant d'identifier les composants

du signal audio portant des informations linguistiques, et de filtrer les informations jugées *non-pertinentes*.

Pour le problème de la RAP, ou de transcription, les paramètres acoustiques de type MFCC (pour *Mel-Frequency Cepstral Coefficients* en anglais) sont privilégiés depuis leur introduction au milieu et à la fin des années 1970 [Mermelstein, 1976, Davis and Mermelstein, 1980].

Pour un signal de parole $\boldsymbol{s}$ en entrée, les paramètres MFCCs sont calculés via les étapes suivantes :

1. Le signal $\boldsymbol{s}$ est tout d'abord passé à un filtre passe-haut afin de relever les hautes fréquences.
2. Le signal résultant est ensuite segmenté en trames de ∼13-20ms avec un chevauchement possible des trames de $\sim \frac{1}{2}$ à $\frac{1}{3}$ de la taille de la trame.
3. Chaque trame est multipliée par une fenêtre de Hamming tel que, considérant le signal d'une trame $s(n)$, le signal résultant d'un fenêtrage d'Hamming $w(n)$ est : $s(n)\, w(n)$. Ce fenêtrage est effectué afin de conserver la continuité entre les premiers et derniers points de la trame lors de la transformée de Fourier.
4. La transformée de Fourier (rapide) est ensuite appliquée sur chaque trame de manière à obtenir les réponses fréquentielles de chacune, les informations sur le timbre se trouvant dans la distribution de l'énergie suivant les fréquences.
5. Les enveloppes des réponses fréquentielles sont ensuite extraites en multipliant les réponses fréquentielles par 20 filtres triangulaires alignés sur l'échelle Mel, échelle psycho-acoustique proposant un écart de fréquence nécessaire pour distinguer deux sons, à chaque fréquence. Nous obtenons donc en sortie la log-energie pour chaque filtre triangulaire. La conversion d'une fréquence de Hertz en mels s'effectue par la formule : $1127 * \ln(1 + \frac{f}{700})$
6. Un passage au domaine fréquentiel est effectué afin d'obtenir les coefficients mel-cepstre, similaire au cepstre, qui nous serviront de paramètres acoustiques. Pour cela, une Transformée en Cosinus Discret (TCD) est appliquée tel que, considérant les log-énergie $E_n$ obtenues précédemment, $N$ le nombre de filtres triangulaires et $L$ le nombre de coefficients mel-spectre en sortie (usuellement 12), la formule de la TCD est :

$$\text{coeff}_L = \sum_{n=1}^{N} \cos[m * (n - 0,5) * \frac{\pi}{N}] * E_n, \quad m = 1, ..., L \tag{2.1}$$

7. L'énergie à l'intérieur de la trame étant facile à acquérir et étant aussi une caractéristique importante, le log-énergie de la trame du signal avant le fenêtrage est ajouté comme treizième paramètre acoustique.

Nous obtenons en sortie un vecteur de 13 dimensions. Dans un souci d'améliorer la reconnaissance automatique, nous incluons en plus les informations relatives à la dynamique du spectre de puissance, c'est-à-dire la trajectoire des MFCC à travers le temps. Ainsi, le nombre de dimensions est étendu à 39, en incluant les coefficients delta (différentiel) et delta-delta (accélération) correspondant à la première et seconde dérivée. Pour certaines langues, comme le japonais, la fréquence fondamentale est estimée et extraite du signal de la parole pour être ajoutée en tant que quarantième dimension.

Différentes caractéristiques peuvent être utilisées en place et lieux des MFCCs comme par exemple : Perceptual Linear Prediction (PLP) utilisant l'échelle Bark [Huang et al., 2001], Linear Prediction Cepstral Coefficient (LPCC) obtenu à partir des spectres LPC [Hariharan et al., 2001], ou encore Power-Normalized Cepstral Coefficients [Kim and Stern, 2016] qui utilise, entre autre, des filtres gammatones. Dans le cadre de cette thèse, les MFCCs et leur équivalent ont été privilégies. Ces derniers étant les banques de filtres mel obtenues après l'application de la fonction log dans le processus décrit précédemment.

## 2.2 Modélisation acoustique

La modélisation acoustique, effectuée par un modèle acoustique, est l'élément principal de la reconnaissance automatique de parole, effectuant la correspondance entre la séquence d'observations acoustiques $O$ et une séquence d'unités élémentaires.

Comme énoncé dans le chapitre précédent, dans le cadre de la RAP traditionnelle, le phonème est l'unité préférée. Cette unité est particulière intéressante pour plusieurs raisons. Premièrement, l'ensemble de phonèmes $Q$ permettant de représenter n'importe quel ensemble de mots $W$ d'une même langue, on peut construire une modélisation acoustique qui est invariante en considération de l'évolution d'une langue et donc l'apparition de nouveaux mots, de néologismes ou encore d'anglicismes. Deuxièmement, une séquence de phonèmes permettant de représenter un mot mais aussi l'ensemble de ses homophones (p. ex., "manger", "mangeait" et "mangé"), ce qui résulte en une représentation plus compacte et un coût en terme de manipulation bien moindre que si on modélisait des caractères ou directement des mots. Finalement, l'utilisation d'une représentation intermédiaire telle que le phonème permet de considérer des informations liés à la prononciation et la coarticulation, rendant la modélisation acoustique plus robuste en considération de changements de locuteurs.

L'estimation des paramètres du modèle acoustique est donc effectuée sur la prédiction des séquences de phonèmes possibles pour chaque séquence de mots connus. En concaténant ensuite les modèles de phonèmes pour une séquence de mots donnée, en considérant bien sur les prononciations possibles pour chaque mot, nous pouvons ainsi construire un modèle acoustique probabiliste pour cette séquence de mots. Pour ce faire, la modélisation acoustique utilisant des modèles de Markov cachés [Baum and Petrie, 1966, Baum and Eagon, 1967, Baum and Sell, 1968, Baum et al., 1970] en association avec des modèles de mélanges gaussiens est privilégiée.

### 2.2.1 Modèle de Markov caché

Un modèle de Markov caché (ou *Hidden Markov Models* en anglais, abrégé HMM) est un modèle statistique permettant de modéliser la temporalité des observations de $O$ pour le problème de la RAP. Le système modélisé est constitué d'un ensemble d'états $(s_t, t)$ et d'un ensemble d'observations $(o_t, t)$ définis selon deux hypothèses :

$$p(s_t|s_1, ..., s_{t-1}) = p(s_t|s_{t-1}) \tag{2.2}$$

$$p(o_t|o_1, ..., o_{t-1}, s_1, ..., s_{t-1}) = p(o_t|s_t) \tag{2.3}$$

La première équation (aussi appelé hypothèse markovienne) définit qu'une transition vers un état suivant $s_{t+1}$ ne dépend que de $s_t$, ou en d'autres termes qu'un état $s_t$ est condi-

tionné seulement par son état précédent $s_{t-1}$. La seconde équation définit quant à elle l'indépendance conditionnelle des observations, tel que l'observation $o_t$ dépend seulement de l'état $s_t$..

Plus formellement, un modèle de Markov caché $\mathcal{H}$ est défini par un 4-tuplet $(E, \pi, A, B)$ où :

- $E = \{e_1, ..., e_N\}$ désigne l'ensemble des états, où un état au temps $t$ est noté par $s_t \in E$.
- $\pi = \{\pi_1, ..., \pi_N\}$ est la matrice d'initialisation des probabilités dans $E$.
- $A = \{a_{ij}\}_{1 \leq i,j \leq N}$ est la matrice des probabilités de transitions entre états, où $a_{ij} = p(e_j|e_i)$
- $B = \{b_j(e_k)\}_{1 \leq j \leq N, 1 \leq k \leq M}$ est la matrice des probabilités d'observation dans les états, où $b_j(k) = p(e_k|e_j)$.

Une illustration d'un HMM à 5 états pour la RAP est donnée par la figure 2.2. Ici, les transitions sont uniquement autorisées dans le sens temporel causal (c.-à.d, de gauche à droite) afin de dénoter de la séquentialité du discours. Parallèlement, des boucles sont aussi admises sur chaque état afin de dénoter de la temporalité des phénomènes acoustiques, ou plus simplement leur durée.

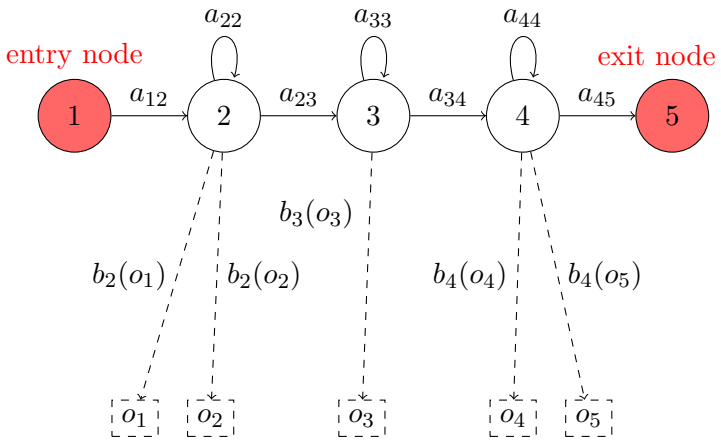



Figure 2.2: Exemple de HMM utilisé pour la reconnaissance de phonèmes.

Typiquement, il y a trois questions que l'on peut poser avec un HMM et qui peuvent être aussi posées pour la RAP. Pour une séquence d'observations $O$ et un modèle HMM $\mathcal{H}$, celles-ci sont les suivantes :

- Quelle est la probabilité $p(O|\mathcal{H})$ que la séquence d'observations $O$ ait été générée par le modèle $\mathcal{H}$?
- Quelle est la séquence d'états $e_1, ..., e_n$ la plus probable ayant généré la séquence $O$?
- Comment optimiser les paramètres du modèle $\mathcal{H}$ afin de maximiser la probabilité d'observation $O$?

Dans le cadre de la RAP, la première question correspond au problème d'évaluation et se résout au travers de l'algorithme *forward* [Rabiner and Juang, 1986], où la réponse unique correspond à la somme des probabilités de voir apparaître la séquence pour chacune des

séquences d'états possibles. La seconde question équivaut ici au problème de décodage et la réponse est obtenue en se servant de l'algorithme de Viterbi que je décris dans la section 2.4. Enfin, la résolution de la troisième question correspond ici à la phase d'apprentissage du modèle et s'effectue via l'algorithme de Baum-Welch, issu des travaux de Baum et Welch [Welch, 2003], permettant d'approcher une solution optimale. Celui-ci sera discuté dans la section 2.2.2.1

### 2.2.2 Modèle de mélange gaussien

Les observations étant de nature continue, il apparaît évident d'utiliser la distribution pour définir la probabilité d'observation de $o_t$ dans un état $s_j$, (c.-à.d., $b_j(o_t)$). Toutefois, en pratique, il est d'usage de considérer qu'une distribution gaussienne n'est pas suffisamment précise et représentative afin de modéliser la probabilité des observations de tout les états d'un HMM. À cela, un modèle de mélange de loi gaussienne à $M$ composantes est préféré, celui-ci ayant la forme :

$$b_j(o_t) = \sum_{m=1}^{M} c_j^m \mathcal{N}(o_t; \mu_j^m, \Sigma_j^m) \tag{2.4}$$

Où $o_t$ est une observation pour le temps $t$. $\mu_j^m$ et $\Sigma_j^m$ sont les paramètres de la composante $m$ de l'état $j$, respectivement le vecteur moyen et la matrice de covariance. Finalement, $c_j^m$ est ici le poids de mélange correspondant à un coefficient positif, où $\sum_{m=1}^{M} c_j^m = 1$.

La figure suivante 2.3 donne un exemple de modèle de mélange gaussien à $M = 2$ composantes.

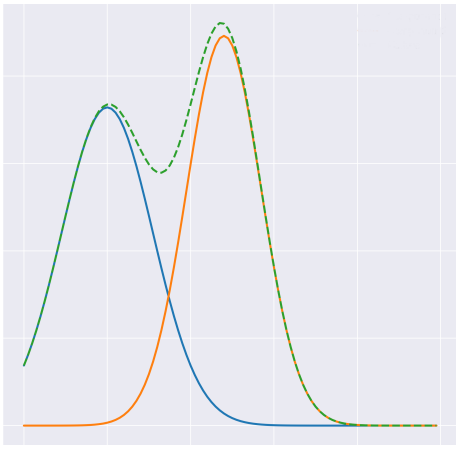


Figure 2.3: Exemple de mélange gaussien 1D (en pointillé) et ses deux composantes (en bleu et vert). Extrait de *Gaussian Mixture Models in PyTorch*[1]

#### 2.2.2.1 Apprentissage du modèle acoustique

L'étape d'apprentissage correspond à la résolution de la troisième question introduite dans la section 2.2.1, à savoir: "Comment optimiser les paramètres du modèle $\mathcal{H}$ afin de

[1] `https://angusturner.github.io/generative_models/2017/11/03/pytorch-gaussian-mixture-model.html`

maximiser la probabilité d'observation $O$". En d'autres termes, pour une séquence $O$ et un nombre d'états E fixe, nous cherchons à estimer les paramètres optimaux du modèle $\mathcal{H} = (\pi, A, B)$ avec $\text{argmax}_{\mathcal{H}} P(O|\mathcal{H})$ quand le chemin d'états est inconnu.

Dans ce cadre, et comme stipulé dans cette même section, l'algorithme le plus couramment utilisé est l'algorithme Baum-Welch qui est basé sur les travaux de L. E. Baum introduit précédemment et ceux de L. Welch [Welch, 2003]. Cet algorithme est une généralisation de l'algorithme d'Espérance-Maximisation [Bilmes, 1997] pour les modèles HMM, et bien qu'il n'existe pas de méthode idéale pour estimer les paramètres du modèle $\mathcal{H}$ à partir d'un nombre limité de séquences observées, celui-ci permet de le faire de manière localement optimale.

$$\gamma_i^d(x) = \frac{\sum_{t|o_t^d=x} \alpha_i^d(t)\beta_i^d(t)}{p(o^d|\mathcal{H})} \tag{2.5}$$

$$\xi_{ij}^d = \frac{\sum_t \alpha_i^d(t)a_{ij}b_i(o_{t+1}^d)\beta_i^d(t+1)}{p(o^d|\mathcal{H})} \tag{2.6}$$

$$\tag{2.7}$$

Étant donné un nombre fixe d'états et un jeu de données constitué de $D$ séquences d'observations où une séquence unique est dénotée par $O^d = (o_1^d, ..., o_t^d)$, l'entraînement du modèle peut être résumé par les étapes suivantes :

1. Les valeurs des paramètres de $\mathcal{H}$ sont tout d'abord initialisées de manière aléatoire ou en utilisant des hypothèses préalables.
2. Une variable, que l'on va nommer *score*, est définie comme étant : score $= \sum_d p(o^d|\mathcal{H})$.
3. Pour chaque séquence d'observations $o^d$, on effectue les étapes suivantes :
   - Le chemin d'états le plus probable $(s_1^d, s_2^d, ...)$ est tout d'abord déterminé.
   - On calcule ensuite les variables *forward* $\alpha^d$ pour $o^d$ en utilisant l'algorithme *forward*. Où, $\alpha_i^d(t)$ désigne la probabilité que le modèle est observé $(o_1^d, o_2^d, ...)$ et se trouve dans l'état $i$ à l'instant $t$.
   - De manière similaire, on calcule aussi les variables *backward* $\beta$ pour $o^d$ en utilisant l'algorithme *backward*. Où $\beta_j^d(t+1)$ dénote de la probabilité d'observer le reste de la séquence si on se trouve dans l'état $j$ à l'instant $t+1$.
   - Enfin, on calcule les variables temporaires $\gamma^d(x)$ et $\xi^d$ en utilisant les équations 2.5 et 2.6 dénotant respectivement de, étant donné une séquence observé $o^d$, la probabilité d'être dans l'état $i$ à l'instant $x$ et la probabilité d'être dans un état $i$ à l'instant $t$ et de faire une transition vers un état $j$ à l'état $t+1$.

4. Les paramètres du modèle $\mathcal{H}$ sont mis à jour avec:

$$\pi_i = \frac{\sum_{d=1}^{D} \gamma_i^d(1)}{D} \tag{2.8}$$

$$a_{ij} = \frac{\sum_{d=1}^{D} \sum_{t=1}^{T} \xi_{ij}^d(t)}{\sum_{d=1}^{D} \sum_{t=1}^{T} \gamma_j^d(t)} \tag{2.9}$$

$$b_i(o_k) = \frac{\sum_{d=1}^{D} \sum_{t=1}^{T} 1(o_t = k)\gamma_j^d(t)}{\sum_{d=1}^{D} \sum_{t=1}^{T} \gamma_j^d(t)} \tag{2.10}$$

5. Un nouveau score est calculé avec les nouveaux paramètres. Si celui-ci ne satisfait pas une condition d'arrêt particulière, on réitère les étapes 3 et 4 jusqu'à ce que la condition soit satisfaite. En général, cette condition correspond, au plus simple, à comparer la différence des scores avec une valeur de seuil déterminée en amont.

Pour une description complète, notamment de l'algorithme *forward-backward* et des calculs relatifs au modèle de mélange gaussien, je dirige le lecteur vers les références données précédemment ainsi que [Gales and Steve, 2008].

De nos jours, la procédure s'accompagne de nombreuses techniques permettant de traiter des problèmes mis en évidence depuis son apparition, d'optimiser la sélection de certains paramètres ou encore de transformer (et adapter) l'espace de caractéristiques et l'espace acoustique. Parmi les problèmes mis en évidence, nous pouvons citer par exemple le coût de l'estimation de la matrice de covariance $\Sigma_j$ (et son inverse) lorsque cette dernière est de grande dimension ou encore la sélection d'un grand nombre de composantes $M$, qui peuvent tous les deux engendrer des problèmes statistiques [Gales and Steve, 2008] notamment liés au problème du "fléau de la dimension" [Bellman, 1957]. Dans le cadre de ce dernier, la sélection d'un grand nombre de composantes $M$ par états est d'ailleurs jugée importante afin de modéliser correctement la variabilité acoustique de la parole, mais cela peut s'accompagner d'un coût important pour certaines opérations et la définition d'un nombre adéquat de composantes peut s'avérer difficile.

Pour la résolution du premier problème, nous pouvons citer l'utilisation de matrices de covariance semi-fixes factorisées [Gales, 2001]. Parallèlement, pour le premier et troisième, nous pouvons citer l'utilisation de *Subspace Gaussian Mixture Model* [Povey et al., 2010] permettant d'obtenir une représentation beaucoup plus compacte, ou encore différentes techniques permettant de réduire le nombre de dimensions [Hu and Zahorian, 2010]. Concernant le second problème de sélection de paramètres, et notamment le nombre de composantes par état, différentes techniques ont vu le jour, basées sur différents critères [Claeskens and Hjort, 2008, Konishi and Kitagawa, 2008, Giraud, 2015]. Ce problème est toutefois à minimiser dans le sens où la démocratisation de boites à outils telles que Kaldi [Povey et al., 2011] facilite la sélection de paramètres adéquats. La raison étant qu'un nombre non-négligeable de recettes (c.-à.-d., un ensemble de scripts et configurations) existent maintenant pour créer des modèles acoustiques HMM-GMM robustes et ce, selon un grand nombre de critères (tel que la langue, le volume de données d'entraînement, le type de parole, l'environnement acoustique, ...).

Concernant les techniques de transformation et d'adaptation de l'espace acoustique (et de l'espace de caractéristiques), ou encore d'amélioration de la qualité du modèle vis à vis de différents critères, je dirige le lecteur vers la section 2.5. Dans celle-ci, un ensemble de techniques seront abordées dans le cadre de la RAP moderne basée Modèles de Markov cachés.

## 2.3 Modélisation linguistique

Dans le cas de la reconnaissance de parole traditionnelle, et de par la nature des unités acoustiques utilisées, la modélisation linguistique est composée de deux modèles : le modèle de prononciation, liant la modélisation acoustique à la représentation orthographique, et le modèle de langage, qui valide les ensembles de mots en considération des règles syntaxiques et orthographiques d'une langue cible.

### 2.3.1 Modèle phonétique

Le lien entre le modèle acoustique et le modèle de langage est effectué par un module lexical transformant la séquence de phonèmes en sortie du modèle acoustique en une séquence de mots dont la représentation phonétique associée est connue, c'est-à-dire présente dans la modélisation phonétique. Ces modèles sont le plus souvent sous la forme d'un dictionnaire phonétique composé d'entrées lexicales auxquelles sont associées leurs descriptions phonétiques données en terme d'unités acoustiques (voir figure 2.4).

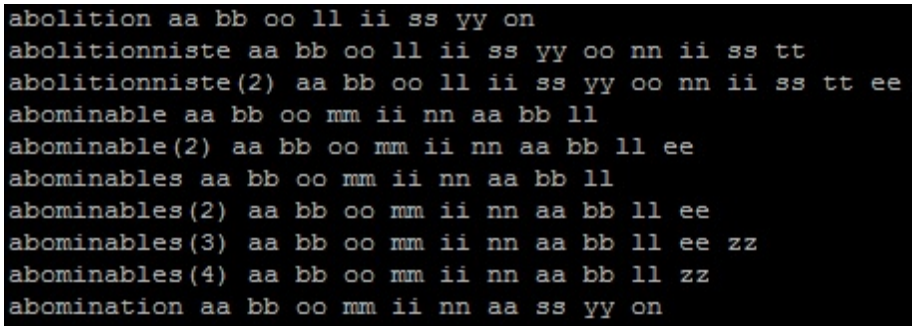



Figure 2.4: Exemple de dictionnaire phonétique pour le français dans Kaldi [Povey et al., 2011]. Le suffixe "$(x)$" désigne ici une prononciation alternative pour le mot préfixe.

Outre les différentes descriptions phonétiques que nous pouvons associer à une même entrée pour dénoter des variations d'élocution, différentes informations peuvent être renseignées afin d'améliorer les performances du système en considération de la langue. En français par exemple, il est utile de dénoter des liaisons entre les mots en ajoutant une entrée lexicale pour un groupe de mots avec liaison (p. ex., "ils_ont") ou en définissant une prononciation alternative, où l'unité acoustique de liaison est ajoutée à la description phonétique du mot (p. ex., le [zz] de "abominables"). En outre, il peut être aussi utile de modéliser les composés lexicaux détachés ou expressions ambiguës par une entrée lexicale commune.

### 2.3.2 Modèle de langage

Un modèle de langage permet de représenter des connaissances linguistiques afin d'amener le décodage vers des hypothèses de phrases cohérentes d'un point de vue syntaxique et grammatical. L'objectif du modèle est de définir les enchaînements de termes, ou mots, possibles dans une langue et réciproquement de prédire l'apparition d'un mot $w$ à une position $i$ dans une phrase.

Deux principales approches existent afin d'effectuer cette modélisation: l'approche utilisant un modèle basé sur l'utilisation de grammaires formelles établies habituellement par des linguistes, et une autre utilisant un modèle statistique construit sur la base d'un

corpus textuel pour produire une description automatique. Dans le cas de la reconnaissance automatique de parole, les modèles statistiques sont privilégiés, en particulier les modélisations basées sur les séquences *N-grammes*, une séquence continue de $N$ éléments représentant un échantillon de texte, ou de parole.

L'idée principale derrière ces dernières modélisations peut être abrégée de la manière suivante : Étant donnée une séquence continue d'éléments, il est possible d'obtenir la fonction de vraisemblance de l'apparition de l'élément suivant. En se servant d'un corpus d'apprentissage quelconque, nous pouvons construire une distribution de probabilité sur les séquences d'éléments observées.

Pour les modèles N-grammes, et dans le cadre des systèmes de RAP traditionnels, les éléments sont généralement des mots et l'obtention de la probabilité du mot $P(w)$ est simplifiée de manière à ce que, pour une séquence de $N$ mots, la probabilité d'apparition d'un mot $w$ à une position $i$ dans une séquence ne dépend que des $N-1$ mots précédents, c'est-à-dire aux positions $i-(N-1)$ à $i-1$. Par exemple, pour la phrase "La reconnaissance de parole est en pleine évolution" et $N=2$, la probabilité du mot "évolution" est calculé par approximation avec le terme "pleine" au lieu de la séquence de mots complète précédente: $P(\text{évolution}|\text{pleine})$.

Plus simplement, une approximation de l'histoire d'un mot est donné en considération avec un contexte restreint à $N$ mots. Le choix de la valeur de $N$ étant fait en considération de la tâche, de la langue, ou encore d'un rapport coût-performance.

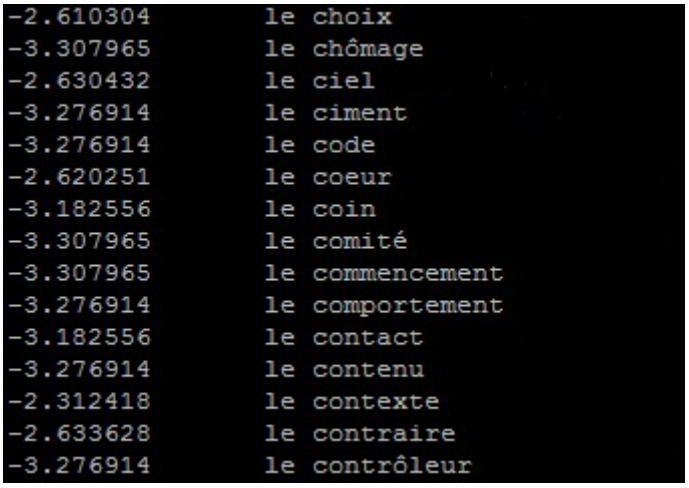



Figure 2.5: Extrait d'un modèle de langage bi-gramme entrainé avec la boite à outils SRILM [Stolcke et al., 2011]. La valeur de gauche étant la log-probabilité associée à ce bi-gramme.

De par l'hypothèse formulée, la probabilité d'un mot dépendant des termes précédents, correspondant à l'hypothèse markovienne, un modèle N-grammes correspond à un modèle de Markov d'ordre $N$. Ainsi, l'approximation de la probabilité conditionnelle du mot $w_i$ dans une séquence de mots se définit par l'équation suivante :

$$P(w_i|w_{1:i-1}) \approx P(w_i|w_{i-(n-1):i-1}) \tag{2.11}$$

Pour obtenir les probabilités de ces n-grammes, un maximum de vraisemblance peut

être utilisé afin d'estimer les paramètres du modèle. Pour cela, étant donné un corpus d'entraînement et un vocabulaire $V$ défini au préalable, les fréquences de chaque terme du corpus appartenant à $V$, seul ou en considération des $N-1$ mots précédents, sont extraites et normalisées par le nombre total de termes afin de générer des valeurs comprises entre 0 et 1. La probabilité d'un N-gramme est ainsi obtenue en divisant la fréquence observée d'une séquence particulière par la fréquence observée d'un préfixe. Celle-ci est définie par le ratio suivant, nommé fréquence relative :

$$P(w_i|w_{i-(n-1):i-1}) = \frac{\text{count}(w_{i-(n-1):i-1}w_i)}{\text{count}(w_{i-(n-1):i-1})} \tag{2.12}$$

En pratique, les probabilités du modèle de langage sont définies sous forme logarithmique (Cf. Fig. 2.5) afin d'éviter un soupassement arithmétique dû à la multiplication d'un grand nombre de faibles probabilités. L'addition dans l'espace logarithmique étant équivalente à la multiplication dans l'espace linéaire, les probabilités logarithmiques sont donc combinées par addition.

Deux problématiques restent en outre à résoudre afin de rendre le modèle utilisable avec de nouvelles données, à savoir : 1) traiter la probabilité des nouveaux termes non-observés lors de l'entraînement, et 2) éviter qu'une probabilité nulle soit assignée à des termes dont le contexte n'aurait jamais été observé.

Afin de résoudre le premier problème, plus connu sous le nom du problème de mot hors-vocabulaire, un pseudo-mot $\langle\text{UNK}\rangle$ peut être défini afin d'entraîner les probabilités des mots hors-vocabulaire. Pour cela, étant donné un vocabulaire $V$, l'ensemble des termes du corpus d'entraînement n'appartenant pas à $V$ peut être remplacé par le pseudo-mot afin que la probabilité d'un mot hors-vocabulaire soit estimé de la même manière que les mots de $V$. Le vocabulaire $V$ peut être défini manuellement ou en se basant sur les fréquences d'apparition de chaque mot dans le corpus d'entraînement.

Pour le second problème, une modification de l'algorithme d'entraînement est habituellement utilisée, nommée "lissage", et consistant à réduire la masse de probabilité de certains événements fréquents afin de l'assigner aux évènements non-observés. Différentes approches existent afin de réaliser ce lissage, la plus communément utilisée de nos jours étant une amélioration de la méthode de lissage Kneser-Ney [Ney et al., 1994] proposée par Chen et Goodman en 1998 [Chen and Goodman, 1998].

La méthode Kneser-Ney considère que la probabilité d'un unigramme ne doit pas être proportionnelle au nombre d'occurrences des mots, mais au nombre de mots différents auquel il succède. Pour ce faire, le concept d'interpolation à "actualisation absolue" est utilisé, où une valeur fixe $\delta$ est soustraite des probabilités des termes d'ordre inférieur afin d'omettre les N-grammes ayant des fréquences plus faibles, tel que pour un modèle d'ordre $N$:

$$\begin{aligned} P_{\text{KN}}(w_i|w_{i-(n-1):i-1}) &= \frac{\max(\text{count}(w_{i-(n-1):i}) - \delta, 0)}{\text{count}(w_{i-(n-1):i-1})} \\ &+ \frac{\delta}{\text{count}(w_{i-(n-1):i-1})} N_{1+}(w_{i-(n-1):i-1})\, P_{\text{KN}}(w_i|w_{i-(n-2):i-1}) \end{aligned} \tag{2.13}$$

Où $N_{1+}(w_{i-(n-1):i-1})$ désigne le nombre de termes uniques qui suivent au moins une fois le contexte désigné tel que, ici :

$$N_{1+}(w_{i-(n-1):i-1}) = |\{w_i : \text{count}(w_{i-(n-1):i}, w_i) > 0\}| \tag{2.14}$$

Partant du postulat que l'actualisation moyenne idéale pour les N-grammes observés moins de trois fois est différente de l'actualisation moyenne idéale pour les N-grammes avec des fréquences plus élevées, Chen and Goodman proposent de remplacer le paramètre $\delta$ par trois paramètres $\delta_1$, $\delta_2$ et $\delta_3$ appliqués respectivement sur les N-grammes de une, deux ou trois et plus occurrences. Ainsi, l'équation 2.13 est réécrite de la manière suivante :

$$P_{\text{CG}}(w_i|w_{i-(n-1):i-1}) = \frac{\text{count}(w_{i-(n-1):i}) - \delta(\text{count}(w_{i-(n-1):i}))}{\text{count}(w_{i-(n-1):i-1})} \tag{2.15}$$

$$+ \Delta(w_{i-(n-1):i-1}) P_{\text{CG}}(w_i|w_{i-(n-2):i-1}) \tag{2.16}$$

Où $\delta(\bullet)$ et $\Delta(w_{i-(n-1):i-1})$ sont respectivement définis par les équations suivantes, avec $N_x$ définit comme précédemment :

$$\delta(\bullet) = \begin{cases} \delta_0 = 0 & \text{if } \bullet = 0 \\ \delta_1 = 1 - 2\frac{N_2}{N_1}\frac{N_1}{N_1+2N_2} & \text{if } \bullet = 1 \\ \delta_2 = 2 - 3\frac{N_3}{N_2}\frac{N_1}{N_1+2N_2} & \text{if } \bullet = 2 \\ \delta_3 = 3 - 4\frac{N_4}{N_3}\frac{N_1}{N_1+2N_2} & \text{if } \bullet \geq 3 \end{cases} \tag{2.17}$$

$$\Delta(w_{i-(n-1):i-1}) = \frac{\delta_1 N_1(w_{i-(n-1):i-1}\ \delta_2 N_2(w_{i-(n-1):i-1}\ \delta_3 N_3(w_{i-(n-1):i-1})}{\text{count}(w_{i-(n-1):i-1})} \tag{2.18}$$

En pratique, les modèles N-grammes d'ordre 3 et 4 sont privilégiés en considération du rapport temps - performance à atteindre. Bien que couramment utilisés de nos jours, ces modèles possèdent toutefois une limitation importante du fait que les liens entre les termes sont exclusivement dépendants de leur position dans la phrase. Ainsi, différentes modélisations ont depuis vu le jour, se proposant de palier cette limitation. Depuis le début des années 2010, l'alternative la plus courante en reconnaissance automatique de parole est la modélisation basée sur l'utilisation de réseaux de neurones récurrents [Mikolov et al., 2011, Mikolov and Zweig, 2012, Mikolov et al., 2013] qui sera abordé en détail dans le chapitre suivant (3).

## 2.4 Processus de décodage

Le processus de décodage de la parole, aussi communément appelé processus de reconnaissance de la parole, correspond à la résolution de la deuxième question évoquée dans la section 2.2.1, à savoir : *étant donné un modèle HMM $\mathcal{H}$ et une séquence d'observations $O$, quelle est la séquence d'états $q_1, ..., q_t$ la plus probable ayant généré la séquence $O$?*. Ainsi, pour une séquence d'observations acoustiques $O$, le processus va chercher à déterminer la séquence de mots $\hat{W}$ la plus probable telle que :

$$\hat{W} = \arg\max_W P(W|O) \cong \arg\max_W P(O|W)P(W)^l \tag{2.19}$$

Où $P(W|O)$ et $P(W)$ correspondent aux probabilités données par, respectivement, le modèle acoustique et le modèle de langage. Du fait d'une différence d'ordre de grandeur entre les deux probabilités, la probabilité linguistique est, en pratique, pondérée par un poids $l$ afin de ne pas négliger la contribution d'un des modèles.

Ici, le processus de reconnaissance doit donc trouver la meilleure séquence d'états de phonèmes qui peut générer la suite d'observations $O = (o_1, ..., o_T)$, extraites du signal audio, correspondant à la prononciation d'un énoncé et permettant de construire une séquence au niveau des mots. Pour cela, les HMMs des phonèmes sont concaténés afin former les modèles HMM des mots, par le biais des informations phonétiques issues du lexique.

Afin d'obtenir la séquence de mots la plus probable, l'algorithme de Viterbi [Viterbi, 1967, Rabiner, 1991] est utilisé récursivement en implémentant une matrice contenant les valeurs $\delta_t(i)$ définissant la vraisemblance du meilleur chemin finissant à l'état $i$ à l'instant $t$:

$$\delta_t(i) = \max_{s_{1:t-1}} P(s_1, ..., s_t = i, o_1, ..., o_t|\mathcal{H}) \tag{2.20}$$

Où $\mathcal{H}$ désigne notre modèle entraîné et $s_t$ un état à l'instant $t$. Une seconde matrice $\psi$ est aussi définie en pratique afin de mémoriser l'état qui a donné le maximum de vraisemblance à chaque pas $t$.

La séquence discrète d'états $S = (s_1, s_2, ...)$, où un état au temps $t$ est désigné par $e_t \in S$, ayant la plus grande vraisemblance $\delta_T$ est obtenue par récursion via la procédure décrite dans Algo. 1. Les données suivantes sont utilisées :

- La matrice de probabilités initiales $\pi = (\pi_1, ..., \pi_N)$ où $\pi_i = P(e_0 = i)$
- $A = \{a_{ij}\}_{1 \leq i,j \leq N}$ est la matrice des probabilités de transitions entre états, où $a_{ij} = p(e_j|e_i)$
- $B = \{b_j(e_k)\}_{1 \leq j \leq N, 1 \leq k \leq M}$ est la matrice des probabilités d'observation dans les états, où $b_j(k) = p(e_k|e_j)$.

**Algorithme 1 :** Algorithme de Viterbi

**Initialisation:**

$\delta_1(i) = \pi_i b_i(o_1)$, où $1 \leq i \leq N$

$\psi_1(i) = 0$

**Récursion:**

$\delta_t(j) = \max_{1,\ldots,N}[\delta_{t-1}(i)a_{ij}]b_j(o_t)$

$\psi_t(j) = \operatorname{argmax}_{1,\ldots,N}[\delta_{t-1}(i)a_{ij}]$

où $2 \leq t \leq T$ et $1 \leq j \leq N$

**Terminaison:**

$P = \max_{1,\ldots,N}[\delta_T(i)]$

$q_T = \operatorname{argmax}_{1,\ldots,N}[\delta_T(i)]$

**Reconstruction de la séquence d'états :**

$q_t^* = \psi_{t+1}(q_{t+1}^*)$

obtenu par marche arrière de $T-1$ à $1$

L'algorithme retourne la séquence de mots ayant la plus grande vraisemblance $\delta_t$, toutefois, une liste de $N$ meilleurs résultats peut être générée, contenant un sous-ensemble de séquences de mots suivant les meilleurs scores de vraisemblance. Pour ce faire, plusieurs passes de re-estimation des scores sont effectuées sur la base d'une première passe contraignant l'espace de recherche via une modélisation restreinte.

## 2.5 Reconnaissance automatique de parole *moderne*

Dans ce chapitre, je me suis concentré sur l'introduction des différents composants d'un système de RAP traditionnel ainsi que leur fonctionnement. Toutefois, dans le cadre des systèmes modernes, des notions importantes sont encore à introduire. Ainsi, nous nous attardons dans cette section à introduire les points suivants : 1) la contextualisation des phonèmes et la création du modèle *triphone*, 2) le décodage avec des transducteurs à états finis et 3) l'utilisation de réseaux de neurones pour la reconnaissance de parole *hybride*.

### 2.5.1 Modèle *triphone* et dépendance au contexte

Dans la section 2.2, nous présentions les deux composants d'un modèle acoustique, à savoir le modèle de Markov caché et le modèle de mélanges gaussiens, servant à modéliser nos séquences de phonèmes. Cependant, nous n'avons pas expliqué que les phonèmes ne sont pas homogènes et qu'entre le début et la fin, les amplitudes des fréquences changent. Pour refléter cela, chaque phonème initial est en pratique représenté par un modèle HMM à trois états dénotant le début, le milieu et la fin du phonème, comme nous pouvons le voir sur la figure 2.6. Dans le cas des phonèmes utilisés pour représenter les silences, bruits ou pauses, ceux-ci sont, en général, représentés par cinq états plutôt que trois du fait de la difficulté de les capturer en comparaison des phonèmes issus d'une langue. C'est en général le système inital HMM-GMM à entraîner et qui est communément appelé modèle *monophone*.

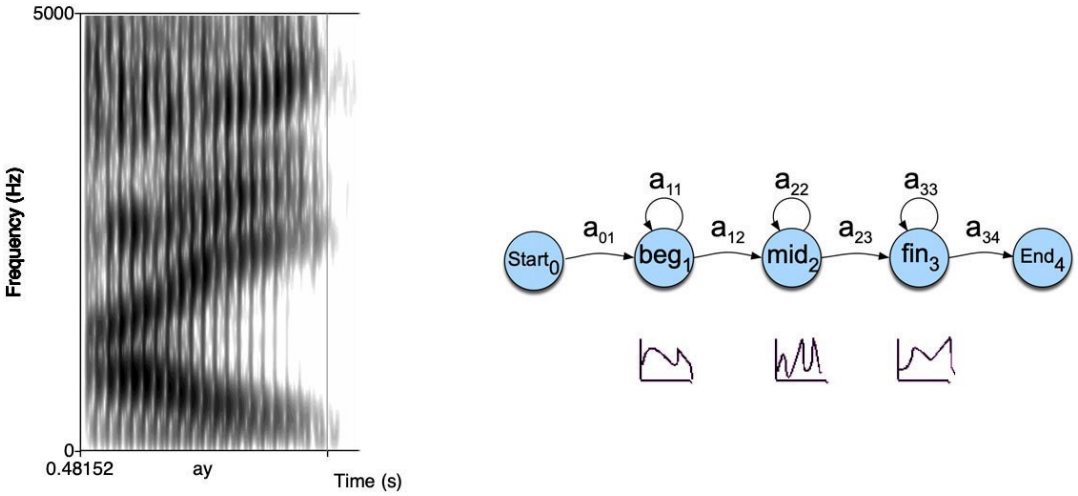



Figure 2.6: Exemple d'un modèle de Markov caché pour un phonème à trois états. *Extrait du cours "CS224S: Spoken Language Processing" de Stanford.*

Parallèlement, un autre phénomène n'est pas encore considéré par le modèle : la coarticulation, dénotant le fait qu'un phonème n'est pas prononcé de la même manière en fonction des phonèmes l'entourant. Ceci est dû au fait que la modification de la configuration du conduit vocal est progressive pour passer d'un phonème à un autre, provoquant au passage une distorsion de ces phonèmes. Lorsque l'on construit un modèle acoustique, il apparaît évident de modéliser aussi les phénomènes liés au contexte phonétique.

Ainsi, un nouveau modèle va être construit sur la base du modèle monophone précédent, le modèle *triphone*, aussi appelé modèle dépendant du contexte. L'idée étant que pour un modèle monophone $\mathcal{H}_y$ donné, un modèle triphone $\mathcal{H}_{x-y+z}$ unique va être créé pour chaque contexte gauche-droite $x - z$ possible, c.-à.-d les phonèmes pouvant apparaître à gauche et à droite du phonème initial. Un exemple de transformation d'un modèle monophone à ses modèles triphone est donné par la figure 2.7. En considérant des paramètres initiaux issus du modèle monophone "racine", le système triphone peut être ensuite entraîné comme précédemment. Outre l'inclusion d'informations sur le contexte, ce modèle présente différents avantages augmentant la robustesse de la modélisation, comme par exemple le fait que chaque modèle est maintenant responsable d'une plus petite région de l'espace acoustico-phonétique.

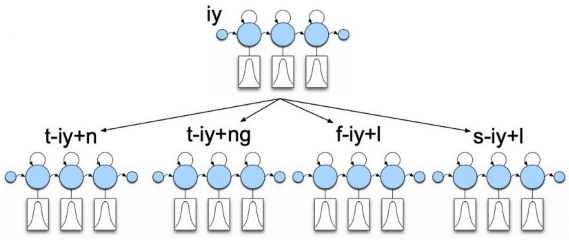



Figure 2.7: Exemple d'un modèle *monophone* et des modèles triphones résultants. *Extrait du cours "CS224S: Spoken Language Processing" de Stanford.*

Ces adaptations ont toutefois un coût non négligeable. Alors qu'auparavant nous avions $N$ états internes pour notre HMM, nous avons maintenant $N^3 \times 3$ états. Nous aurions

donc besoin d'un très grande volume de données d'entraînement pour entraîner le modèle. Il est donc nécessaire de trouver un moyen de réduire le nombre d'états tout en conservant les informations liées au contexte. En partant du principe que le nombre de types de triphones possibles est beaucoup plus important que le nombre d'étiquettes triphones observées, différentes approches ont ainsi été proposées pour résoudre ce problème en retravaillant la modélisation des triphones peu fréquents notamment. Parmi elles, deux types d'approches sont majoritairement utilisées : l'approche par lissage et l'approche par partage de paramètres.

L'approche par lissage correspond ici à modifier les modèles triphones de manière à ce qu'une représentation plus ou moins spécifique soit utilisée à la place selon les observations rencontrées. Dans ce cas, deux techniques sont principalement utilisées en combinaison : le lissage par repli et le lissage par interpolation. La première, appelé aussi lissage *back-off*, consiste à remplacer les modèles par des modèles moins spécifiques lorsque que nous ne possédons pas assez de données d'observation pour un triphone ou biphone particulier. Si un triphone n'est pas observé, le modèle triphone correspondant est remplacé par un modèle biphone. Parallèlement, si un nombre d'occurrences restreint de biphone est observé, le modèle biphone correspondant est remplacé par un modèle monophone. Par exemple :

$$\text{triphone} \rightarrow \text{biphone} : \text{bb-oo+nn} \rightarrow \text{oo+nn}$$
$$\text{(cascade) triphone} \rightarrow \text{biphone} \rightarrow \text{monophone} : \text{bb-oo+nn} \rightarrow \text{oo+nn} \rightarrow \text{oo}$$

Où le remplacement est défini par le nombre d'occurrences de chaque observation, calculé en se servant d'une portion du jeu de données d'entraînement. La seconde technique de lissage consiste quant à elle à faire l'interpolation des paramètres d'un modèle triphone $\mathcal{H}^{\text{tri}}$ avec ses modèles biphones $\mathcal{H}^{\text{bi}}$ et monophone $\mathcal{H}^{\text{mono}}$ tel quel :

$$\mathcal{H}^{\text{tri}} = \alpha_3\mathcal{H}^{\text{tri}} + \alpha_2\mathcal{H}^{\text{bi}} + \alpha_1\mathcal{H}^{\text{mono}} \tag{2.21}$$

Où $\alpha_1, \alpha_2, \alpha_3$ sont des paramètres d'interpolation estimés en se basant sur la *deleted interpolation* [Huang et al., 1996]. En définitive, ces techniques nous permettent d'assurer que le modèle soit bien entraîné selon les données disponibles (lissage par repli) tout en renforçant l'estimation des modèles triphones en partageant les données provenant d'autres contextes (lissage par interpolation).

L'approche préférée reste toutefois celle par partage de paramètres. Celle-ci peut être réalisée à différents niveaux selon la technique employée. Par exemple, des mélanges liés [Bellegarda and Nahamoo, 1990, Huang, 1992] peuvent être utilisés pour partager les paramètres gaussiens, où toutes les distributions partagent le même ensemble de gaussiennes mais ont des poids de mélange différents. Si nous voulons partager des modèles, la construction d'un modèle triphone généralisé peut être envisagée [Lee, 2010], où les modèles dépendant d'un contexte similaire sont fusionnés en considération d'une mesure de similarité quelconque.

Parmi tous les niveaux de partage existants, le partage d'états reste le plus utilisé. L'idée est ici de partager des données d'apprentissage entre les états des (HMMs) triphones. Pour ce faire, une technique de *regroupement d'états* est utilisée afin de décider quels états doivent être liés ensemble [Young and Woodland, 1994].

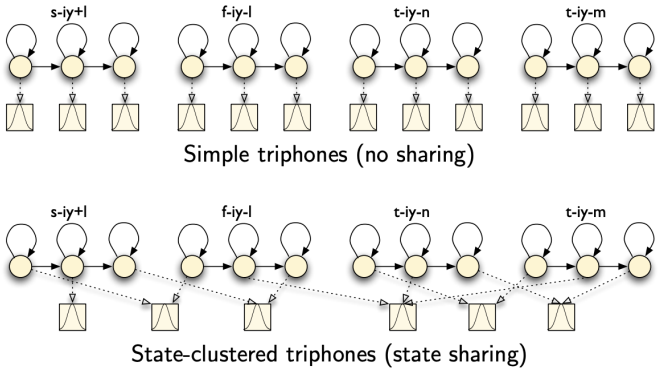



Figure 2.8: Exemple de transformation d'un modèle triphone en un modèle triphone avec états partagés.

Traditionnellement, ce regroupement est effectué en exploitant des arbres de décision phonétiques construits à partir des données d'entraînement [Young et al., 1994]. L'idée étant ici de créer pour chaque état de chaque phonème un arbre de décision phonétique qui va permettre de regrouper tous les états similaires des triphones. Pour chaque arbre, une question binaire relative à l'articulation du phonème parent par rapport à son contexte phonétique est posée à chaque noeud, par exemple : "Est ce que le phonème de droite est une fricative", "Est-ce que le phonème de gauche est une voyelle" ou encore "Est-ce que le phonème de droite est le phonème X". Lorsque deux états se retrouvent dans une même feuille de l'arbre, c.-à.d. lorsqu'ils fournissent la même réponse pour toutes les questions de l'arbre, alors les paramètres d'états peuvent être partagés (voir figure 2.8).

Concernant la construction des arbres et la sélection des questions pour chaque noeud d'un arbre, ceux-ci sont effectués et sélectionnés de manière à maximiser la vraisemblance des modèles avec les données d'apprentissage. En pratique, cette approche permet de réduire le nombre d'états sans aucune dégradation en performance. Pour une description complète du processus, j'oriente le lecteur vers [Young et al., 1994] ainsi que les cours de Daniel Povey[2] pour la reconnaissance de parole avec la boite à outils Kaldi.

Le modèle triphone résultant de ces adaptions est ensuite entraîné en suivant la procédure d'entraînement standard décrite précédemment (voir section 2.2). A partir d'ici, un ensemble d'algorithmes d'entraînement pour modèle triphone peut être utilisé impliquant notamment des techniques de transformation de l'espace de caractéristiques en amont. Parmi les techniques utilisées, les principales consistent à entraîner un modèle triphone avec *Linear Discriminant Analysis – Maximum Likelihood Linear Transform* (LDA-MLLT) [Saon et al., 2000] et un modèle triphone avec SAT *Speaker Adaptive Training* [Anastasakos et al., 1996]. La première approche consiste à créer les états HMM dans un espace de caractéristiques réduit au travers d'une analyse discriminante linéaire. L'espace de caractéristiques est ensuite passé à une transformation linéaire entraînée par maximum de vraisemblance [Gales, 1998] qui dérive une transformation unique pour chaque locuteur. La seconde technique vise quant à elle à compenser les variations inter-locuteurs dans le processus d'estimation des paramètres HMM. L'idée est d'utiliser des transformations MLLR (*Maximum Likelihood Linear Regression*) ou fMLLR (*Feature space Maximum Likelihood Linear Regression*) [Gales, 1998] estimées pour chaque locuteur de l'ensemble d'entraînement, puis de réestimer les paramètres du modèle (moyenne, variance et poids

[2] `https://www.danielpovey.com/kaldi-lectures.html`

de mélanges) en se servant de ses transformations.

### 2.5.2 Transducteurs finis pondérés

Dans la section 2.4, j'ai introduit le processus de décodage utilisant l'algorithme de décodage de Viterbi appliqué à un modèle de Markov, issu de la concaténation et la composition de différents HMMs. Toutefois, dans le cadre d'un système à large vocabulaire modélisant un grand nombre de termes, les modèles HMMs s'avèrent trop complexes pour être décodés efficacement. Afin de résoudre ce problème, les systèmes de RAP modernes s'appuient sur l'utilisation de "transducteurs finis pondérés" (ou *weighted Finite State Transducers* en anglais, abrégé wFST) [Mohri et al., 1996] [Mohri et al., 2000] [Mohri et al., 2002]. Ceux-ci fournissent une représentation commune et naturelle pour modéliser les HMMs ainsi que les différents composants du système : la dépendance au contexte, les dictionnaires de prononciations, ou encore les modèles de langage ou grammaires.

Dans les sous-sections suivantes, je donne une brève description des transducteurs finis pondérés ainsi que leur utilisation dans le cadre du processus de décodage en RAP. Avant de continuer, il est toutefois important de rappeler des notions relatives au *demi-anneau* et aux automates finis.

#### 2.5.2.1 Demi-anneau

Un demi-anneau est une structure algébrique $(\mathbb{K}, \bigoplus, \bigotimes, \overline{0}, \overline{1})$ ayant les propriétés suivantes :

- $(\mathbb{K}, \bigoplus, \overline{0})$ est un monoïde commutatif.
- $(\mathbb{K}, \bigotimes, \overline{1})$ est un monoïde.
- $\bigotimes$ est distributif par rapport à $\bigoplus$.
- $\overline{0}$ est absorbant pour $\bigotimes$: $\forall x \in \mathbb{K} : x \bigotimes \overline{0} = \overline{0} \bigotimes x = \overline{0}$

En fonction de l'ensemble manipulé, les deux lois de composition internes et éléments neutres associés sont *adaptés*. Ainsi, les entiers naturels forment par exemple un demi-anneau $(\mathbb{N}, +, \times, 0, 1)$ et le demi-anneau des probabilités est défini de manière similaire par $(\mathbb{R}, +, \times, 0, 1)$. Dans le cadre de la reconnaissance de parole, les poids associés aux transitions représentant le plus souvent des log probabilités, il est d'usage d'utiliser le log demi-anneau défini par $(\mathbb{R}, +, \cdot, 0, 1)$. Toutefois, lorsque des log probabilités sont utilisées dans le cadre d'une approximation de Viterbi, le demi-anneau *tropical* est utilisé à la place : $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$.

#### 2.5.2.2 Automate fini

Un automate fini est un modèle définissant une machine possédant un nombre fini d'états et de transitions entre ses états. Le passage d'un état à un autre est effectué au travers d'une transition et en réponse à une entrée, définie comme une *étiquette* (ou condition de transition). La figure 2.9 est un exemple de machine à 2 états illustrant l'état dans lequel se trouve une porte, c'est-à-dire soit ouvert soit fermé.

Ici, les transitions sont étiquetées par des actions appartenant à l'ensemble $\mathcal{A}$ définit par $\mathcal{A} = \{\text{ouvert}, \text{fermé}\}$. Toutefois, un ensemble quelconque fini non-vide de symboles peut être défini à la place.

[3] `https://fr.wikipedia.org/wiki/Automate_fini`

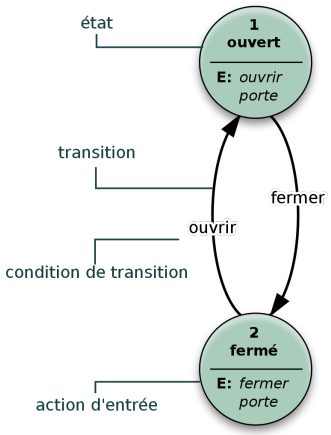



Figure 2.9: Exemple d'automate fini. Extrait de Wikipédia[3].

#### 2.5.2.3 Accepteur fini pondéré

Un accepteur fini est un automate fini produisant une sortie binaire, qui indique si la séquence en entrée est acceptée ou non. Pour cela, un accepteur possède un ensemble d'états particuliers appelés *états finaux*. Si l'état courant, après la lecture de la séquence d'entrée est un état final, l'entrée est acceptée, sinon elle est rejetée. La figure 2.10 donne un exemple d'accepteur fini pondéré à 5 états représentant un modèle de langage à états finis pour un vocabulaire restreint. Ici, chaque état est associé à une valeur unique $x \in \mathbb{N}$. L'état initial est conventionnellement défini en gras et correspond à l'état 0. L'état final est lui dénoté par un double cercle, comme pour l'état 5 sur la figure.

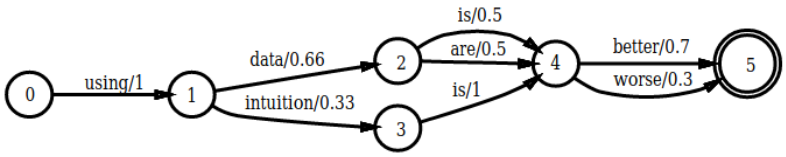



Figure 2.10: Exemple d'accepteur fini pondéré. [Mohri et al., 2002]

L'accepteur est ici dans sa version pondérée, où une valeur, appelée *poids*, est associée à l'étiquette pour chaque transition afin de définir le coût pour passer d'un état à un autre lorsque la transition est effectuée. L'automate modélisant un modèle de langage, les poids représentent ici les probabilités associées à chaque mot-étiquette. Ainsi, une séquence de mots terminant dans un état final (p. ex., "using data is better") spécifie une séquence de mot légale et le produit des probabilités pour le chemin associé donne la probabilité de la séquence de mots (p. ex., $1 \times 0.66 \times 0.5 \times 0.7$).

Plus formellement, un accepteur fini pondéré dépend de la structure algébrique d'un demi anneau $\mathbb{K}$ et se défini par un 7-tuple $(Q, \sum, E, i, F, \lambda, p)$, où:

- Q est un ensemble fini d'états.
- $\sum$ est l'alphabet d'entrée représenté par un ensemble fini de symboles.
- E est un ensemble fini de transitions où:
  $E \subseteq Q \times (\sum \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times \mathbb{K} \times Q$
- i est l'état initial, où $i \in Q$

- F est un ensemble d'états finaux, où $F \subseteq Q$
- $\lambda$ est le poids initial.
- $p$ est une fonction de pondération finale.

Définissons maintenant une transition $t$ comme étant un 4-tuple $(t^-, l(t), w(t), t^+) \in E$, où $t^-$ désigne l'état source, $t^+$ l'état de destination avec l'étiquette $l(t)$ et un poids $w(t)$. Un chemin au travers d'un accepteur fini pondéré est défini comme la séquence de transitions consécutives $t_1, ..., t_n$ avec $t_i^+ = t_{i+1}^-$. Un chemin est défini comme accepté si celui-ci termine dans un état final $f \in F$. L'étiquette associée au chemin $\pi$ est le résultat de la concaténation des étiquettes pour chaque transition du chemin, soit $l(\pi) = l(t_1, ..., l(t_n)$. Parallèlement, une séquence de symboles est définie comme acceptée s'il existe un chemin $\pi$ étiqueté par x : l($\pi$) = x.

En nous appuyant sur les lois du demi-anneau $\mathbb{K}$, nous pouvons calculer le poids associé au chemin $\pi$ et à la séquence $x$ en nous servant de respectivement le produit-$\bigotimes$ et la somme-$\bigoplus$ tel que :

$$w(\pi) = \lambda \bigotimes w(t_1) \bigotimes ... w(t_n) \bigotimes p(t_n^+) \tag{2.22}$$

$$w(x) = \bigoplus_{\pi \in P(i,x,F)} \lambda(p(\pi)) \bigotimes w(\pi) \bigotimes p(t_n^+) \tag{2.23}$$

#### 2.5.2.4 Transducteur fini pondéré

Un transducteur fini est aussi une forme particulière d'automate fini ayant comme particularité d'avoir une étiquette d'entrée et une étiquette de sortie associées à chaque transition. Ici, la machine effectue donc la correspondance entre les séquences de symboles d'entrée et de sortie. Puisque de nombreuses sources d'information utilisées pour le problème de reconnaissance automatique de parole impliquent des correspondances stochastiques à états finis entre des séquences de symboles, les transducteurs s'avèrent être un choix naturel de représentation.

De manière similaire à l'accepteur fini pondéré, le transducteur fini pondéré peut être formalisé par un 8-tuple $(Q, \sum, \Omega, E, i, F, \lambda, p)$ où:

- Q est un ensemble fini d'états
- $\sum$ est l'alphabet d'entrée représenté par un ensemble fini de symboles.
- $\Omega$ est l'alphabet de sortie représenté par un ensemble fini de symboles.
- E est un ensemble fini de transitions où:
  $E \subseteq Q \times (\sum \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times \mathbb{K} \times Q$
- i est l'état initial, où $i \in Q$
- F est un ensemble d'états finaux, où $F \subseteq Q$

Ici, une transition est définie par $(t^-, l_i(t), l_o(t), w(t), t^+) \in E$, où $t^-$ où $l_i(t)$ et $l_o(t)$ désignent respectivement les étiquettes d'entrée et de sortie. L'ensemble des définitions données précédemment pour l'accepteur à états finis pondéré est aussi valable pour le transducteur à états finis. Une distinction est toutefois faite entre le chemin d'étiquettes d'entrée et le chemin d'étiquettes de sortie, où ce dernier correspond à la concaténation

des étiquettes de sortie pour les transitions du chemin. La figure 2.11 donne un exemple de transducteur à états finis pour représenter un dictionnaire phonétique pour deux termes data et dew définis par respectivement deux séquences de phonèmes et une séquence de phonèmes.

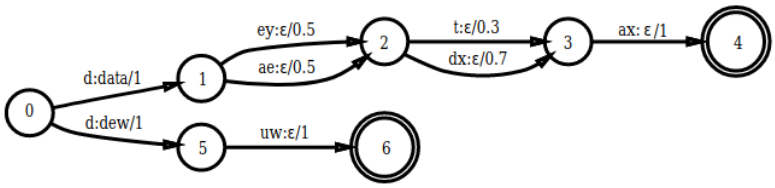



Figure 2.11: Exemple d'accepteur fini pondéré. [Mohri et al., 2002]

Ici, les étiquettes d'entrée sont donc des phonèmes et les étiquettes de sortie sont des mots. Les transducteurs des différentes entrées du dictionnaire sont combinées tout en conservant l'identité de chaque mot. Cette caractéristique peut être étendue aux autres modèles du système de RAP que nous voulons manipuler. Ainsi, si nous considérons les transducteurs pour des structures HMMs à la place, ceux-ci peuvent être combinés en un seul transducteur tout en permettant la conservation de l'identité du modèle phonétique et de partager les distributions des sous-séquences lorsque cela est possible.

#### 2.5.2.5 Opérations pour transducer fini pondéré

Un ensemble d'opérations peuvent être appliquées à un transducteur fini pondéré et cela indépendamment de la nature des données représentées ou du type de demi-anneau utilisé. Dans le cadre de la RAP, les principales opérations utilisées sont :

- La **somme** pour effectuer l'union de plusieurs transducteurs. Cela permet par exemple de représenter l'ensemble des prononciations possibles pour chaque mot du vocabulaire en faisant l'union des transducteurs de chaque entrée de notre dictionnaire phonétique.
- Le **produit** afin de concaténer plusieurs transducteurs. Ainsi, les transducteurs de mots peuvent être concaténés pour former des phrases par exemple.
- La **fermeture de Kleene** pour former une structure avec répétitions. Une séquence de mots peut ainsi être créée à partir de mots individuels répétés.
- La **composition** permettant de construire une structure hiérarchique à partir des transducteurs issus de nos différents composants.
- La **déterminisation** afin de rendre les transducteurs déterministes.
- La **minimisation** servant à réduire le nombre d'états d'un automate fini pondéré déterministe.

Parmi ces opérations, les trois dernières sont les plus importantes afin de traiter efficacement les éléments constitutifs dans le cadre du processus de décodage. Pour la composition, celle-ci se définit plus précisément par :

$$[[A \circ B]](x, y) = \bigoplus_{z} [[A]](x, z) \bigotimes [[B]](z, y) \tag{2.24}$$

Où A et B sont deux transducteurs finis pondérés. Lorsque l'un des transducteurs contient des transitions comportant des étiquettes nulles ($\epsilon$), des opérations sont utilisées afin de s'assurer que le transducteur final inclue des chemins valides et uniques. Pour cela, un transducteur filtre est habituellement employé afin de supprimer les chemins redondants ou inutiles.

#### 2.5.2.6 Décodage avec transducteurs finis pondérés

Dans le cadre du processus de décodage, les transducteurs représentant les différents composants du système de RAP sont utilisés afin de transformer les flux d'états des HMMs en une séquence de mots. Ces transducteurs sont nommés H, C, L et G, et définissent respectivement :

- La structure HMM donnant en sortie la séquence de phonèmes dépendant au contexte étant donnée une séquence d'états HMMs en entrée.
- Le processus de réétiquetage en fonction du contexte qui, pour une séquence de phonèmes dépendants au contexte en entrée, produit la séquence de phonèmes indépendants au contexte.
- Le dictionnaire phonétique (ou lexique) transformant une séquence de phonèmes en mots.
- Le modèle de langage (ou grammaire), qui étant donné une séquence de mots en entrée, produisent une séquence de mots autorisées. Pour ce faire, le composant est construit comme un automate fini acceptant.

La sortie de chaque transducteur correspondant à l'entrée du transducteur suivant dans la liste, une composition des différents transducteurs est effectuée afin de former un graphe de recherche, ou *recognizer*, appelé HCLG. Cette composition est effectuée de la manière suivante, où det et min sont respectivement des étapes de déterminisation et minimisation pouvant contenir des adaptations particulières (p. ex., gestion de transitions nulles) et permettant d'optimiser l'espace de recherche :

$$H \circ C \circ L \circ G = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G)))))) \tag{2.25}$$

Pour un énoncé de parole de $N$ trames en entrée, le chemin le plus probable représentant notre énoncé peut être ensuite obtenu en appliquant l'algorithme de décodage de Viterbi à un nouvel espace de recherche $X$. Ce nouvel espace est obtenu par la composition du transducteur HCLG avec un automate fini $E$ acceptant modélisant les transitions HMM de l'énoncé, où chaque transition est étiquetée par l'état HMM pour l'instant $t$ et la log-vraisemblance acoustique correspondante.

En pratique, le graphe résultant de cette composition est trop grand pour être manipulé efficacement. Ainsi, différentes optimisations sont apportées pour accélérer le processus et contraindre l'espace de recherche, tel qu'un élagage par faisceau associé à des étapes de déterminisation et de poussée de poids. Cette dernière opération permet de redistribuer les poids vers l'état initial ou les états finaux en les poussant, ce qui est particulièrement pratique dans le cadre de l'élagage par faisceau puisque les poids peuvent être anticipés. Pour plus d'informations, j'oriente le lecteur vers [Mohri et al., 2000] qui est l'une des références principales pour l'utilisation des wFSTs en RAP.

### 2.5.3 Reconnaissance de parole *hybride*

Comme il a été dit dans le chapitre d'introduction du problème de la RAP, et plus précisément la section 1.6 donnant un bref historique du domaine jusqu'à 2016, l'utilisation des réseaux de neurones pour la modélisation acoustique remonte aux années 90. Ceux-ci présentent de nombreux avantages comme la capacité de modéliser un contexte large, la facilité d'intégration de différentes sources en entrée ou encore le fait de proposer une meilleure généralisation des distributions complexes avec un plus petit nombre de paramètres. Si leur utilisation fut tout d'abord limitée pour différentes raisons (p. ex., puissance de calcul, volume de données d'entraînement, complexités de certaines opérations, etc.), les réseaux de neurones sont maintenant couramment utilisés en RAP, formant ce qui s'appelle les approches *hybrides* dans le cadre de la RAP traditionnelle. Parmi ces approches, deux sont principalement utilisées:

1. La RAP hybride *tandem* où le réseau de neurones fonctionne comme un extracteur de caractéristiques fournissant des données d'entrée au modèle HMM-GMM. Ce dernier utilisant les caractéristiques dérivées des réseaux de neurones entraînés, concaténées avec les caractéristiques acoustiques originales. [Hermansky et al., 2000]

2. La RAP hybride *classique* se proposant d'utiliser un réseau de neurones profond pour estimer les probabilités d'observation des HMM à partir des alignements générés d'un modèle HMM-GMM. Les réseaux de neurones remplaçant donc la modélisation GMM. [Hinton et al., 2006, Hinton et al., 2012a]

De nos jours, les systèmes de RAP traditionnels s'appuient majoritairement sur l'approche hybride dite *classique* et ont permis une avancée significative du domaine [Hinton et al., 2012a]. Depuis leur démocratisation au milieu des années 2000, un ensemble considérable de travaux ont permis de démontrer l'efficacité des réseaux de neurones pour le problème de RAP ainsi que de traiter des problématiques liés à leur utilisation. Parmi ces travaux, il semble important d'en citer quelques uns :

- La technique *Dropout* [Hinton et al., 2012b] permettant la régularisation d'un réseau et de palier ce problème de sur-apprentissage.
- La technique d'augmentation de données *SpecAugment* [Park et al., 2019] et *Speed Perturbation* [Ko et al., 2012].
- L'approche *Deep Belief Network* [Hinton et al., 2006] utilisée comme initialisation d'un réseau de neurones profonds pour les systèmes de RAP modernes [Mohamed et al., 2009].
- Les nouvelles architectures neuronales tels que le réseau de neurones convolutionnels (ou *Convolutional Neural Network* en anglais, abrégé CNN) ou le réseau de neurones à délai (ou *Time Delay Neural Network* en anglais, abrégé TDNN) [Waibel et al., 1989] mis à jour dans une forme factorisé [Peddinti et al., 2015].
- La technique d'entraînement discriminante *Lattice-Free MMI* [Povey et al., 2016] qui permet un entraînement basé purement sur la séquence et repense le calcul de l'approximation de la somme du dénominateur défini par la fonction objective MMI sans l'utilisation de lattice.

A l'exception de *Deep Belief Network*, l'ensemble de ces travaux ont été utilisés dans le cadre de ma thèse. Plus précisément, ceux-ci ont permis la réalisation du système de RAP traditionnel pour le français utilisé comme système de référence dans le chapitre 6

et le chapitre 8. Dans ce cadre, les techniques d'augmentation de données, le réseau de neurones à délai (TDNN) et la technique d'entraînement *Lattice-Free MMI* se sont avérés être des éléments importants.

## 2.6 Conclusion

Dans ce chapitre, j'ai décrit les composants principaux d'un système de RAP traditionnel basé phonèmes ainsi que leur fonctionnement. En outre, j'introduis les différentes architectures, techniques et approches couramment utilisées pour l'entraînement et le décodage de systèmes de RAP dits modernes. Ce type de système de RAP est encore aujourd'hui couramment utilisé, grâce notamment à des boites à outils telles que Kaldi [Povey et al., 2011] qui ont acquis une maturité au fil du temps permettant la création de systèmes compétitifs et déployables aussi bien dans un contexte académique qu'industriel. Dans le cadre de mes travaux sur la RAP en français (voir chapitre 6), ces systèmes sont utilisés en tant que systèmes de référence et constituent un objectif à atteindre lorsque de nouvelles approches sont évaluées, telle que la RAP bout-en-bout (voir chapitre 3. Pour finir, dans le cadre de mes projets au sein d'Airudit que je décris dans le chapitre 8, je donne des exemples d'utilisation de ces systèmes pour le français dans un contexte industriel et évalue leur capacité à généraliser à de nouvelles données issues d'environnements acoustiques divers.

# Chapter 3

# Systèmes de RAP bout-en-bout

Comme vu dans le chapitre précédent, la reconnaissance automatique de parole (RAP) utilise traditionnellement des Modèles de Markov Cachés (HMMs), décrivant la variabilité temporelle, combinés à des Modèles de Mélanges Gaussiens (GMMs), calculant les probabilités d'émission à partir des états des GMMs, pour la modélisation acoustique. Cette modélisation associant les caractéristiques acoustiques à des unités phonétiques, le modèle est habituellement indissociable d'un dictionnaire phonétique et d'un modèle de langage, le plus souvent basé mots. Par la suite, la réintroduction de réseaux de neurones profonds remplaçant les GMMs a permis d'améliorer considérablement la qualité de la modélisation acoustique par rapport aux systèmes précédents. Cependant, la construction et l'entraînement de ces systèmes peuvent s'avérer complexes et de nombreuses connaissances et étapes de pré-traitement sont nécessaires.

Apparues peu à peu entre le milieu des années 2000 et 2020, des approches plus directes, appelées méthodes bout-en-bout (ou *end-to-end* en anglais), dans lesquelles les architectures neuronales sont entraînées pour modéliser directement des séquences de caractéristiques acoustiques en séquences d'unités de nature orthographique, ont été proposées [Graves et al., 2006, Graves, 2012b, Bahdanau et al., 2014]. Basées sur l'utilisation des réseaux de neurones récurrents remplaçant le duo HMM-GMM/DNN pour la modélisation acoustique, ces systèmes ont suscité un grand intérêt de la part de la communauté des chercheurs ainsi que des développeurs non-experts ces dernières années. Ceci est dû à la simplicité des architectures et à la possibilité de créer un système de RAP complet sans avoir besoin de connaissances spécialisées. En outre, le fait de pouvoir prédire des cibles indépendantes du contexte telles que le caractère ou la syllabe, permet de résoudre le problème de mot hors-vocabulaire rencontré dans les systèmes de RAP traditionnels qui rend difficile la généralisation des systèmes ou leur maintenance dans le temps.

Ces méthodes sont de nos jours couramment utilisées par les grandes entreprises du secteur tel que Google [He et al., 2019], Microsoft [Kim and Seltzer, 2018] ou encore Facebook [Schneider et al., 2019], et sont intensivement étudiées par les chercheurs pour différents problèmes de traitement de la parole tel que la traduction vocale [Bérard et al., 2016, Bérard et al., 2018, Inaguma et al., 2019] ou encore la synthèse de parole [Wang et al., 2017, Ping et al., 2019]. Dans le cadre de ma thèse, ces approches se sont avérées particulièrement utiles pour la résolution de problématiques scientifiques mais aussi industrielles, que je rapporte dans la suite de ce manuscrit. Ainsi, je me focalise dans ce chapitre à introduire et décrire les différentes architectures et approches existantes pour la mise en place d'un système de RAP bout-en-bout.

Dans la première section, une introduction des réseaux de neurones récurrents et des architectures neuronales utilisées de nos jours pour la modélisation acoustique sera donnée. Couplés avec les modèles séquence-à-séquence que nous décrivons dans la même section, ces notions sont les fondations ayant permis le développement des différentes approches bout-en-bout. Dans la section suivante, j'introduis les trois principales approches bout-en-bout existantes de nos jours. Cette liste sera étendue par une courte description d'approches bout-en-bout hybrides ou multi-tâches en dernière section. Je clos ce chapitre en introduisant les architectures neuronales ayant remplacées les réseaux de neurones dans ces approches mais aussi dans une grande majorité de domaines traitants de problèmes temporels ou séquentiels : le Transformer.

## 3.1 Architectures neuronales

Cette section présente les architectures neuronales principalement utilisées au cours de ma thèse. Cette liste sera étendue au fil du manuscrit, et notamment dans le chapitre 5, par des architectures ou approches récentes ayant fait l'objet d'une attention particulière pour certains de mes travaux.

### 3.1.1 Réseaux de neurones récurrents

Un réseau de neurones récurrents (ou *Recurrent Neural Networks* en anglais, abrégé RNN) est un type de réseau de neurones artificiels permettant de traiter des problèmes séquentiels ou temporels tels que la reconnaissance automatique de parole, le traitement du langage naturel ou encore la traduction automatique de langue. Contrairement aux réseaux de neurones profonds qui supposent que les entrées et les sorties sont indépendantes les unes des autres, la sortie d'un RNN est conditionnée par les éléments précédents, constituant ce qui est appelé *l'historique*, dans la séquence. Pour cela, chaque cellule possède un ensemble d'opérations permettant de faire persister l'information dans le temps.

Habituellement, les RNNs sont représentés sous une forme repliée (à gauche sur la figure 3.1), où une boucle est admise pour représenter le flux d'informations réutilisés, ou une forme dépliée (à droite sur la figure 3.1), illustrant simplement le réseau traitant la séquence complète.

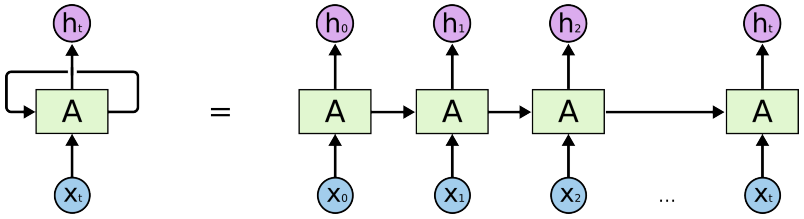



Figure 3.1: Schéma d'un réseau de neurones récurrents dans sa forme simple (à gauche) et sa forme déroulée (à droite). *C. Olah*, 2015

Ainsi, pour une séquence $X = (x_1, ..., x_t)$ de longueur $T$, le réseau parcourt successivement les entrées $x_1$ à $x_t$. À l'instant $t$, la t-ème cellule combine l'entrée courante $x_t$ avec la prédiction au pas précédent, $h_{t-1}$ pour calculer une sortie $h_t$. La séquence de sortie du réseau est donc $H = (h_1, ..., h_t)$

D'une manière plus formelle, une couche RNN définit donc une relation de récurrence: $h_t = f(x_t, h_{t-1})$. La fonction $f(\cdot)$ est définie ici comme une sigmoïde $\sigma$ prenant en entrée

la concaténation de l'entrée $x_t$ pour l'instant $t$ et de l'état historique $h_{t-1}$ pour l'instant précédent, tel que :

$$h_t = \sigma(Ux_t h_{t-1} + Vh_{t-1} + b_h) \tag{3.1}$$

Où $U$ et $V$ sont des matrices de transformation linéaire, appelés communément poids, liés respectivement à l'état historique $h_{t-1}$ et à l'entrée $x$. $b$ définit le biais, une valeur permettant à la fonction d'activation de modéliser un espace de données qui est centré autour d'un point autre que l'origine. Les paramètres entre toutes les cellules d'une même couche RNN sont ici partagés, ainsi la même fonction est appliquée à chaque pas de temps $t$. Ceci permet de représenter d'une manière homogène les éléments de la séquence tout en permettant de réduire le nombre de poids à apprendre.

La sortie est quant à elle formulée en considération de l'information contextuelle $h_t$ à laquelle nous appliquons une transformation linéaire ayant pour matrice $W$ et un biais $b$ :

$$p_t = Wh_t + b_p \tag{3.2}$$

La séquence complète $P = (p_1, ..., p_t)$ peut être calculée en commençant à $t = 1$ et en appliquant récursivement les équations précédentes, avec $t$ incrémenté à chaque itération.

#### 3.1.1.1 Entraînement

L'entraînement des RNNs s'effectue, de manière similaire aux réseaux *feed-forward*, via un algorithme de rétropropagation. Toutefois, du fait que les paramètres sont partagés, le calcul du gradient à chaque pas de temps dépend aussi des calculs des pas de temps actuels et précédents. Ainsi, il est nécessaire de rétropropager en plusieurs étapes et additionner les gradients correspondants. Ce processus est appelé rétropropagation dans le temps (ou *BackPropagation Through Time* [Williams and Zipser, 1995], abrégé BPTT) est s'effectue de la manière suivante.

Définissons tout d'abord une fonction de coût $l$ évaluant l'écart entre la sortie $p_t$ et l'étiquette cible $y_t$. La fonction $\mathcal{L}$ est définie comme la somme des fonctions pour tous les pas de temps $1 : T$:

$$\mathcal{L} = \sum_{t=1}^{T} l(y_t, p_t) \tag{3.3}$$

Une fois la passe avant effectuée via la procédure décrite dans la section précédente pour le calcul de $P$, les paramètres du modèle $U$, $V$ et $W$ peuvent être entraînés en minimisant la fonction de coût précédente. Cela implique donc le calcul du gradient par rapport à ces paramètres, respectivement $\frac{\partial \mathcal{L}}{\partial U}$, $\frac{\partial \mathcal{L}}{\partial V}$ et $\frac{\partial \mathcal{L}}{\partial W}$.

La première étape est de calculer, en se servant de l'équation 3.3, le gradient de $\mathcal{L}$ par rapport au paramètre $W$ de la couche de sortie :

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial p_t} h_t^\intercal \tag{3.4}$$

Où $^\intercal$ désigne une transposée et $\frac{\partial \mathcal{L}}{\partial p_t}$ est la fonction de coût par rapport à la sortie du modèle à n'importe quel pas de temps $t$, définit par :

$$\frac{\partial \mathcal{L}}{\partial p_t} = \frac{\partial l(y_t, p_t)}{T \cdot \partial p_t} \tag{3.5}$$

Ensuite, nous devons calculer le gradient de $\mathcal{L}$ par rapport à $h_t$. À l'instant $T$, la fonction $\mathcal{L}$ dépend seulement de $h_T$ via $o_T$. Toutefois, pour un pas $1 \leq t \leq T$, celle-ci dépend de $h_t$ au travers de $o_t$ et de $h_{t+1}$ comme nous pouvons le voir avec les équations 3.1 et 3.2. Ainsi, le gradient de $\mathcal{L}$ par rapport à $h_t$, où $1 \leq t \leq T$ est calculé récursivement de la manière suivante :

$$\frac{\partial \mathcal{L}}{\partial h_t} = \sum_{i=t}^{T} {V^\intercal}^{T-i} \, W^\intercal \frac{\partial \mathcal{L}}{\partial p_{T-t+i}} \tag{3.6}$$

Ce calcul met en évidence deux problèmes des RNNs liés à la gestion de longues séquences et la dépendance à long terme : dans le cas de grandes puissances pour $V^\intercal$, les valeurs propres inférieures à 1 disparaissent et les valeurs propres supérieures à 1 divergent, ce qui peut provoquer des instabilités numériques. Cela donne lieu à deux phénomènes appelés: *vanishing gradient* et *exploding gradient* [Hochreiter, 1991, Bengio et al., 1994]. Dans les sections suivantes, ce problème sera adressé.

Finalement, nous pouvons calculer le gradient de $\mathcal{L}$ par rapport à $U$ et $V$. Pour cela, on se sert de $\frac{\partial \mathcal{L}}{\partial h_t}$ obtenu récursivement via l'équation 3.6 afin de calculer $\frac{\partial \mathcal{L}}{\partial U}$ et $\frac{\partial \mathcal{L}}{\partial U}$ tels que :

$$\frac{\partial \mathcal{L}}{\partial V} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial h_t} h_{t-1}^\intercal \tag{3.7}$$

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial h_t} x_t^\intercal \tag{3.8}$$

$$\tag{3.9}$$

Lors de l'entraînement du modèle RNN, les phases de propagation avant et arrière (ou rétropropagation) dans le temps sont effectuées de manière alternée. Lors de cette dernière phase, les gradients calculés pour les paramètres intermédiaires à un instant $t$ sont stockés pour éviter d'effectuer des calculs redondants à l'instant $t-1$.

### 3.1.2 *Long-Short Term Memory*

Afin de palier au problème de dépendance à long terme décrit précédemment, [Hochreiter and Schmidhuber, 1997] proposent une amélioration des RNNs appelée mémoire court et long terme (ou *Long-Short Term Memory* en anglais, abrégé LSTM) permettant de contrôler l'information à transmettre au travers de différentes couches internes. Au fil du temps, les LSTMs se sont vu apporter de nombreuses améliorations et sont de nos jours encore utilisés pour de nombreux problèmes séquentiels et temporels.

La différence majeure entre un RNN et un LSTM peut se résumer à quatre composants : une cellule état $c_t$, s'occupant du transport de l'information, deux portes contrôlant

l'information entrante, $e_t$ (c.-à-d., porte d'entrée), et l'information sortante $s_t$ (c.-à-d., porte de sortie) et pour finir, une porte définissant les informations à conserver ou oublier, $o_t$ (c.-à-d., porte d'oubli). Pour chaque porte, des paramètres partagés $U_x$, $W_x$ sont définis et représentent respectivement les matrices de transformation linéaire pour l'entrée $x_t$ et l'état précédent $h_{t-1}$. La somme des informations issue de ces transformations est ensuite passée au travers d'une fonction d'activation ($\sigma$ ou tanh), en incluant un biais $b_x$, comme pour les RNNs.

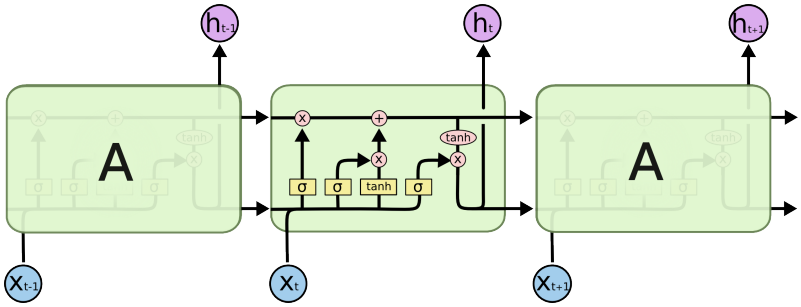



Figure 3.2: Représentation interne d'une cellule LSTM. *C. Olah*, 2015

La première étape consiste à activer les portes $e_t$, $o_t$, $s_t$ servant à contrôler l'information de différentes manières, respectivement : l'ajout et la dissipation de l'information à l'intérieur de la cellule, et à filtrer l'information sortante de la cellule. Étant donnés $x_t$ et $h_{t-1}$ comme entrée, ces opérations sont formulées de la manière suivante :

$$e_t = \sigma(U_e x_t + W_e h_{t-1} + b_e) \tag{3.10}$$
$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o) \tag{3.11}$$
$$s_t = \sigma(U_s x_t + W_s h_{t-1} + b_s) \tag{3.12}$$

L'étape suivante consiste à désigner l'information à ajouter à une cellule état $c_t$, qui va nous servir ensuite à pondérer les informations. Pour se faire, les entrées $x_t$ et $h_{t-1}$ sont passées ici au travers d'une fonction d'activation de type tangente hyperbolique afin de générer $n_t$:

$$n_t = \tanh(U_n x_t + W_n h_{t-1} + b_n) \tag{3.13}$$

Une fois ces opérations effectuées, les valeurs de la cellule état $c_t$ sont mis à jour en pondérant, avec un produit matriciel de Hadamard $\circ$, la porte d'entrée $e_t$ et la porte d'oubli $o_t$ sur respectivement le paramètre $n_t$ et la cellule état au pas de temps précédent $c_{t-1}$:

$$c_t = e_t \circ n_t + o_t \circ c_{t-1} \tag{3.14}$$

La sortie est ensuite calculée à partir de ce nouvel état $c_t$ en filtrant les valeurs de l'état à sortir au travers d'une nouvelle sigmoïde. La matrice résultante est ainsi passée au travers d'une tangente hyperbolique afin de contrôler l'information à conserver pour la nouvelle représentation $h_t$ :

$$h_t = s_t \circ tanh(c_t) \tag{3.15}$$

L'ajout de ces opérations ne modifiant pas le processus d'entraînement, la rétropropagation au travers du temps est aussi utilisée pour les LSTMs.

### 3.1.3 *Gated Recurrent Units*

Afin de résoudre aussi le problème de la disparition du gradient qui accompagne un réseau de neurones récurrents standard, [Cho et al., 2014] propose l'utilisation de ce qu'il appelle l'unité récurrente à portes (ou *Gated Recurrent Unit* en anglais, abrégé GRU). De manière similaire au LSTM, un GRU se caractérise par l'utilisation de portes permettant de contrôler des informations à transmettre à la sortie, à savoir : une porte de mise à jour $m_t$ et une porte de réinitialisation $r_t$. Comme précédemment, des paramètres partagés $U_x$ et $W_x$ sont définis, représentant les matrices de transformation linéaire liées respectivement à une entrée $x_t$ et un état historique précédent $h_{t-1}$, et le biais est ajouté pour l'activation de chaque porte.

Ici, la première étape est de décider de l'information à passer et à supprimer. Pour se faire, l'état historique précédent $h_{t-1}$ et le vecteur d'entrée $x_t$ sont passés au travers des portes $m_t$ et $r_t$ tel que :

$$m_t = \sigma(U_m x_t + W_m h_{t-1} + b_m) \tag{3.16}$$
$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r) \tag{3.17}$$

En se servant de l'entrée $x_t$ et de l'activation de la porte de réinitialisation $r_t$ afin de conserver les informations pertinentes de l'état historique précédent $h_{t-1}$, l'information à propager $n_t$ est ensuite calculée au travers d'une fonction de tangente hyperbolique :

$$n_t = \tanh(U_n x_t + W_n(r_t h_{t-1}) + b_n) \tag{3.18}$$

La sortie de la cellule $h_t$ est ensuite calculée en pondérant l'activation de la porte de mise à jour $m_t$ avec respectivement $n_t$ et l'état historique précédent $h_{t-1}$ de la manière suivante :

$$h_t = m_t \circ n_t + (1 - m_t) h_{t-1} \tag{3.19}$$

Malgré le fait que les cellules GRUs sont réputées moins performants que les LSTMs [Su and Kuo, 2019, Shewalkar, 2019], ces premières restent toutefois intéressantes pour leur caractère compact, accélérant le processus d'entraînement et d'inférence d'un RNN. Dans un contexte industriel, l'utilisation des GRUs comparés aux LSTMs peut s'avérer bénéfique en considération du rapport performance - temps du processus d'inférence.

#### 3.1.3.1 Bi-directionnalité

Introduit par [Schuster and Paliwal, 1997] afin d'augmenter le volume d'informations accessible dans les réseaux de neurones récurrents, un RNN bidirectionnel se caractérise par l'utilisation de deux couches cachées de directions opposées connectées à la même sortie : une première dirigée vers l'avant de l'axe temporel et une seconde évoluant dans le sens inverse. Contrairement aux RNN unidirectionnels n'utilisant qu'une direction temporelle, les informations sur l'état passé et futur pour une trame à l'instant $t$ peuvent être considérées.

L'image 3.3 donne une description de la structure, où de bas en haut nous avons respectivement : les entrées, la couche avant, la couche arrière et les sorties du réseau. Chaque sortie $y_t$ est connectée par $\overrightarrow{h_t}$ et $\overleftarrow{h_t}$ obtenu respectivement par la couche avant et la couche arrière. Les sorties ne sont toutefois pas connectées aux entrées pour la couche de sens opposé.

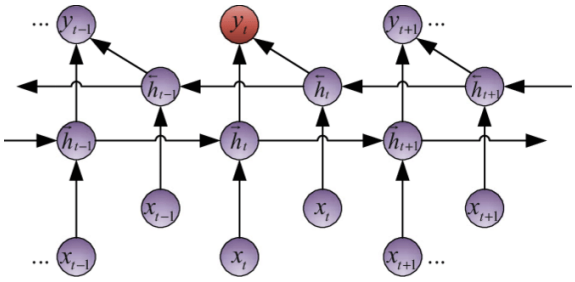
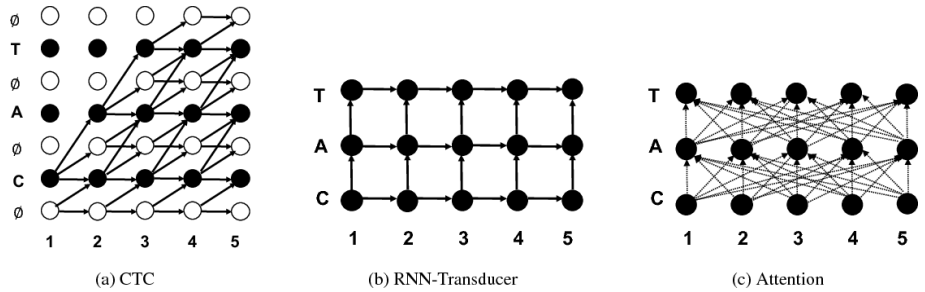



Figure 3.3: Structure interne d'un RNN bidirectionnel [Messner et al., 2018].

Concernant l'entraînement du modèle, la procédure décrite précédemment pour les RNNs unidirectionels peut être utilisés car les cellules des deux directions n'interagissent pas entre elles. Toutefois, lors de la rétropropagation au travers du temps, des calculs intermédiaires sont effectués pour mettre à jour les couches d'entrée et de sortie, ce qui ne peut pas être effectué simultanément comme pour les RNNs simples.

## 3.2 Approches bout-en-bout

En association avec les architectures précédemment présentées, différentes approches et fonctions de coût ont récemment été proposées afin de faire la correspondance directe entre la séquence audio d'entrée et la séquence d'étiquettes en sortie au travers d'un seul réseau neuronal. Pour se faire, et contrairement aux approches traditionnelles (Cf. chapitre 2), le modèle tente ici de prédire des séquences de nature orthographique (p. ex. caractères ou syllabes) directement à partir des observations acoustiques. La représentation intermédiaire phonétique est donc omise.




(a) CTC (b) RNN-Transducer (c) Attention

Figure 3.4: Illustration des transitions de probabilité pour les trois méthodes bout-en-bout sur un énoncé de 5 trames et étiqueté "CAT" [Battenberg et al., 2017]. Le noeud à $(t, u)$ représente la probabilité d'avoir sorti les $u$ premiers éléments de la séquence de sortie à l'instant $t$. Une flèche verticale représente la prédiction de plusieurs étiquettes à un pas de temps $t$ (interdit pour la CTC). Une flèche horizontale représente la prédiction de caractères répétitifs (CTC) ou la prédiction d'étiquettes nulles (RNN-Transducer). Les flèches en gras représentent des alignements durs (CTC ou RNN-Transducer) et des alignements souples (Attention).

De nos jours, trois approches bout-en-bout sont principalement utilisées: 1) La *Connectionnist Temporal Classification* (CTC) [Graves et al., 2006] qui utilise des hypothèses de Markov (c.-à-d., indépendance conditionnelle entre les prédictions à chaque pas de temps) pour résoudre efficacement des problèmes séquentiels par programmation dynamique, 2) RNN-Transducer [Graves, 2012a] qui est une extension de la CTC qui modélise en plus les dépendances entre les sorties à différentes étapes par le biais d'un modèle prédictif analogue similaire un modèle de langage, et 3) l'approche Encodeur-Décodeur basé attention proposé par [Bahdanau et al., 2014] qui repose sur l'utilisation d'un mécanisme d'attention pour effectuer un alignement non-monotone entre les trames acoustiques et les unités orthographiques reconnues. En résumé, celles-ci diffèrent donc sur les hypothèses formulées pour résoudre le problème de RAP, et plus notamment sur trois points:

- L'approche CTC considère une indépendance conditionnelle entre les prédictions à différents pas de temps. RNN-Transducer et Attention modélisent l'inter-dépendance entre les étiquettes prédites.
- Les alignements entre les séquences d'entrée et les séquences de sortie sont monotoniques pour la CTC. Ceux-ci peuvent aussi être imposés pour l'approche RNN-Transducer [Tripathi et al., 2019].
- Les approches CTC et RNN-Transducer considèrent les alignements entre l'entrée et la sortie comme des variables latentes et marginalisent sur tous les alignements, tandis que l'approche Encodeur-Décodeur basé Attention modélise un alignement souple entre chaque pas de sortie et chaque pas d'entrée.

Ces différences sont illustrées dans la figure 3.4 qui représente les alignements effectués entre une séquence d'observation acoustique d'entrée et une séquence d'étiquettes orthographiques en sortie pour ces trois approches. Dans le cadre de ma thèse, et notamment du projet ESPnet décrit dans le chapitre 5, je me suis intéressé à l'ensemble de ces méthodes avec un focus plus notable sur l'approche RNN-Transducer et l'approche CTC ensuite. Ainsi, je donne une description plus détaillée de ces approches dans la suite de ce chapitre.

### 3.2.1 *Connectionist Temporal Classification*

La *Connectionnist Temporal Classification*, ou CTC, est la première approche bout-en-bout utilisée pour le problème de RAP. Proposée par [Graves et al., 2006], celle-ci repose sur l'utilisation d'un réseau de neurones récurrents associé à une fonction Softmax en sortie afin d'effectuer la correspondance en entre une séquence d'observations acoustiques $x = \{x_1, ..., x_U\}$ et une séquence d'étiquettes orthographiques $y = \{y_1, ..., l_U\}$ appartenant à un ensemble $L$. Pour se faire, et en considération de la différence de longueur des séquences en entrée et en sortie, l'approche introduit une étiquette spéciale, appelée *blank* ou $\varnothing$, qui est rajoutée à l'ensemble d'étiquettes $L$ pour former l'ensemble $L' = L \cup \{\varnothing\}$.

Ainsi, étant donnée une séquence d'observations acoustiques $x$ et le réseau CTC, chaque sortie du réseau, va représenter la probabilité $p(\pi_t|x)$ d'observer un chemin $\pi_t \in L^{'}$ supposé conditionnellement indépendant des autres chemins. En multipliant les probabilités $p(\pi_t|x)$, nous pouvons obtenir la probabilité conditionnelle d'observer un chemin $\pi = \{\pi_1, ..., \pi_T\}$ tel que:

$$p(\pi|x) \approx \prod_{t=1}^{T} p(\pi_t|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \quad \forall \pi \in L'^T \tag{3.20}$$

Où $y_k^t$ désigne l'activation de la sortie $k$ à l'instant $t$, c'est-à-dire la probabilité d'observer l'étiquette $k$ à l'instant $t$. La probabilité conditionnelle d'une étiquette $l \in L^{\leq T}$ est ensuite définie comme la somme des probabilités de tous les chemins qui lui correspondent:

$$p(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} \prod_{t=1}^{T} y_{\pi_t}^t \tag{3.21}$$

Où $\mathcal{B}^{-1}(y)$ désigne une transformation $B : L'^T \rightarrow L^{\leq T}$ permettant de convertir une séquence d'étiquettes de $L'$, incluant donc l'étiquette $\varnothing$, de longueur $T$ en une séquence d'étiquettes de $L$ de longueur $\leq T$.

#### 3.2.1.1 Entraînement

Tout comme la RAP traditionnelle utilisant des modèles de Markov cachés et la programmation dynamique, les probabilités conditionnelles indépendantes $p(l|x)$ peuvent être calculées efficacement au travers de l'algorithme *forward-backward* [Rabiner, 1991]. Ici, l'idée étant que la somme de tous les chemins correspondant à un étiquetage $l$ peut être décomposée en une somme itérative des chemins correspondants aux préfixes de cet étiquetage. Toutefois, du fait de l'introduction de l'étiquette $\varnothing$, le raisonnement doit être ici formulé sur une séquence $l'$ incluant l'étiquette $\varnothing$ en début et fin de séquence et entre chaque étiquette non-*blank* de la séquence $l$.

Étant donnée une séquence de référence $l^*$, définissons tout d'abord la fonction de coût $\mathcal{L}_{ctc}$ à minimiser :

$$\mathcal{L}_{\text{ctc}} \triangleq -\log p(l^*|x) \tag{3.22}$$

$$\text{où } p(l|x) = \sum_{u=1}^{|l'|} \frac{\alpha_t(u)\beta_t(u)}{y_{l'_u}^t} \tag{3.23}$$

Afin de rétropropager au travers du réseau, nous devons calculer le gradient de la fonction $\mathcal{L}_{\text{ctc}}$ par rapport aux probabilités de sortie non normalisées $z_k^t$ de la manière suivante:

$$\frac{\partial \mathcal{L}_{\text{ctc}}}{\partial z_k^t} = y_k^t - \gamma_k^t \tag{3.24}$$

$$\text{où } \gamma_k^t \triangleq p(u_t = k|l', \lambda) = \frac{1}{y_k^t \, p(l|x)} \sum_{u \in \text{lab}(l,k)} \alpha_t(u)\beta_t(u) \tag{3.25}$$

Où $\text{lab}(l, k)$ définit l'ensemble des positions où l'étiquette k apparaît comme suit : $\text{lab}(l, k) = u : l'_u = k$, et qui est utilisé pour la différentiation de $p(l|x)$ par rapport à $y_{l'_u}^t$, du fait que la même étiquette (incluant $\varnothing$) peut être répétée plusieurs fois pour un même étiquetage $l$. Le calcul des variables *forward* $\alpha_t(u)$ et *backward* $\beta_t(u)$ est quant à lui effectué par récurrence, en considérant l'initialisation suivante pour chaque variable:

$$\alpha_1(1) = y_\varnothing^1, \quad \alpha_1(2) = y_{l_1}^1, \quad \forall u > 2 : \alpha_1(u) = 0 \tag{3.26}$$

$$\beta_T(|l'|) = y_\varnothing^T, \quad \beta_T(|l'| - 1) = y_{l_{|l|}}^T, \quad \forall u < |l'| - 1 : \beta_T(u) = 0 \tag{3.27}$$

Une fois l'initialisation effectuée, le calcul de $\alpha_t(u)$ $\forall u : 1 < u < |l'|$ et le calcul de $\beta_t(u)$ $\forall u : |l'| < u \leq 1$ peuvent être effectués récursivement via les équations suivantes:

$$\alpha_t(u) = \begin{cases} [\alpha_{t-1}(u-1) + \alpha_{t-1}(u)] \; y_{l'_u}^t & \text{si } l'_u = \varnothing \text{ ou } l'_{u-2} = l'_u \\ [\alpha_{t-1}(u-2) + \alpha_{t-1}(u-1) + \alpha_{t-2}(u)] \; y_{l'_u}^t & \text{sinon} \end{cases} \tag{3.28}$$

$$\beta_t(u) = \begin{cases} [\beta_{t+1}(u-1) + \beta_{t+1}(u)] \; y_{y'_u}^t & \text{si } l'_u = \varnothing \text{ ou } l'_{u+2} = l'_u \\ [\beta_{t+1}(u-2) + \alpha_{t+1}(u-1) + \alpha_{t+1}(u)] \; y_{l'_u}^t & \text{sinon} \end{cases} \tag{3.29}$$

Dans le cas de $\alpha_t(u)$, la terminaison est déterminée par :

$$p(l|x) = \alpha_T(|l'|) + \alpha_T(|l'| - 1) \tag{3.30}$$

#### 3.2.1.2 Inférence

Une fois le modèle entraîné, le processus de décodage peut être effectué de deux manières: via un algorithme glouton ou via un algorithme de recherche en faisceau. Dans le premier cas, l'algorithme suppose que le chemin le plus probable correspond à l'étiquetage le plus probable $\mathcal{B}(\pi^*)$ où $\pi^* = \arg\max p(\pi|x)$, c'est à dire la concaténation des activations des sorties à chaque pas de temps, avec les répétitions consécutives enlevées et les étiquettes $\varnothing$ enlevés (dans cet ordre).

Dans le deuxième cas, une recherche par préfixe [Graves et al., 2006], est ajoutée au processus de recherche en faisceau afin de traiter les alignements multiples qui correspondent à une même séquence. Dans ce cas, à chaque étape de la recherche et étant donnée une séquence $l$ d'un ensemble de recherche $A$, les probabilités de chaque séquence de $A$ sont modifiées en fonction des préfixes trouvés, tel que :

$$p(l_i) = p(l_i) + \sum_{\hat{l}} p(\hat{l}) * p(l_i|\hat{y}) \tag{3.31}$$

Où $\hat{l} \in pref(l_i) \cap A$.

Parallèlement, il est d'usage, dans le cadre de la phase d'inférence, d'utiliser un modèle de langage externe afin de rajouter les informations concernant l'inter-dépendance entre les étiquettes pour améliorer les performances du modèle. Pour cela, la probabilité du LM est ajoutée à chaque extension de la séquence si le LM modélise les mêmes unités que le modèle CTC (p. ex., des caractères). Sinon, un algorithme multi-niveau est utilisé [Hori et al., 2017a].

### 3.2.2 RNN-Transducer

L'approche RNN-Transducer a été introduite pour la première fois par [Graves, 2012b] pour remédier à la principale limitation de l'approche CTC proposée précédemment par le même auteur : celle-ci ne peut pas modéliser les inter-dépendances entre les étiquettes car une indépendance conditionnelle entre les prédictions à différents pas de temps est supposée. Afin de résoudre ce problème, les auteurs proposent une approche schématisée par la figure 3.5. Celle-ci s'appuie sur l'utilisation de trois réseaux ayant chacun un rôle déterminé.

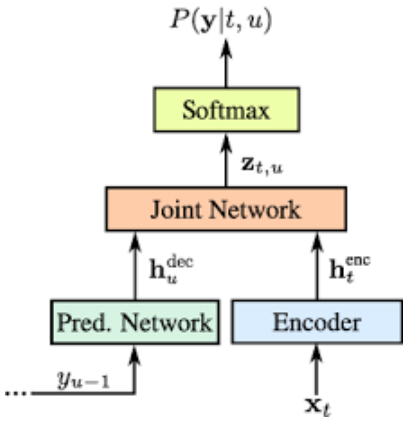



Figure 3.5: Structure d'un RNN-Transducer [Wang et al., 2019].

Avant de les décrire, définissons tout d'abord une séquence d'entrée $x = (x_1, ..., x_T)$ et une séquence $y = (y_1, ..., y_U)$ comme appartenant respectivement à l'ensemble $\mathcal{X}*$ de toutes les séquences sur un espace d'entrée $\mathcal{X}$ et l'ensemble $\mathcal{Y}*$ de toutes les séquences sur un espace de sortie $\mathcal{L}$. Comme pour la CTC, l'étiquette spéciale $\varnothing$ est incorporée afin de définir un ensemble de sortie étendu $\mathcal{Y}' = \mathcal{Y} \cup \varnothing$. Connaissant la séquence $x$, le modèle va définir une distribution conditionnelle $p(a \in \mathcal{Y}'^*|x)$ où $a$ désigne un alignement, nommé ainsi du fait que la position de $\varnothing$ permet de déterminer l'alignement entre la séquence d'entrée et de sortie. La distribution est ensuite transformée au travers d'une fonction $\mathcal{B} : Y'^U \rightarrow Y^{\leq U}$ qui enlève l'étiquette nulle :

$$p(y \in \mathcal{Y}^*|x) = \sum_{a \in \mathcal{B}^{-1}(y)} p(a \in \mathcal{Y}'^*|x) \tag{3.32}$$

Où $p(a \in \mathcal{Y}'^{*}|x)$ est obtenu au travers des réseaux suivants.

Le premier réseau, en vert sur la figure 3.5, est appelé réseau de prédiction (*prediction network* en anglais), ou Décodeur, et peut être vu comme analogue à un modèle de langage prédisant l'étiquette suivante. Le réseau prend en entrée un vecteur $\hat{y} = \varnothing, y_1, ..., y_U)$ de longueur $U + 1$. Celui-ci est tout d'abord transformé en une représentation *one-hot* où chaque élément de $\hat{y}$ est transformé en un vecteur de $\mathcal{K}$ dimension, où $K$ désigne toutes les étiquettes existantes de $\mathcal{Y}$. Si $y_u = k$, le vecteur résultant est constitué de valeurs nulles sauf pour la $k^{\text{ième}}$ dimension contenant 1. Dans le cas de $\varnothing$, l'étiquette est encodée comme un vecteur de $K$ dimensions ne contenant que des valeurs nulles. La représentation est ensuite passée au travers d'un réseau de neurones récurrents générant une séquence $h^{\text{dec}} = (h_0^{\text{dec}}, ..., h_U^{\text{dec}})$ de longueur $U + 1$. Pour cela, les équations 3.1 et 3.2 sont utilisées avec $h_u^{\text{dec}}$ et $y_u$ remplaçant respectivement $h_t$ et $p_t$. La séquence $h^{\text{dec}}$ est obtenue en commençant à $u = 0$ et en itérant récursivement les équations, avec $u$ incrémenté à chaque itération.

Le second réseau, nommé Encodeur (ou *Encoder* en anglais, en bleu sur la figure 3.5), peut être considéré comme un équivalent au réseau CTC sans la couche de sortie Softmax. Celui-ci encode la séquence d'observations acoustiques $x$ afin de générer une représentation $h_{\text{enc}} = (h_1^{\text{enc}}, ..., h_T^{\text{enc}})$. Pour cela, un réseau de neurones récurrent bidirectionnel est employé, générant la séquence par itération de $t = 1$ à $T$ au travers des représentations cachées avant et arrière (voir section 3.1.3.1). Comme pour le réseau précédent, la couche de sortie de l'Encodeur considère l'étiquette nulle, générant des vecteurs $h_t^{enc}$ de taille $K + 1$.

Le dernier composant de l'approche, en orange sur la figure 3.5, est intitulé réseau conjoint (ou *joint network* en anglais). Celui-ci combine les représentations sortantes du premier et second réseau afin de générer, par passage au travers d'une fonction d'activation *Act* (p. ex., une tangente hyperbolique) et d'une couche linéaire, la représentation $h_{joint}$ de dimension $(T, U+1, |Y'|)$. Connaissant une étiquette $k \in \mathcal{Y}'$, la fonction de densité suivante peut être définie correspondant à la sortie du réseau, où $k$ en tant qu'exposant désigne le $k$-ième élément du vecteur correspondant:

$$h_{k,t,u}^{joint} = \text{Act}(h_t^{enc^k} + h_u^{dec^k}) \tag{3.33}$$

Ce qui permet de générer la distribution conditionnelle suivante, par normalisation de la fonction de densité précédente :

$$p(k \in \mathcal{Y}'|t, u) = \frac{h_{k,t,u}^{joint}}{\sum_{k' \in \mathcal{Y}'} h_{k',t,u}^{joint}} \tag{3.34}$$

$p(k|t, u)$ est utilisé par le modèle afin de déterminer les probabilités de transition dans un treillis, comme par exemple celui de la figure 3.6, où $y(t, u)$ et $\varnothing(t, u)$ est définit par les équivalences suivantes :

$$y(t, u) \equiv p(y_{u+1|t,u}) \tag{3.35}$$

$$\varnothing(t, u) \equiv p(\varnothing|t, u) \tag{3.36}$$

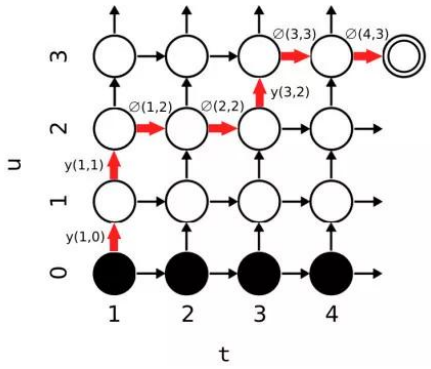



Figure 3.6: Probabilité de sortie d'un treillis défini par $p(k|t,u)$ [Graves, 2012b]. Le chemin en rouge, commençant au noeud en bas à gauche et terminant au noeud terminal en haut à droite, désigne un alignement possible entre la séquence d'entrée et la séquence de sortie. Chaque alignement commence avec une probabilité de 1 et la probabilité finale correspond au produit des probabilités de transition du chemin.

L'ensemble des alignements entre une séquence $x$ et une séquence $y \in \mathcal{Y}^*$, c.-à-d. l'ensemble $Y'^* \cap \mathcal{B}^{-1}(y)$, correspond ici à l'ensemble des chemins possibles dans le treillis (voir description figure 3.6). La somme des probabilités des alignements correspondant à $p(y|x)$ comme définit par l'équation 3.32. Ainsi, puisqu'un treillis similaire peut être obtenu $\forall y \in Y^*$ fini, $p(k|t,u)$ définit donc une distribution sur toutes les séquences de sortie possibles, étant donné une séquence d'entrée $x$.

#### 3.2.2.1 Entraînement

Étant donné une séquence d'entrée $x$ et une séquence cible $y^*$, définissons tout d'abord la fonction de coût à minimiser pour l'entraînement du modèle :

$$\mathcal{L}_{\text{rnnt}} = -\log p(y^*|x) \tag{3.37}$$

$$\text{Où } p(y^*|x) = \sum_{(t,u):t+u=n} \alpha(t,u)\beta(t,u) \tag{3.38}$$

Les variables forward $\alpha$ et backward $\beta$ définissent respectivement ici la probabilité d'émettre $y_{1:u}$ durant $h^{\text{enc}}_{1:t}$ et la probabilité d'émettre $y_{u+1:U}$. Ces variables sont initialisés par, respectivement, $\alpha(1,0) = 1$ et $\beta(T,U) = \varnothing(T,U)$. La variable $\alpha$ au noeud terminal $(t,u)$ permettant d'obtenir la probabilité totale de la séquence de sortie :

$$p(y^*|x) = \alpha(T,U)\varnothing(T,U) \tag{3.39}$$

Les variables $\alpha$ et $\beta$ sont ensuite calculées récursivement par les équations suivantes :

$$\alpha(t,u) = \alpha(t-1,u)\varnothing(t-1,u) + \alpha(t,u-1)y(t,u-1) \tag{3.40}$$

$$\beta(t,u) = \beta(t+1,u)\varnothing(t,u) + \beta(t,u+1)y(t,u-1) \tag{3.41}$$

Afin d'effectuer la rétroprogation, nous devons tout d'abord définir le gradient de $\mathcal{L}_{\text{rnnt}}$ par rapport au postérieur final $p(k|t,u)$ pour une étiquette $k \in \mathcal{Y}$ :

$$\frac{\partial \mathcal{L}_{\text{rnnt}}}{\partial p(k|t,u)} = -\frac{\alpha(t,u)}{p(y^*|x)} \begin{cases} \beta(t,u+1) & \text{si } k = y_{u+1} \\ \beta(t+1,u) & \text{si } k = \varnothing \\ 0 & \text{sinon} \end{cases} \tag{3.42}$$

Ce qui permet de calculer le gradient de $\mathcal{L}_{\text{rnnt}}$ par rapport aux sorties respectives des réseaux $h^{\text{enc}}$ et $h^{\text{dec}}$ :

$$\frac{\partial \mathcal{L}_{\text{rnnt}}}{\partial h_t^{\text{enc}^k}} = \sum_{u=0}^{U} \sum_{k' \in \mathcal{Y}'} \frac{\partial \mathcal{L}_{\text{rnnt}}}{\partial p(k'|t,u)} \frac{\partial p(k'|t,u)}{\partial h_t^{\text{enc}^k}} \tag{3.43}$$

$$\frac{\partial \mathcal{L}_{\text{rnnt}}}{\partial h_u^{\text{dec}^k}} = \sum_{t=1}^{T} \sum_{k' \in \mathcal{Y}'} \frac{\partial \mathcal{L}_{\text{rnnt}}}{\partial p(k'|t,u)} \frac{\partial p(k'|t,u)}{\partial h_u^{\text{dec}^k}} \tag{3.44}$$

Où, à partir de l'équation 3.34:

$$\frac{\partial p(k'|t,u)}{\partial h_t^{\text{enc}^k}} = \frac{\partial p(k'|t,u)}{\partial h_u^{\text{dec}^k}} = p(k'|t,u)[\delta_{kk'} - p(k|t,u)] \tag{3.45}$$

Les gradients par rapport aux poids de chaque réseau du modèle peuvent ensuite être calculés indépendamment en utilisant une rétroprogation au travers du temps.

#### 3.2.2.2 Inférence

Habituellement, une recherche en faisceau, admettant une recherche par préfixe comme pour la CTC [Graves et al., 2006], est utilisée pour le processus de décodage des modèles RNN-Transducer. Toutefois, contrairement à la CTC qui est synchrone dans le temps, l'algorithme proposé par [Graves, 2012b] admet ici deux types de transitions, à travers des axes $T$ et $U$, et peut donc émettre plusieurs étiquettes pour un même pas de temps $t$ :

**Algorithme 2 :** Recherche en faisceau pour le modèle RNN-Transducer [Graves, 2012b]

**Entrées :** $h^{\text{enc}}_{1:T}$, $N_{\text{bs}}$ and $N_{\text{best}}$
**Initialisation :** $B = \{(\varnothing, 1, \text{state}^{\text{dec}}_0)\}$
**pour** $t$ = *1 ... T* **faire**
  $A = \text{SortedByLength}(B)$
  $B = \{\}$
  **pour** $(y_i, \delta(y_i)) \in A$ **faire**
    $\delta(y_i) \mathrel{+}= \sum_{\hat{y}} \delta(\hat{y}) * \delta(y_i|\hat{y}, t)$
      **where** $\hat{y} \in pref(y_i) \cap A$
  **fin**
  **tant que** $B$ *contient moins de* $N_{bs}$ *éléments plus probables que l'élément le plus probable de* $A$ **faire**
    $A_{\max} = \text{ExtractMostProb}(A)$
      où $A_{\max} = (y, \delta_{u-1,t-1}(y), \text{state}^{\text{dec}}_{u-1})$
    $\text{DeleteFromSet}(A,\ A_{\max})$
    $h^{dec}_u, \text{state}^{\text{dec}}_u = \text{Decoder}(y, \text{state}^{\text{dec}}_{u-1})$
    $p^{\text{pr}} = \text{Softmax}(\text{JointNetwork}(h^{\text{enc}}_t, h^{\text{dec}}_u))$
    $\delta_{t,u-1}(y) = \delta_{t-1,u-1}(y) + p^{\text{pr}}(\varnothing)$
    $A = A \cup \{(y_{u-1}, \delta_{t,u-1}(y), \text{state}^{\text{dec}}_{u-1})\}$
    **pour** $k \in \mathcal{Y}$ **faire**
      $\delta_{t-1,u}(y+k) = \delta_{t-1,u-1}(y) + p^{\text{pr}}(k)$
      $B = B \cup \{(y+k, \delta_{t-1,u}(y+k), \text{state}^{\text{dec}}_u)\}$
    **fin**
  **fin**
  $B = \text{SortedByScore}(B)[:N_{\text{bs}}]$
**fin**
**Retourne:** $\text{SortedByScore}(B)[:N_{\text{best}}]$

Où un ensemble d'hypothèses $X$ est désigné par un triplet contenant: la séquence d'étiquettes $y$, le score de la séquence $\delta_x(\cdot)$ pour l'alignement $x$ et l'état caché du décodeur $h^{\text{dec}}_u$ pour un pas de $u$. $\mathcal{Y}$ définit l'ensemble des étiquettes existantes, où $k \in \mathcal{Y}$. $N_{\text{bs}}$ et $N_{\text{best}}$ désignent respectivement le nombre d'éléments du faisceau et le nombre d'hypothèses à retourner.

L'algorithme est exécuté en utilisant deux ensembles d'hypothèses $A$ et $B$, respectivement l'ensemble des hypothèses considérées au pas de temps $t$, contenant l'hypothèse $\varnothing$ à $t = 0$, et l'ensemble des hypothèses pour $t + 1$, se terminant par une transition nulle au pas de temps $t$. À chaque pas de temps $t$, les hypothèses de $A$ sont d'abord déplacées vers $B$. La meilleure hypothèse de $B$ est extraite et étendue avec une transition nulle ou une transition non-nulle. Les hypothèses se terminant par une transition nulle sont stockées dans $B$ tandis que les autres sont envoyés vers $A$, où le score de chaque hypothèse est mis à jour avec le score de la transition nulle ou non-nulle correspondant. La procédure est ensuite répétée jusqu'à ce que $B$ contienne au moins $N_{\text{bs}}$ hypothèses plus probables que la plus probable dans $A$, où bs désigne la taille du faisceau choisi. Lorsque la condition est remplie, $B$ est élagué à $N_{\text{bs}}$ hypothèses et passé à $t + 1$. À la fin de la procédure, les N meilleures hypothèses de $B$ sont retournées triées par score décroissant.

Des algorithmes de recherche alternatifs sont depuis apparus se proposant de contrôler l'extension d'étiquettes de différentes manières [Saon et al., 2020, Tripathi et al., 2019].

L'objectif principal de ces méthodes étant de supprimer l'une des limitations de l'algorithme standard qui est l'impossibilité de vectoriser les hypothèses d'un même faisceau du fait de la présence de la boucle *Tant que*. Ces algorithmes seront abordés et décrits dans la section 5.4.4. J'introduirai en outre un algorithme de recherche synchrone dans le temps, développé dans le cadre du projet ESPnet.

### 3.2.3 Encodeur-Décodeur basé attention

Basé sur la technique Encodeur-Décodeur utilisé alors pour résoudre le problème de la traduction automatique de texte [Sutskever et al., 2014, Cho et al., 2014], l'approche "Encodeur-Décodeur basé attention" fut proposée pour la première fois par [Bahdanau et al., 2014] afin que le modèle apprenne à aligner et traduire simultanément grâce à l'introduction d'un mécanisme d'attention. Les premières applications de cette approche pour le problème de la RAP se focalisent tout d'abord sur la modélisation phonétique [Chorowski et al., 2014, Chorowski et al., 2015, Bahdanau et al., 2016] puis pour la modélisation de séquences orthographiques ensuite, avec notamment *Listen, Attend and Spell* [Chan et al., 2016] que la figure 3.7 décrit.

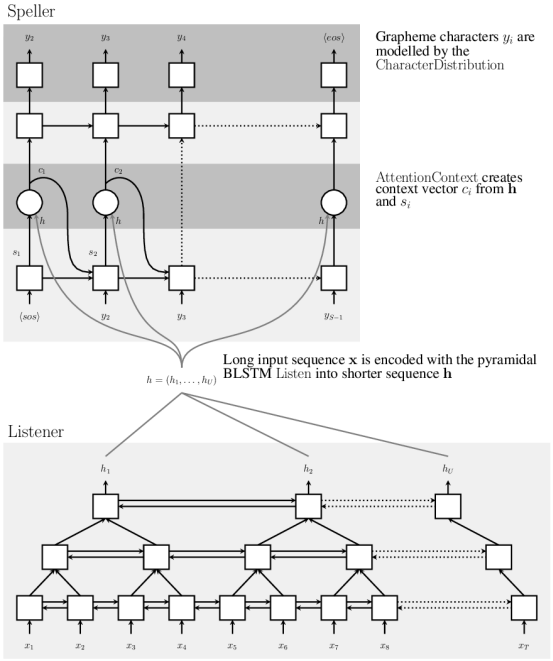



Figure 3.7: Modèle Listen, Attend and Spell (LAS) proposé par [Chan et al., 2016]. Le *Listener*, en bas, encode une séquence $x$ en une représentation haut-niveau $h^{\text{enc}}$. Le *Speller*, en haut, génère la séquence orthographique $y$ à partir de $h^{\text{enc}}$ et à l'aide d'un mécanisme d'attention.

Contrairement à la CTC, l'approche Encodeur-Décodeur basé attention (abrégée ici Attention), ne suppose pas l'indépendance conditionnelle entre les prédictions à différents pas de temps et ne nécessite pas de représentation intermédiaire. Ainsi, la distribution postérieure $p(y|x)$ est directement calculée en modélisant chaque étiquette $y_u$ comme une distribution conditionnelle sur les étiquettes précédentes $y_{1:u-1}$ et la séquence d'observations acoustiques $x$, en utilisant la règle de dérivation en chaîne :

$$p(y|x) = \prod_{u=1}^{U} p(y_u|y_{1:u-1}, x) \tag{3.46}$$

Ici, $p(y|x)$ est obtenu au travers de deux réseaux de neurones récurrents:

- L'Encodeur, similaire au module du même nom dans l'approche CTC et RNN-Transducer, encode une séquence d'observations acoustiques $x_{1:T^{\text{input}}}$ en une représentation haut-niveau $h^{\text{enc}}_{1:T}$ où $T \leq T^{\text{input}}$
- Le DécodeurAttention qui produit la distribution de probabilités $y_u$ conditionnée par $h^{enc}_{1:T}$ et les étiquettes observées précédemment $y_{1:u-1}$.

Ou plus formellement, $p(y|x)$ est calculé via $y_u$ obtenu selon les équations récursives suivantes :

$$h^{\text{enc}} = \text{Encodeur}(x) \tag{3.47}$$
$$y_u \sim \text{DécodeurAttention}(h^{enc}, y_{1:u-1}) \tag{3.48}$$

La partie Encodeur ayant été décrite précédemment dans le cadre des autres approches, intéressons nous à la partie DécodeurAttention. Celle-ci correspond à un réseau de neurones récurrents unidirectionnel incorporant un mécanisme d'attention, suivi d'une couche de sortie Softmax. L'activation d'un état caché au pas de temps $u$ étant défini par l'équation suivante :

$$s_u = \text{Récurrence}(s_{u-1}, c_u, y_{u-1}) \tag{3.49}$$

Où $s_u$ et $s_{u-1}$ définissent les états cachés pour les pas de temps $u$ et $u-1$, $y_{u-1}$ correspond à l'étiquette de sortie pour le pas de temps $u-1$. $c_u$ est ici un vecteur de contexte, aussi appelé *glimpse*, correspondant à la somme pondérée des sorties de l'Encodeur :

$$c_u = \sum_t \alpha_{u,t} h_t \tag{3.50}$$

Les poids, ou matrice d'attention, $\alpha_{u,t}$ associés à chaque sortie de l'Encodeur étant obtenus par le calcul suivant :

$$\alpha_{u,t} = \frac{\exp(\gamma e_{u,t})}{\sum_t \exp(\gamma e_{u,t})} \tag{3.51}$$

Où $e_{u,t}$, l'énergie, mesure la correspondance entre les entrées autour de la position $t$ et la sortie à la position $u$. Cette mesure peut être effectuée de différente manière, en se basant sur le contenu seulement [Bahdanau et al., 2014] ou en incorporant des informations de localisation [Chorowski et al., 2015] par le biais des alignements produits au pas précédents. Dans les deux cas, un réseau de neurones *feed-forward* est utilisé prenant en entrée l'état caché $s_{u-1}$ du Décodeur RNN avant émission de $y_{u-1}$ et la $t^{\text{ième}}$ de la séquence encodé $h^{enc}_t$ :

$$e_{u,t} = \begin{cases} \text{basé contenu :} \\ \quad w^T \tanh(W s_{u-1} + V h_t^{\text{enc}} + b) \\ \text{basé position :} \\ \quad f_u = F \star \alpha_{u-1} \\ \quad w^T \tanh(W s_{u-1} + V h_t^{\text{enc}} + U f_{u,t} + b) \end{cases} \tag{3.52}$$

Où $\star$ désigne une opération de convolution, tanh une fonction d'activation de type tangente hyperbolique et $s_{u-1}$ l'état du décodeur pour le pas $u-1$. $w, W, V, F, U$ et $b$ désignent quant à eux des paramètres entraînables. $w$ et $b$ sont des vecteurs tandis que $W, V, F, U$, sont des matrices, respectivement: la matrice pour l'état du décodeur $s_{u-1}$, la matrice pour l'entrée $h^{\text{enc}}$, la matrice de filtres de convolution et enfin la matrice pour le vecteur de convolution pour la position (u, t).

La sortie $y_u$ pour le pas $u$ est ensuite calculée par une fonction *Generate* définissant un réseau de neurones *feed-forward* prenant en entrée le vecteur $c_u$ et l'état caché $s$ pour le pas $u-1$, et où Act désigne une fonction d'activation :

$$y_u \sim \text{Generate}(c_u, s_{u-1}) = \text{Act}(W_c c_u + W_s s_{u-1}) \tag{3.53}$$

#### 3.2.3.1 Entraînement

Étant donné une séquence d'entrée $x$ et une séquence cible $y^*$, la fonction de coût à minimiser dans le cadre de l'entraînement du modèle Attention est définie comme telle :

$$\begin{aligned} \mathcal{L}_{\text{att}} &= -ln\, p(y^*|x) \\ &\text{Où } p(y^*|x) = p(y_u^*|x, y_{1:u-1}^*) \end{aligned} \tag{3.54}$$

Où $y_{1:U}^*$ est ici une étiquette appartenant à l'ensemble $\mathcal{Y} \cup \{\text{sos/eos}\}$. L'étiquette spéciale sos/eos, pour *start-of-sentence/end-of-sentence*, que nous n'avons pas encore introduite, est une composante primordiale des architectures Encodeur-Décodeur. Le décodeur progressant en utilisant des étiquettes émises précédemment en entrée (avec la représentation $h^{enc}$), celui-ci a besoin d'une étiquette initiale pour initier le processus. Parallèlement, si nous voulons que le Décodeur puisse émettre des séquences de longueur arbitraire, celui-ci doit pouvoir indiquer quand il a fini d'émettre des étiquettes. L'étiquette sos/eos est donc ajouté à l'ensemble des étiquettes pour ces raisons et s'utilise comme étiquette de départ de la séquence d'entrée, $y_0$ et comme étiquette de fin de séquence de sortie $y_{|U|}^*$.

L'architecture Encodeur-Décodeur peut être vue comme un réseau complet et les deux parties peuvent donc être entraînées conjointement par rétropropagation. Le gradient circule donc de la sortie du décodeur, où $\mathcal{L}_{\text{att}}$ est calculé, vers les poids d'entrée de l'Encodeur, où la séquence $x$s est consommée. Lors de la phase d'entraînement, la séquence d'observations $x$ est tout d'abord introduite à l'Encodeur afin de générer la représentation $h^{\text{enc}}$. Celle-ci est ensuite passée au Décodeur avec la séquence cible. Il suffit ensuite de calculer la fonction de coût à chaque pas de temps, d'en faire la somme et d'effectuer la rétropropagation comme nous l'avons présenté dans la section 3.1. Les poids d'attention étant traités comme les poids de la couche canonique.

#### 3.2.3.2 Inférence

Contrairement à la CTC et à RNN-Transducer, le mécanisme d'attention aligne de manière déterministe les séquences d'entrée et de sortie, ce qui permet de simplifier la procédure de recherche. Ainsi, étant donné une séquence $h^{\text{enc}}$ et une étiquette sos en entrée, il nous suffit de prédire l'étiquette la plus probable pour chaque pas $u$ jusqu'à ce que eos soit prédit. Parallèlement, une recherche en faisceau standard basé, sur la probabilité maximale, peut être utilisée en gardant la trace de $N$ hypothèses à chaque pas de temps.

Toutefois, cette simplification s'accompagne de plusieurs problèmes. Tout d'abord, puisque n'importe quelle partie de la séquence $h^{\text{enc}}$ peut être utilisée pour la prédiction de l'étiquette suivante, le Décodeur peut prédire prématurément une étiquette sos. Parallèlement, celui-ci peut prédire de manière répétée la même étiquette, en se focalisant sur la même partie de $h^{\text{enc}}$ que pour les prédictions précédentes, ce qui peut résulter en de longues séquences d'étiquettes. Deuxièmement, la recherche en faisceau va favoriser les séquences de petite taille, du fait que les probabilités des étiquettes sont multipliées à chaque pas de temps, rendant les longues séquences peu probables en comparaison. Pour finir, la procédure d'inférence ne peut pas être effectuée en temps réel, puisque la séquence d'entrée complète doit être disponible avant que le décodeur puisse l'utiliser.

Afin de palier à ces problèmes, différentes techniques ont été introduites depuis comme par exemple l'utilisation de termes de couverture [Wu et al., 2016, Chorowski and Jaitly, 2017] pour traiter le premier problème ou encore l'incorporation de la normalisation par longueur de séquence pour le second problème [Wu et al., 2016]. Concernant le décodage en temps réel, différentes méthodes ont vu le jour afin d'entraîner le modèle à procéder la séquence $h^{\text{enc}}$ par portion en considération du mécanisme d'attention utilisé [Chiu and Raffel, 2018, Kim et al., 2019].

## 3.3 Autres approches et architectures notables

En marge des réseaux de neurones récurrents et des approches de RAP bout-en-bout présentées précédemment, de nouvelles techniques sont venues, au cours de ma thèse, enrichir l'état de l'art en RAP. Dans cette section, j'introduis deux travaux dont je me suis servi durant ma thèse et ayant contribué à faire avancer considérablement le domaine durant cette période : l'entraînement multi-tâches combinant CTC et Attention tout d'abord, et l'architecture Transformer venue remplacer les RNNs pour finir.

### 3.3.1 Approche *Joint CTC-Attention*

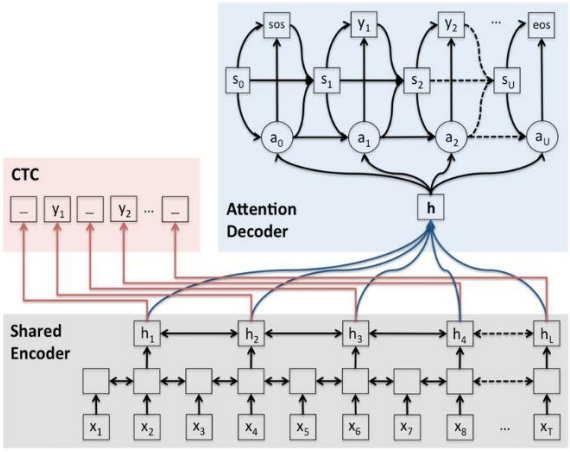



Figure 3.8: Architecture *Joint CTC Attention* [Kim et al., 2017]. L'encodeur est entraîné simultanément par la CTC et la fonction de coût du modèle Attention.

L'idée derrière l'apprentissage conjoint de la CTC et de la modélisation basée attention proposée par [Kim et al., 2017] est relativement simple. En entraînant simultanément l'Encodeur avec le mécanisme d'attention et la fonction de coût CTC comme tâche auxiliaire, des alignements monotones entre les séquences d'observations acoustiques et les séquences d'étiquettes de sortie peuvent être appliqués. Ainsi les problèmes d'alignement décrits dans la section 3.2.3.2 peuvent être réduits sans avoir recours à différentes adaptations du processus de décodage.

Pour se faire, une architecture Encodeur-Décodeur basée attention est utilisée, décrite par la figure 3.8. L'Encodeur est connecté simultanément au Décodeur (en bleu sur la figure) et à une couche de sortie $Softmax$ pour la modélisation CTC (en rouge sur la figure).

#### 3.3.1.1 Entraînement

La fonction objective à maximiser durant l'apprentissage conjoint, ou multi-tâches (abrégé MTL) du modèle, est donnée par l'équation suivante : où $\lambda : 0 \leq \lambda \leq 1$ est un paramètre modifiable pondérant la contribution de chaque méthode :

$$\mathcal{L}_{MTL} = \lambda \mathcal{L}_{ctc} + (1 - \lambda)\mathcal{L}_{att} \tag{3.55}$$

$$= \lambda \log p_{ctc}(y|x) + (1 - \lambda) \log p_{att}(y|x) \tag{3.56}$$

La phase d'entraînement consiste donc à simplement à utiliser le processus décrit dans la section pour l'Encodeur-Décodeur et le processus *forward-backward CTC* décrit dans la section 3.2.1.1 en considérant les poids donnés à chaque approche.

#### 3.3.1.2 Inférence

Initialement, les algorithmes de recherche utilisés pour l'approche Encodeur-Décodeur basé attention 3.2.3.2 étaient aussi employés pour le décodage des modèles *Joint CTC-Attention*, et ceux-ci n'incluaient donc pas les prédictions de la CTC. Proposée à la suite,

Hori et al. [Hori et al., 2017b] introduisent donc une méthode de décodage conjointe afin de considérer les prédictions de la CTC dans le processus de décodage du modèle Attention. Compte tenu de la difficulté de combiner les scores respectifs, le décodeur basé sur l'attention effectuant la recherche de faisceau de manière synchrone avec les caractères et la CTC l'effectuant de manière synchrone avec les trames, deux méthodes ont été proposées.

La première méthode est un processus de décodage simple en deux passes. Lors de la première passe, les hypothèses complètes sont tout d'abord obtenues en utilisant une recherche gloutonne ou recherche par faisceau et le modèle Attention seul. Les scores d'hypothèses sont ensuite recalculés avec les probabilités données par les deux méthodes, où les probabilités de la CTC sont obtenus via l'algorithme *forward* décrit dans la section 3.2.1. Le résultat de la seconde passe est défini par l'équation suivante, où $\lambda$ est une valeur pondérant la contribution de la CTC par rapport au modèle Attention :

$$\hat{y} = \arg\max_{y \in \mathcal{Y}*} \{\lambda \log\ p_{\text{ctc}}(y|x) + (1-\lambda) \log\ p_{\text{att}}(y|x)\} \tag{3.57}$$

Ou, lors de la recherche, le Décodeur calcule un score $\alpha$ pour chaque hypothèse partielle. Avec le modèle Attention, celui-ci étant calculé récursivement avec l'équation suivante, où $p_u$ désigne l'hypothèse partielle et $e$ définissant la dernière étiquette de $p_u$ (c.-à-d., $p_u = g_{u-1} \cdot e$) :

$$\alpha_{\text{att}}(p_u) = \alpha_{\text{att}}(p_{u-1}) + \log p(e|p_{u-1}, x) \tag{3.58}$$

La deuxième méthode est une approche en seule passe. Celle-ci s'appuie sur l'utilisation de la probabilité de préfixe CTC [Graves, 2012a] qui est définie comme la probabilité cumulée de toutes les séquences d'étiquettes ayant $p_u$ comme préfixe :

$$p(p_u, ...|x) = \underset{v \in (\mathcal{Y} \cup \{\text{eos}\})^+}{P} (p_u \cdot v|x) \tag{3.59}$$

Où $v$ représente toutes les séquences d'étiquettes possibles avec l'étiquette de fin de séquence eos et sans l'étiquette nulle $\varnothing$.

Le score de CTC ne pouvant être obtenu récursivement comme pour le modèle Attention, celui-ci est calculé en conservant les probabilités *forward* sur la séquence de trames d'entrée pour chaque hypothèse partielle. Le score $\alpha_{\text{ctc}}$ peut ensuite être combiné avec $\alpha_{\text{att}}$ en pondérant avec $\lambda$, comme pour l'équation 3.57.

### 3.3.2 Architecture Transformer

L'architecture Transformer a été introduite pour la première fois par un ensemble de chercheurs de Google Brain et Google Research [Vaswani et al., 2017] pour différentes tâches de traitement automatique du langage naturel. Cette architecture est maintenant utilisée massivement pour la majorité des tâches transformant une séquence d'entrée de longueur donnée en une autre séquence, comme la reconnaissance automatique de parole [Dong et al., 2018].

Initialement, un Transformer est présenté comme une architecture de type Encodeur-Décodeur (3.9) et peut être vu comme une évolution naturelle de l'architecture Encodeur-Décodeur basé attention présentée dans la section 3.2.3. L'idée étant ici de remplacer les couches de neurones récurrents par des couches de type *feed-forward* associées à des mécanismes d'attention afin de capturer les dépendances temporelles dans les séquences. Ce faisant, le modèle peut ainsi palier aux problèmes de dépendance à long terme (en traitant des séquences entières plutôt que étiquette par étiquette) et réduire le risque de perte d'information en se servant de techniques décrites en suivant, permettant d'apprendre les relations entre les étiquettes.

Dans cette partie, je me focalise à présenter les composants principaux de la partie Encodeur de l'architecture. La raison étant que, contrairement à la proposition de base, la partie Décodeur peut admettre plusieurs formes selon l'approche bout-en-bout utilisée en association. Pour la CTC par exemple, la partie Décodeur est exclue. Dans le cadre de l'approche RNN-Transducer, la partie Décodeur est indépendante et n'incorpore pas les informations de l'Encodeur. Ainsi, on préfère de nos jours distinguer l'architecture Transformer, dont je vais présenter les composants, de l'architecture Transformer complète 3.9) que nous pouvons considérer comme un système ou une approche à part entière.

Parallèlement, les notions relatives à l'entraînement et l'inférence seront omises pour ces mêmes raisons, celles-ci dépendant de l'approche et de la structure utilisées.

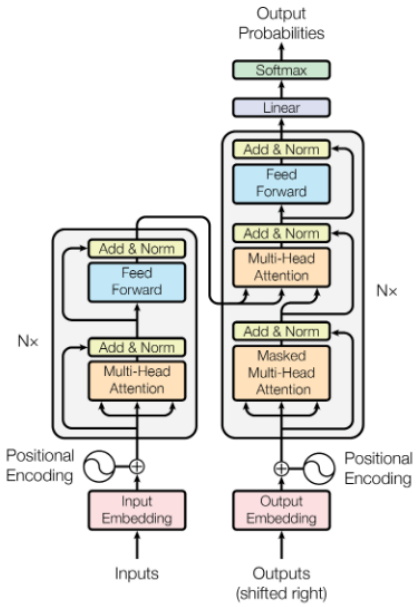



Figure 3.9: Structure d'une architecture Transformer. *Google*, 2017

L'architecture Transformer repose en pratique sur l'utilisation de trois composants : un module d'encodage de position, des couches de *feed-forward* par position et un mécanisme d'attention à plusieurs têtes. Les deux derniers composants, associés à différentes opérations, représentent ce qu'on appelle une couche Transformer et peuvent être répétées $N$ fois. L'entrée d'un bloc $N_i$ correspond ici à la sortie d'un bloc $N_{i-1}$ sauf pour $N_0$ recevant la sortie du module d'encodage de position.

#### 3.3.2.1 Encodage de position

Comme l'architecture ne contient ni récurrence ni convolution, des informations sur la position absolue des étiquettes de la séquence d'entrée doivent être injectées dans le vecteur passé en entrée au bloc Transformer. Pour cela, un module d'encodage de position (ou *positional encoding* en anglais) est ajouté en entrée, permettant de calculer $PE$ tel que définit par les équations suivantes:

$$PE_{(\text{pos},2i)} = \sin(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}) \tag{3.60}$$

$$PE_{(\text{pos},2i+1)} = \cos(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}) \tag{3.61}$$

Où $d_{\text{model}}$ est la dimension du modèle, équivalant à la dimension du vecteur intégré, pos est la position de l'étiquette et $i$ est la dimension. Ici, les fréquences sont décroissantes le long de la dimension du vecteur et encodées via une paire de fonctions sinus-cosinus, formant une progression géométrique de $2\pi$ à $10000 - 2\pi$ sur les longueurs d'ondes. Toutefois, d'autres variantes existent et sont encore aujourd'hui étudiées.

Les informations sont ensuite intégrées à la séquence d'observations d'entrée par simple somme et passées au module suivant.

#### 3.3.2.2 Mécanisme d'attention à plusieurs têtes

Le mécanisme d'attention utilisé ici peut être résumé comme une mise en correspondance d'une requête et d'un ensemble de paires clé-valeur avec une séquence de sortie, où la requête, les clés, les valeurs et la séquence de sortie sont tous des vecteurs. La sortie est calculée comme une somme pondérée des valeurs, où le poids attribué à chaque valeur est calculé par une fonction vérifiant la compatibilité de la requête avec la clé correspondante.

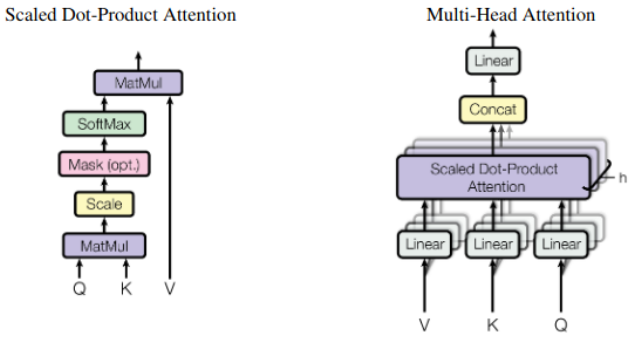



Figure 3.10: A gauche, le mécanisme d'attention *scaled dot-product*. A droite, le mécanisme d'attention composé de plusieurs couches d'attention actant en parallèle. *Google*, 2017

Le mécanisme principal est appelé *scaled dot-product attention* (à gauche sur la figure 3.10) et consiste à calculer simultanément un ensemble de requêtes, clés et valeurs représentés par, respectivement, des matrices $Q$, $K$ et $V$. Plus précisément, le calcul suivant est employé, où $d_k$ définit un facteur d'ajustement :

$$\text{Attention}(Q_i, K_i, V_i) = \text{Softmax}(\frac{Q_i K_i^T}{\sqrt{d^k}})V_i \tag{3.62}$$

Dans le cas de l'architecture Transformer, ce mécanisme est utilisé comme "tête" dans le cadre d'un autre mécanisme de plus haut-niveau (à droite, sur la figure 3.10) appelé "attention à plusieurs têtes" (ou *Multi-head attention* en anglais, abrégé MHA) afin d'accéder simultanément à des informations provenant de différents sous-espaces de représentation, et ce, à différentes positions $i$.

Ainsi, au lieu de calculer la fonction d'attention présentée précédemment avec $Q$, $K$ et $V$ de dimension $d_{model}$, chaque requêtes, clés et valeurs sont projetées $h$ fois au travers de différentes projections linéaires vers, respectivement, $d_k$, $d_k$ et $d_v$ dimensions. Ensuite, la fonction d'attention précédente est calculée en parallèle sur chacune des projections. Les valeurs projetées sont ensuite concaténées et une nouvelle projection linéaire est effectuée. Le processus est résumé par l'équation suivante :

$$\begin{aligned} MHA(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{où head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3.63}$$

Où $W_i^Q$, $W_i^K$, $W_i^V$ sont des matrices de projection de dimension $d_k$, $d_k$ et $d_v$.

La sortie est calculée comme une somme pondérée des valeurs, où le poids attribué à chaque valeur est calculé par une fonction de compatibilité de la requête avec la clé correspondante. Des connexions résiduelles sont ensuite utilisées [He et al., 2016] afin d'ajouter les informations de la sortie du module au vecteur d'entrée et une normalisation de couche LayerNorm est ensuite appliquée tel que : $x = LayerNorm(x + MHA(x))$

#### 3.3.2.3 Couche de propragation avant par position

Afin de propager et conserver les informations de position, chaque bloc de l'architecture contient un réseau *feed-forward* $FF$ entièrement connecté, qui est appliqué à chaque position séparément et de manière identique. Ce réseau est constitué de deux transformations linéaires avec une fonction d'activation Act (habituellement une fonction ReLU) entre les deux transformations :

$$\text{FF}(x) = \text{Act}(W_1 + b_1)W_2 + b_2 \tag{3.64}$$

Où $W_n$ et $b_n$ désignent respectivement les matrices de poids et le vecteur de biais de la première et seconde couche. La dimension reste la même d'une position à une autre mais peut être modifiée d'un bloc à un autre.

Tout comme pour le module d'attention précédent, des connexions résiduelles sont ensuite utilisées suivi d'une normalisation par couche.

## 3.4 Conclusion

Dans ce chapitre, j'ai introduit les principales architectures et approches bout-en-bout utilisées pour le problème de la RAP. Celles-ci ont été utilisées au cours de ma thèse dans

le cadre des travaux présentés dans les chapitres 6 et 7 qui proposent respectivement une comparaison des approches bout-en-bout pour la langue française pour le premier, incluant notamment une évaluation objective avec différentes métriques usuelles tel que le WER ainsi qu'une analyse sur l'unité (orthographique ou phonétique) à modéliser; et une analyse approfondie des erreurs formulées en français par chacune des méthodes pour le problème de la RAP pour le second, auquel j'ajoute une mise en correspondance avec le problème de compréhension du langage parlé. À noter toutefois que l'architecture Transformer n'est pas abordée dans ces chapitres, celle-ci n'ayant été proposée et démocratisée pour la RAP que plus tard.

L'ensemble de ces approches ont aussi été utilisées, ou mises en place pour certaines, dans le cadre du projet ESPnet [Watanabe et al., 2018a] auquel je participe. Le projet ainsi que mes contributions dans ce cadre sont présentés dans le chapitre 5. Dans ce cadre, de nouvelles architectures s'appuyant sur les éléments présentés dans ce chapitre ainsi que des techniques pour l'entraînement et l'inférence des modèles Transducer seront introduites.

# Chapter 4

# Bases de données

Dans ce chapitre, je décris les corpus de parole principaux utilisés pour la réalisation des travaux présentés dans les chapitres suivants. Leur nombre s'élève à quatre et couvre trois langues (français, italien et vietnamien), deux types de parole (lue et conversationnelle) et trois types d'environnements acoustiques (radio-diffusé, enregistrement domestique et conditions laboratoire). Le volume de données pour ces corpus varie entre 16 heures et quelques centaines d'heures.

Dans chaque section, je décris brièvement le contexte de création du corpus (ainsi que le contexte historique si celui-ci existe) et les données orales et écrites mises à disposition pour l'entraînement des différents modèles d'un système de RAP et l'évaluation de ce dernier. Si des modifications ont été apportées dans le cadre de mes travaux, celles-ci sont précisées. Dans le cadre de l'entraînement des modèles linguistiques, je donne une description succincte des données textuelles utilisées en complément de chaque corpus, si elles existent.

Pour finir, je rapporte les systèmes de référence pour chaque corpus au moment de leur acquisition ainsi que les performances obtenues, en terme de taux d'erreurs (de caractères, syllabes et/ou mots).

## 4.1 ESTER (Français)

L'ensemble des expériences principales pour le Français a été mené avec les données fournies lors de la première et la deuxième campagne d'évaluation *Évaluation des systèmes de transcription enrichis de Broadcast News* (ESTER) [Gravier et al., 2004, Galliano et al., 2009], et regroupées en deux corpus nommés respectivement ESTER1 [Gravier et al., 2006] et ESTER2 [Galliano et al., 2009]. Ces campagnes, menées en 2004-2005 et 2008-2009, sont nées d'une volonté d'offrir aux chercheurs des ensembles d'évaluation objectifs et des matériaux de recherche francophones jusqu'alors manquants pour deux axes de recherche : 1) l'évaluation à grande échelle de systèmes de transcription d'émissions radiophoniques et 2) l'évaluation de méthodes d'extraction automatique de contenu, incluant notamment la détection des entités nommées dans des transcriptions automatiques. Ces campagnes font écho aux campagnes réalisées par le *Defense Advanced Research Projects Agency* (DARPA) et le *National Institute of Standards and Technology* (NIST) considérées jusqu'alors comme des références pour les chercheurs francophones sur ces axes de recherche, bien que uniquement en anglais.

La première campagne ESTER menée en 2004-2005 a été mise en place dans le cadre du projet EVALDA par la Délégation Générale pour l'Armement (DGA), l'Association française de la communication parlée (AFCP) regroupant différents laboratoires en pointe sur le domaine, et the European Language Resources Association (ELRA) pour la distribution des données aux participants et plus généralement maintenant, aux chercheurs intéressés par la reconnaissance de parole en français. Lors de cette première campagne, seule l'évaluation de systèmes de transcription d'émissions radiophoniques fut effectuée. Cette évaluation comprend toutefois différentes mesures concernant les performances de la détection d'événements sonores particuliers ou des systèmes de segmentation en tours de paroles.

Dans la continuité, une seconde campagne est effectuée entre janvier 2008 et avril 2009 par les mêmes organismes. L'objectif étant d'évaluer les progrès effectués depuis la première campagne, mais aussi d'initier de nouveaux axes de recherche comme notamment la détection et l'étiquetage par entités nommées.

À ce jour, les deux corpus résultant de cette campagne sont encore parmi les corpus les plus utilisés pour l'entraînement et l'évaluation des systèmes de RAP conversationnel en Français.

### 4.1.1 Données utilisées

Pour l'entraînement des modèles acoustiques, j'utilise les corpus d'entraînement délivrés lors des deux premières campagnes d'évaluation, correspondant respectivement à 90 et 75 heures de parole. En complément, j'ajoute 90 heures provenant d'un ensemble de données non transcrites fourni lors de la campagne ESTER1 auquel j'associe les transcriptions manuelles fournies dans le corpus EPAC [Esteve et al., 2010]. Les segments contenant moins de 1,5 secondes de parole transcrite sont exclus du corpus final, tout comme les énoncés incluant des segments audio de plus de 3000 trames ou des phrases de plus de 400 caractères. Cette opération est effectuée afin de contourner les limitations mémoires rencontrées pour l'entraînement de certains systèmes de RAP, notamment bout-en-bout. Additionnellement, du fait de la présence de certaines paires (segment audio - transcriptions) irrégulières, une étape de re-segmentation des données d'entraînement est effectuée en utilisant un modèle HMM-GMM décrit dans [Boyer and Rouas, 2019]. Lors de cette étape, seules les parties de l'audio correspondant aux transcriptions ont été sélectionnées. Cela porte le volume final de données d'entraînement à environ 231h. Pour l'entraînement des réseaux de neurones, j'applique de plus une perturbation de vitesse "triple" [Ko et al., 2012] et une perturbation de volume avec un facteur de volume aléatoire entre 0,25 et 2, résultant en un volume total d'environ 700 heures.

Pour le modèle de langage français, j'utilise les transcriptions manuelles provenant du corpus d'entraînement utilisé pour la modélisation acoustique. Cet ensemble est étendu avec des transcriptions sélectionnées manuellement à partir d'autres sources de parole telles que : le corpus BREF (cf. Sec. 4.2), les interventions orales provenant du corpus EuroParl de 96-'06 [Koehn, 2005] et une petite partie de transcriptions provenant de projets internes (env. 10% du volume total). Le corpus final est composé de 2 041 916 phrases, pour un total de 46 840 583 mots.

Les évaluations des systèmes sont effectuées sur les ensembles de développement et de test donnés lors de la seconde campagne. Les détails de l'ensemble de données, correspondant respectivement à 5h46 et 6h34 de parole, sont décrits dans [Galliano et al., 2009]. Lors de mes expériences, les mêmes règles de normalisation et de notation que dans le plan

d'évaluation de la campagne ESTER 2 sont appliquées, à l'exception qu'aucun dictionnaire d'équivalence n'est utilisé en complément et que les mots partiellement prononcés sont évalués de la même manière que des mots entiers/complets.

### 4.1.2 Résultats initiaux

Au moment de l'obtention du corpus, soit quelques mois après le début de cette thèse, les meilleurs systèmes évalués pour la tâche de transcription correspondent toujours aux systèmes reportés durant la seconde phase de campagne d'ESTER [Galliano et al., 2009]. Les trois premiers systèmes obtenant un taux d'erreurs de mots sur le corpus de test de respectivement : 12.1% pour le système du LIMSI, 15.1% pour le système du VR (Vecsys Research) et 17.8% pour celui proposé par le LIA.

## 4.2 BREF (Français)

Dans le cadre des travaux préliminaires pour la RAP en français, détaillés dans la première partie du chapitre 6, un corpus de parole lue a aussi été utilisé : BREF [Lamel et al., 1991]. Celui-ci a été conçu et développé par des chercheurs du LIMSI au début des années 90 avec l'appui financier de différents acteurs : le GDR-PRC Communications Homme-Machine, l'Agence de coopération culturelle et technique (ACCT), la CEE (projet ESPRIT Polyglot) et enfin l'Aupelf-Uref.

L'objectif du projet est de proposer des matériaux oraux en français similaires à ceux proposés dans le cadre de projets de la DARPA pour l'anglais (tel que TIMIT ou encore *Resource Management*) et de permettre le développement de *machines de dictée* pour la langue. En définitive, le corpus BREF est l'un des premiers corpus oral en français dit large vocabulaire et permettant : 1) l'entraînement, le développement et l'évaluation de systèmes de RAP continus pour le français, et 2) l'étude et la modélisation des variations phonologiques pour cette langue.

Pour ce faire, les auteurs s'appuient sur des articles du journal Le Monde (dans une période de trois mois pour limiter les thématiques) afin de constituer les textes de références à lire. À partir de là, un ensemble de phrases a été extrait sur la base des critères de sélection tels que la maximisation du contexte phonémique ou encore le nombre total de mots uniques. Un ensemble de locuteurs est ensuite sélectionné en considération de critères précis, afin de produire un nombre de mots défini en amont. Chaque locuteur est ici enregistré dans un environnement contrôlé (chambre sourde).

Bien que le corpus est de nos jours peu utilisé en comparaison à d'autres corpus tel qu'ESTER, il s'avère toutefois intéressant pour initier nos travaux de RAP en français (tâche considérée simple, à savoir parole lue en condition laboratoire) et pour l'enrichissement des données d'entraînement servant à d'autres systèmes de RAP. Notamment, les textes de références comportent de nombreux noms communs, propres, homophones, etc., qui s'avèrent adéquats pour améliorer la qualité de certaines modélisations linguistiques, telles que celles utilisées dans le cadre du système ESTER.

### 4.2.1 Données utilisées

Le corpus complet contient 100 heures de parole lue produites par 120 locuteurs français (45% d'hommes et 55% de femmes) et natifs de la région parisienne, à l'exception de 6 locuteurs venant du Maroc et du Luxembourg. Le nombre de phrases prononcées est

d'environ 165 000 pour un nombre total de mots distincts de l'ordre de 90 000. Chaque locuteur prononce entre 5000 (70 locuteurs) et 10 0000 mots (50 locuteurs).

Un certain nombre d'irrégularités et d'erreurs existant dans les transcriptions de référence, notamment pour les mots ayant une fréquence d'apparition inférieure à 20 (voir [Lamel et al., 1991]), un travail a été effectué en amont afin d'améliorer la qualité du corpus, par correction manuelle ou élimination des paires audio-transcription problématiques. En définitive, le corpus final utilisé contient environ 97 heures d'enregistrement.

De ce volume, 91,1 heures de parole (et leurs transcriptions correspondantes) sont extraites pour l'entraînement du modèle acoustique. Les éléments restants sont assignés aux ensembles de développement et d'évaluation, formant des ensembles de respectivement 2h30 et 3h30. Cette répartition diffère quelque peu de celle proposée par les auteurs dans [Lamel et al., 1991] et qui donnent une description de BREF-90, un sous-ensemble du corpus final comprenant 90 locuteurs au lieu de 120. Une attention particulière est toutefois donnée afin de conserver une répartition similaire en terme de locuteurs mais aussi de propriétés distributives sur les mots et sous-mots (voir Table 6 [Lamel et al., 1991]).

Pour le modèle de langage français, les transcriptions de référence provenant de l'ensemble d'entraînement sont utilisées pour l'entraînement du modèle. Contrairement au modèle de langage décrit dans la section précédente, je n'utilise pas de données supplémentaires.

### 4.2.2 Résultats initiaux

Bien que des corpus de développement et d'évaluation soient définis par les auteurs dans le cadre du projet BREF, les systèmes de RAP de l'état de l'art entraînés avec le corpus BREF (seul ou en association avec d'autres données) sont le plus souvent évalués sur différents ensembles selon l'époque., comme par exemple BDSONS [Descout et al., 1986], ou encore les ensembles de tests d'ESTER1 et d'ESTER2.

Il existe toutefois un ensemble d'évaluation pour BREF datant de février 1994 qui est utilisé pour l'évaluation du système de dictée de référence du LIMSI [Gauvain et al., 1994]. Dans ce cadre, les meilleurs systèmes proposés obtiennent un taux d'erreurs de mots de respectivement 5,8% pour le système limité à 5000 mots et 9.6% pour le système à "large vocabulaire" (plus de 20 000 mots).

## 4.3 Voxforge (Italien)

Afin d'entraîner et évaluer les systèmes pour l'italien, je m'appuie sur le corpus libre proposé dans le cadre du projet VoxForge [Voxforge.org, 2019]. Initié en 2006, l'objectif du projet est la collecte de parole transcrite afin de créer un corpus de parole libre (sous licence GPL) pour l'entraînement de solutions de traitement de parole. Pour cela, une plateforme collaborative a été mise en place permettant aux participants du projet de sélectionner des phrases ou textes parmi une liste d'ouvrages ou extraits libres de droits afin d'enregistrer leur lecture. Depuis 2007, le projet s'appuie de plus sur les données audio disponibles dans le projet LibriVox, aussi utilisé dans le cadre du corpus Librispeech.

Parallèlement, le projet distribue des outils pour le traitement des données, des dictionnaires phonétiques, un étiquetage d'entités nommées pour différents segments de la parole ainsi que des modèles acoustiques pré-entraînés, utilisables avec différentes boites à outils de reconnaissance de parole telles que CMU Sphinx, HTK, Julius et plus récemment Kaldi. Ces ressources sont mises à jour régulièrement en considération de différents critères tel

que le volume de données actuel en comparaison de la version du corpus publiée actuellement ou encore des améliorations de performances dénotés pour de nouveaux systèmes utilisant ces données.

Initié au préalable pour la langue anglaise, le projet s'est rapidement étendu à d'autres langues pour atteindre un nombre total de 17 au moment de l'écriture de ce manuscrit. Les principales langues, outre l'anglais, étant le français, l'espagnol, l'allemand et l'italien qui nous intéresse ici.

Malgré le fait que la problématique liée à ce corpus soit relativement simple (parole lue enregistrée en condition domestique et non-bruité) et que la validation initiale des données soit souvent débattue par la communauté (du fait de conditions d'enregistrements hétérogènes et de la présence de doublons entre les sous-ensembles), celui-ci reste pertinent de par : 1) son volume comparé aux autres corpus étudiés (env. 20 heures) et 2) le nombre restreint d'études pour la reconnaissance de parole utilisant ce corpus et généralement en italien. En outre, et en lien avec le volume de données et l'accessibilité du corpus, la grande majorité des travaux et des systèmes de RAP entraînés sur ce corpus s'avère facilement reproductible du fait d'un faible coût, temporel et matériel.

### 4.3.1 Données utilisées

La version du sous-corpus italien dont je me sers est datée de 2017 et contient environ 20 heures d'enregistrements de parole. Le découpage des sous-corpus d'entraînement, validation et test n'étant pas fixé dans le cadre du projet VoxForge, je suis la procédure de sélection utilisée par la boite à outils ESPnet. Celle-ci est incluse dans la "recette" VoxForge[1] et est appliquée systématiquement quelle que soit la langue sélectionnée, la recette étant la même pour tous les corpus issus de Voxforge. Dans un premier temps, les énoncés correspondant à des segments audio de plus de 2000 trames ou moins de 20 trames ainsi que les transcriptions contenant plus de 400 caractères sont enlevés. Chaque paire est ensuite assignée aléatoirement à un sous-ensemble en considérant une répartition définie empiriquement : 80% pour l'entraînement, 10% pour le développement et 10% pour l'évaluation. Le corpus contenant des énoncés pouvant être dupliqués, l'assignation à la fois au sous-corpus d'entraînement et au sous-corpus de test est proscrite. Un même locuteur peut toutefois être connu lors de l'entraînement et retrouvé dans l'ensemble de test.

Pour l'entraînement d'un modèle de langage en italien, j'utilise une nouvelle fois les phrases provenant du corpus d'entraînement. Ce qui correspond à environ 9000 phrases pour 100 000 mots. Malgré la redondance des informations, certains systèmes modélisant un modèle de langage "interne" basé sur les transcriptions des données d'entraînement, et le volume restreint d'exemples, l'utilisation d'un modèle de langage entraîné sur ces données s'est avéré bénéfique lors de nos expériences.

### 4.3.2 Résultats initiaux

Les travaux en relation avec ce corpus ont été initiés courant 2019. À cette date, le meilleur système de RAP connu correspond au système bout-en-bout proposé par la boite à outils ESPnet. Celui-ci est un modèle bout-en-bout entraîné conjointement avec CTC et Attention, et utilisant une architecture RNN pour l'encodeur et le décodeur. Le système

[1] `https://github.com/espnet/espnet/tree/master/egs/voxforge/asr1`

obtient un taux d'erreurs de caractères de 12.6% sur le corpus de test défini via la procédure décrite précédemment. Le taux d'erreurs de mots n'est quant à lui pas communiqué.

## 4.4 VIVOS (Vietnamien)

Le dernier corpus sélectionné est utilisé pour la construction de systèmes de RAP en vietnamien et se nomme VIVOS[2]. Rendu publique en 2016, celui-ci est constitué de 16 heures d'enregistrements de parole en haute qualité. Ces enregistrements ont été obtenus lors de différentes sessions effectuées en laboratoire et sont séparés en deux sous-ensembles servant à l'entraînement (env. 15 heures) et à l'évaluation (env. 50 minutes) des systèmes de RAP en vietnamien.

Motivé par le développement actif de différentes boites à outils telles que Kaldi, les auteurs proposent, en parallèle de cette collection de données, des recettes "simples et directes" [Luong and Vu, 2016] pour tous les chercheurs voulant créer un système de RAP traditionnel pour la langue vietnamienne. Cette simplicité provient de leur préparation, les différents composants des systèmes de RAP étant basés sur la forme écrite de la langue plutôt que sur des connaissances spécialisées en linguistique et en phonologie.

Ce corpus est particulièrement intéressant sur plusieurs aspects. De par le volume de données relativement faible et le manque de ressources pour la langue cible tout d'abord permettant de 1) effectuer une comparaison entre les systèmes bout-en-bout basés sur les réseaux de neurones et les systèmes traditionnels dans le cadre de la RAP dite *low ressources*, et 2) mettre en perspective le postulat disant que les modélisations neuronales sont réputées trop gourmandes en ressources et moins performantes dans les conditions citées par rapport aux systèmes traditionnels basés HMM. D'autre part, les études sur le choix des unités à modéliser (phonémique ou graphémique) ainsi que l'importance des informations spécialisées (exemple: pitch, contextualisation) est particulièrement intéressant pour le vietnamien, comme démontré dans [Luong and Vu, 2016], mais aussi en considération des autres langues. Le vietnamien comportant des différences avec des langues indo-européennes romanes (Français) ou germanique (Anglais), cela justifie une évaluation approfondie en considération des points précédents mais aussi dans le cadre du problème de compréhension de la parole : usage du ton, tempo de la parole par rapport au taux d'informations ou encore unité minimale phonologique.

Dans le cadre des travaux présentés dans ce manuscrit, ce corpus est toutefois utilisé en priorité pour l'évaluation de fonctionnalités ajoutées au projet ESPnet. Une étude portant sur l'évaluation des aspects cités précédemment a été initiée avec les auteurs du corpus mais n'a toutefois pas mené à des travaux complémentaires ou publications à ce jour.

### 4.4.1 Données utilisées

Contrairement aux corpus précédents, j'utilise les données telles que présentées, c'est-à-dire qu'aucune étape d'augmentation de données ou de suppression est appliquée. La collection de données ne définissant pas un sous-ensemble de validation, j'extrais aléatoirement 10% des échantillons du corpus d'entraînement pour constituer un corpus de validation de quelques minutes afin de contrôler l'entraînement des modèles acoustiques.

Par manque de données pertinentes, et d'expertise sur la langue, je n'utilise pas de modèle de langage dans les systèmes de RAP pour le vietnamien. Les données textuelles provenant

[2]https://ailab.hcmus.edu.vn/vivos

du corpus d'entraînement ne permettent pas d'améliorer significativement les performances du système et, au contraire, les dégradent.

### 4.4.2 Résultats initiaux

Les travaux en relation avec ce corpus ont débuté en même temps que ceux utilisant le corpus VoxForge, c'est-à-dire 2019. A cette date, le meilleur système proposé reste le système rendu disponible dans la boite à outils Kaldi [Luong and Vu, 2016], obtenant un taux d'erreurs sur les syllabes (TES) de 13.34% dans sa version de base, 9.55% par incorporation du *pitch* dans les caractéristiques de la parole et 9.48% par ajout des informations de tons dans l'étape de création de l'arbre de décision graphémique. Une autre recette utilisant cette fois la boite à outils ESPnet a depuis été proposée par l'un des auteurs du corpus, H.-T. Luong, dans un souci d'évaluer les systèmes bout-en-bout pour la langue vietnamienne. Ce système utilise la méthode CTC et des réseaux de neurones récurrents. Il obtient un TES de 54.7% dans sa version originale, et un SyER de 42.4% par utilisation d'un modèle de langage externe basé syllabes. Malgré un écart de performance flagrant, de nombreuses modifications peuvent être apportées en terme d'approches ou d'architecture par exemple, afin de se rapprocher des performances des systèmes de RAP traditionnels. Ce dernier point étant confirmé dans le chapitre 5.

# Chapter 5

# Le projet ESPnet

Ce chapitre est consacré à la description de la boite à outils ESPnet (*End-to-end Speech Processing toolkit*) [Watanabe et al., 2018b] et les contributions que j'ai apportées à celle-ci. Par souci de clarté, ce chapitre est découpé en trois parties. Tout d'abord, une présentation générale de l'architecture et des fonctionnalités principales est donnée. Je me focalise ensuite sur les contributions apportées au projet portant sur trois thématiques distinctes, les opérations dites *d'affinage*, les modèles entraînés avec la fonction de perte RNN-T [Graves, 2012b] et les *recettes* permettant la création de modèles de RAP entraînés avec différents corpus. Je conclus ce chapitre en introduisant plusieurs papiers basés sur les contributions présentées dans la seconde section ainsi que les axes de travail futurs.

Du fait de la constante évolution de la boite à outils et du nombre de tâches couvertes à ce jour par cette dernière (reconnaissance automatique de parole, traduction automatique -orale et écrite- ou encore synthèse de la parole par exemple), il apparaît difficile de couvrir ici l'ensemble des fonctionnalités, modèles et algorithmes disponibles. A cela, je me concentre principalement sur la tâche de la reconnaissance automatique de parole bout-en-bout et les fonctionnalités associées qui sont disponibles à la date de février 2021.

En outre, je dirige le lecteur vers le chapitre 3 contenant l'ensemble des notions nécessaires à la compréhension des architectures et fonctionnalités présentées dans ce chapitre, notamment pour la section 5.2 et la section 5.4.3

## 5.1 Présentation

ESPnet (*End-to-end Speech Processing toolkit*) est une boite à outils, ou plateforme, open-source pour la reconnaissance automatique de parole bout-en-bout introduite en 2018 par Watanabe et al. [Watanabe et al., 2018b]. Elle s'appuie sur les bibliothèques logicielles open-source d'apprentissage machine PyTorch, développé par Facebook, et Chainer, dont le développement est dirigé par une société japonaise, Preferred Networks, en partenariat avec IBM, Intel, Microsoft et Nvidia. La majorité des fonctionnalités de ESPnet sont implémentées sur la base de chaque bibliothèque, l'utilisateur a donc le choix de la brique logicielle sur laquelle s'appuyer pour ses travaux. Dans le cas de ce manuscrit, je fais toutefois abstraction de cette notion, les implémentations basés sur Chainer étant abandonnées petit à petit dans le projet.

Contrairement aux boites à outils open source traditionnelles basées sur des architectures DNN-HMM hybrides, ESPnet vise à fournir une architecture unique, basée sur les

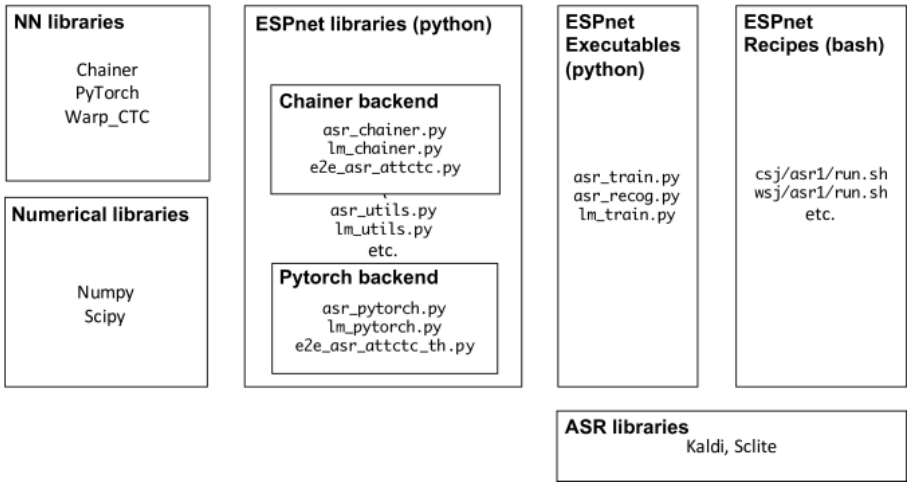



Figure 5.1: Architecture de la boite à outils au moment de sa publication. [Watanabe et al., 2018b]

réseaux de neurones, pour effectuer la reconnaissance vocale d'une manière *bout en bout*. A cela, ESPnet exploite pleinement les avantages de deux approches bout-en-bout majeures à savoir la *Connectionist Temporal Classification* et les encodeurs-décodeurs utilisant des mécanismes d'attention (Cf. chapitres 3.2.1, 3.2.3, 3.3.1). En outre, une méthode d'apprentissage multi-tâches est proposée permettant d'entraîner de manière conjointe un modèle avec les deux approches précédentes. Celle-ci, proposée par S. Kim et deux des auteurs de ESPnet [Kim et al., 2017], a suscité un grand intérêt de la communauté, faisant de ESPnet une référence de nos jours, aux côtés d'autres projets tel que Kaldi.

La figure 5.1 met en évidence les composants et fonctions principales proposées initialement et que je vais couvrir dans ce chapitre.

## 5.2 Architecture générale du modèle de RAP

Contrairement aux boites à outils pour la RAP bout en bout à cette époque majoritairement basés sur la CTC comme par exemple DeepSpeech[1], ESPnet est basé sur une architecture de type encodeur-décodeur utilisant un mécanisme d'attention pour effectuer un entraînement multi-tâches avec la méthode CTC. L'utilisation d'un décodeur est omis pour l'entraînement avec la CTC seule.

#### 5.2.0.1 Encodeur

Étant donnée une source $x$ de $T$ trames, représentée par une séquence de caractéristiques de type *log-mel filterbank* de dimension 80 (ou 83 si les caractéristiques de hauteur sont ajoutées), l'encodeur transforme $x$ en une séquence de caractéristiques encodée $h^{\text{enc}}_{1:t'} \in \mathcal{R}^{T^{\text{sub}}}$ au travers de deux fonctions, EncInput$(\cdot)$ et EncBody$(\cdot)$.

$$h^{\text{enc}}_{1:t"} = \text{EncInput}(x_{1:t}), \tag{5.1}$$

$$h^{\text{enc}}_{1:t'} = \text{EncBody}(h^{\text{enc}}_{1:t"}) \tag{5.2}$$

[1] https://github.com/mozilla/DeepSpeech

Où EncInput(·) est généralement un réseau de couches de convolutions transformant $x$ de longueur $T$ en $h^{\text{enc}}_{1:t"}$ de longueur $T^{\text{sub}}$, où sub est un facteur défini en considération des paramètres du réseau. Selon la nature de EncBody(·), EncInput(·) peut être soit sous la forme d'un réseau de type VGG avec opération de *max pooling* [Hori et al., 2017b], ou un réseau composé de deux couches de convolutions avec 256 canaux, une taille de stride de 2 et une taille de kernel de 3 [Dong et al., 2018]. Dans le second cas, lorsque les couches de EncBody(·) n'utilisent pas de neurones récurrents, une opération dite "d'encodage de position" (*positional encoding* en anglais) est associée pour représenter la localisation temporelle de chaque trame de la séquence en sortie de EncInput(·).

EncBody(·) est par défaut un réseau de neurones récurrents de type LSTM bidirectionnel à plusieurs couches. Étant donnée la séquence d'entrée $h^{\text{enc}}_{1:t"}$, le réseau extrait une représentation $h^{\text{enc}}_{1:t'}$ de longueur $T"^{\text{sub}}$ telle que :

$$h^{\text{enc}}_{1:t'} = \text{BLSTM}(h^{\text{enc}}_{1:t"}) \tag{5.3}$$

Où $t' < t"$ si BLSTM est de type pyramidal [Chan et al., 2016], $t' = t"$ sinon.

Plus récemment, l'utilisation de blocs de type Transformer est venue remplacer peu à peu les architectures de type RNN à plusieurs couches dans ESPnet. Dans ce cas, l'équation 5.3 est remplacée par :

$$X_i^{'} = X_i + \text{MHA}_i(X_i, X_i, X_i) \tag{5.4}$$

$$X_{i+1} = X_i^{'} + \text{FF}_i(X_i^{'}) \tag{5.5}$$

Où $i = 0, ..., l_{\text{enc}} - 1$ désigne un numéro de bloc, MHA une couche d'attention à plusieurs têtes (*MultiHead Attention* et FF un réseau de type *feed-forward* à deux couches. L'entrée $X_i$ pour le premier bloc est ici $h^{\text{enc}}_{t"}$ tandis que la sortie du dernier bloc, $X_{i+1}$, correspond à $h^{\text{enc}}_{1:t'}$.

#### 5.2.0.2 Décodeur

Étant donné $h^{\text{enc}}_{1:t'}$, la sortie de l'encodeur, et une séquence d'étiquette $y_{1:u-1}$ de longueur $U - 1$, le décodeur génère la représentation $h^{\text{dec}}_{1:u}$ au travers de deux fonctions DecInput(·) et DecBody(·). Les équations suivantes dénotent le processus :

$$h^{\text{dec}}_{1:u-1} = \text{DecInput}(y_{1:u-1}) \tag{5.6}$$

$$h^{\text{dec}}_{1:u} = \text{DecBody}(h^{\text{enc}}_{1:t'}, h^{\text{dec}}_{1:u-1}) \tag{5.7}$$

Où DecInput(·) est par défaut une couche d'intégration (*embedding layer* en anglais) encodant une séquence d'entrée de dimension $D^{\text{feats}}$, en une séquence de dimension $D^{\text{classes}}$, le nombre de classes cibles défini par l'utilisateur, et correspondant aussi à la dimension d'entrée de DecBody(·). Si DecBody(·) est de type Transformer, une opération d'encodage de position est utilisée pour représenter la localisation de chaque entrée de la séquence $y_{u-1}$.

DecBody(·), tout comme EncBody(·), peut être sous deux formes : soit un réseau composé de couches de type RNN unidirectionnel, soit un réseau composé de blocs de type Transformer. Dans le premier cas, le décodeur contient un réseau à une ou plusieurs couches de

type LSTM unidirectionnel associé à un mécanisme d'attention, généralement basé sur la position [Chorowski et al., 2015, Bahdanau et al., 2016]:

$$h_{1:u}^{\text{dec}} = \text{AttentionLSTM}(h_{1:t'}^{\text{enc}}, h_{1:u-1}^{\text{dec}}) \tag{5.8}$$

Dans le second cas, le décodeur basé Transformer est composé de blocs contenant deux couches d'attention à plusieurs têtes (MHA) permettant d'obtenir respectivement, la matrice d'attention pour l'entrée du décodeur et la matrice d'attention entre l'entrée du décodeur et la sortie de l'encodeur:

$$Y_j[f] = Y_j[f] + \text{MHA}_j^{\text{self}}(Y_j[f], Y_j, Y_j) \tag{5.9}$$

$$Y_j^{''} = Y_j + \text{MHA}_j^{\text{src}}(Y_j^{'}, h_{1:t'}^{\text{enc}}, h_{1:t'}^{\text{enc}}) \tag{5.10}$$

Où $j = 0, ..., l_{\text{dec}} - 1$ désigne un numéro de bloc et $u$ est l'index d'une trame cible. De part, l'utilisation d'un décodeur unidirectionnel, un masquage est utilisé dans ESPnet afin que les matrices d'attention de la trame cible $f$ ne soient pas connectées aux trames suivantes. La sortie est ensuite passée à un réseau *feed-forward* (FF) comme pour l'encodeur de type Transformer :

$$Y_{j+1} = Y_j^{''} + \text{FF}_j(Y_j^{''}) \tag{5.11}$$

Ici, l'entrée $Y_j$ pour le premier bloc est équivalent à $h_{1:u-1}^{\text{dec}}$ tandis que la sortie du dernier bloc, $Y_{j+1}$, correspond à $h_{1:u}^{\text{dec}}$.

## 5.3 Fonctionnalités

Cette section décrit les fonctionnalités et modules principaux liés au problème de la reconnaissance de parole automatique dans ESPnet.

### 5.3.1 Pré-traitement des données

Du fait de la maturité des outils déjà proposés dans la boite à outils Kaldi et de par leur usage démocratisé, l'ensemble des fonctionnalités relatives au pré-traitement des données [2] a été repris dans ESPnet afin de faciliter la prise en main pour les acteurs du domaine. Parallèlement, cela permet à n'importe quel chercheur ou développeur de migrer facilement les recettes pour un corpus d'une plateforme à une autre ainsi que d'effectuer une comparaison équitable des performances obtenues entre les systèmes hybrides de Kaldi et les systèmes bout en bout de ESPnet.

Outre la préparation des données, comprenant la définition, la vérification et la normalisation des données des différents sous-ensembles d'un corpus cible, ESPnet s'appuie aussi le module d'extraction de caractéristiques mis à disposition dans Kaldi dans la majorité des recettes disponibles. A noter toutefois que certains cas particuliers, comme la création de systèmes bout-en-bout multi-canaux, s'appuient sur l'utilisation de réseaux ou techniques particulières *maison*.

[2] https://kaldi-asr.org/doc/data_prep.html

Plus récemment, dans le cadre du projet ESPnet2, des travaux sur le renforcement de la parole ont été initiés. La première phase des ces travaux consiste à mettre en oeuvre une technique de base autonome d'amélioration et de séparation de la parole pour, dans un second temps, appeler de manière flexible cette technique en tant que sous-réseau de traitement frontal de la reconnaissance de parole tout en maintenant le coût computationnel du graphe de calcul dans le cadre de l'entraînement conjoint de la CTC et de la méthode basée attention.

### 5.3.2 Entraînement

ESPnet possède trois modes d'entraînement: CTC et attention encoder-decoder et entraînement multi-tâches (ou *Multi Task Learning* en anglais, abrégé MTL). Les deux premiers modes s'appuient sur la distribution postérieure par trame de $y_{1:u}$ étant donné $x_{1:t}$ obtenue par respectivement le module CTC et le décodeur. Le dernier mode est quant à lui basé sur l'utilisation combinée des deux premiers modes, définissant la fonction de perte comme la somme pondérée des logarithmes négatifs des distributions précédentes. Les équations suivantes décrivent les fonctions de perte pour chaque mode, où $\alpha$ est un paramètre contrôlant l'importance donnée à une méthode par rapport à l'autre.

$$\mathcal{L}_{\text{ctc}} = -\log p_{\text{ctc}}(y_{1:u}|x_{1:t}) \tag{5.12}$$

$$\mathcal{L}_{\text{att}} = -\log p_{\text{att}}(y_{1:u}|x_{1:t}) \tag{5.13}$$

$$\mathcal{L}_{\text{MTL}} = -\alpha \log p_{\text{att}}(y_{1:u}|x_{1:t}) - (1-\alpha) \log p_{\text{ctc}}(y_{1:u}|x_{1:t}) \tag{5.14}$$

### 5.3.3 Inférence

Pour la phase d'inférence, ESPnet propose un mode unique pour l'ensemble de ses modèles reposant sur l'utilisation d'une recherche en faisceau. Dans le cadre de la CTC, une procédure de recherche par préfixe [Graves et al., 2006] est introduite. Pour le décodage du modèle *attention* seul, des paramètres de contrôle de longueur de séquence ou des termes de couvertures [Wu et al., 2016] peuvent aussi être utilisés pour réduire le nombre d'alignements irréguliers causés par de long sauts ou répétitions pour la même trame [Bahdanau et al., 2016]. Dans le cadre d'une procédure de décodage avec un modèle CTC-attention conjoint, l'étiquette suivante est obtenue en fonction de la représentation $h^{\text{enc}}_{1:t'}$ de la séquence source $x_{1:t}$ et des étiquettes précédemment prédites lors de la recherche en faisceau qui combine les scores des différentes méthodes comme suit :

$$\hat{y} = \arg\max_{y \in \mathcal{Y}^*} \{\lambda \log p_{\text{att}}(y|h^{\text{enc}}_{1:t'}) + (1-\lambda) \log p_{\text{ctc}}(y|h^{\text{enc}}_{1:t'})\} + \gamma \log p_{\text{LM}}(y|y_{1:u-1}) \tag{5.15}$$

Où $\mathcal{Y}^*$ désigne l'ensemble des séquences possibles en sortie et $\lambda$ est un paramètre de pondération similaire à $\alpha$ utilisé lors de l'entraînement. Pour chaque mode, les log-probabilités d'un modèle de langage [Hori et al., 2017b] peuvent être incorporées dans l'équation, le paramètre $\gamma$ permettant de pondérer la contribution.

## 5.4 Contributions apportées

Dans cette section, je décris les contributions que j'ai apporté à la boite à outils ESPnet et qui portent sur quatre sujets distincts : les techniques d'affinage, l'architecture

conformer, les modèles entrainés avec la fonction de perte RNN-T et différentes recettes pour l'entraînement de modèles bout-en-bout sur des corpus de différentes langues (Cf. chapitres 4). Des travaux ponctuels de corrections et d'optimisations pour d'autres fonctionnalités d'ESPnet ont aussi été effectués.

### 5.4.1 Affinage

Deux techniques d'affinage ont été ajoutés à ESPnet pour la tâche de RAP : le transfert de connaissance [Pan and Yang, 2009, Rao et al., 2017] et le gel de paramètres. Chaque opération est applicable aux différentes parties de l'architecture (encodeur ou décodeur) de manière indépendante. Les paramètres, c'est-à-dire les poids et biais, des différents modules à transférer ou geler pour chaque partie du modèle sont définis par une liste donnée par l'utilisateur.

Le code Python pour ces deux techniques est consultable à l'adresse suivante : `https://github.com/espnet/espnet/blob/master/espnet/asr/pytorch_backend/asr_init.py`. La documentation associée à leur utilisation est quant à elle disponible à cette adresse : `https://espnet.github.io/espnet/tutorial.html#how-to-use-finetuning`.

#### 5.4.1.1 Transfert de connaissance

Au cours de la phase d'apprentissage, les paramètres des différentes couches composant le modèle sont modifiés jusqu'à ce que la valeur associée à la fonction de perte ou à un critère d'entraînement ne satisfasse plus des critères de convergence. Pour certains modèles tels que les modèles entraînés avec une fonction de perte RNN-T et réputés difficile à entraîner, l'utilisation d'un modèle pré-entraîné pour initialiser les paramètres d'un nouveau modèle est fortement souhaité afin réduire le temps d'entraînement et atteindre des performances jugées satisfaisantes [Rao et al., 2017, Battenberg et al., 2017]. En outre, le choix du modèle pré-entraîné et la sélection de la connaissance à transférer, c'est-à-dire les paramètres des différentes couches, peuvent s'avérer cruciaux en considération de la tâche ou de la langue cible.

Traditionnellement, pour les modèles entraînés avec la fonction de perte RNN-T, l'encodeur est initialisé avec un modèle pré-entrainé avec la fonction de perte CTC [Rao et al., 2017] et le réseau de prédiction avec un modèle de langage [Hu et al., 2020].

Dans le cadre de ESPnet, l'apprentissage par transfert est rendu disponible pour l'ensemble des approches de RAP mais aussi les approches de synthèse de la parole, de traduction orale et bien d'autres. Le choix des modèles sources et cibles ainsi que les paramètres à transférer sont pilotables par l'utilisateur. Les seules limitations étant: 1) le modèle source doit avoir été entraîné avec la boite à outils ESPnet et 2) les paramètres transférés du modèle source doivent être équivalent à ceux du modèle cible.

#### 5.4.1.2 Gel de paramètres

Lorsqu'un transfert de connaissance est effectué, l'utilisation d'une fonction de gel peut être bénéfique afin de s'assurer que l'information transférée n'est pas détruite lors d'un nouvel entraînement [Kunze et al., 2017]. Plus précisément, celle-ci permet de bloquer les paramètres des couches transférées afin qu'ils ne soient pas mis à jour lors du nouvel entraînement. En excluant les paramètres lors de l'étape de rétro-propagation du gradient, l'entraînement du nouveau modèle peut être, en outre, accéléré.

Dans le cadre de ESPnet, à partir d'un modèle pré-entraîné à transférer, la procédure standard d'utilisation de la fonction de gel est la suivante :

1. Transfert des couches ou paramètres de couches du modèle pré-entrainé au modèle cible..
2. Gel des paramètres transférés.
3. Entraînement des paramètres des couches non-incluses dans le transfert.

D'autres procédures incluant cette fonction sont possibles en considération du modèle, de la tâche ou encore du nombre de classes entre le modèle pré-entraîné et le nouveau modèle. L'ensemble des procédures simples connues utilisant la technique de gel de paramètres, à notre connaissance, sont couvertes dans ESPnet par la fonction de gel proposée.

### 5.4.2 Conformer

L'architecture Transformer (voir section 3.3.2) a suscité un immense intérêt de la communauté scientifique et est devenue ces dernières années l'architecture dominant en raison de son efficacité pour diverses tâches de séquence à séquence. Le succès de l'architecture s'explique par le fait que les couches d'auto-attention à plusieurs têtes permettent un meilleur apprentissage du contexte global à long terme par rapport aux réseaux de neurones récurrents. Cependant, pour les tâches de traitement de la parole, le contexte global mais aussi les informations locales sont cruciales pour capturer certaines propriétés particulières de la parole, comme la co-articulation et la monotonie. Les réseaux neuronaux à convolution (CNN), d'autre part, permettent d'extraire des caractéristiques locales très fine.

Récemment, Gulati et al. [Gulati et al., 2020] ont proposé une nouvelle architecture appelée Conformer et proposant de combiner l'auto-attention et la convolution pour la modélisation acoustique. Dans celle-ci, la couche d'auto-attention apprend le contexte global tandis que le module de convolution capture efficacement les corrélations locales de manière synchrone. La figure 5.2 décrit l'architecture d'un bloc Conformer où: FFN est une couche *positionwise feed-forward* (en fonction de la position), MHA est un module d'attention à plusieurs têtes et CONV est un module de convolution. Pour chaque bloc, une normalisation de couche peut être appliquée avant ou après chaque FFN et MHA et une fonction Dropout peut être utilisée. En outre, un mécanisme d'encodage positonnel relatif pour la séquence source est utilisé afin de générer un vecteur d'intégration de la position utilisé ensuite par le module d'attention MHA.

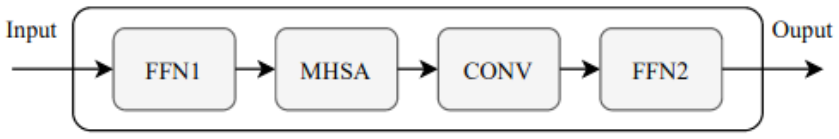



Figure 5.2: Aperçu d'un block Conformer.

Les différences entre un bloc Conformer et un bloc Transformer dans ESPnet peuvent donc se résumer à l'incorporation de trois nouveaux composants : un mécanisme d'encodage positionnel relatif, un module de convolution intégré et enfin, une paire de couches *feed-*

*forward* dans le style *Macaron-Net* [Lu et al., 2020]. Ceux-ci sont décrits dans les parties suivantes.

#### 5.4.2.1 Encodage positionnel relatif

Étant donné un vecteur d'entrée $h^{\text{enc}}_{1:t}$, issu de EncInput$(\cdot)$, le mécanisme d'encodage positionnel relatif génère un vecteur $x^{\text{emb}}$ intégrant des informations sur la position de la séquence d'entrée de différentes longueurs grâce à la procédure proposé par Dai et al. [Dai et al., 2019] (Cf. annexe B). Où $x^{\text{emb}} \in \mathcal{R}^{L_{\max} \times d}$ et la $i$-ème colonne $x^{\text{emb}}_i$ indique la distance relative de $i$ entre deux positions. Le vecteur est ensuite passé au module $MHA$ afin d'injecter dynamiquement les distances relatives dans le calcul du score d'attention, permettant au vecteur *query* de distinguer les représentations de $x_{t,j}$ et $x_{t+1,j}$ pour différentes distances. L'équation suivante décrit l'utilisation du vecteur par le module $MHA$ et remplace l'équation 5.4 donnée pour un bloc Transformer :

$$X_i^{'} = X_i + \text{MHA}_i(X_i, X_i, X_i, X^{emb}) \tag{5.16}$$

#### 5.4.2.2 Module de convolution

La figure 5.3 illustre les détails du module de convolution (CONV) de l'architecture conformer. Celui-ci introduit une couche de convolution 1D ponctuelle, une couche de convolution 1D en profondeur et deux nouvelles fonctions d'activations de type *gated-linear unit* (GLU) et *Swish* [Ramachandran et al., 2018], et pour finir une couche de normalisation par *batch* nommée BN. Ici, la couche de convolution 1D ponctuelle double le nombre de canaux d'entrée, la fonction GLU divise l'entrée selon la dimension du canal et effectue un produit par éléments, et la couche de convolution 1D en profondeur effectue une convolution en profondeur où les canaux sont considérés séparément, suivi d'une convolution ponctuelle mélangeant les canaux.

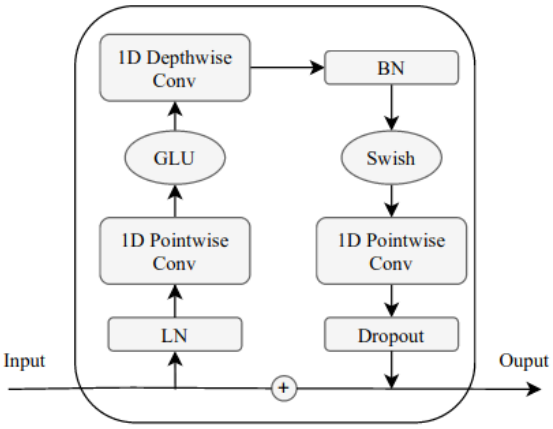



Figure 5.3: Description du module de convolution (CONV).

#### 5.4.2.3 Positionwise feed-forward et Macaron-net

Contrairement au réseau *feed-forward* (FF) de l'architecture Transformer qui utilise une fonction d'activation de type ReLU, le FF pour l'architecture Conformer utilise une fonction d'activation Swish. Cette fonction est identique à celle utilisée dans le cadre du

module de convolution précédent et est définie par :

$$Swish(x) = x \cdot \text{Sigmoid}(\beta\, x) \tag{5.17}$$

Où $\beta$ est une constante ou un paramètre entrainable. Si $\beta = 1$, la fonction est équivalente à la fonction *Sigmoid-weighted Linear Unit* proposé par Elfwing et al. [Elfwing et al., 2018]. Concernant le réseau FF, celui-ci est définit par :

$$\text{FF}(x) = W_2\ \text{Swish}(W_1 x + b_1) + b_2 \tag{5.18}$$

Où $W_1 \in \mathcal{R}^{d^{\text{att}}\ \text{x}\ d^{\text{FF}}}$, $W_2 \in \mathcal{R}^{d^{\text{att}}\ \text{x}\ d^{\text{FF}}}$ sont des matrices de projection linéaire et $d^{att}\ x\ d^{\text{FF}}$ décrit la dimension de la couche linéaire.

En outre, inspiré par le *Macaron-Net* [Lu et al., 2020], l'architecture Conformer suit un schéma en demi-étapes où l'entrée est passée au travers de deux demi-couches *feed-forward* FF prenant en sandwich un module d'auto-attention à plusieurs têtes MHA suivi d'un module de convolution $CONV$. La procédure inclut en outre des fonctions optionnelles de normalisation par couche $LN$ et le module $MHA$ considère ici un encodage de position relatif. Ainsi, 5.4 et 5.5 sont remplacées par les équations suivantes :

$$X_i^{'} = X_i^{'} + \frac{1}{2}\,\text{x}\,\text{FF}_i(X) \tag{5.19}$$

$$X_i^{''} = \text{CONV}_i(X_i^{'} + \text{MHA}_i(X_i^{'}, X_i^{'}, X_i^{'}, X^{emb}) \tag{5.20}$$

$$X_{i+1} = X_i^{''} + \frac{1}{2}\,\text{x}\,\text{FF}_i(\text{CONV}_i(X_i^{''})) \tag{5.21}$$

A la suite de cette proposition, différents travaux ont été initiés par P. Guo et al. dans le cadre du projet ESPnet afin de proposer une étude incluant :

- Une extension de l'architecture Conformer à différents problèmes de traitement de parole bout-en-bout
- Une comparaison étendue avec les architectures Transformer et les architectures RNN pour l'ensemble des tâches de traitement de parole couvertes.
- Des guides pratiques pour l'entraînement de modèles basés Conformer comprenant des indications sur la sélection des paramètres comme la taille du noyau du bloc de convolution ou encore la définition de l'architecture pour différents modèles.
- Un ensemble de recettes et configurations pour reproduire les modèles utilisés pour nos expériences sur plus de 20 corpus. Cela inclut aussi les modèles pré-entrainés.

Le papier [Pengcheng et al., 2021], accepté dans le cadre de la conférence ICASSP 2021, résume les travaux effectués. Ma contribution dans ce projet se focalise sur la tâche de RAP avec, en autres, l'incorporation des Conformer pour les systèmes basés sur la fonction de perte RNN-T, la mise en place de différents guides pour l'entraînement des modèles RNN-Transducer avec les architectures Conformer couplées à différents réseaux de neurones (p. ex., TDNN, VGG, etc) et enfin la mise en place de recettes pour l'entraînement et le décodage de modèles basés sur la fonction de perte RNN-T ou Joint CTC-Attention pour différents corpus.

### 5.4.3 Modèles entraînés avec fonction de perte RNN-T

La première version du modèle Transducer[3] a été rendue disponible à la date du 20 août 2019 dans la boite à outils ESPnet. Initiée en collaboration avec Mingkun Huang, le créateur de la librairie warp-transducer[4] implémentant la fonction de perte RNN-T pour PyTorch et Tensorflow. Cette première version est alors équivalente à la seconde version du modèle proposé par Graves [Graves et al., 2013].

Au travers des avancées relatives à la RAP bout-en-bout et à ce type d'approche, différentes versions du modèle ont vu le jour et de nombreuses fonctionnalités ont été modifiées ou ajoutées au fil des années. Dans cette section, je me focalise sur les architectures, modes d'entraînement et algorithmes d'inférence disponibles dans la version 0.9.7 de la boite à outils ESPnet, datant du 15 janvier 2020.

Le code Python relatif à la majorité des implémentations pour les différentes modélisations, architectures et fonctionnalités présentés dans cette section est consultable à l'adresse suivante : `https://github.com/espnet/espnet/tree/master/espnet/nets/pytorch_backend/transducer`. Parallèlement, la documentation associée à l'entraînement et au décodage de type de modèle avec ESPnet est disponible à cette adresse : `https://espnet.github.io/espnet/tutorial.html#transducer`. À noter que la majorité des fonctionnalités et travaux présentés dans cette section sont présentés dans le papier [Watanabe et al., 2021] qui a été accepté dans le cadre du IEEE Data Science & Learning Workshop.

#### 5.4.3.1 Définition

Un modèle Transducer dans ESPnet correspond à une variante de l'architecture de type encodeur-decodeur présenté précédemment. Le modèle est composé d'un encodeur transformant une séquence source $x_{1:t}$ en une séquence encodée $x^{1:t"}$ où $T \leq T''$:

$$x^{\text{enc}}_{1:t"} = \text{EncInput}(x_{1:t}) \tag{5.22}$$
$$x^{\text{enc}}_{1:t'} = \text{EncBody}(x^{\text{enc}}_{1:t"}) \tag{5.23}$$

D'un décodeur, analogue à un modèle de langage et prédisant la représentation $h^{\text{dec}}_{1:u}$, étant donnée une séquence $y_{1:u-1}$, où $u$ désigne l'index de l'étiquette cible :

$$h^{\text{dec}}_{1:u-1} = \text{DecInput}(y_{1:u-1}) \tag{5.24}$$
$$h^{\text{dec}}_{1:u} = \text{DecBody}(h^{\text{dec}}_{1:u-1}) \tag{5.25}$$

Et d'un réseau combinant les représentations de l'encodeur et du décodeur pour générer la représentation $h^{\text{joint}}_{t,u}$ qui, passé au travers d'une fonction Softmax, produit la distribution des probabilités de la cible actuelle $P(y_{t,u}|x_{1:t}, y_{1:u-1})$:

$$h^{\text{joint}}_{t,u} = \text{JointNetwork}(x^{\text{enc}}_{1:t'}, h^{\text{dec}}_{1:u}) \tag{5.26}$$
$$P(y_{t,u}|x_{1:t}, y_{1:u-1}) = \text{Softmax}(h^{\text{joint}}_{t,u}) \tag{5.27}$$

[3]Du fait de la possibilité d'utiliser différentes architectures hors RNN pour la définition du modèle, le terme Transducer est préféré ici.

[4]https://github.com/HawkAaron/warp-transducer

Où JointNetwork est composé de deux fonctions linéaires projetant les représentations $x_{1:t'}^{\text{enc}}$ et $h_{1:u}^{\text{dec}}$ dans un espace commun afin d'être combinées. La représentation combinée étant ensuite projetée sur l'espace de classes.

Une différence en comparaison des modèles présentés précédemment est l'introduction d'un type *Custom* pour les modules EncBody(·) et DecBody(·) lorsque l'architecture n'utilise pas de couches de type RNN. Alors que la définition des couches et paramètres associés pour les autres modèles se fait de manière "globalisée" (c.-à-d., une seul type de couche est défini par module, qui sera ensuite propagé $L$ fois, où $L$ est le nombre de couches), la définition de l'architecture est plus flexible pour les modèles Transducer. Ainsi, l'utilisateur a la possibilité de définir les modules EncBody(·) et DecBody(·) et les paramètres couche par couche. En outre, il est possible de définir des architectures hybrides où l'encodeur et le décodeur contiennent des types de réseaux de neurones différents. Comme par exemple un encodeur de type Transformer et un décodeur de type RNN. La figure 5.4 illustre les différences entre une définition classique et une définition "libre".

```
# network architecture
## encoder related
etype: blstmp
elayers: 4
eunits: 320
eprojs: 320
subsample: "1_2_2_1_1"
dropout-rate: 0.0
## decoder related
dtype: lstm
dlayers: 1
dec-embed-dim: 300
dunits: 300
dropout-rate-decoder: 0.1
dropout-rate-embed-decoder: 0.1
## joint network related
joint-dim: 300
```

```
# network architecture
## encoder related
etype: custom
custom-enc-input-layer: vgg2l
enc-block-arch:
        - type: transformer
          d_hidden: 320
          d_ff: 320
          heads: 4
          att-dropout-rate: 0.4
enc-block-repeat: 8
## decoder related
dtype: lstm
dlayers: 1
dec-embed-dim: 512
dunits: 512
dropout-rate-decoder: 0.2
dropout-rate-embed-decoder: 0.1
## joint network related
joint-dim: 128
```

Figure 5.4: A gauche une définition d'architecture RNN standard avec des paramètres globaux, et à droite une définition d'architecture personnalisée.

Enfin, des types de couches supplémentaires ont été ajoutés pour être utilisés avec les blocs Transformer ou Conformer, afin d'apporter un peu plus de flexibilité aux utilisateurs. Ces couches sont introduites dans la partie suivante.

#### 5.4.3.2 Architecture

EncInput(·) peut être ici un réseau de couches de convolutions si EncBody(·) est de type Custom, ou un réseau de type VGG avec opération de *max pooling* si EncBody(·) est de type RNN. Contrairement à l'encodeur de la section précédente, l'utilisation d'un réseau de type VGG à deux couches avec opération de *max pooling* est aussi possible si *EncInput*(·) est de type *Custom*. Cet ajout a été motivé par le système Transformer-Transducer de Google [Yeh et al., 2019] incluant un réseau VGG. Celui-ci a aussi démontré son efficacité avec les architectures de type Conformer lors de différents travaux.

EncBody(·) est quant à lui un réseau de neurones récurrents de type RNN bidirectionnel à plusieurs couches par défaut. Dans ce cas, $h_{1:t'}^{\text{enc}}$ est obtenu de manière équivalente aux

autres modèles par l'équation 5.3 de la sous-section 5.2. Si EncBody(·) est de type Custom, l'obtention de $h^{\text{enc}}_{1:t'}$ peut prendre différentes formes : si cette partie de l'architecture contient exclusivement des blocs de type transformer ou conformer alors les équations 5.4 et 5.5 ou 5.19, 5.20 et 5.21 sont utilisées. Si celle-ci admet des blocs de différents types, l'équation définissant EncBody(·) doit être formulée en considération des blocs constituant la fonction. Par exemple, si des couches de type TDNN sont insérées avant les blocs de Transformer, l'équation est ré-écrite de la manière suivante :

$$X_i^{'} = \text{TDNN}(X_i) \tag{5.28}$$
$$X_i^{''} = X_i^{'} + \text{MHA}_i(X_i^{'}, X_i^{'}, X_i^{'}) \tag{5.29}$$
$$X_{i+1} = X_i^{''} + \text{FF}_i(X_i^{''}) \tag{5.30}$$

A noter que le réseau de neurones de type $TDNN$ est seulement accessible dans le cadre des modèles Transducer. Leur utilisation, ainsi que leur ajout dans ESPnet, est motivé par les travaux récents de Tencent AI Lab [Weng et al., 2019].

Concernant la partie décodeur, DecInput(·) désigne ici une couche d'intégration transformant une séquence d'entrée de dimension $D^{\text{labels}}$ en une séquence de dimension $D^{\text{classes}}$. Dans le cas où DecBody(·) utilise des réseaux de neurones récurrents, l'obtention de $h^{\text{dec}}_{1:u}$ est effectué par l'équation $h^{\text{dec}}_{1:u} = \text{LSTM}(h^{\text{dec}}_{1:u-1})$. Si DecBody(·) est de type transformer et contient exclusivement des blocs de type Transformer, les équations suivantes sont utilisées en place et lieu des équations 5.9, 5.10 et 5.11:

$$Y_j^{'}[f] = Y_j[f] + \text{MHA}^{\text{self}}_j(Y_j[f], Y_j, Y_j) \tag{5.31}$$
$$Y_{j+1} = Y_j^{'} + \text{FF}_j(Y^{'}) \tag{5.32}$$

Parallèlement, DecBody(·) doit être formulé en considération des blocs constituant la fonction si ceux-ci ne sont pas tous de type Transformer. Ainsi, si un décodeur est constitué de couches de convolutions causales à 1 dimension suivies par des blocs Transformer par exemple, les équations précédentes sont remplacées par :

$$Y_j^{i}[f] = \text{CausalConv1D}(Y_j) \tag{5.33}$$
$$Y_j^{''}[f] = Y_j^{'}[f] + \text{MHA}^{\text{self}}_j(Y_j^{'}[f], Y_j^{'}, Y_j^{'}) \tag{5.34}$$
$$Y_{j+1} = Y_j^{''} + \text{FF}_j(Y^{''}) \tag{5.35}$$

Tout comme pour les réseaux de type TDNN, l'utilisation de couches de convolution causale n'est disponible que pour les modèles Transducer et seulement pour le module Décodeur. Leur utilisation, ainsi que leur ajout dans ESPnet, ont également été motivés par les travaux de Tencent AI Lab cité précédemment [Weng et al., 2019].

Finalement, le réseau conjoint défini dans l'équation 5.26 permettant d'obtenir $h^{\text{joint}}_{t,u}$ à partir de $h^{\text{enc}}_{1:t}$, issu de l'encodeur, et $h^{\text{dec}}_{1:u}$, issu de l'encodeur, est défini dans ESPnet par :

$$h^{\text{dec}}_{1:u} = \text{Act}(\text{Lin}(h^{\text{enc}}_{1:t}) + \text{Lin}(h^{\text{enc}}_{1:u}) \tag{5.36}$$
$$\tag{5.37}$$

Où Lin est une fonction linéaire permettant de transposer les vecteur $X_e$ et $Y_d[1:u]$ dans l'espace commun avant concaténation et Act désigne une fonction d'activation controlable par l'utilisateur (par défaut ReLU est utilisé).

#### 5.4.3.3 Entraînement

ESPnet possède historiquement deux modes d'entraînement pour les modèles Transducer : le mode standard, définissant l'utilisation d'un décodeur classique tel que définit par l'équation 5.25 et le mode "Transducer attention" définissant l'utilisation d'un décodeur avec mécanisme d'attention considérant la source $h^{\text{enc}}_{1:t}$ tel que définit par l'équation 5.7. Toutefois, par manque d'utilisation et dans un souci d'accélérer le processus de décodage partagé pour tous les modèles, ce second mode a été enlevé. Il fut toutefois mentionné et utilisé dans le cadre des travaux couverts dans le chapitre 6.

A noter que dans le cadre du développement de la nouvelle version de la boite à outils, et de par la volonté d'unifier l'entraînement de l'ensemble des modèles, la fonction de perte utilisée pour l'entraînement multi-tâches a été étendue pour inclure la fonction de perte RNN-T. :

$$\mathcal{L}_{\text{MTL}} = -\alpha_1 \log p_{\text{att}}(y_{1:u}|x_{1:t}) - \alpha_2 \log p_{\text{ctc}}(y_{1:u}|x_{1:t}) - \alpha_3 \log p_{\text{trans}}(y_{1:u}|x_{1:t}) \tag{5.38}$$

Où $\alpha_1$, $\alpha_2$ et $\alpha_3$ désignent respectivement les poids donnés aux approches CTC, attention encodeur-décodeur et Transducer.

### 5.4.4 Algorithmes de recherche pour le modèle Transducer

Contrairement aux modèles basés sur la CTC et le mécanisme d'attention ne proposant qu'un seul algorithme de recherche en faisceau pour effectuer l'étape d'inférence, 3 algorithmes de recherche en faisceau sont proposés dans ESPnet pour les modèles Transducer. Le premier, notre algorithme par défaut (Sec. 5.4.4.1), étend la séquence d'étiquettes de manière non restreinte, la seconde (Sec. 5.4.4.2) agit de manière synchrone avec les étiquettes et les deux dernières stratégies (Sec. 5.4.4.3 et Sec. 5.4.4.4) fonctionnent selon l'axe du temps. Les sous-sections suivantes donnent une description succinctes de ces procédures.

#### 5.4.4.1 Recherche en faisceau par défaut

La stratégie de décodage par défaut pour le modèle Transducer dans ESPnet est basée sur l'algorithme de recherche de faisceau proposé par Graves dans [Graves, 2012b] et décrite dans la section 3.2.2.2.

Sur la base des expériences initiales, la partie de recherche de préfixe proposé par Graves a été supprimée. Il a été constaté que celle-ci ne garantit pas nécessairement que l'espace de recherche ne soit pas redondant si des vérifications supplémentaires de duplication ne sont pas incluses et que, au final, les hypothèses les plus probables sont toujours retenues.

Ainsi, la suppression de cette partie a permis de réduire le temps de décodage sans sacrifier la précision de la recherche.

#### 5.4.4.2 *Alignment-length synchronous decoding*

**Algorithme 3 :** Alignment-length synchronous decoding [Saon et al., 2020]

**Input:** $h^{\text{enc}}_{1:T}$, $U_{\max}$, $N_{\text{bs}}$ and $N_{\text{best}}$
$B = \{\varnothing, 1, \text{state}^{\text{dec}}_0\}$; $F = \{\}$
**for** $i = 1 \ldots T + U_{\max}$ **do**
  $A = \{\}$
  **for** $(y, \delta_{i-1}(y), \text{state}^{\text{dec}}_{u-1}) \in B$ **do**
    $u = |y|$
    t = i - u + 1
    **if** $t > T$ **then**
      continue
    **end**
    $h^{\text{dec}}_u, \text{state}^{\text{dec}}_u = \text{Decoder}(y, \text{state}^{\text{dec}}_{u-1})$
    $p^{\text{pr}} = \text{Softmax}(\text{Joint}(h^{\text{enc}}_t, h^{\text{dec}}_u))$
    $\delta_i(y) = \delta_{i-1}(y) * p^{\text{pr}}(\varnothing)$
    $A = A \cup \{(y, \delta_i(y), \text{state}^{\text{dec}}_{u-1})\}$
    **if** $t == T$ **then**
      $F = F \cup \{(y, \delta_i(y))\}$
    **end**
    **for** $k \in Y$ **do**
      $\delta_i(y + k) = \delta_{i-1}(y) * p^{\text{pr}}(k)$
      $A = A \cup \{(y + k, \delta_i(y + k), \text{state}^{\text{dec}}_u)\}$
    **end**
  **end**
  $B = \text{PruneAndRecombineHyps}(A)[:N_{\text{bs}}]$
**end**
**Return:** $\text{SortedByScore}(F)[:N_{\text{best}}]$

*Alignment-length synchronous decoding* est la procédure proposée dans [Saon et al., 2020] s'effectuant le long de l'axe $\{1, ..., U\}$ et utilise le paramètre $U_{\max}$, une fraction de $T$ définie par l'utilisateur, pour estimer la longueur maximale de la séquence de sortie. La procédure conserve la trace de 2 ensembles d'hypothèses $A$ et $B$ pour les étapes d'alignement $i$ et $i - 1$. À l'étape $i$, pour chaque hypothèse de $B$, le nombre de trames couvertes par la séquence de sortie $y$ est calculé en soustrayant la longueur de l'hypothèse de $i + 1$. L'hypothèse est ensuite ajoutée à $A$ avec son score ($\delta_x$) mis à jour en ajoutant le score de la transition nulle. Si la dernière trame est atteinte, l'hypothèse est également placée dans l'ensemble des hypothèses finales $F$. Ensuite, les hypothèses $y$ de $A$ sont étendues par chaque étiquette de sortie (minus $\varnothing$), et chaque nouvelle hypothèse est ajoutée à $A$ avec son score correspondant et l'état du réseau du décodeur mis à jour. Enfin, un élagage est appliqué à l'ensemble $A$ et les duplications d'hypothèses sont fusionnées avec leur score respectif additionnés. L'ensemble des hypothèses uniques, réduit à la taille du faisceau ($N_{bs}$), devient l'ensemble $B$ pour l'étape d'alignement suivante. À la fin de la procédure, les N meilleures ($N_{\text{best}}$) hypothèses de $F$ sont retournées triées par score décroissant. La procédure complète est donnée dans Algo. 3.

#### 5.4.4.3 *Time-synchronous decoding*

**Algorithme 4 :** Time synchronous decoding [Saon et al., 2020]

**Input:** $h^{\text{enc}}_{1:T}$, max_sym_exp, $N_{\text{bs}}$ and $N_{\text{best}}$
$B = \{\varnothing, 1, \text{state}^{\text{dec}}_0\}$
**for** *t = 1 ... T* **do**
  $A = \{\}$
  $C = B$
  **for** *v = 1 ...* max_sym_exp **do**
    $D = \{\}$
    **for** $(y, \delta_{t-1,u-1}(y), \text{state}^{\text{dec}}_{u-1})$ **do**
      $h^{\text{dec}}_u, \text{state}^{\text{dec}}_u = \text{Decoder}(y, \text{state}^{\text{dec}}_{u-1})$
      $p^{\text{pr}} = \text{Softmax}(\text{Joint}(h^{\text{enc}}_t, h^{\text{dec}}_u))$
      **if** $y \notin A$ **then**
        $\delta_{t,u-1}(y) = \delta_{t-1,u-1}(y) * p^{\text{pr}}(\varnothing)$
        $A = A \cup \{(y, \delta_{t,u-1}(y), \text{state}^{\text{dec}}_{u-1})\}$
      **else**
        $\delta_{t-1,u}(y) = \delta_{t-1,u-1}(y) * p^{\text{pr}}(k)$
      **end**
      **if** $v <$ max_sym_exp **then**
        **for** $k \in Y$ **do**
          $\delta_{t-1,u}(y+k) = \delta_{t-1,u-1}(y) * p^{\text{pr}}(k)$
          $D = D \cup \{(y+k, \delta_{t-1,u}(y+k), \text{state}^{\text{dec}}_u)\}$
        **end**
      **end**
    **end**
    $C = \text{PruneHyps}(D,\ N_{\text{bs}})$
  **end**
  $B = A$
**end**
**Return:** SortedByScore(B)[:$N_{\text{best}}$]

*Time-synchronous decoding*, ou décodage synchronisé dans le temps en français, est une procédure également proposée par Saon et al. [Saon et al., 2020]. Celle-ci s'exécute sur l'axe $\{1, ..., T\}$ et utilise un paramètre max_sym_exp pour contrôler le nombre d'extensions d'hypothèses à chaque pas de temps. L'algorithme 4 montre la procédure complète et peut être décrit comme suit. Ici, la procédure conserve la trace de 4 ensembles d'hypothèses, où $A$ et $B$ stockent les hypothèses pour les temps $t$ et $t-1$ et $C$ et $D$ stockent les hypothèses pour les étapes d'extension $v-1$ et $v$. À chaque pas de temps, en cas de transition vide, les hypothèses de $C$ sont mises dans $A$ avec le score de la transition nulle ajouté à leur score respectif si $y \notin A$, sinon les scores ($(\delta_x)$ sont additionnés. Pour une transition non-nulle, les hypothèses de $C$ sont développées avec chaque étiquette de sortie (minus $\varnothing$) et sont ajoutées à l'ensemble $D$ avec leur score mis à jour. Ensuite, l'ensemble $D$ est élagué pour définir un nouvel ensemble $C$ limité à $N_{\text{bs}}$ hypothèses. La procédure est ensuite répétée max_sym_exp fois. Lorsque $v$ atteint max_sym_exp, les hypothèses de $A$ sont stockées dans $D$ et la procédure est répétée pour chaque pas de temps. À la fin de la procédure, les N meilleures ($N_{\text{best}}$) hypothèses de $B$ sont retournées triées par score décroissant.

#### 5.4.4.4 N-step Constrained beam search (NSC)

L'algorithme *N-Step Constrained beam search* (abrégé NSC), est ma proposition d'extension de l'algorithme de J. Kim and Y. Lee [Kim and Yoonhan, 2020] qui se propose de contraindre la recherche en faisceau pour les modèles Transducer à une seule et unique émission d'étiquette (plus $\varnothing$) par pas de temps. Cette contrainte est basée sur l'hypothèse qu'un chemin de décodage optimal existe vraisemblablement dans la région d'un treillis de sortie proche de sa diagonale. L'hypothèse sous-jacente étant donc que l'alignement temporel entre les étiquettes de sortie et les trames acoustiques est presque linéaire. Ainsi, des contraintes fortes peuvent être appliquées au nombre d'extensions d'étiquettes à chaque pas de temps, comme par exemple, ici, une formulation symétrique par rapport à l'extension du temps et des étiquettes de sortie.

Bien que les auteurs démontrent l'efficacité de leur algorithme pour différentes tâches de RAP et étudient le nombre d'étiquettes émises à chaque pas de temps pendant la recherche par faisceau standard pour les tâches présentées, nous avons trouvé la contrainte initiale trop stricte, résultant en deux faiblesses notables :

- Dans le cas des tâches dites de "faibles ressources", le nombre d'extension d'étiquettes nécessaire à chaque pas de temps pour atteindre des performances satisfaisantes peut être supérieur à 1, tel qu'on peut le voir avec le tableau 5.1. Ici, notre investigation sur le nombre d'extensions par pas de temps a été menée sur deux corpus plus petits, VIVOS (15 heures) et Voxforge (20 heures), et un nombre significatif d'extensions supérieures à 1 a été observé par rapport à l'investigation initiale (+4.24% pour Voxforge et +7.28% pour VIVOS).
- Si aucune contrainte équivalente n'est appliquée lors de l'entraînement du modèle [Tripathi et al., 2019], les contraintes appliquées lors de l'inférence peuvent pénaliser les hypothèses formulées pour certains cas, comme par exemple les longues séquences.

Afin de répondre aux problématiques citées, je propose donc l'algorithme *N-Step Constrained beam search*. L'algorithme est similaire à TSD (Sec. 5.4.4.3) et étend l'algorithme OSC original à $N$ étapes d'extensions par pas de temps (plus $\varnothing$) par le biais d'une boucle supplémentaire contrôlée par un paramètre $N_{\text{step}}$. Pour surmonter le deuxième problème, une nouvelle condition est ajoutée pour l'étape d'extension finale dans NSC. Si $N_{\text{step}} = 1$ et auto-$N_{\text{step}} > 1$, alors je permets aux hypothèses incomplètes d'être transmises au pas de temps suivant sans ajouter le score de transition nulle. Le paramètre auto-$N_{\text{step}}$ est obtenu par une méthode de comptage, construite sur la recherche de faisceau par défaut (voir Sec. 5.4.4.1), qui calcule le nombre attendu d'expansions nécessaires. La procédure complète est donnée par l'algorithme 5, où les lignes en rouge font référence à nos modifications apportées, et les paramètres $N_{\text{bs}}$ et $N_{\text{best}}$ définissent respectivement la taille du faisceau et les $N$ meilleures hypothèses.

Le tableau 5.2 compare les différentes stratégies synchrones dans le temps sur le corpus de test Voxforge pour les cas problématiques identifiés. On peut observer ici que l'augmentation du nombre d'extension permet une amélioration relative du taux d'erreurs de caractères d'environ 23.5% au détriment d'une augmentation d'une augmentation du facteur temps-réel (+0.39). Dans le cas où le nombre d'extension par pas de temps est fixé à 1, la condition mise en place à un fort impact sur le taux d'erreurs de mots obtenus, diminuant d'environ 27% le taux d'erreurs de caractères. Concernant l'algorithme TSD, nous pouvons voir que dans le cas où l'algorithme émet à chaque pas de temps une seule étiquette non-nulle unique suivi d'une étiquette nulle, les performances sont affectées de la

**Algorithme 5 :** N-step constrained beam search.

**Input:** $h^{\text{enc}}$, $N_{\text{step}}$, auto-$N_{\text{step}}$, $\alpha$, $N_{\text{bs}}$ and $N_{\text{best}}$
$h^{\text{dec}}_{\text{batch}}, \text{state}^{\text{dec}}_{\text{batch}} = \text{BatchDecoder}(\varnothing, \text{Duplicate}(\text{state}^{\text{dec}}_{0}, N_{\text{bs}}))$
$B = \{\varnothing, 1, h^{\text{dec}}_{\text{batch}_0}, \text{state}^{\text{dec}}_{\text{batch}_0}\}$
**for** $t = 1 \ldots T$ **do**
  $A = \text{SortedByLength}(B)$; $B = \{\}$
  **for** $(y_i, \delta(y_i)) \in A$ **do**
    $\delta(y_i) \mathrel{+}= \sum_{\hat{y}} \delta(\hat{y}) * \delta(y_i|\hat{y}, t)$
      **where** $\hat{y} \in pref(y_i) \cap A$ and $|y_i| - |\hat{y}| < \alpha$
  **end**
  $S = \{\}$
  $V = \{\}$
  **for** $n = 1 \ldots N_{\text{step}}$ **do**
    $h^{\text{dec}}_{A} = \{h^{\text{dec}}_{y_1}, ..., h^{\text{dec}}_{y_{N_{\text{bs}}}}\} \in A$
    $p^{\text{pr}} = \text{Softmax}(\text{Joint}(h^{\text{enc}}_{\text{t}}, h^{\text{dec}}_{A}))$
    **for** $(y_i, \delta(y_i), h^{\text{dec}}_{y_i}, \text{state}^{\text{dec}}_{y_i}) \in A$ **do**
      $\delta(y_i) = \delta(y_i) * p^{\text{pr}}(\varnothing)$
      $S = S \cup \{(y_i, \delta(y_i), h^{\text{dec}}_{y_i}, \text{state}^{\text{dec}}_{y_i})\}$
      **for** $k \in Y$ **do**
        $\delta(y_i + k) = \delta(y_i) * p^{\text{pr}}(k)$
        $V = V \cup \{(y_i + k, \delta(y_i + k), h^{\text{dec}}_{y_i}, \text{state}^{\text{dec}}_{y_i})\}$
      **end**
    **end**
    $V = \text{SubstractSet}(\text{SortedByScore}(V),\ B)[:N_{\text{bs}}]$
    $y_{\text{batch}} = \{y_1, ..., y_{N_{\text{bs}}}\} \in V$
    $\text{state}^{\text{dec}}_{\text{batch}} = \{\text{state}^{\text{dec}}_{y_1}, ..., \text{state}^{\text{dec}}_{y_{N_{\text{bs}}}}\} \in V$
    $h^{\text{dec}}_{\text{batch}}, \text{state}^{\text{dec}}_{\text{batch}} = \text{BatchDecoder}(y_{\text{batch}}, \text{state}^{\text{dec}}_{\text{batch}})$
    **if** $n < N_{\text{step}} - 1$ **then**
      **for** $(y_i, h^{\text{dec}}_{y_i}, \text{state}^{\text{dec}}_{y_i}) \in V$ **do**
        $h^{\text{dec}}_{y_i} = h^{\text{dec}}_{\text{batch}_i}$
        $\text{state}^{\text{dec}}_{y_i} = \text{state}^{\text{dec}}_{\text{batch}_i}$
      **end**
      $A = V$
    **else**
      $p^{\text{pr}} = \text{Softmax}(\text{Joint}(h^{\text{enc}}_{\text{t}}, h^{\text{dec}}_{\text{batch}}))$
      **for** $(y_i, \delta(y_i), h^{\text{dec}}_{y_i}, \text{state}^{\text{dec}}_{y_i}) \in V$ **do**
        **if** $N_{step} = 1$ *and auto-*$N_{step} = 1$ **then**
          $\delta(y_i) = \delta(y_i) * p^{\text{pr}}(\varnothing)$
        **end**
        $h^{\text{dec}}_{y_i} = h^{\text{dec}}_{\text{batch}_i}$
        $\text{state}^{\text{dec}}_{y_i} = \text{state}^{\text{dec}}_{\text{batch}_i}$
      **end**
    **end**
  **end**
  $B = \text{SortedByScore}(S + V)[:N_{\text{bs}}]$
**end**
**return:** $\text{SortedByScore}(B)[:N_{\text{best}}]$

Table 5.1: Nombre d'extensions par pas de temps (%) durant la phase d'extension.

| | VIVOS | Voxforge | TIMIT [Kim and Yoonhan, 2020] |
|---|---|---|---|
| Num. exp | % exp. | % exp. | % exp. |
| 1 | 89.62 | 93.18 | 97.73 |
| 2 | 9.52 | 6.48 | 2.24 |
| 3 | 0.86 | 0.34 | 0.03 |

même manière que pour l'algorithme OSC (+6.2% TEC). Ce qui confirme notre hypothèse initiale et la mise en place de la condition dans le cadre de l'algorithme NSC.

| Algorithme (param) | TEC | FTR |
|---|---|---|
| default | 12.4 | 0.127 |
| OSC (nstep = 1) | 22.6 | 0.105 |
| NSC (nstep = 1) | 16.2 | 0.098 |
| NSC (nstep = 2) | 12.4 | 0.144 |
| NSC (nstep = 3) | 12.1 | 0.188 |
| TSD (max_sym_exp = 2) | 22.4 | 0.09 |

Table 5.2: Comparaison du taux d'erreurs de caractères (TEC) et Facteur Temps-Réel pour les algorithmes OSC et NSC sur le sous-corpus de test de Voxforge.

### 5.4.5 Recettes pour les modèles Transducer

L'ensemble des corpus associés aux recettes présentées dans cette sous-section est décrit dans le chapitre 4. Il est toutefois important de noter que les recettes présentées ici sont données afin de démontrer les contributions précédentes et ne représentent qu'une partie des recettes disponibles dans ESPnet pour les modèles Transducer.

#### 5.4.5.1 VIVOS

Le corpus VIVOS [Luong and Vu, 2016], contenant une vingtaine d'heures d'enregistrement de la parole en vietnamien, est le corpus principalement utilisé pour valider les fonctionnalités et architectures associées aux modèles Transducer dans ESPnet. Pour entraîner les différents modèles, je me base sur les scripts mis à disposition dans ESPnet par les auteurs du corpus. Une modification a toutefois été apportée afin d'utiliser la technique de *checkpoint averaging* proposée par Vaswani & al. [Vaswani et al., 2017] lorsque des blocs de type Transformer sont utilisés.

De nombreuses configurations d'entraînement pour ce corpus ont été proposées et regroupées à cette adresse `https://github.com/espnet/espnet/tree/master/egs/vivos/asr1/conf/tuning/transducer`. Les configurations permettent l'entraînement de modèles Transducer avec les architectures Encodeur-Décodeur suivantes :

- Encodeur RNN et Décodeur RNN.
- Encodeur Transformer et Décodeur Transformer
- Encodeur Transformer et Décodeur RNN
- Encodeur TDNN + Transformer et Décodeur causal-convolution 1D + Transformer
- Encodeur Conformer et Décodeur Transformer

- Encodeur Conformer et Décodeur RNN

Pour effectuer l'inférence avec les différents modèles entraînés, des configurations pour l'ensemble des algorithmes de recherche introduits dans 5.4.4 sont aussi mises à disposition à l'adresse précédente.

L'ensemble des modèles pré-entraînés est par ailleurs disponible sur une plateforme mise à disposition pour le projet. Les liens des modèles ainsi que les performances obtenues sur les corpus de développement et de test sont regroupés au lien suivant : `https://github.com/espnet/espnet/blob/master/egs/vivos/asr1/RESULTS.md`. Le tableau suivant présente les résultats obtenus en terme de taux d'erreurs de caractères (TEC) et de taux d'erreurs de syllabes (TESy) sur le corpus de test de VIVOS pour différents types d'architecture. A noter toutefois, qu'au moment d'écrire ce manuscrit, ces résultats peuvent avoir été modifiées, du fait de la constante évolution des modèles. Les architectures pour la partie encodeur et la partie décodeur sont séparées par un slash.

| Modèle | Unité | LM | TEC | TESy |
|---|---|---|---|---|
| HMM-DNN [Luong and Vu, 2016] | phonème | Oui | X | 9.48 |
| RNN CTC [5] | caractère | Non | 22.2 | 54.7 |
| RNN/RNN Transducer | caractère | Non | 18.4 | 35.2 |
| TDNN-Transformer/RNN Transducer | caractère | Non | 16.2 | 33.9 |
| TDNN-Conformer/RNN Transducer | caractère | Non | 14.0 | 31.5 |

Malgré une amélioration significative du TEC et du TESy comparé au modèle bout-en-bout initial, de nombreuses optimisations sont encore nécessaires afin d'atteindre les performances obtenues avec un système traditionnel HMM-DNN.

#### 5.4.5.2 Voxforge

Afin de démontrer l'utilisation des méthodes d'affinage et leur efficacité pour les modèles Transducer, un ensemble de scripts et de configurations ont été créés dans ESPnet. Le corpus Voxforge italien (Cf Sec. 4.3), comprenant une vingtaine d'heures d'enregistrement de la parole, a été utilisé pour l'entraînement et l'évaluation des différents modèles.

Cet ensemble est composé des fichiers suivants :

- **train_transducer.yaml**: le fichier de configuration définissant un modèle Transducer avec un encodeur RNN et un Décodeur RNN.
- **decode_*.yaml**: les fichier de configuration définissant le type d'algorithme et les paramètres associés pour effectuer l'inférence. L'ensemble des algorithmes définis dans la section 5.4.4 sont disponibles.
- **prep_transducer_finetuning.sh**: un script Bash qui, étant donné des paramètres d'affinage en entrée, renvoie un ensemble de configurations pour effectuer le pré-entraînement d'un ou plusieurs modèles ainsi que l'entraînement du modèle basé transducer avec une phase pré-initialisation.
- **run_rnnt.sh**: le script principal effectuant la préparation des données, l'extraction des données et permettant l'entraînement d'un ou plusieurs modèles selon les options d'affinage sélectionnées.

La particularité de cette recette en comparaison avec les autres recettes existantes dans la boite à outils ESPnet est le fait que, au travers d'un fichier de configuration unique et d'options définissant le type de transfert de connaissance à effectuer (appliqué soit

à l'encodeur, au décodeur ou aux deux parties), le processus complet est automatisé. L'ensemble des configurations pour les modèles servant au transfert de connaissance est généré à la volée et les différentes phases d'entraînement sont effectuées automatiquement. Cela permet en outre de s'assurer de la compatibilité entre le modèle Transducer principal et les modèles servant à la pré-initialisation des paramètres du premier.

Le tableau suivant présente les performances obtenues en terme de taux d'erreurs de caractères (TEC) sur le corpus de test de Voxforge pour différents modes d'entraînements, avec et sans transfert d'apprentissage, et architectures. Par souci de comparaison, j'associe aussi les résultats pour les modèles basés RNN et basés Conformer développés par différents collaborateurs du projet ESPnet. Ces résultats correspondent aux derniers résultats obtenus avec la dernière version du corpus Voxforge datant de janvier 2021.

| Modèle | Pre-init (enc, dec) | TEC |
|---|---|---|
| RNN/RNN CTC-Att. [Karita et al., 2019] | Non, Non | 13.7 |
| T/T CTC-Att. [Karita et al., 2019] | Non, Non | 9.6 |
| C/T CTC-Att. [Pengcheng et al., 2021] | Non, Non | 8.8 |
| RNN/RNN Transducer | Non, Non | 12.4 |
| RNN/RNN Transducer | CTC, Non | 12.0 |
| RNN/RNN Transducer | CTC, LM | 11.8 |
| T/RNN Transducer | Non, Non | 9.8 |
| C/RNN Transducer | Non, Non | 8.6 |
| C/RNN Transducer | CTC, Non | 8.3 |

Où T désigne une architecture Transformer, C désigne une architecture Conformer, CTC-Att un modèle entrainé en multi-tâches avec CTC et mécanisme d'attention, et Att. un modèle basé attention seul.

A noter qu'à partir de la version 0.9.3 de la boite à outils, le choix a été fait d'arrêter le support de cette recette, ou tout du moins le support des scripts permettant l'automatisation du processus décrit précédemment. La raison est double : d'une part car la maturité actuelle du module pour les opérations d'affinage et la documentation associée permettent d'effectuer facilement les opérations manuellement; et d'une autre part car le développement lié aux modèles Transducer est très actif, demandant un travail considérable pour le maintien de ces scripts.

## 5.5 Travaux annexes utilisant mes contributions

Un certain nombre de publications, de travaux et de projets s'appuient sur les contributions présentées dans la section précédente. Je relève ici les plus notables (soumis en dehors du projet ESPnet) et connus au moment de l'écriture de ce manuscrit. Une introduction succincte des modèles et fonctionnalités utilisées est donnée.

### 5.5.1 Étude de la RAP bout-en-bout pour le français

La première mention des modèles Transducer de ESPnet et leur utilisation à des fins de recherche correspond à mon étude des systèmes de RAP bout-en-bout pour le français conversationnel [Boyer and Rouas, 2019] et dont un chapitre de ce manuscrit lui est consacré 6. Durant cette étude, j'ai pu mettre en évidence pour la première fois les performances

de ce type de modèle par rapport aux autres approches bout-en-bout disponibles dans ESPnet ainsi que les approches traditionnelles basés HMM.

### 5.5.2 Étude de la RAP bout-en-bout pour le russe

Inspiré par mes travaux sur le français, Andrei Andrusenko et al. [Andrusenko et al., 2020a] ont proposé en 2020 une exploration des systèmes de RAP bout-en-bout pour le Russe sur la base du corpus OpenSTT [6]. Ce corpus propose différents types de parole (lu, conversationnel, commande) ainsi que différents sources d'acquisition (appels téléphoniques, vidéos YouTube, livres audio, etc).

L'étude comprend une comparaison similaire à celle proposée dans [Boyer and Rouas, 2019], où trois systèmes bout-en-bout (RNN Transducer, RNN joint CTC-Attention et Transformer) sont entraînés pour modéliser chacun deux types d'unités acoustiques cibles (caractère et sous-mots). La comparaison inclut aussi, comme système de référence, un système TDNN-HMM basé phonème entrainé avec LF-MMI.

Le modèle RNN-Transducer proposé par les auteurs s'avère comparable au modèle CTC-Attention avec architecture RNN que cela soit avec des caractères ou des sous-mots mais reste en deçà des performances proposées par le modèle Transformer entraîné avec entropie croisée. Sans l'utilisation de modèle de langage, le modèle Transducer obtient un taux d'erreurs de mots (TEM) nettement inférieur sur le sous-ensemble YouTube, avec respectivement : 21.7% pour le modèle basé caractère contre 23.2% pour le modèle CTC-Attention basé caractère, et 20.3% modèle basé sous-mots contre 23.0% pour le modèle CTC-Attention basé sous-mots. Toutefois, les auteurs ont observé des dégradations significatives par l'incorporation des probabilités d'un modèle de langage externe dans le processus d'inférence. Ils émettent en outre des réserves sur la manière dont les probabilités sont combinées ("soft fusion" dans le cas d'ESPnet) notamment pour le modèle entraîné avec fonction de perte RNN-T. Une discussion avec les auteurs du papier et des expériences complémentaires sont en cours sur ce point.

### 5.5.3 Challenge CHIME-6

Motivés par les résultats obtenus sur le corpus OpenSTT, les précédents auteurs ont décidé d'explorer l'utilisation de modèles bout-en-bout pour une tâche plus complexe, à savoir la reconnaissance de parole dans des environnements bruyants et avec peu de ressources [Andrusenko et al., 2020b]. Pour se faire, les auteurs utilisent les données distribuées dans le cadre du challenge CHIME-6.

Ils montrent ainsi que l'utilisation d'un modèle RNN-Transducer, couplé à des techniques d'augmentation et amélioration de la parole, permet d'obtenir des résultats compétitifs par rapport à des systèmes traditionnels hybrides et autres variantes bout-en-bout utilisant aussi bien des architectures RNN que Transformer. Leur système RNN-Transducer obtient un taux d'erreurs de mots de 55% sur l'ensemble de test tandis que les autres systèmes obtiennent respectivement : 82.1% pour le modèle combinant CTC et Attention, 80.7% pour le modèle basé réseaux de neurones convolutionnels multi-canaux et 51.7% pour le modèle hybride TDNN-F entraîné avec LF-MMI.

[6] `https://github.com/snakers4/open_stt`

### 5.5.4 Boite à outils NeMo

Plus récemment, des membres de NVidia ont proposé, dans le cadre de la boite à outils NeMo[7] permettant la création d'applications d'IA conversationnelle, un modèle Transducer pour la tâche de RAP. Si la définition du modèle ainsi que le mode d'entraînement sont quelque peu différents de ceux proposés dans ESPnet, les auteurs proposent aussi l'ensemble des algorithmes de recherche mentionnés dans la section 5.4.4, sur la base de ce que j'ai proposé dans ESPnet.

À ce jour, je n'ai toutefois pas eu de retours concernant l'expérience utilisateur ou l'utilisation de ces algorithmes dans le cadre de travaux par NVidia ou autres entités utilisant NeMo.

## 5.6 Développements en cours et futurs

Plusieurs axes de recherche et de développement sont en cours ou à venir en ce qui concerne mon travail dans le projet ESPnet.

Au moment de l'écriture de ce manuscrit, le développement prioritaire concerne le portage des modèles Transducer vers la seconde version du projet, appelé sobrement ESPnet2, et dont les principaux changements sont les suivantes :

- Arrêt du support du backend Chainer.
- Extraction de caractéristiques et pré-traitement de données textuelles à la volée. Dans ce cadre, des outils internes sont développés pour remplacer l'utilisation d'outils provenant de Kaldi. Ce dernier n'étant plus un pré-requis pour l'installation d'ESPnet.
- Support de l'entraînement distribué en parallèle.
- Et enfin, refactorisation majeure du code comprenant notamment une interface unifiant l'ensemble des encodeurs et décodeurs existants.

En ce qui concerne mon travail sur ce portage, celui-ci est double : d'une part repenser l'implémentation des modèles Transducer en considération de l'interface unifiée, et d'une autre part adapter la définition personnalisée d'architecture pour l'ensemble des modèles. Au moment de l'écriture de ce manuscrit, différentes propositions ont été faites et des versions fonctionnelles sont actuellement à l'étude. Des discussions internes sont toujours en cours concernant l'unification des différentes modélisations en considération de l'impact sur les modèles et ce, sur différents aspects: performance, taille, temps de traitement, etc.

A la suite de ce portage, la priorité est donnée au développement d'une librairie Torch, avec support CUDA/GPU, pour l'entraînement et l'inférence des modèles Transducer. Si des librairies existent déjà pour l'entraînement, sur lesquelles je m'appuie d'ailleurs dans ESPnet, notre volonté est de pouvoir proposer un ensemble de techniques pour palier aux limitations observées avec ces modèles, à savoir : accélérer l'entraînement, réduire l'utilisation mémoire ou encore permettre d'utiliser des modes d'entraînement alternatifs. Les principales techniques retenues pour ces points étant, dans le désordre: Monotonique RNN-T [Tripathi et al., 2019], contraignant le nombre d'étiquettes émises par pas de temps à 1 lors de l'entraînement, Neural Transducer [Jaitly et al., 2015], permettant d'effectuer des prédictions incrémentielles à mesure que de nouvelles données arrivent sans besoin de re-calcul, et enfin, une approche non-nommée [Li et al., 2019], permettant de combiner

[7] https://github.com/NVIDIA/NeMo

efficacement les sorties de l'encodeur du réseau de prédiction qui est l'étape la plus coûteuse en terme de mémoire.

En ce qui concerne la partie inférence, l'objectif principal est de pouvoir accélérer les algorithmes de recherche alternatifs déjà implémentés dans ESPnet et notamment la procédure de vectorization pour traiter plusieurs hypothèses du faisceau en parallèle. L'implémentation actuelle, faites au niveau de l'API python, souffre de ralentissement lors de l'utilisation de ces techniques. Ceci s'explique par le fait que de nombreuses boucles ou étapes de tri sont nécessaires pour les modèles Transducer, dû notamment au fait que le chemin de décodage évolue au travers de deux types de transitions. La solution optimale, outre des optimisations algorithmiques, est de déporter ces opérations au niveau du backend de Pytorch, en les incluant dans la librairie Torch en développement que j'ai mentionné dans le paragraphe précédent.

Le dernier axe de travail concerne le processus de décodage en continu et le déploiement sur des machines à faible ressources pour les modèles Transducer. En s'appuyant sur les travaux de Google [He et al., 2019] ou Microsoft [Chen et al., 2020] sur le sujet, nous souhaitons proposer une solution compétitive en terme de temps de traitement par rapport à des solutions matures comme celles issues de Kaldi, et utilisables sur des appareils mobiles ou des machines avec de faibles ressources. Notre objectif étant de démocratiser l'utilisation des modèles bout-en-bout pour les non-experts et permettre le déploiement de solutions industrielles encore rare à ce jour pour ces modèles.

# Chapter 6

# Modèles bout-en-bout pour le français

Ce chapitre décrit les travaux conduits sur l'étude des systèmes de RAP bout-en-bout pour le français. Celui-ci ce découpe en deux parties : la première partie se focalise sur les travaux initiaux s'appliquant à la résolution du problème de mots hors-vocabulaire dans le cadre des systèmes de RAP traditionnel, au travers des modules linguistiques, tandis que la seconde, présente l'étude effectuée sur les systèmes de RAP bout-en-bout en Français, qui, par nature, ne sont pas affectés par ce problème. Pour cette dernière partie, un article portant sur cette étude est aussi disponible au format numérique sur la plateforme libre arXiv [Boyer and Rouas, 2019].

## 6.1 Introduction

Un des défis principaux en reconnaissance automatique de parole est la reconnaissance d'un vocabulaire ouvert, c'est-à-dire comportant des mots partiellement vus ou non. Le système de RAP admettant un vocabulaire connu, défini au travers du dictionnaire (Sec. 2.3.1) associant des représentations de plus bas niveau correspondant aux classes modélisées par le modèle acoustique, le plus souvent de nature phonétique. Celui-ci n'est pas conçu pour reconnaître les mots hors de ce périmètre. Ce problème, bien connu, est appelé problème des mots hors-vocabulaire (ou *Out-Of-Vocabulary* en anglais, abrégé OOV). Habituellement, ces mots sont corrélés avec leur fréquence d'apparition dans une langue, correspondant souvent à des noms propres ou communs, termes empruntés à une autre langue, etc. [Ferrer-i-Cancho and Vole, 2002]

Afin de palier à ce problème, de nombreuses méthodes sont apparues au fil des années, se focalisant principalement sur les modélisations linguistiques du système (dictionnaire et modèle de langage), que cela soit de manière isolée ou couplée. Au début des années 2010, la majorité des systèmes de RAP traditionnels reposent sur un vocabulaire hybride contenant des mots complets et des représentations intermédiaires, le plus souvent des sous-mots [M. A. Basha Shaik et al., 2012, Kozielski et al., 2013]. Bien que ces approches soient efficaces pour reconnaître correctement la plupart des mots hors vocabulaire, elles ne parviennent pas, du fait du vocabulaire intermédiaire limité, à reconnaître des mots ayant de nombreuses formes dont certaines rares ou évoluant constamment en considération de la langue. De plus, pendant la reconnaissance, le remplacement d'un mot hors vocabulaire entraîne régulièrement des erreurs dans le voisinage.

Plus récemment, la reconnaissance de parole bout-en-bout (Cf. Chap. 3) dans laquelle les architectures neuronales sont entraînées pour modéliser directement des séquences de caractéristiques en représentations orthographiques ont été intensivement étudiées pour le problème des mots hors-vocabulaire. La capacité naturelle des méthodes bout-en-bout à représenter les termes d'une langue au travers d'une représentation directe (c'est-à-dire ne demandant pas des connaissances et transformations intermédiaires -phonétiques- pour restituer une transcription orthographique) permettant de s'affranchir du problème de mots hors-vocabulaire. L'ensemble des caractères d'une langue cible pouvant par exemple représenter l'ensemble des mots actuels ou futurs de celle-ci.

Durant ma thèse, je me suis focalisé dans un premier temps sur l'étude et le développement de méthodes servant à reconnaître les mots hors-vocabulaire dans les systèmes de RAP traditionnels basés phonèmes et représentés au travers de transducteurs à états finis (Sec. 2.5.2). Toutefois, malgré des avancées prometteuses sur ce sujet, je me suis rapidement tourné vers les approches bout-en-bout du fait de leur capacité naturelle à traiter le problème des mots hors-vocabulaire et aussi de par l'intérêt important de la communauté scientifique pour ces modèles.

La section suivante, faisant office de préambule, présente succinctement les approches étudiées avec les systèmes de RAP traditionnels d'une manière générale (sans dépendance au langage). Les sections suivantes se focalisent sur mon étude des systèmes de RAP bout-en-bout pour le français.

## 6.2 Reconnaissance de parole traditionnelle

Les travaux effectués dans cette partie se focalisent sur les modèles linguistiques d'un système de RAP traditionnel, à savoir le dictionnaire de prononciation et le modèle de langage, correspondant respectivement à $L$ et $G$ dans un contexte de représentation basée transducteurs à états finis pondérés (Sec. 2.5.2). Les corpus utilisées pour ces travaux sont : ESTER1 (Sec. 4.1), BREF (Sec. 4.2) et le corpus Voxforge anglais [1].

Pour chaque corpus, je définis comme mot hors-vocabulaire l'ensemble des mots définis n'étant pas représenté dans les dictionnaires utilisés par les systèmes développés, c'est-à-dire le dictionnaire du LIUM[2] pour la langue française et contenant environ 60K mots lors de son acquisition, et le dictionnaire du CMU pour la langue anglaise et contenant environ 120K mots au moment de son acquisition. Les mots définis comme hors-vocabulaire sont alors remplacés par un symbole spécial: ⟨UNK⟩. A partir de là, une mesure sur la fréquence d'occurrence de chaque mot est effectuée afin de faire un classement pour chaque corpus. Tous les termes ayant une fréquence de 1 sont définis comme étant hors-vocabulaire et sont remplacés. Le pourcentage final de mots hors-vocabulaire pour chaque corpus est alors : 3.8% pour ESTER1, 2.7% pour BREF et 1.6% pour Voxforge.

Dans le cadre du corpus ESTER, une partie des analyses complémentaires sur les mots hors-vocabulaires a été effectuée. Ces analyses ont servi dans le contexte de l'évaluation des erreurs formulées pour la RAP bout-en-bout en français présenté dans le chapitre suivant (Cf. Sec. 7).

[1] Disponible à cette adresse : `http://www.repository.voxforge1.org/downloads/SpeechCorpus/Trunk/`

[2] Disponible via le projet CMU Sphinx : `https://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/French/`

Une note importante pour le lecteur, dans la section suivante aucun résultat en terme de taux d'erreurs de mots et de couverture de mots hors-vocabulaire ne sera donné. La raison étant que les disques durs du serveur utilisé pour ces travaux ont été endommagés durant ma thèse, et je ne peux consciemment rapporter des résultats que je ne peux vérifier sans ré-itérer les expériences. Des commentaires généraux sur la finalité des travaux effectués, provenant de mes rapports manuscrits, seront toutefois donnés à titre indicatif.

### 6.2.1 Expansion de lexique

L'approche la plus directe afin de diminuer le nombre de mots hors-vocabulaire est de simplement augmenter en amont la couverture du lexique utilisé dans le système de RAP traditionnel. Pour cela, une expertise humaine ou des méthodes automatiques peuvent être utilisées. Pour des raisons évidentes, la seconde catégorie est privilégiée mais en pratique, notamment dans le cadre industriel, celle-ci s'accompagne d'une vérification et validation humaine à posteriori.

Habituellement, les modélisations de type "graphème à phonème" (ou *grapheme-to-phoneme* en anglais, abrégé g2p) sont privilégiées afin de représenter les mots hors-vocabulaire. Dans ce cadre, j'ai choisi la méthode proposée par M. Bisani et H. Ney [Bisani and Ney, 2005] et qui est mise à disposition dans le cadre du projet libre Sequitur [3] afin d'étendre la couverture des différents dictionnaires. Pour ce faire, la procédure d'augmentation suivante est utilisée:

1. À partir du dictionnaire phonétique de la langue cible, un modèle g2p d'ordre 1 est entraîné.
2. Un dictionnaire de test est crée manuellement pour chaque langue cible, comprenant quelques milliers de mots.
3. Le modèle g2p d'ordre supérieur est entraîné par itération en se servant du modèle d'ordre inférieur jusqu'à atteindre une précision satisfaisante (à savoir 80%, choisi empiriquement) sur le corpus de test créé. Cela résulte en un modèle d'ordre 4 pour le français et d'ordre 3 pour l'anglais.
4. Pour chaque corpus (ESTER1 et BREF en français et Voxforge en anglais), les représentations phonétiques des mots désignés hors-vocabulaire sont générées.
5. À partir de ces représentations, une transposition de la procédure est effectuée (c.-à-d., phonème à graphème) afin de re-générer les représentations orthographiques.
6. Une mesure de similarité, basée sur le taux d'erreurs de caractères, est effectuée entre les représentations graphémiques d'origine et générées.
7. Si le taux d'erreur obtenu est inférieur à 10%,, la paire "mot d'origine - séquences de phonèmes générées" est ajoutée au dictionnaire phonétique.

Au final, le pourcentage de mots hors-vocabulaire pour le corpus d'entraînement d'ESTER1 et de Voxforge est diminué d'environ 38% et 30% d'après mes rapports. Dans le cadre de BREF, la diminution observée est relativement faible ( 11%), cela s'explique par le nombre conséquent de noms propres et communs contenant des lettres muettes ou des orthographes rares dans le corpus. Dans ce cadre, la majorité des paires "mots - représentation phonétique" générées ne passent pas la dernière étape de la procédure.

[3] `https://github.com/sequitur-g2p/sequitur-g2p`

### 6.2.2 Modèle de langage hybride

Une autre approche possible afin de réduire le nombre de mots hors-vocabulaire dans les systèmes de RAP traditionnels est d'utiliser un modèle de langage hybride, où le modèle de langage basé mots est utilisé pour les termes du vocabulaire connu en association avec un autre modèle de langage modélisant une représentation intermédiaire (habituellement des caractères) pour les termes hors-vocabulaire. Ainsi, durant la phase de décodage, lorsque le symbole spécial ⟨UNK⟩ est rencontré, le processus de décodage s'appuie sur la représentation du second modèle. Si la procédure est attrayante, celle-ci s'accompagne toutefois par une limitation importante dans le cadre des systèmes traditionnels représentant les modélisations au travers des transducteurs à états finis pondérés: le modèle de langage intermédiaire doit être associé pour représenter chaque occurrence du symbole spécial, rendant la composition du dictionnaire et du modèle de langage (dénoté L ∘ G) extrêmement lourde en terme d'utilisation mémoire.

Afin de palier à ce problème, et rendre la composition possible à la volée, différentes méthodes ont été proposées au fil des années pour la tâche de RAP et la reconnaissance de mots écrits. Celles-ci s'appliquent à réduire le nombre de chemins accessibles durant la composition et ainsi, réduire la mémoire utilisée [M. A. Basha Shaik et al., 2012, Kozielski et al., 2013, Messina and Kermovant, 2014]. Dans ce cadre, la méthode introduite par R. Messina et C. Kermorvant [Messina and Kermovant, 2014] pour la tâche de reconnaissance de mots écrits a retenu mon attention. Sur la base de leurs travaux, je propose de développer une méthode équivalente pour la tâche de RAP dont la procédure générale est la suivante.

1. Une liste de mots est tout d'abord générée, constituée des mots hors-vocabulaires définis pour chaque corpus d'entraînement, à partir des lexiques associés à chaque système.
2. À partir de cette liste, un modèle de langage N-gramme (Cf. Section 2.3.2) d'ordre 3 pour le français et d'ordre 2 pour l'anglais sont entraînés.
3. Chaque transition de $LG$ (composition du transducteur du dictionnaire phonétique $L$ et du transducteur du modèle de langage $G$) rentrante ou sortante de l'état inconnu et ayant une ⟨UNK⟩-transition est modifiée tel que : 1) si le symbole ⟨UNK⟩ apparaît comme unigramme, les chemins sont combinés, 2) s'il apparaît comme un bigramme ou un trigramme, les chemins sont combinés pour chaque position respective (gauche-droite pour un bigramme et gauche-milieu-droite pour un trigramme).
4. Le transducteur du modèle de langage intermédiaire, ici $G_c$, est ensuite branché à l'état du symbole inconnu et des états relatifs aux mots environnants au travers de transitions spontanées (c.-à-d., $\epsilon$-transition, une transition ne consommant pas un symbole d'entrée), remplaçant les ⟨UNK⟩-transitions précédentes.
5. Le transducteur représentant le système de RAP (c.-à-d., $H \circ C \circ L \circ G$) est ensuite recomposé: $H \circ C \circ L \circ G_m$.

L'utilisation de cette procédure a permis en définitive de réduire le taux d'erreurs de mots de respectivement : 17% pour ESTER1, 13% pour Voxforge et 8% pour BREF. Toutefois, contrairement à ce que les auteurs rapportent, l'utilisation mémoire dénotée est bien plus grande dans mon cas (augmentation de l'ordre de 76%), ce qui m'a empêché de considérer une composition à la volée dans le cadre de ces travaux.

## 6.3 Reconnaissance de parole bout-en-bout

Les systèmes bout-en-bout sont de nos jours largement étudiés en RAP et cela pour un grand nombre de langues comme l'anglais, le mandarin ou le japonais. Cependant, pour une langue comme le français, ce type de système reste encore peu étudié en comparaison, aussi bien pour la reconnaissance de parole que les autres tâches. Ce qui peut apparaître surprenant en considération des caractéristiques du français: le grand nombre de lettres muettes, d'homophones ou d'anglicismes par exemple, rendant la comparaison entre les unités orthographiques et phonétiques pertinente.

Dans ce contexte, j'étudie les trois principaux types d'architectures qui ont donné des résultats prometteurs en comparaison des systèmes traditionnels : 1) La classification temporelle connexionniste (CTC, cf. section 3.2.1), 2) les méthodes basées attention (Cf. section 3.2.3) et 3) RNN-Transducer (Cf. section 3.2.2). Je compare ces approches avec un système de RAP traditionnel basé HMM-DNN, qui est ici le système de référence et qui propose des performances comparables aux systèmes de l'état de l'art pour ce corpus (voir section 4.1).

Afin de compléter l'étude, j'ajoute deux autres systèmes bout-en-bout hybrides : 1) Joint CTC-Attention et 2) RNN-Transducer augmenté avec un mécanisme d'attention. En outre, afin de faire le lien avec les approches de RAP traditionnelles et bout-en-bout, je propose d'inclure dans la comparaison un système entrainé avec la méthode *end-to-end lattice-free MMI* qui est une variante *quasi*-bout-en-bout du système traditionnel de référence et qui modélise, en place et lieu des phonèmes, des caractères.

### 6.3.1 Systèmes bout-en-bout hybrides pour la RAP

Cette section décrit succinctement les deux approches bout-en-bout hybrides qui n'ont pas encore présentées dans les chapitres précédents et que j'utilise dans le cadre de mon étude.

#### 6.3.1.1 RNN-Transducer avec mécanisme d'attention

L'architecture RNN-Transducer, augmentée par un mécanisme d'attention, a été mentionnée pour la première fois, à notre connaissance, dans [Prabhavalkar et al., 2017]. Ici, le réseau de prédiction décrit dans section 3.2.2 est remplacé par un réseau utilisant un mécanisme d'attention, similaire à celui présenté dans la section 3.2.3, et aussi utilisé dans la méthode d'entraînement conjoint CTC-attention. Cette modification permet au décodeur d'accéder à des informations acoustiques parallèlement aux séquences prédites précédentes. En pratique, cela se traduit par le remplacement de l'équation 5.25 du chapitre 5 par l'équation suivante, où DecBody$(\cdot)$ est un réseau AttentionLSTM:

$$h^{\text{dec}}_{1:u} = \text{DecBody}(h^{\text{enc}}_{1:t}, h^{\text{dec}}_{1:u-1}) \tag{6.1}$$

Comme le calcul des représentations en sortie n'est pas affecté par cette modification (le calcul des sorties du décodeur et des sorties jointes ne dépendent pas d'un choix particulier de segmentation), l'architecture peut être entraînée avec le même algorithme forward-backward que celui utilisé pour l'architecture RNN-Transducer standard. Enfin, contrairement à la procédure hybride précédente, l'étape d'inférence peut être effectuée par le biais d'un algorithme de recherche glouton ou de faisceau standard.

#### 6.3.1.2 *End-to-end lattice-free MMI*

*End-to-end lattice-free MMI* [Hadian et al., 2018] est, comme son nom l'indique, la version bout-en-bout du critère introduit par Povey et al. [Povey et al., 2016] pour l'entraînement discriminant par séquence des modèles hybrides HMM-DNN. Dans cette version, une méthode de démarrage "à plat" est adoptée afin de supprimer 1) la nécessité d'entraîner un HMM-GMM initial pour l'obtention des alignements et 2) le *pipeline* de construction d'arbres contextuels.

Bien que l'approche ressemble plus à une adaptation à plat de la méthode initiale qu'à une approche bout en bout en terme d'architecture, et bien qu'elle ne bénéficie pas de la propriété de vocabulaire "ouvert" permettant la construction des mots non-définis au préalable par rapport aux méthodes présentées précédemment, nous ajoutons cette approche dans notre études pour les raisons suivantes : 1) celle-ci permet l'utilisation de différentes unités acoustiques, dont des unités orthographiques, et 2) elle propose des performances comparables à l'approche originale, et ce, quelles qu'en soient les unités acoustiques choisies. Ainsi, cela nous permet d'effectuer une comparaison portant sur les différentes unités acoustiques, dont la relation n'est pas directe, ainsi que sur les productions des systèmes ouverts et des systèmes plus contraints, comme les systèmes hybrides, où la relation entre les unités acoustiques et une représentation au niveau du mot est limitée.

## 6.4 Base de données

L'ensemble des expériences a été mené sur la base des données fournie lors de la campagne d'évaluation ESTER (*Évaluation des systèmes de transcription enrichie d'émissions radiophoniques*) (Sec. 4.1). L'évaluation est effectuée sur le jeu de test fourni dans le cadre de la seconde campagne. Les détails de l'ensemble de données, correspondant à 6h34 de parole, sont donnés dans [Galliano et al., 2009]. J'utilise les mêmes règles de normalisation et de notation que dans le plan d'évaluation de la campagne ESTER 2, à l'exception du dictionnaire d'équivalence proposé qui n'est pas utilisé et que les mots partiellement prononcés sont évalués comme des mots entiers.

Pour l'entraînement des modèles acoustiques et modèles linguistiques, j'utilise l'ensemble des données décrites dans 4.1. Pour rappel, le volume total de données pour le modèle acoustique est de 231 heures avant la phase d'augmentation de données et 700 heures après. Pour le modèle de langage, le corpus final utilisé contient environ 2 millions de phrases pour un total d'environ 46 millions de mots, ce qui reste toutefois relativement faible en considération des ressources utilisées par les participants de la campagne d'évaluation [Galliano et al., 2009].

## 6.5 Implémentations

Tous les systèmes utilisés ici partagent une optimisation équivalente – aucune technique de re-calcul des hypothèses ou de post-traitement n'est appliquée – ainsi qu'une utilisation équivalente des ressources pour l'entraînement. Chaque système est donné dans sa forme initiale, c'est-à-dire sans aucune phase d'entraînement supplémentaire en plus du système reporté).

### 6.5.1 Unités acoustiques

Trois types d'unités acoustiques ont été sélectionnés pour cette étude: phonèmes, caractères et sous-mots. Les systèmes basés phonèmes sont utilisés comme base de référence et utilisent les 36 phonèmes utilisés normalement pour la RAP en français. Les systèmes CTC, attention et bout-en-bout hybrides ont chacun deux versions : une version pour les caractères, avec 41 classes (26 lettres de l'alphabet latin, 14 lettres avec une diacritique et un signe de ponctuation, l'apostrophe) et une autre pour les sous-mots pour lequel le nombre de classes est fixé à 500, l'ensemble final de sous-mots utilisé pour l'entraînement étant obtenu en utilisant un algorithme de segmentation des sous-mots basé sur un modèle de langage unigramme [Kudo, 2018]. Celui-ci est disponible dans la boîte à outils SentencePiece de Google [Kudo and Richardson, 2018].

Pour la variante bout-en-bout du modèle de référence, les phonèmes et les caractères sont utilisés comme unités, formant ainsi deux systèmes distincts, avec le même ensemble précédemment décrit, c'est-à-dire constitué de respectivement 36 et 41 classes.

### 6.5.2 Systèmes de base

Pour l'entraînement du modèle de référence et de sa variante bout-en-bout, j'utilise la boite à outils Kaldi [Povey et al., 2011].

Le modèle de RAP traditionnel a une architecture de type TDNN-HMM et est entrainé avec la fonction objective LF-MMI. Le réseau de neurones est constitué d'un réseau de neurones à retardement (ou *time-delay neural networks*, abrégé TDNN) sous-échantillonné composé de 7 couches et 1024 unités dans chaque couche. La valeur du pas de temps associée aux TDNNs étant fixée à 1 dans les trois premières couches, 0 dans la quatrième et 3 dans les dernières.

La version bout-en-bout du modèle précédent est entraîné de la même manière, avec la fonction objective LF-MMI, mais avec l'architecture utilisée quelque peu différente. Le réseau est composé d'une couche LSTM projetée [Sak et al., 2014] ayant 512 unités suivie par 2 couches de type TDNN ayant 512 unités chacune - ces trois premières couches étant répétées deux fois - et d'une autre couche LSTM avec projection de 512 unités lorsque le caractère est utilisé comme unité. La valeur du délai dans les connexions récurrentes des couches LSTM projetées est fixée à 3.

En entrée des modèles, j'utilise des vecteurs de caractéristiques de type MFCC à "haute résolution" ayant 40 dimensions (c'est-à-dire transformée linéaire des banques de filtres) auxquels j'applique une normalisation de type CMV (Cepstral-Mean and Variance) pour le modèle standard entraîné avec lattice-free MMI et sa variante bout-en-bout. Pour le système traditionnel de réference basé phonèmes, une seconde version a également été entraînée avec, en entrée, le précédent vecteur MFCC à 40 dimensions concaténé ici avec un i-vecteur [Gupta and P. Kenny, 2014] à 100 dimensions comme entrée pour évaluer l'impact des caractéristiques dépendantes du locuteur.

Pour la partie linguistique, j'entraîne un modèle de langage basé mots de type 3-grammes en utilisant la méthode de comptage de n-grammes du SRILM [Stolcke, 2002] utilisant la méthode de lissage Kneser–Ney. Pour le lexique, je m'appuie sur le dictionnaire phonétique fourni par le LIUM, ainsi, le vocabulaire associé au modèle de langage est limité aux 50.000 mots les plus fréquents trouvés dans nos textes d'entraînement et également présents dans leur dictionnaire. Les autres termes étant remplacés par le symbole désignant le terme inconnu ⟨UNK⟩.

Pour la variante bout-en-bout du système traditionnel modélisant des caractères, le lexique phonétique est remplacé par un lexique orthographique avec les mêmes entrées, où la représentation orthographique correspondante est une séquence de caractères obtenue par insertion d'un espace entre chaque caractère constituant le mot représenté.

### 6.5.3 Systèmes bout-en-bout

J'utilise la boite à outils ESPnet [Watanabe et al., 2018a] pour l'entraînement des cinq systèmes bout-en-bout restants. La version de ESPnet utilisée ici correspond à la version de septembre 2019, incluant la première version du modèle RNN-Transducer. Pour chaque méthode, deux modèles sont construits modélisant chacun une unité orthographique : caractère et sous-mot. Dix *epochs* sont utilisées pour entraîner chaque modèle.

Pour toutes les méthodes citées, les modèles acoustiques entraînés partagent la même architecture composée d'un goulot d'étranglement de type VGG [Hori et al., 2017b] suivi d'un LSTM bi-directionnel à 3 couches avec 1024 unités dans chaque couche et chaque direction. Pour les modèles utilisant un mécanisme d'attention, j'utilise comme décodeur un LSTM à 1 couche avec 1024 unités couplé à un mécanisme basé localisation avec 10 filtres de convolution centrés de largeur 100 pour l'extraction des caractéristiques convolutionnelles. Lors de l'entraînement conjoint de la CTC avec le module d'attention, $\lambda$ a été fixé à $0,3$ sur la base d'expériences préliminaires. Pour le modèle RNN-Transducer, l'espace commun entre l'encodeur et le décodeur a été fixé à 1024 dimensions.

Les caractéristiques d'entrée pour ces modèles sont des vecteurs de type *filterbank* à 80 dimensions normalisés avec une normalisation de type CMN (pour *Cepstral-Mean Normalization* en anglais).

Pour les expériences impliquant l'utilisation de modèles de langage, j'entraîne trois modèles avec une architecture de type LSTM en me servant du module disponible dans ESPnet : un modèle basé caractères, un autre basé sous-mots et un dernier avec des mots complets pour une combinaison à plusieurs niveaux lorsque les caractères utilisés pour la modélisation acoustique. Chaque modèle est incorporé au moment de l'inférence en utilisant la fusion "peu profonde" [Kannan et al., 2018], sauf dans le cas de l'utilisation du LM basé mots qui repose sur le décodage multi-niveaux décrit dans [Hori et al., 2017a]. L'architecture principale des modèles de langage est un RNN à 1 couche, le nombre d'unités dans chaque couche dépendant de l'unité cible : 650 pour les sous-mots et les caractères, et 1024 pour les mots. Contrairement aux systèmes hybrides décrits précédemment, le vocabulaire associé au LM basé mots a été limité en fonction des seuls textes d'entraînement.

Afin de comparer directement les systèmes de référence aux systèmes bout-en-bout utilisant différents types de modélisation linguistique pour les mots (c.-à-d., N-gram et RNN), un autre modèle de langage a été entraîné à l'aide des outils mis à disposition à cet effet dans Kaldi. Le modèle partage la même architecture que le RNN-LM basé mots construit avec ESPnet et a été entraîné avec des paramètres équivalents. En suivant l'approche de re-calcul des treillis proposée dans [Xu et al., 2018], un décodage pour tous les systèmes de base a ensuite été effectué avec ce nouveau modèle de langage. Une amélioration maximale du taux d'erreurs de mots (TEM) a été dénotée sur le corpus de développement et de test par rapport aux systèmes reposant sur les modèles de type 3-grammes, respectivement : 0,12% et 0,16%. En ajoutant à cela une différence de couverture de moins de 1,3% entre les mots des vocabulaires respectifs des modèles de langue pour les systèmes de base et les systèmes bout-en-bout, nous pouvons considérer minime l'impact pour la comparaison effectuée.

### 6.5.4 Décodage

Pour mesurer les meilleures performances, la taille du faisceau a été fixée à 30 durant la procédure de décodage, et cela quelles que soient les conditions et le type de modèle. Lors du décodage avec le modèle basé attention seul, je n'utilise pas de paramètres de contrôle de la longueur de la séquence tels que le terme de couverture ou les paramètres de normalisation de la longueur [Wu et al., 2016]. Lors du décodage conjoint, le paramètre $\lambda$ est fixé à $0,2$ sur la base d'expériences préliminaires. Pour les expériences avec la CTC et l'attention impliquant un modèle de langage, le poids associé à la contribution du modèle de langage, lors de la fusion avec les probabilités du modèle acoustique, est fixé respectivement à $0,3$ pour le modèle basé caractères et sous-mots, et à $1,0$ pour le LM basé mots. Pour les systèmes utilisant la fonction de perte RNN-Transducer, durant la phase de décodage avec un modèle de langage basé mots, l'importance donnée à la modélisation linguistique a été réduite, la valeur étant fixée à $0,3$.

## 6.6 Résultats

Les résultats des expériences en termes de taux d'erreur sur les caractères (TEC) et de taux d'erreur sur les mots (TEM) sur l'ensemble de test sont rassemblés dans le tableau 6.1. Pour le TEC, les erreurs sous-jacentes suivantes sont aussi notifiées: caractères corrects, substitués, insérés et supprimés.

### 6.6.1 Systèmes de base

Le modèle de référence basé phonèmes entraîné avec le critère LF-MMI obtient un TEM de $14,2\%$ sur l'ensemble de test. L'ajout de caractéristiques de type i-vecteurs améliore encore les performances de notre modèle, conduisant à un TEM de $13,7\%$.

Par rapport au meilleur système rapporté durant la campagne ESTER (TEM $12,1\%$, LIMSI [Galliano et al., 2009]), les performances montrent une dégradation relative de $14,8\%$. Mon système reste toutefois compétitif comparé aux deux meilleurs systèmes suivants, à savoir : celui de VR (TEM 15.1%) et du LIUM (TEM: 17.8%). Bien que les systèmes comparés reposent sur l'utilisation d'architectures de type GMM-HMM, il convient de noter qu'un re-calcul à plusieurs passes (+ post-traitement) est appliqué, que les systèmes de référence ont un nombre conséquent de paramètres comparé à mes systèmes et qu'une quantité substantielle de données est utilisée pour l'entraînement des différents modèles linguistiques (jusqu'à plus de 11 fois le volume utilisé ici).

Pour la variante bout-en-bout basé phonèmes du système, une faible dégradation du TEM de l'ordre de $0,2\%$ est observée par rapport au système original (sans i-vecteurs), ce qui est un bon compromis si l'on considère la suppression de l'entraînement initial d'un système GMM-HMM. En utilisant des caractères comme unités acoustiques, un TEC de $7,6\%$ est obtenu, ce qui correspond à un TEM de $14,8\%$. Le rapport détaillé montre que le volume d'erreurs rencontrées par type est équilibré, avec toutefois un nombre plus élevé de suppressions. Le système reste compétitif même avec les unités orthographiques, malgré une faible correspondance entre la représentation phonétique et la représentation orthographique avérée pour le français. De même, une simple conversion du lexique phonétique en un lexique basé sur les graphèmes ne semble pas impacter négativement les performances. Cela est surprenant si l'on considère l'utilisation normalisée d'une représentation phonétique alternative en français pour indiquer, par exemple, les *liaisons* possibles (la prononciation

Table 6.1: Taux d'erreur sur les caractères (TEC) avec rapport détaillé et taux d'erreur sur les mots (TEM) pour toutes les méthodes évaluées sur le jeu de test d'ESTER2. Les valeurs en italique indiquent les erreurs sur les sous-mots. Les valeurs en gras indiquent les meilleurs résultats pour chaque section.

| Modèle | Unités | Lexique | LM | Corr. | Sub. | Del. | Ins. | TEC | TEM |
|---|---|---|---|---|---|---|---|---|---|
| chain LF-MMI | phone | 50K | word 3-gram | | | | | | 14.2 |
| chain LF-MMI *(i-vectors)* | | | | | | | | | **13.7** |
| e2e chain LF-MMI | phone | 50K | phone 4-gram + word 3-gram | | | | | | 14.4 |
| | char | | char 4-gram + word 3-gram | **94.3** | 2.6 | **2.0** | 3.0 | **7.6** | 14.8 |
| CTC | char | None | None | 87.4 | 4.9 | 7.7 | 3.0 | 15.5 | 42.3 |
| | | | char RNNLM | 89.5 | 4.4 | 6.1 | 2.8 | 13.3 | 31.0 |
| | | 50K | word RNNLM | 89.8 | 4.3 | 5.8 | 2.7 | 12.8 | 27.3 |
| | subword | None | None | *81.2* | *9.2* | *9.6* | ***1.4*** | *20.1* | 28.4 |
| | | | subword RNNLM | *85.7* | *9.1* | *6.1* | *2.3* | *17.5* | 21.2 |
| Attention *(location-based)* | char | None | None | 89.8 | 3.2 | 6.7 | 3.3 | 13.2 | 24.4 |
| | | | char RNNLM | 90.1 | 3.2 | 6.4 | 3.2 | 12.8 | 23.6 |
| | | 50K | word RNNLM | 90.2 | 3.3 | 6.2 | 3.2 | 12.7 | 23.0 |
| | subword | None | None | *84.1* | *12.3* | *3.6* | *3.6* | *19.5* | 22.7 |
| | | | subword RNNLM | *85.0* | *11.1* | *3.4* | *3.3* | *18.4* | 21.8 |
| RNN-Transducer | char | None | None | 93.9 | 2.8 | 3.3 | 2.4 | 8.5 | 19.7 |
| | | | char RNNLM | 94.0 | 2.6 | 3.4 | 2.2 | 8.2 | 18.8 |
| | | 50K | word RNNLM | 94.1 | 2.5 | 3.4 | **2.1** | **8.0** | 18.1 |
| | subword | None | None | *87.1* | *8.9* | *4.1* | *2.5* | *15.5* | 18.5 |
| | | | subword RNNLM | ***87.4*** | ***8.2*** | *4.3* | *2.2* | *14.7* | 17.4 |
| Joint CTC-Attention *+ joint decoding* | char | None | None | 91.7 | 2.9 | 5.4 | **2.1** | 10.4 | 22.1 |
| | | | char RNNLM | 92.2 | 2.9 | 5.0 | 2.2 | 10.1 | 20.6 |
| | | 50K | word RNNLM | 92.8 | 3.1 | 4.1 | 2.4 | 9.6 | 18.6 |
| | subword | None | None | *87.3* | *9.3* | *3.5* | *2.5* | *15.3* | 18.7 |
| | | | subword RNNLM | ***87.4*** | *8.8* | ***3.3*** | *2.4* | ***14.5*** | 17.8 |
| RNN-Transducer w/ att. *(location-based)* | char | None | None | 94.1 | 2.7 | **3.2** | 2.3 | 8.2 | 19.1 |
| | | | char RNNLM | 94.1 | 2.5 | 3.4 | **2.1** | 8.0 | 18.3 |
| | | 50K | word RNNLM | **94.3** | **2.4** | 3.3 | **2.1** | **7.8** | 17.6 |
| | subword | None | None | *87.1* | *9.0* | *4.1* | *2.5* | *15.6* | 18.4 |
| | | | subword RNNLM | *87.3* | *8.3* | *4.4* | *2.2* | *14.9* | 17.5 |

de la consonne finale d'un mot immédiatement avant le son de la voyelle suivante dans le mot précédent).

### 6.6.2 Systèmes bout-en-bout

Dans un souci de clarté, et du fait du volume non-négligeable de données présentées pour chaque système, cette sous-section est séparée en deux parties : la première traitant des modélisations acoustiques utilisant les caractères comme unités et la seconde traitant des modèles utilisant des sous-mots.

#### 6.6.2.1 Modèles basés caractères

Alors que, sans modèle de langage, le modèle basé sur l'attention surpasse comme prévu le modèle CTC, les performances des modèles basés RNN-Transducer dépassent mes estimations initiales, battant les modèles précédents en termes de TEC et TEM. Le modèle standard RNN-Transducer surpasse ces mêmes modèles couplés avec le modèle linguistique, quel que soit le niveau de connaissance linguistique inclus (c'est-à-dire caractères et mots). Le TEC obtenu avec ce modèle est de $8,5\%$ alors que le TEM est de $19,7\%$. Cela représente une diminution relative de près de 40% pour le TEC et de 17% pour le TEM par rapport au modèle basé sur l'attention couplé au modèle de langage basé mot,

qui est le deuxième meilleur système *classique* bout-en-bout rapporté. Par rapport à la variante bout-en-bout du système basé caractères du modèle, une faible différence de $0,9\%$ pour le TEC est observée, ce qui correspond à une augmentation relative de $4,9\%$. Bien que le TEC obtenu soit compétitif, les erreurs au niveau des mots semblent indiquer des difficultés à modéliser les limites entre les mots par rapport aux systèmes de référence.

En étendant la comparaison aux modèles hybrides, seul le RNN-Transducer avec mécanisme d'attention obtient des résultats similaires voire meilleurs que sa version standard. Bien que la procédure CTC-Attention soit utile pour corriger certaines limitations des approches individuelles, le système ne peut atteindre qu'un TEC de $10,4\%$ pour un TEM de $22,1\%$. Toutefois, en ajoutant le modèle de langage basé mots et en utilisant un décodage à plusieurs niveaux, le système peut obtenir un TEM se rapprochant des performances des systèmes RNN-Transducer présentés ($18,6\%$) malgré une différence significative en terme de TEC ($9,6\%$ contre $8.0\%$ pour les RNN-Transducer). Pour le RNN-Transducer hybride utilisant un module d'attention supplémentaire, l'ensemble des performances est amélioré par rapport à la version simple, atteignant un TEC de $8,2\%$ et un TEM de $19,1\%$ sans modèle de langage.

En ce qui concerne les meilleurs systèmes, il convient de noter que les performances du RNN-Transducer sont encore améliorées grâce à l'incorporation d'un modèle de langage, permettant d'obtenir un TEC de $8,0\%$, proche des performances de la variante bout-en-bout du système de référence basé caractères, et utilisant un LM basé mots, sur la même métrique ($7,6\%$ de TEC pour rappel). En terme de TEM, cela représente une amélioration relative de $8,5\%$ par rapport aux résultats précédents sans modèle de langage. Cela reste cependant encore loin des performance indiquées par le système de base et sa variante sur cette mesure, respectivement $13.7\%$ avec phonèmes et $14,8\%$ avec caractères. Pour le RNN-Transducer avec attention, les résultats obtenus sont améliorés un peu plus, avec un TEC de $7,8\%$ pour un TEM de $17,6\%$. C'est donc le système bout-en-bout basé caractères, outre la variante du système de référence, le plus performant durant cette étude.

Parallèlement, en se concentrant sur le rapport du TEC, plusieurs observations peuvent être faites pour compléter l'étude:

Les erreurs d'insertion sont plus faibles pour les modèles CTC que pour les systèmes basés sur l'attention, avec l'ajout de modèles linguistiques inclus. Les systèmes basés sur l'attention sont censés comporter un nombre plus élevé de suppressions ou d'insertions en fonction de la différence de longueur entre les séquences d'entrée et de sortie, il est cependant surprenant d'observer un nombre aussi élevé d'erreurs de suppression.

Suite à cette dernière observation, les erreurs de suppression faites par le modèle utilisant seulement un module d'attention ont été examinées. La raison principale est l'existence de paires de segments-énoncés irrégulières dans l'ensemble de données (c'est-à-dire ayant une correspondance faible). L'utilisation de termes de couverture, de pénalité ou de rapport de longueur a permis la réduction des erreurs faites pour les paires problématiques mais a dégradé les performances globales sur le corpus de test, les paires régulières courtes ou longues étant affectées.

L'ajout d'un modèle linguistique diminue l'ensemble des erreurs formulées par les systèmes de CTC alors que seules les erreurs de suppression diminuent pour le système basé attention seulement. Couplé à un modèle de langage basé mots, les erreurs de substitution sont encore plus élevées pour le modèle d'attention.

Des observations similaires peuvent être faites pour le modèle RNN-Transducer. Si une légère diminution des erreurs d'insertion est observée avec l'ajout d'un modèle de langage, je constate également une légère augmentation des erreurs de suppression. Cependant, la réduction du nombre d'insertions semble impacter positivement les performances du système, qui s'accompagne par une diminution du nombre de substitutions et une augmentation du nombre de mots correctement orthographiés.

Malgré des performances similaires en terme de TEC entre le modèle CTC avec le LM basé mot et le modèle attention seul et n'importe quel type de modélisation linguistique, le premier système ne semble pouvoir atteindre le TEM du second. Il apparaît plus avantageux de modéliser l'information linguistique en même temps que l'information acoustique plutôt que dans un modèle de langage externe basé caractères ou mots, bien que les deux niveaux puissent être combinés pour atteindre de meilleures performances. Cependant, il est important de considérer que les données d'entraînement du modèle acoustique sont les mêmes que les données utilisées pour entraîner le modèle de langage, augmentées d'un volume équivalent à moins d'un quart du nombre de phrases dans le corpus initial.

En comparant la variante bout-en-bout du système de référence modélisant des caractères au RNN-transducteur couplé avec un LM, plusieurs informations utiles peuvent être extraites. Les erreurs de suppression faites par le RNN-Transducer sont plus influentes au niveau des mots que les erreurs d'insertion faites par le système de base. En étudiant les hypothèses formulées (c.-à-d., les transcriptions) par le système de base, il a pu être observé que les erreurs d'insertion se produisent surtout sur des formes verbales ambiguës ou des formes à accorder en genre et en nombre non-connues. Pour le RNN-Transducer, le même comportement est observé mais les erreurs de suppression au niveau des caractères se produisent surtout sur des mots de petite taille (comme les articles) ainsi que les noms communs et les noms propres qui sont nombreux dans le corpus. Ces derniers comportant majoritairement des lettres ou formes muettes.

Bien qu'un plus petit nombre de substitutions au niveau des caractères soit observé par rapport au système de base pour le RNN-Transducer avec ou sans module d'attention, les erreurs de substitution impactent un plus grand nombre de mots en comparaison. Ces erreurs sont principalement dues aux problèmes décrits précédemment, tandis que les substitutions dans les systèmes de base sont plus localisées en raison notamment de la présence de mots hors vocabulaires et d'homophones.

Compte tenu des observations précédentes, il conviendrait de procéder à une étude plus approfondie afin de comparer et de classer les erreurs au niveau des caractères et des mots formulées par chaque système et d'évaluer également la valeur ou l'impact de ces erreurs. Les erreurs de caractères signalées pour le RNN-transducteur avec attention devraient être une motivation suffisante : un nombre inférieur d'erreurs de substitution et d'insertion couplées à un nombre équivalent de mots corrects est observé malgré un écart important de performance en terme de taux d'erreurs de mots.

#### 6.6.2.2 Modèles basés sous-mots

Le remplacement des caractères par des unités de type sous-mot améliore les performances pour l'ensemble des méthodes bout-en-bout. Ce gain est particulièrement important pour le modèle CTC qui voit une réduction du TEM de $42,3\%$ à $28,4\%$ sans utilisation de modèle de langage. Le gain observé lors de l'ajout du modèle linguistique pour la CTC est impressionnant avec une amélioration relative de près de 28% par rapport au TEM (de $28,4\%$ à $21,2\%$). Pour le système reposant uniquement sur l'utilisation d'un module

d'attention, le TEM est encore amélioré que cela soit avec ou sans l'ajout d'un modèle linguistique. Toutefois, contrairement à la version avec caractères, le modèle est surpassé par le modèle basé CTC et cela sur les deux métriques utilisées. Bien qu'un TEC proche entre les deux méthodes soit observé, je constate également une différence significative en termes de caractères corrects et de TEM (près de 6%). Le module d'attention fait surtout des erreurs consécutives sur les mêmes mots ou groupes de mots (en particulier au début et à la fin des énoncés) alors que la CTC tend à reconnaître une partie des mots comme étant séparés, identifiant incorrectement les limites de certains mots. Si le système RNN-Transducer est ajouté à la comparaison, les deux méthodes précédentes sont dépassées, aussi bien en terme de TEC ($20,1\%$ pour la CTC, $17,5\%$ pour l'attention et $15,2\%$ pour RNN-Transducer) qu'en terme de TEM ($21,1\%$ pour la CTC, $21,8\%$ pour l'attention et $18,4\%$ pour le RNN-Transducer). En ajoutant un modèle de langage externe, les performances en terme de TEC et de TEM sont encore améliorés, avec une diminution relative de respectivement $5,5\%$ et $6,0\%$. Il convient aussi de noter que le modèle RNN-Transducer sans modèle de langage obtient de meilleurs résultats que la CTC et l'attention utilisant un modèle de langage basé sous-mot.

En incorporant les systèmes hybrides dans notre comparaison, on peut noter quelques différences par rapport aux systèmes basés sur les caractères. Le système basé RNN-Transducer n'est pas amélioré par le mécanisme d'attention et est même légèrement dégradé en terme de TEC et de TEM. Les mêmes observations peuvent être faites avec et sans ajout du modèle de langage basé sous-mots. Il semble que le mécanisme d'attention ait plus de difficultés à modéliser les relations intra-sous-mots que les relations intra-caractères. Cependant, des travaux supplémentaires doivent être effectués pour étendre la comparaison avec différents mécanismes d'attention, tels que l'attention à plusieurs têtes ajoutant des informations de position, et aussi, estimer l'influence de l'architecture en fonction des dimensions et des représentations de sortie.

En ce qui concerne le dernier système hybride, le système utilisant conjointement CTC et attention semble plus adapté à l'utilisation de sous-mots que de caractères, atteignant des performances comparables aux systèmes RNN-Transducer sans modèle de langue externe : 18.7% contre 18.5% pour RNN-transducteur et 18.4% pour RNN-Transducer avec attention. Bien que ce dernier soit considéré comme le meilleur système, il faut noter que le système CTC-attention atteint des performances égales ou supérieures en terme d'erreur sur les sous-mots. En ne considérant que les mesures conventionnelles pour la reconnaissance automatique de parole, les deux systèmes hybrides et le modèle RNN-Transducer standard sont équivalents, pour cette tâche, en terme de performance en utilisant des unités de type sous-mot.

Comme dans la section précédente, je rapporte en détails les types d'erreurs rencontrés pour chaque modèle. En l'occurrence, après observation approfondie, je dénote quelques différences par rapport aux observations précédentes pour les types d'erreurs faites sur les caractères:

Comme pour les observations précédentes avec les caractères, les erreurs d'insertion sont plus faibles pour les modèles basés CTC ($1,4\%$) que pour les modèles basés sur l'attention ($3,6\%$) avec des sous-mots. Cependant, ici, le nombre d'insertions pour la CTC est encore plus faible que pour toutes les autres méthodes, les systèmes de type RNN-Transducer et hybrides présentant un taux d'erreur d'insertion moyen de $2,5\%$.

Il a été noté précédemment qu'il fallait s'attendre à un nombre plus élevé de suppressions ou d'insertions avec le modèle utilisant le mécanisme d'attention seul. Avec les unités de

sous-mots, on observe cependant un nombre équilibré de suppressions et d'insertions bien qu'un nombre important de substitutions soit aussi dénoté. Suite à cette nouvelle observation, j'ai également étudié les productions orthographiques des deux modèles avec les différentes unités modélisées (caractères et sous-mots). Je note que la principale limitation du modèle basé attention est en partie levée par l'utilisation de sous-mots, les séquences de sous-mots étant déroulées ou arrêtées correctement. Cependant, cela se traduit aussi par un très grand nombre de substitutions, certains sous-mots composant les mots à plus haut niveau étant répétés ou coupés.

Bien que je rapporte un nombre plus élevé de mots corrects et un nombre plus faible d'erreurs pour le système utilisant CTC et attention de manière conjointe, cette méthode hybride obtient un TEM plus élevé que le RNN-Transducer et sa version hybride. En analysant les hypothèses formulées et la distribution des erreurs faites par les deux systèmes, je n'ai cependant pas pu extraire d'informations pertinentes expliquant le nombre de mots affectés par les erreurs au niveau des caractères.

Sur ce même point, il convient de noter la différence suivante : les modèles RNN-Transducer ont un nombre inférieur de substitutions et un nombre d'insertions équivalent ou inférieur alors que les modèles basés CTC-attention ont un nombre inférieur de suppressions et un nombre équivalent ou supérieur de caractères corrects. En dehors des étiquettes correctes, seule la CTC présente une distribution d'erreurs similaire.

Dans le cas du modèle Joint CTC-Attention, on peut constater que la CTC en tant que fonction auxiliaire apporte certains avantages : le nombre de substitutions et d'insertions étant encore réduit par rapport au modèle utilisant le mécanisme d'attention seul. En outre, le nombre de suppressions est maintenu dans la même fourchette alors qu'un nombre élevé de suppressions est observé pour le modèle utilisant seulement la CTC. Dans le cas de l'utilisation d'un module d'attention supplémentaire pour le système RNN-Transducer, bien que le modèle basé attention présente un nombre inférieur d'erreurs de suppression ($3,6$ contre $4,1$ pour le RNN-Transducer standard), l'inclusion du mécanisme ne contribue pas à réduire le nombre total d'erreurs dans cette catégorie. La distribution des erreurs est la même avec et sans mécanisme d'attention. Il convient également de noter que RNN-Transducer avec attention a une performance équivalente que cela soit avec des caractères ou des sous-mots.

En ajoutant des modélisations linguistiques, le nombre d'erreurs pour chaque catégorie est réduit. Les seules exceptions étant: le nombre d'insertions pour la CTC (de $1,4\%$ à $2,3$), le nombre de suppressions pour RNN-transducteur (de $4,1\%$ à $4,3$) et son homologue hybride (de $4,1$ à $4,4$). Dans ces cas, comme dans le cas de l'utilisation de caractères, il peut être observé que le taux d'erreur d'une catégorie (par exemple : insertion) diminue lorsque l'autre (par exemple : suppression) augmente.

## 6.7 Conclusion

Dans ce chapitre, j'ai pu montrer que les approches bout-en-bout ainsi que l'utilisation d'unités orthographiques semblent adaptés pour la tâche de RAP et la langue française. Avec les caractères, le modèle RNN-transducteur s'est avéré particulièrement compétitif par rapport aux autres approches bout-en-bout. Parmi les deux unités orthographiques, l'utilisation de sous-mots s'est avéré bénéfique pour la plupart des méthodes afin de résoudre les problèmes décrits dans la section 6.6.2 et afin de retenir les informations sur les motifs ambigus en français. En étendant le système avec des modélisations linguis-

tiques externes, des résultats prometteurs ont pu être obtenus par rapport aux systèmes traditionnels basés phonèmes. Concernant le système les plus performants avec les unités de type caractères, celui ci est le RNN-transducer avec module d'attention, atteignant $7,8\%$ en termes de TEC et $17,6\%$ sur le TEM. Pour les systèmes modélisant des sous-mots, les systèmes RNN-transducer classique, RNN-transducer avec attention et Joint CTC-Attention présentent des performances comparables en termes de taux d'erreur sur les sous-mots et de TEM, le premier étant légèrement meilleur en terme de TEM ($17,4\%$) et le dernier ayant un taux d'erreur plus faible sur les sous-mots ($14,5\%$). Cependant, des différences notables concernant le type d'erreurs produites pour chaque méthode a été observé, ayant un impact différent au niveau du mot selon l'approche ou les unités utilisées. A cela, les travaux suivants se concentrent sur l'analyse de la sortie orthographique de ces systèmes de deux manières : 1) étudier les erreurs produites par les méthodes bout-en-bout et explorer plusieurs approches pour corriger les erreurs courantes faites en français et 2) comparer les méthodes bout-en-bout dans un contexte de compréhension du langage parlé et évaluer la valeur sémantique des mots produits *partiellement* corrects.

# Chapter 7

# Erreurs en RAP et conceptualisation des erreurs pour la compréhension du langage parlé

Dans ce chapitre, je décris les travaux conduits entre novembre 2019 et janvier 2020 qui se concentrent sur (i) l'identification des erreurs produites par les systèmes de RAP bout-en-bout présentés dans le chapitre précédent et sur (ii) les perspectives de travaux répondant à ces erreurs. Le deuxième point sera particulièrement abordé en considérant une analyse sémantique des mots erronés produits. Ce chapitre s'appuie sur les travaux décris dans le chapitre précédent 6 et a donné lieu a un article soumis à ICASSP 2020 qui a malheureusement été rejeté malgré des examinateurs jugeant celui-ci acceptable (*weak acceptance*). Une version mise à jour du papier est en préparation et sera finalisée à la suite de l'écriture de ce manuscrit.

## 7.1 Introduction

De nos jours, les approches neuronales bout-en-bout pour les tâches telles que la Reconnaissance Automatique de Parole (RAP) [Bahdanau et al., 2016, Amodei et al., 2016] ou la Compréhension du Langage Parlé (CLP) [Serdyuk et al., 2018, Tomashenko et al., 2019] ont suscité un intérêt croissant, que ce soit pour la communauté des chercheurs mais également pour les développeurs qui y ont vu la possibilité de mettre au point des systèmes complets sans connaissances expertes en technologies vocales. Pour la tâche de compréhension, l'utilisation d'une chaîne de traitement bout-en-bout est particulièrement intéressante pour traiter deux problèmes : En premier lieu, la maintenance de la chaîne de traitement est facilitée puisque les deux parties, RAP et Compréhension du Langage Naturel (CLN), peuvent être optimisées conjointement. L'interprétation sémantique sera également impactée puisque le sens peut être directement inféré des paramètres acoustiques sans la nécessité d'une représentation textuelle intermédiaire. Même si la conception de tels systèmes parait prometteuse, ils sont toutefois encore peu étudiés [Tomashenko et al., 2019, Serdyuk et al., 2018]. Ainsi, la majorité des chercheurs n'étudie que la chaîne de traitement traditionnelle, où les systèmes de RAP et de CLN sont optimisés séparément.

Lorsque l'on s'intéresse aux systèmes de CLP conventionnels, on peut également noter que peu de travaux considèrent l'impact des erreurs de transcription lors de la conception du

module de compréhension [Simonnet et al., 2017, Simonnet et al., 2018]. Avec l'émergence des système de RAP bout-en-bout intégrés dans les systèmes de CLP pouvant permettre la création "à la volée" de représentations orthographiques sans vocabulaire connu, il y a désormais un besoin grandissant pour l'évaluation de la production d'un système de RAP. Les mots produits, même incorrects, peuvent toutefois contenir des informations sémantiques importantes. Parallèlement, les métriques conventionnelles telles que le taux d'erreur de caractères (TEC) et le taux d'erreur de mots (TEM), utilisés pour estimer les performances des systèmes de RAP, peuvent s'avérer insuffisantes afin d'estimer les performances du dit système dans un contexte de système de CLP complet.

C'est dans ce contexte que je propose d'étendre mon étude des méthodes de RAP bout-en-bout pour la langue française à l'analyse de la production orthographique et des erreurs formulées par ces méthodes. Cette étude est intéressante du fait de la correspondance faible entre les représentations orthographiques et phonétiques en Français, dûe notamment à la présence de nombreux homophones verbaux, et non verbaux, la présence de lettres ou groupes de lettres muettes mais aussi l'emploi occasionnel d'argots ou anglicismes de nos jours. Ceci est d'autant plus intéressant dans le cadre de la RAP bout-en-bout peuvent prédire des unités acoustiques, ici orthographiques, de manière indépendante ou dépendant du contexte environnant.

Dans la section suivante, j'introduis les systèmes de RAP bout-en-bout utilisés dans cette étude ainsi que les résultats obtenus avec les métriques conventionnelles. Ensuite, je présente les différentes catégories d'erreurs ainsi que les analyses pour chacune des méthodes. Pour finir, je discute dans la dernière partie l'impact que ces erreurs peuvent avoir pour l'interprétation sémantique par le biais d'une expérience préliminaire que nous proposons ici.

## 7.2 Systèmes bout-en-bout utilisés

Les travaux présentés dans ce chapitre étant dans la continuité des travaux du chapitre 6, les mêmes approches bout-en-bout sont utilisées ici, à savoir : 1) La CTC (Sec. 3.2.1), 2) les méthodes basés attention (Sec. 3.2.3) et 3) les modèles Transducer (Sec. 3.2.2). S'ajoutent à cela, les deux approches hybrides bout-en-bout aussi utilisées dans le chapitre précédent: l'approche multi-tâche appelée "Joint CTC-Attention" (abrégé "JCA") (Sec. 3.3.1) et 2) l'extension du modèle Transducer avec un mécanisme d'attention (Sec. 6.3.1.1).

Pour rappel, chaque approche est entraînée pour modéliser deux unités orthographiques : caractères ou sous-mots (plus communément appelés en anglais *byte-pair units*). Tous ces systèmes partagent des optimisations équivalentes – pas de technique de re-score ni de traitement *a posteriori* – ainsi que des ressources similaires. Notons également que contrairement à nos travaux précédents, nous ne considérons ici que les systèmes RAP seuls, excluant ainsi les modèles de langage.

Nous référons le lecteur au chapitre précédent (6) pour la description des systèmes utilisés et des unités acoustiques considérées. Concernant le corpus de test, nous utilisons le corpus d'évaluation proposé durant la campagne d'évaluation d'ESTER qui est décrit dans la section 4.1. Les modalités d'évaluation tel que la normalisation des hypothèses et l'utilisation de ressources tierces sont aussi données dans le chapitre précédent.

## 7.3 Résultats

Les résultats des expériences en termes de taux d'erreur de caractères (TEC) ou taux d'erreur de sous-mots (TES), et de taux d'erreur de mots (TEM) obtenus sur l'ensemble de test d'ESTER sont rassemblés dans le tableau 7.1, sous-tableau de 6.1 de la section 6.6. Pour le taux d'erreur de caractères, j'indique également le pourcentage pour les sous-catégories suivantes : caractères correctement reconnus, caractères substitués, caractères additionnels insérés et caractères supprimés.

Table 7.1: TEC, avec le détail des erreurs, et TEM pour l'ensemble des méthodes évaluées sur l'ensemble de test d'ESTER2. Les valeurs en italique dénotent d'une évaluation sur les sous-mots. Les valeurs en gras indiquent les meilleurs résultats obtenus pour chaque section.

| **Model** | **Unit** | **Cor.** | **Sub.** | **Del.** | **Ins.** | **TEC** | **TEM** |
|---|---|---|---|---|---|---|---|
| CTC | char | 87.4 | 4.9 | 7.7 | 3.0 | 15.5 | 42.3 |
| | subword | *81.2* | *9.2* | *9.6* | ***1.4*** | *20.1* | 28.4 |
| Att. | char | 89.8 | 3.2 | 6.7 | 3.3 | 13.2 | 24.4 |
| | subword | *84.1* | *12.3* | *3.6* | *3.6* | *19.5* | 22.7 |
| RNN-T | char | 93.9 | 2.8 | 3.3 | 2.4 | 8.5 | 19.7 |
| | subword | *87.1* | ***8.9*** | *4.1* | *2.5* | *15.5* | *18.5* |
| JCA | char | 91.7 | 2.9 | 5.4 | **2.1** | 10.4 | 22.1 |
| | subword | ***87.3*** | *9.3* | ***3.5*** | *2.5* | ***15.3*** | 18.7 |
| RNN-T-Att. | char | **94.1** | **2.7** | **3.2** | 2.3 | **8.2** | **19.1** |
| | subword | *87.1* | *9.0* | *4.1* | *2.5* | *15.6* | **18.4** |

Pour l'utilisation d'unités de type caractère, les systèmes Transducer utilisant des architectures RNN et leur version hybride obtiennent les meilleures performances que ce soit pour le TEC ou le TEM. Le système CTC obtient des performances décevantes sans l'utilisation d'un modèle de langage. Lorsque des unités de type sous-mots sont utilisées, une amélioration significative des résultats en terme de TEM pour l'ensemble des méthodes est observée, particulièrement marquée pour le système modélisant conjointement CTC et attention, dont les performances atteignent celles des modèles Transducer. Notons toutefois que même si les performances du système basé CTC sont améliorées ici, elles restent toutefois inférieures en comparaison des autres systèmes modélisant des caractères ou des sous-mots.

## 7.4 Analyse des erreurs

Dans cette section je présente l'ensemble des analyses d'erreurs effectuées. Celles-ci sont regroupées dans trois thématiques portant sur : les accents, la similarité entre les mots à différents niveaux et l'analyse au regard de la compréhension du langage naturel. Au travers de ces expériences, j'espère remettre en perspective la notion d'erreur et les limites d'une évaluation purement syntaxique et orthographique. En outre, j'espère mettre en évidence les avantages des systèmes bout-en-bout en comparaison des systèmes hybrides traditionnels modélisant des unités phonémiques pour des langues comme le Français.

### 7.4.1 Accents

Contrairement à l'anglais, la langue Française emploie de nombreux signes diacritiques pour dénoter différentes prononciations ou pour distinguer des homonymes. En tout, cinq signes sont couramment utilisés en Français (c.-à-d., accent aigu, accent grave, accent circonflexe, tréma et cédille) résultant en 14 caractères accentués.

Des conventions d'écriture sont mises en place pour la bonne utilisation des accents en considération du caractère. Cependant pour le cas des mots inconnus ou des homonymes possédant plusieurs orthographes (p. ex., Thibaut et Thibault), la transcription peut être complexe, même pour un humain. Afin d'évaluer les performances de chacune des méthodes pour ce cas précis, nous proposons une détection et un classement des termes accentués mal reconnus. Ces résultats sont répertoriés dans le tableau 7.2.

Table 7.2: Nombre de mots uniques et total d'occurrences pour les erreurs sur les diacritiques.

| **Units** | | **CTC** | **Att.** | **RNN-T** | **JCA** | **RNN-T-Att.** |
|---|---|---|---|---|---|---|
| char | unique | 113 | 54 | 58 | 52 | 50 |
| | total | 266 | 224 | 208 | 197 | 199 |
| subword | unique | 56 | 55 | 55 | 62 | 62 |
| | total | 179 | 215 | 182 | 225 | 210 |

Pour l'ensemble des systèmes, le nombre de mots uniques entraînant la totalité des erreurs est en dessous de soixante-deux, avec toutefois l'exception notable d'un système (CTC modélisant des caractères) produisant le double de mots uniques ayant des erreurs d'accent. Les systèmes bout-en-bout hybrides génèrent le plus faible nombre d'erreurs sauf lorsque l'unité est les sous-mots. L'ensemble de sous-mots utilisé est probablement trop petit pour inclure certaines utilisations rares d'accentuation.

De cet ensemble, j'extrais également les cinq erreurs les plus fréquentes pour une évaluation complémentaire. Ici, j'observe que les mots accentués les plus couramment transcris incorrectement sont toujours (en nombre décroissant d'erreurs) : 'à', 'égypte', 'où', 'éric' et 'ça'. Parmi ces cinq mots, trois se trouvent être des homophones (p. ex., 'à/a', 'où/ou', 'ça/sa') tandis que deux sont des noms propres (p. ex., égypte, éric). Ces cinq mots sont cependant responsables de plus de 50% du nombre total d'erreurs d'accentuation, quelque soit le système utilisé.

Comme il ne semble pas y avoir un comportement particulier d'un système par rapport aux autres vis à vis de ces erreurs, je fais l'hypothèse qu'elles sont principalement dues à la nature homophonique de ces termes et le faible nombre d'exemple ne permettant pas la désambiguïsation lexicale. Dans le cas des noms propres, il s'est avéré, après vérification, qu'un grand nombre de mots ont leur première lettre capitalisé sans accent dans les transcriptions de référence des corpus d'entraînement (p. ex., Egypte) provoquant des ambiguïtés lors de l'évaluation sur le corpus de test.

### 7.4.2 Couverture des erreurs

Comme il a été montré précédemment que des différences significatives existent entre les taux d'erreur de caractères des différents systèmes que nous étudions, j'effectue une étude supplémentaire sur la couverture des erreurs sur la base du calcul de la fertilité des erreurs [Béchet and Favre, 2014]. Ce calcul permet de représenter le nombre d'erreurs (ou mots erronés dans l'hypothèse formulée) résultant d'un même mot dans transcription de référence (en excluant les erreurs de suppression). Un mot de référence séparé en deux parties dans l'hypothèse formulée par le système pouvant par exemple engendrer une erreur de substitution suivie d'une erreur d'insertion.

Tous les systèmes étudiés ont obtenus une fertilité d'erreur comprise en 1.09 et 1.20. Les systèmes possédant les valeurs les plus élevées sont ceux reposant uniquement sur le mécanisme d'attention, et cela, quel qu'en soit l'unité orthographique utilisée (caractère ou sous-mot).

Parallèlement, je calcule également la couverture des erreurs (donnée en terme de pourcentage) représentant le nombre d'erreurs de même nature au sein d'une même phrase. Ainsi, je peux observer que dans le cas du système basé Attention seul, la plupart des erreurs de suppression sont dues à des hypothèses partiellement formulées voire non formulées par le décodeur. En effet, environ 150 phrases ont plus de 70% des mots les constituant supprimés. Ce comportement peut être attendu [Chorowski et al., 2015, Bahdanau et al., 2016] et corrigés en utilisant des termes de couverture, voire des termes de pénalité ou l'utilisation d'un ratio en considération de longueur. Cependant, bien que ces paramètres de contrôle permettent de corriger certaines phrases problématiques, les performances globales sont toutefois dégradées par la génération de nouvelles erreurs. J'ai également pu observer que plus de 1500 phrases, ayant une moyenne de 14.4 mots, ont vu au moins 10% de leurs mots supprimés. Dans ce cas, la nature ou la fonction des mots impactés est diverse, avec toutefois une proportion non-négligeable de mots ou sous-mots monosyllabiques (27%). En ce qui concerne les erreurs de substitution, quelle que soit la couverture, les systèmes ayant les moins bonnes performances sont toujours ceux s'appuyant sur la fonction de perte CTC.

### 7.4.3 Similarité

Pour le système CTC utilisant les caractères comme unités, ainsi que pour tous les autres modèles employant les sous-mots, on peut observer un nombre significatif de substitutions comparé aux autres types d'erreur. Cela m'a conduit à évaluer la similarité entre les mots erronés produits par les différents systèmes et les mots correspondants dans les transcriptions de référence. Afin de mesurer la similarité, je propose de procéder de la manière suivante. Tout d'abord, je limite l'analyse aux mots des transcriptions générées et qui sont associés le plus souvent à des erreurs de substitution et à des erreurs composées (c'est-à-dire des substitutions associées à une ou plusieurs insertions ou suppressions), en excluant toutefois les mots déjà traités dans l'analyse des diacritiques (7.4.1). En outre, je ne considère que les mots ayant généré au moins trois erreurs. En appliquant la Distance Orthographique de Levenshtein [Vergara-Martinez and Swaab, 2012], et étant donné un ensemble références-hypothèses, j'extrais les hypothèses appartenant au voisinage orthographique des mots références correspondants. Précisement, je définis qu'une hypothèse est dans le voisinage de référence lorsque la distance par paire est inférieure à 3. Par exemple, si le mot de référence est "ces", les hypothèses "ses" ou "c'est" sont considérées proches, contrairement à "cette" ou "c'était". À partir de là, j'assigne une étiquette à toutes les paires en utilisant le POS-tagger de Stanford [Toutanova et al., 2003] préalablement entraîné avec le corpus arboré du Français [Abeillé and Barrier, 2004]. Chaque mot des transcriptions de références et d'hypothèses sont ainsi étiquetés en prenant en compte le contexte de la phrase. Pour les mots des transcriptions de référence n'ayant pas été annotés, une seconde passe manuelle est effectuée par des linguistes. Pour les hypothèses générées par les systèmes de RAP, l'étiquette ⟨UNK⟩ est utilisée dans ce cas. Finalement, grâce à un dictionnaire "maison", nous ajoutons également pour les mots étiquetés comme verbe la forme infinitive du verbe, ainsi que la racine du mot pour les mots étiquetés comme nom commun et nom propre.

Le tableau 7.3 rassemble les résultats pour les 4 classes de mots les plus fréquentes en terme de nombre d'erreurs, et ce pour l'ensemble de nos systèmes. Pour des raisons de clarté, nous avons décidé d'exclure la classe "multiple" englobant différents contextes dans les transcriptions de référence. Les termes "art", "nc", "np", "adj" and "prep" désignent respectivement les classes suivantes : article, nom commun, nom propre, adjectif

et préposition.

Table 7.3: Top 4 des classes POS-taggées, classés par nombre d'occurences.

| **Model** | **Unit** | **Ranking** |
|---|---|---|
| CTC | char | art (262), adj (242), nc (213), prep (129) |
| | subword | nc (370), adj (225), art (143), verb (99) |
| Att. | char | art (193), adj (128), nc (120), np (86) |
| | subword | art (173), adj (152), nc (101), verb (76) |
| RNN-T | char | art (182), nc (182), adj (141), np (100) |
| | subword | nc (153), art (146), adj (114), verb (81) |
| JCA | char | art (191), adj (149), nc (143), np (104) |
| | subword | art (169), nc (141), adj (134), verb (81) |
| RNN-T-Att | char | art (178), nc (160), adj (136), np (82) |
| | subword | art (150), adj (140), nc (118), verb (97) |

A l'exception du système CTC utilisant les unités de sous-mots, la classe "article" est celle qui contient le plus grand nombre d'erreurs, et ce quelque soit la méthode utilisée. Dans cette classe, les paires référence/hypothèse les plus fréquentes sont : "le-les" "l'-le/la", "des-de" or "au-aux". Tandis que le dernier exemple est un homophone, les erreurs pour les autres cas sont principalement dues à la prononciation ou une altération causée par une liaison. En ce qui concerne les autres classes de mots générant le plus d'erreurs, les classes "adjectif" et "noms communs" terminent en deuxième et troisième place tandis que les classes "noms propres" et "verbes" terminent la liste. Nous attirons par ailleurs l'attention du lecteur sur le fait qu'en dehors de la classe "article", les autres classes contiennent de nombreux homophones.

Pour compléter ces analyses, et dans le but d'évaluer également les différences entre les termes de référence et les termes produits au niveau acoustique, une autre étiquette a été attribuée à chaque paire afin de désigner le type d'erreur selon l'ensemble suivant : homophone, bordure, variante et prononciation. Nous définissons comme "bordure" les mots d'hypothèse ayant une représentation orthographique ou phonétique incluse dans la référence et un reste dans les mots environnants, comme "variante" les mots ayant une forme alternative mais autorisé (ex: "VIII" et "huit"), et enfin comme "prononciation" les erreurs dues au locuteur. Pour l'étiquetage, nous avons utilisé une combinaison constituée d'un lexique phonétique fourni par le LIUM, et utilisé dans notre système de RAP traditionnel basé phonèmes, le dictionnaire d'homophones Dichto[1] ainsi qu'une vérification manuelle par écoute. Les balises les plus couramment attribuées pour les catégories homophones et bordures sont reportées dans le tableau 7.4.

Table 7.4: Pourcentage d'erreur sur les homophones et bordures de mots.

| **Cause** | **Units** | **CTC** | **Att.** | **RNN-T** | **JCA** | **RNN-T-Att.** |
|---|---|---|---|---|---|---|
| homophn. | char | 32.3 | 49.9 | 54.5 | 48.1 | 52.3 |
| | subword | 49.2 | 54.1 | 52.7 | 50.9 | 53.5 |
| boundary | char | 16.2 | 16.5 | 13.7 | 16.4 | 15.3 |
| | subword | 12.8 | 12.9 | 12.2 | 12.8 | 14.1 |

Cette analyse permet de confirmer qu'un grand nombre d'erreurs est bien dû à l'homophonie. Dans ce cadre, j'ai pu observer que dans le cas des classes produisant le plus d'erreur, la plupart des différences se situent au niveau des marques d'accord en genre (p. ex., "assigné/assignée") et en nombre (p. ex., "mangeait/mangeaient") alors que la racine du mot

[1] https://sites.google.com/site/ledictho/les-homonymes

ou la forme de l'infinitif (dans le cas du verbe) est la même pour la référence et l'hypothèse. En outre, la plupart de ces erreurs se retrouvent sur des caractères muets. Cela nous a conduit à approfondir notre étude sur les formes et fonctions attribuées par l'étiquetage POS afin de vérifier l'impact de ces erreurs dans le contexte de la compréhension d'une phrase complète.

### 7.4.4 Compréhension

Une des perspectives de ce travail étant l'évaluation de l'impact de la qualité des transcriptions automatiques sur les systèmes de compréhension, je propose en complément une expérience préliminaire, similaire dans sa philosophie à celle menée par Simonnet et al. [Simonnet et al., 2017]. Celle-ci utilise la métrique du taux d'erreur de concept-valeur (Concept-Value Error Rate ou CVER). Cette métrique, principalement utilisée pour l'évaluation des systèmes de dialogue, a pour objectif de mesurer conjointement le taux d'erreurs d'un point de vue orthographique et le taux d'erreurs du point de vue de leur fonction "haut-niveau" (p. ex., le concept "Ville" et les valeurs "Paris", "Bordeaux" et "Toulouse").

Tout d'abord, à partir de l'ensemble de test d'ESTER2, j'extrais environ 1700 phrases ayant des thématiques liées et j'associe à chaque mot d'une phrase un concept. L'annotation a été faite manuellement par un groupe de 11 linguistes afin de valider l'expertise. Du fait qu'un nombre important de thèmes et de mots existent dans l'ensemble de données, seuls les concepts de haut niveau ont été annotés selon la liste suivante : sujet, objet, temps, cible, mesure, nombre, lieu, négation, modalité et conduite (comme les marques de politesse). Chaque concept se voit attribuer comme valeur le mot de référence associé. Pour les mots composés, les concepts sont segmentés en fonction du nombre de valeurs/mots.

Je retire de cet ensemble les énoncés ne contenant aucune erreur de mot et jugés non-pertinents pour une étude sur les erreurs, ainsi que les énoncés ayant une faible correspondance mot-concept (c'est-à-dire moins d'un concept non nul pour quatre mots) du fait de la segmentation initiale dans ESTER. En parallèle, je demande à chaque linguiste d'évaluer indépendamment chaque phrase (avec uniquement un concept ou une valeur) afin de vérifier qu'elles puissent toutes être compréhensibles lorsque qu'elles ne contiennent que les mots de contenu. Durant cette seconde partie, pour des raisons évidentes de biais, chaque linguiste effectue l'évaluation sur un ensemble de phrases qu'il n'a pas annoté durant la première partie.

En définitive, notre corpus est constitué de 831 énoncés et 4663 concepts-valeurs annotés (à l'exclusion des mots marqués comme *null*), pour 2390 paires uniques. J'indique le taux d'erreur de mots pour chaque système sur ce sous-ensemble dans le tableau 7.5.

Table 7.5: TEM obtenu sur le sous-set du corpus de test

| **Units** | **CTC** | **Att.** | **RNN-T** | **JCA** | **RNN-T-Att.** |
|---|---|---|---|---|---|
| char | 34.1 | 17.8 | 15.8 | 16.9 | 15.3 |
| subword | 23.5 | 18.2 | 14.7 | 14.7 | 14.8 |

Ensuite, j'applique une évaluation en deux passes avec la métrique CVER sur le sous-ensemble final, tel que : le concept est qualifié de correct si la valeur correspond à l'hypothèse dès la première passe, il est également qualifié de correct si la valeur ou un mot similaire correspond à l'hypothèse lors de la seconde passe. Dans le cas de mots similaires, les mots du voisinage tels qu'ils sont définis dans la section précédente sont utilisés pour la comparaison avec la valeur de référence.

Les résultats pour chaque passe sont rassemblés dans tableau 7.6.

Table 7.6: CVER en deux étapes sur le corpus de test.

| Pass | Units | CTC | Att. | RNN-T | JCA | RNN-T-Att. |
|---|---|---|---|---|---|---|
| 1 | char | 39.8 | 22.6 | 22.2 | 22.3 | 22.0 |
| | subword | 32.1 | 24.1 | 21.4 | 20.5 | 21.0 |
| 2 | char | 39.0 | 22.5 | 21.8 | 21.9 | 21.7 |
| | subword | 31.3 | 23.8 | 21.0 | 20.3 | 20.7 |

On peut observer que la hiérarchie définie sur la base du TEM est aussi respectée sur la métrique CVER, à l'exception notable du système utilisant conjointement la CTC et l'attention : alors que le meilleur système utilisant le sous-mot comme unité de base est le même pour les deux métriques (à savoir RNN-T avec attention), le système "JCA" est le plus performant selon la métrique CVER. Lorsque l'unité est le caractère, malgré une différence significative (+1%) en termes de TEM entre ces deux systèmes, ils obtiennent des performances comparables en termes de CVER.

En outre, la deuxième passe effectuée sur l'évaluation du CVER est toujours bénéfique, même si l'amélioration est relativement faible. Compte tenu de la petite taille du sous-corpus utilisé pour cette expérience, il s'agit toutefois d'un résultat encourageant qui montre que la prise en compte d'un nombre même faible de paires similaires uniques (650) permet de corriger des erreurs impactant la compréhension de l'intitulé.

## 7.5 Conclusion

Outre des difficultés à modéliser certains aspects spécifiques du français mis en évidence pour plusieurs méthodes, les systèmes de RAP bout-en-bout ne semblent pas parvenir à résoudre le problème principal déjà rencontré par les systèmes phonétiques traditionnels et qui est : comment modéliser les nombreuses variantes orthographiques possibles pour un mot énoncé en Français. D'après les expériences effectuées, la prise en compte des homophones les plus courants peut aider mais ne suffit pas à améliorer significativement les performances que ce soit en termes de transcription que de compréhension.

Cependant, ce constat est mis en suspens jusqu'à ce que le corpus décrit dans la section précédente soit complété et qu'une évaluation à grande échelle puisse être réalisée. Aussi, je souhaite reproduire les travaux effectués avec la métrique CVER en utilisant différents systèmes automatiques de CLN afin de 1) valider l'expérience cognitive et 2) vérifier la robustesse aux erreurs d'un système automatique en comparaison avec des humains.

# Chapter 8

# Travaux et projets industriels

Au cours de ma thèse, j'ai eu le plaisir de travailler à la conception de différents systèmes pour le traitement de la parole ainsi que de participer à différents projets industriels. Dans ce chapitre, je donne un aperçu des systèmes mis en place et des thématiques abordées. Toutefois, dans un souci de secret professionnel, de nombreux détails concernant la description des systèmes et des techniques mises en place, ainsi que les bases de données utilisées seront omis. Concernant ces dernières, nous pouvons toutefois citer l'utilisation du corpus ESTER (cf. chapitre 4) pour la réalisation d'un système *proof-of-concept* durant la première phase du projet Man Machine Teaming. Ce projet est animé par Thalès et Dassault, sous la direction de la DGA qui fut aussi l'un des collaborateurs pour la campagne d'évaluation ESTER. L'utilisation du corpus pour ce projet a été autorisée par le LaBRI, qui a fait l'acquisition initiale du corpus, et par la DGA.

## 8.1 Systèmes de RAP actuellement en place à Airudit

Afin de pouvoir répondre à des demandes de mise en place rapide et être aussi apte à proposer une solution dans le cas où le volume de données utilisable ne serait pas suffisant pour créer un système dédié, j'ai mis en place une méthode *hybride* pour nos projets. Cette méthode est basée sur l'utilisation d'une modélisation acoustique traditionnelle (de type HMM-DNN) construite et entraînée avec la boite à outils Kaldi. Une attention particulière a été donnée sur la modélisation acoustique et sa robustesse à différents environnements acoustiques bruités ainsi que différents accents en français que nous pouvons rencontrer lors de nos projets. Cette modélisation est ensuite utilisée comme base pour la construction de différents systèmes de RAP dont leur différence réside dans le vocabulaire employé ainsi que la modélisation linguistique associée. Ainsi, ceux-ci peuvent admettre un vocabulaire fermé, pour la mise en place de systèmes basés sur la commande vocale ou la reconnaissance de mots clés par exemple, ou un vocabulaire ouvert, pour la reconnaissance automatique de parole à large vocabulaire et la réalisation d'assistants conversationnels. Les différences de vocabulaire et de modélisation linguistique étant les suivantes :

- **Système de base**: Utilise un vocabulaire de plus de 80.000 mots connus de la langue française modélisés par un ensemble de plusieurs millions de phrases provenant de dialogues oraux radio-télédiffusés mais aussi de journaux du début des années 2000.
- **Système fermé**: Utilise un vocabulaire réduit composé des termes métiers utilisés dans le cadre d'un projet. Un ensemble réduit de phrases contextualisant l'utilisation

de ces termes métiers est défini pour l'entraînement d'un modèle de langage *fermé*.

- **Système ouvert**: Issu de la combinaison des deux modèles précédents. Le vocabulaire métier et le vocabulaire du langage courant sont concaténés et un nouveau modèle de langage est créé par interpolation du modèle de langage métier avec le modèle de langage large.

L'idée de cette *approche* est, outre la possibilité d'effectuer une mise en place rapide du système, de pouvoir distinguer la modélisation linguistique du jargon métier, qui est difficile à définir en amont des projets sans une connaissance préalable du domaine ou du métier, de la modélisation du langage courant qui demande un effort à long terme pour obtenir une couverture satisfaisante en terme de vocabulaire et contexte associé. À cela, il est nécessaire de pouvoir modifier et faire évoluer ces modélisations linguistiques de manière découplée sans que l'intervention du client ou de l'utilisateur final soit nécessaire.

### 8.1.1 Améliorations proposées

Au cours de ces dernières années, différents travaux ont été effectués afin d'améliorer et faciliter la construction, la mise en place et l'évaluation des systèmes de RAP développés pour nos projets. Dans cette sous-section, je donne une description succincte des améliorations les plus notables mises au point.

#### 8.1.1.1 Outils pour la création et la maintenance des systèmes de RAP

Afin de faciliter la création, l'évaluation continue et le déploiement des systèmes de RAP, différents outils ont été mis au point. Utilisés majoritairement en interne, une partie de ces outils sont aussi consultables et utilisables pour des utilisateurs externes sous notre supervision. À moyen terme, nous comptons les incorporer dans des solutions à plus grande échelle, en association avec d'autres modules ou outils, afin de permettre à nos clients d'améliorer eux-mêmes les systèmes sans que des connaissances particulières soient demandées. Parmi les outils notables développés au cours de ma thèse, nous pouvons citer :

- La mise en place d'une plateforme pour la création de corpus d'entraînement et de corpus d'évaluation pour les systèmes de RAP. Celle-ci s'accompagne d'une convention de transcription et d'annotation maison basée sur les documents fournis lors de la campagne d'évaluation ESTER ainsi que l'expérience acquise lors des différents projets.
- Le développement d'une interface pour l'évaluation de nos différents systèmes de RAP permettant la génération automatique de rapport sur la base de plusieurs métriques tel que le taux d'erreur de mots et le facteur temps-réel.
- La mise en place d'un ensemble de scripts et de programmes permettant de *plug-and-play* les différents composants constituants un système de RAP. Outre la possibilité d'évaluer et utiliser les composants séparément, cela nous permet de mettre rapidement à jour des composants spécifiques sans devoir passer par des étapes de ré-entraînement ou d'adaptation.
- La création de processus de décodage alternatifs pour le déploiement sur des systèmes d'exploitations non-supportés nativement par la boite à outils Kaldi (comme par exemple les décodeurs temps-réel reposant sur l'utilisation de bibliothèques POSIX, non compatibles avec Windows) ou des systèmes embarqués (Raspberry, Yocto).

- Le développement d'un processus d'écoute continue simple avec détection d'activité vocale et de détection de mot clé.
- Et enfin, la création d'un *widget* permettant de tester en temps réel le système de RAP couplé au système de compréhension du dialogue développé par Airudit. A ce titre, des scénarios d'utilisation pour chaque projet sont définis.

Cet ensemble de travaux nous permet à ce jour de pouvoir répondre rapidement aux demandes clients, de permettre un maintien et une amélioration des systèmes au travers du temps à moindre coût, mais aussi d'améliorer tout simplement l'expérience de l'utilisateur final.

#### 8.1.1.2 Transcription en temps-réel

Initialement, le processus de décodage mis en place avec nos systèmes de RAP prenait en entrée un fichier audio, ce qui demandait que l'utilisateur termine d'abord son tour de parole afin de produire ce fichier pour démarrer le processus. Malgré des temps de réponses convenables, cette approche comporte certaines limitations :

1. Les différents modèles constituant le système sont chargés en mémoire pour chaque nouvelle étape de transcription.
2. Le processus de décodage est dépendant de la fin de l'enregistrement : on ne peut dont pas transcrire l'audio tant que l'utilisateur n'a pas fini de le produire.
3. Les transcriptions partielles ne sont pas visibles pour l'utilisateur, c'est-à-dire que l'utilisateur ne perçoit pas de transcription lors de son tour de parole. Nous avons pu mesurer au travers de nos projets que cela se traduit par un impact cognitif non négligeable pour l'utilisateur, celui-ci ne sachant pas si le système est actif ou si un problème particulier ayant un impact sur la transcription a émergé durant le tour de parole.

Du fait, nous avons développé un processus augmenté en nous basant sur l'utilisation d'un protocole TCP similaire à celui proposé dans le projet Kaldi. Celui-ci est composé d'un serveur local, d'un client sous forme de widget, et d'un ensemble de techniques pour manipulation du flux audio. Cela nous permet de nous affranchir de ces limitations et de proposer un décodage instantané. De manière plus formelle, étant donné un serveur local lancé au préalable, le nouveau protocole prend directement en entrée le flux audio brut provenant d'un microphone et produit des transcriptions par le biais de deux étapes:

1. À chaque période de rafraîchissement une transcription partielle de l'audio est proposée. La transcription est formulée en considérant les $N$ dernières trames avant le dernier rafraîchissement, où $N$ est un paramètre pilotable défini en fonction de la tâche. La transcription partielle est affichée à l'écran avec une séquence préfixée (c.-à-d., "...") pour que l'utilisateur sache que le traitement n'est pas terminé.
2. Lorsqu'une limite d'émission est détectée, c'est-à-dire lorsque l'enregistrement est défini comme terminé par différents critères (comme la détection de fin de parole représentée par un long silence ou une faible activité de parole), le processus est arrêté. Dès lors, une transcription finale est proposée et affichée à l'écran.

Ce processus nous permet ainsi de réduire de manière drastique le temps de latence de manière objective, en effectuant la transcription à la volée sans attendre la fin de l'enregistrement, et de manière subjective, en permettant à l'utilisateur de visualiser la

transcription au fil de l'enregistrement. Celui-ci a été utilisé pour la première fois dans le cadre de la seconde phase du projet Man Machine Teaming, dont je présente les résultats obtenus dans la sous-section suivante.

### 8.1.2 Résultats du projet MMT et MMT2

Pour mesurer les performance des systèmes au court du temps et pour chaque projet, un rapport de performances est effectué de manière régulière sur la base de trois mesures objectives : le taux d'erreurs de mots (TEM), le facteur temps réel (TRF) et la latence estimée lors d'un processus de décodage en ligne. La troisième mesure définit le délai entre la fin de parole et la restitution de la transcription lors d'un processus de décodage.

Je donne ci-dessous les résultats obtenus sur ces trois métriques pour le projet MMT1, effectué en 2019, et le projet MMT2 effectué en 2020. Ce dernier ayant bénéficié de nombreuses améliorations, cela nous permet de voir l'évolution du système au cours de ces deux années. A noter toutefois qu'une comparaison objective directe est difficile : le corpus de test de MMT1 est constitué de 90 énoncés pour 628 mots et prononcés par 6 locuteurs différents, tandis que le second est composé de 479 énoncés pour 4462 mots prononcés par 11 locuteurs. Les enregistrements acquis pour la constitution des deux corpus de test ont été faits en condition réaliste, le locuteur, un pilote, peut donc poser des questions de manière libre sur des sujets métiers divers ou suivre un scénario de mission pré-établi. Dans les deux cas, le locuteur peut être mis face à un ordinateur (environnement calme) ou un simulateur de vol (environnement plus réaliste). Chaque acquisition de donnée a été effectuée via un micro-casque Sennheiser PC373D mis à disposition par Dassault.

| Corpus | TEM | RTF | Latence |
|---|---|---|---|
| Système large MMT1 (vocabulaire: 65k mots) | 4.9% | 0.228 | 8.68 |
| Système large MMT2 (vocabulaire: 70k mots) | 3.83% | 0.278 | 0.048 |

Table 8.1: Taux d'erreurs de mots (TEM), facteur temps-réel (RTF) et latence effective pour les systèmes des projets MMT1 et MMT2 sur leur ensemble de test respectif.

Malgré une augmentation significative du vocabulaire dans le cadre du second projet et un ensemble de test plus conséquent et riche en terme de variations, nos différentes optimisations ont permis d'améliorer le rapport performance/temps-réel lors du second projet tout en permettant de rendre quasiment nulle la latence du processus de décodage lors de son utilisation en mode en-ligne. Dans ce cadre, nous avons dénoté une charge cognitive inférieure lors de la deuxième phase du projet, renforçant ainsi l'expérience utilisateur. Des mesures complémentaires lors d'autres projets ont toutefois été effectuées afin de valider cette observation.

L'ensemble des mesures à été effectué sur une machine virtuelle sous Ubuntu 16.04 dont l'utilisation des ressources et des processus concurrents ont été limités à un seul CPU et un seul thread. Dans ce cadre, les caractéristiques de cette machine sont les suivantes :

- CPU : Intel Core Processor (Haswell): 4 cores - 2Ghz; 6.91 avg. cores/computer, 2.1 GFLOPS/core, 14.5 GFLOPS/computer.
- RAM : VPS 2016 Cloud RAM 3 (1x16Go + 1x8Go DDR2)

Concernant le taux de mots hors-vocabulaire dans les corpus de test par rapport au lexique de nos systèmes, celui-ci est équivalent: environ 4% de mots pour le premier corpus et 6% pour le second).

## 8.2 Relation entre le système de RAP et le système de CLN

Dans cette section, je présente les travaux effectués en rapport avec la mise en relation du système de RAP et du système de compréhension du langage naturel (abrégé CLN) développé par la société Airudit. Cette association permet la formation d'un système de compréhension du langage parlé (abrégé CLP, ou *Spoken Language Understanding* en anglais) avec restitution d'une réponse à l'utilisateur. Ces travaux ont pour but de traiter un problème inhérent existant de l'association de ces deux systèmes dans un contexte traditionnel : les systèmes de RAP traditionnel basé phonèmes et les systèmes de CLN évoluent et interagissent de manière disjointe. En effet, chaque système est optimisé indépendamment de l'autre, et la sortie de l'un est formulée sans considération du problème traité par l'autre. Cette relation disjointe enlève donc des mécanismes observés dans une communication naturelle comme par exemple la connaissance a priori du contexte, l'inférence du sens malgré le manque d'informations ou encore la gestion de l'information lors de la présence d'erreurs. Ainsi, afin de résoudre ces problèmes, nous proposons une évolution de cette relation au travers de ce que nous appelons désormais un système de *Speech to Context*.

La première sous-section de cette section est dédiée à l'introduction du système de CLN développé par la société Airudit. La seconde se focalise à introduire le brevet déposé au cours de l'année 2019 et proposant une solution au problème de relation précédemment formulé, reposant sur l'incorporation de connaissances a priori dans le système de RAP et de mécanismes de restitution du sens dans un dialogue.

### 8.2.1 Système de CLN développé par Airudit

Le domaine de la recherche et du développement de systèmes de CLN et de CLP propose une littérature riche et un vaste champ d'application. Un ensemble non négligeable d'approches peuvent être proposées, basées sur le traitement automatique de langage (abrégé TAL) et l'extraction par mots clés [Rose et al., 2010] en association avec des systèmes de RAP, ou encore l'apprentissage profond avec l'utilisation de modèles Transformer [Devlin and M.-W. Chang, 2019] et d'approches bout-en-bout [Serdyuk et al., 2018, Lugosch et al., 2019] se proposant de résoudre les problèmes de RAP et de CLN au travers d'un seul et même système.

L'approche choisie par Airudit depuis sa création repose quant à elle sur l'utilisation d'ontologies, des structures informatiques permettant de modéliser un contexte métier et expliquées en section 8.2.1.1, pour traiter le problème de CLN de manière distincte. Ici, l'ensemble d'un domaine métier est modélisé et sert de base de connaissances pour la compréhension d'un énoncé.

#### 8.2.1.1 Ontologies et formalisme de création

Les ontologies sont des modélisations informatiques qui tendent à représenter un fonctionnement humainement observable. Elles sont majoritairement utilisées dans les domaines de la biologie, de la chimie et de la médecine afin de modéliser, ou cartographier, des observations, tant au niveau conceptuel (inventaire des éléments et de types de relations qui existent) qu'au niveau relationnel (inventaire des interactions entre les éléments).

Il existe différents types d'ontologies. Le type le plus basique est appelé taxonomie et définit une organisation d'un ensemble de concepts par une méthode de catégorisation. Cette ontologie présente l'intérêt de pouvoir référencer les concepts d'un domaine avec

une certaine proximité au langage : chaque concept est nommé et peut donc comporter des synonymes. La limite de la taxonomie est l'impossibilité de relier les concepts par des relations différentes que leur relation de parenté. On ne peut toutefois pas représenter les temporalités, dépendances, les causes, les effets.

Les ontologies formelles sont un type particulier d'ontologies qui permettent une utilisation des éléments modélisés par des processus informatiques. En effet, une méthode systématique pour la création des concepts et de leurs relations va permettre l'exploitation par un système informatique. L'utilisation de composants réutilisables rend même l'utilisation des ontologies portable d'un système à un autre [Coral et al., 2010].

Concernant Airudit, les ontologies développées couvrent généralement les éléments suivants :

- Classes : ensembles, ou types d'objets ;
- Individus : occurrence des classes, connaissances, objets ;
- Attributs : propriétés, fonctionnalités, caractéristiques ou paramètres que les objets peuvent posséder et partager ;
- Relations : les liens que les objets peuvent avoir entre eux ;
- Évènements : changements subis par des attributs ou des relations ;
- Métaclasse (web sémantique) : des collections de classes qui partagent certaines caractéristiques.

#### 8.2.1.2 Exemples de systèmes de CLN basés sur les ontologies

Différents travaux de l'état de l'art proposent une approche reposant sur l'utilisation d'une ou des ontologies, modélisant des bases de connaissances, afin de concevoir un assistant personnel dans différents cas de figure ou domaine, et ayant des connaissances basées sur une ou des ontologies. Ici, je donne quelques exemples ayant servi de références pour la conception du système de CLN d'Airudit.

Un premier exemple est le travail de [Cimiano et al., 2014] qui proposent de créer un système d'interprétation sémantique des énoncés utilisateurs basé sur l'utilisation d'une ontologie associée à un pont Lexique-Ontologie [McCrae et al., 2011] et l'utilisation de la base SPARQL, regroupant un langage de requête et un protocole pour la manipulation des données. L'approche propose une décomposition sémantique de l'énoncé utilisateur au travers de techniques de TAL afin de faire la correspondance entre des éléments de syntaxe et des éléments conceptuels de l'ontologie. Bien que performante, cette approche s'avère peu tolérante aux écarts dialogiques tel que l'utilisation de disfluences ou la présence de fautes grammaticales et syntaxiques. En outre, le coût computationnel pour la résolution d'énoncés complexes rend difficile la mise en production de la méthode, le nombre de détails demandé pour la modélisation étant exponentiel en considération du spectre métier couvert.

Autre exemple, les travaux de [Bello et al., 2018] proposant la mise en place d'un système de réponse automatique spécifique à un centre d'appel téléphonique. Ce système repose sur l'utilisation d'une ontologie de type thésaurus créée sur une base de connaissance issue d'échanges téléphoniques et une recherche de connaissance basée sur une technique d'évaluation par combinaison de mots-clés. Contrairement aux travaux précédents, les auteurs proposent une approche simplifiée d'un point de vue linguistique et ontologique

permettant la mise en place rapide d'un système dans un contexte métier donné. Ainsi, la création d'un thésaurus référençant les définitions des différents concepts métier est le seul pré-requis pour décrire l'énoncé utilisateur et délivrer une réponse correspondante.

D'autres travaux comme ceux réalisés par [Al-Zubaide and Issa, 2011] ou [Altinok, 2018] peuvent être cités. Ici, l'énoncé de l'utilisateur (formulé dans un langage naturel) est soumis à un traitement comportant des étapes de compréhension, d'interprétation et de restitution de la donnée avant d'être mis en correspondance avec les différents concepts d'une ontologie spécifique. Bien que proposant de traiter des requêtes en langage naturel, ces systèmes se focalisent sur la compréhension de requêtes simples, les relations entre les concepts au sein d'un même énoncé sont donc limités. En outre, les ontologies sont composées d'un vocabulaire restreint, se proposant de modéliser un contexte d'utilisation connu a priori et relativement fermé, rendant le coût performance/rapidité intéressant au détriment d'une utilisation d'une portabilité faible.

#### 8.2.1.3 Problématiques liées aux ontologies

L'utilisation de modélisations basées sur les ontologies dans les systèmes de CLN permet particulièrement de pouvoir traiter des énoncés complexes sur un domaine métier précis. La connaissance étant portée par l'ontologie, les relations sémantiques et classes spécifiques au domaine d'étude sont rapidement reconnues et mises en commun pour une compréhension rapide et efficace. Toutefois, l'utilisation d'ontologies présente également certaines problématiques parmi lesquelles on peut citer :

- la dépendance du système à la structure et au format de l'ontologie.
- le manque de réalisme et de robustesse pour des environnements et des domaines trop riches (p. ex., l'ensemble des spécificités de tous les services de l'administration française).
- la nécessité de coupler la méthode de traitement basée ontologie à des systèmes de TAL dans le but de pallier le manque de traitement syntaxique et grammatical absent d'une structure ontologie.

L'approche proposée par Airudit, décrite dans la sous-section suivante tente de répondre à ces différentes problématiques dans le but de proposer un assistant personnalisé pour le métier de ses clients. L'idée générale étant de proposer un système de CLN s'appuyant sur des briques de TAL performantes et une ontologie métier dans laquelle l'ensemble des processus et concepts métiers sont modélisés.

#### 8.2.1.4 Proposition de système de CLN par Airudit

Airudit propose donc une solution de CLN dans le but de mettre à disposition de ses clients un système de dialogue capable de traiter des énoncés simples et complexes dans un périmètre métier connu et modélisé par avance. L'accent étant mis, en outre, sur l'utilisation d'un langage naturel.

La structure du système actuel est la suivante :

1. À la réception d'un énoncé textuel en entrée, une **première phase** de TAL est effectuée dans le but d'identifier des éléments de langage tel que la présence d'entités nommées ou de négation, et de détecter une forme syntaxique singulière ou des informations contextuelles pertinentes à la compréhension de la requête de l'utilisateur.

2. Cette première phase terminée, le système bénéficie d'une première analyse de l'énoncé et dispose d'un ensemble de métadonnées pertinentes et utilisables dans le but de comprendre l'intention de l'utilisateur et de former au besoin une requête vers une base de données externe pour récupérer des connaissances et ainsi formuler la réponse. Lors de la **seconde étape**, une attention particulière est portée sur les entités nommées présentes dans l'énoncé dans le but de faire le lien entre ces entités et les concepts (classes) présents dans l'ontologie métier. L'objectif ici étant de mettre en évidence et d'isoler la connaissance pertinente et adaptée à la requête de l'utilisateur.

3. À ce niveau, deux états sont possibles : (1) la requête de l'utilisateur correspond à une requête de recherche d'informations (l'utilisateur cherche à obtenir une donnée), ou (2) la requête de l'utilisateur donne lieu à un scénario (l'utilisateur va interagir avec la machine dans le but de produire une action par exemple). Dans ce deuxième cas, la machine va prendre la main sur le dialogue en posant diverses questions à l'utilisateur. Distinguons un instant la démarche de traitement de ces deux états afin d'observer l'utilisation de l'ontologie dans ceux-ci.

    - Dans le premier cas, l'utilisateur cherche à acquérir une information. L'objectif est donc de générer, à partir de la connaissance précédemment identifiée comme pertinente, une requête vers un système d'information tiers. Cette requête va s'appuyer sur les éléments de l'ontologie pour définir les classes des individus recherchés et les attributs associés. Un exemple d'énoncé utilisateur menant à ce type de requête serait : "À quelle distance se trouvent les bureaux de poste les plus proches". Dans ce cas, la requête générée indiquera une recherche d'individu de classe "Bureau de poste" et d'un attribut "distance". Il faut noter également que l'information à récupérer peut, dans certains cas, être uniquement un éditorial de réponse pour des questions de type "Foire aux Questions" (abrégé FAQ).

    - Dans le second cas, l'utilisateur a besoin de mettre à jour une information, de déclencher une action ou encore d'être assisté, ou guidé, par la machine. Par exemple, l'utilisateur demande une aide au dépannage de son imprimante, ou demande au système de changer la station de radio, etc. L'approche d'Airudit dans la mise en place de scénario est la suivante : le système s'enrichit de la connaissance métier identifiée pertinente pour une question donnée, et s'adresse à l'utilisateur pour obtenir les informations manquantes. Dans le cadre d'un dépannage, il suit par exemple la liste des vérifications à demander à l'utilisateur et déroule celle-ci. Dans le cas d'une modification de fréquence radio, le système demandera des précisions sur les paramètres manquants (quel poste, ou quelle fréquence) avant de générer l'action. En fin de scénario, la requête vers le système pour mettre à jour une potentielle information est fortement similaire à celle précédemment décrite pour l'obtention d'une information (cf. point précédent).

4. Une fois l'information récupérée, mise à jour ou le système mis en action, Airudit propose de formaliser des réponses créées à la volée. Ici encore, le générateur de réponse s'appuiera sur les classes de l'ontologie pour former les phrases et accorder les différents thèmes qui la composent.

Cette présentation du système actuel rend donc compte de la prise en charge d'une des problématiques précitées dans la sous-section précédente, à savoir : le couplage d'un

système basé ontologie avec une pile de TAL suffisante. De plus, il est à noter qu'Airudit construit l'ensemble de ces ontologies selon un formalisme commun appelé *Basic Formal Ontology* [Arp et al., 2015]. En effet, le formalisme proposé par Barry Smith et ses collègues permet de représenter un grand nombre de domaines diversifiés selon une même architecture globale et des règles strictes. Par exemple, les classes représentant des objets persistants dans le temps (tels les objets matériels) sont rangés sous la classe dite des *continuants* tandis que des objets non persistants seront rangés sous la classe dite des *occurrents*. L'utilisation de ce formalisme permet, entre autre, de maintenir une cohérence dans le traitement des requêtes des utilisateurs quel que soit le domaine métier modélisé, et cela au travers du temps.

Enfin, afin de répondre à la troisième problématique mise en avant, à savoir, le manque de réalisme d'un tel système sur des environnements métier trop riches et complexes, Airudit travaille aujourd'hui sur la mise en place de réseaux d'ontologies. Cette structure permettrait de pouvoir scinder les différents sous-périmètres métier en ces différentes ontologies tout en conservant une efficacité de traitement des requêtes utilisateurs. En outre, des travaux sur l'augmentation et la mise à jour à la volée des ontologies en considération de l'expérience utilisateur, et notamment le langage employé non-couvert, sont une des priorités de l'entreprise.

### 8.2.2 Connaissance a priori et mécanismes de restitution du sens dans un dialogue

Cette section présente le brevet européen déposé en 2019 et visant à traiter la relation disjointe entre un système de RAP traditionnel et un système de CLN basé sur les ontologies. Au moment de l'écriture de ce manuscrit, ce brevet a été accepté sous le nom "Procédé et dispositif d'obtention d'une réponse à partir d'une question orale posée à une interface homme-machine" (en français) et est consultable sur la plateforme WIPO [1]. Une copie du document est aussi mis à disposition en Annexe A

Pour des soucis de secret industriel, cette section se focalise sur l'aspect général du brevet et les propositions apportées mais omet certains détails techniques concernant la réalisation du brevet dans un contexte industriel.

#### 8.2.2.1 Problématique

Un système de RAP traditionnel est doté d'un espace de vocabulaire, comportant tout le vocabulaire admis par ledit système, c'est-à-dire l'ensemble des éléments lexicaux désignés par le dictionnaire phonétique et le modèle de langage. En parallèle, un système de CLN, muni d'une ontologie, est aussi doté d'un espace de vocabulaire associé à l'ontologie, c'est-à-dire l'ensemble des individus, attributs, relations, etc., précédemment désignés.

Le brevet s'inscrit dans le cadre général d'une réponse par un système de SLU, composé donc d'un système de RAP traditionnel et d'un système de CLN basé sur les ontologies, à une question orale posée. Dans l'état de l'art antérieur, les procédés visant à répondre à la question de l'utilisateur comportent une étape de génération par un moteur de RAP d'une transcription désignant un signal de parole en entrée puis une génération d'une réponse par un moteur de CLN à partir de la question formée par le signal de parole transcrit. Chacun des systèmes de RAP et de CLN étant dotés d'un vocabulaire indépendant de

[1] `https://patentscope.wipo.int/search/fr/detail.jsf?docId=WO2020260797`

l'autre, on comprend que la chaîne de traitement mise en place ne peut fonctionner que pour le lexique commun aux deux vocabulaires.

Dans l'immédiat, il est envisageable de compléter chacun des vocabulaires des systèmes par le lexique connu seulement de l'autre des deux moteurs. Toutefois, une telle solution est difficile à mettre en oeuvre, car elle nécessite d'augmenter chacun des vocabulaires dès lors qu'un nouveau terme apparaît. En outre, cette solution pose un problème pour la pérennité du système en considération de l'évolution d'un métier, d'un domaine, ou plus généralement de l'évolution constante d'un langage.

#### 8.2.2.2 Proposition

L'objectif du brevet est donc de remédier à l'inconvénient pré-cité, à savoir le traitement du vocabulaire appartenant au complémentaire de l'intersection des ensembles A, le vocabulaire du système de RAP, et B, le vocabulaire du système du système de CLN. En outre, différents modes de fonctionnement sont proposés en considération de différents critères que nous détaillons.

Le brevet propose un procédé de génération d'une réponse à un signal de parole à partir d'un système de RAP, comportant un espace de vocabulaire A, générant une transcription étant donné un signal de parole en entrée et d'un moteur de CLN, doté d'au moins une ontologie comportant un espace de vocabulaire B, générant la réponse à la requête. La réponse est générée à partir d'un pré-traitement de la transcription comportant, pour chacun des mots de la dite transcription n'appartenant pas à l'espace de vocabulaire B, une étape de plongement lexical des mots dans l'espace de vocabulaire B, une détermination d'une mesure de similarité entre ledit plongement lexical dudit mot et chacun des mots de l'espace de vocabulaire B, une étape de sélection d'au moins un mot dans ce même espace et enfin, une étape de remplacement du mot de la transcription par le mot sélectionné pour reformer la question initiale.

Le plongement lexical du mot n'appartenant pas à l'espace de vocabulaire B est un plongement lexical à la volée par composition [Ling et al., 2015]. Le procédé de modification peut comporter une étape intermédiaire entre l'étape de plongement lexical du mot n'appartenant pas à l'espace de vocabulaire cible et l'étape de sélection, cette étape intermédiaire comportant une modélisation statistique des connexions sémantiques faites entre les termes utilisateurs et les concepts définis dans le système de CLN en se basant sur un corpus de requêtes d'utilisateurs. Dans ce cas, les connexions sémantiques correspondent aux relations sémantiques entre les classes et individus de l'ontologie. L'utilisation d'un corpus de requête utilisateur permettant à la fois de générer de nouvelles relations mais également de pondérer ces relations. Le poids assigné à chaque relation peut être estimé par une méthode de remise, en se servant de la fréquence relative de chaque concept par exemple. De ce fait une relation rarement représentée dans les requêtes utilisateurs sera ainsi faiblement pondérée, et inversement. De préférence, la méthode de remise et d'interpolation entre les poids est déterminée de sorte qu'une relation avec une pondération très faible puisse être considérée. Une pondération peut être obtenue par un modèle probabiliste simple. Ainsi, si on considère pour la phase d'entraînement un ensemble de questions que des utilisateurs ont déjà posé à un système de dialogue Homme-Machine, en utilisant un modèle de type word2vec [Mikolov et al., 2013], la pondération peut tenir compte de la moyenne des vecteurs de mots pour chaque phrase afin de regrouper les phrases similaires. De cette manière, un modèle probabiliste peut être défini en calculant la probabilité d'occurrence de chaque phrase.

L'étape de sélection d'au moins un mot de l'espace de vocabulaire cible peut comporter un algorithme de sélection prenant en entrée les N mots connus du moteur sémantique les plus proches sémantiquement du mot inconnu et proposant un ou des termes de remplacement étant donné plusieurs critères contrôlables par l'utilisateur, comme par exemple la probabilité de la demande/question en considération de l'expérience utilisateur, et le coût estimé pour la résolution de la question/requête en considération des connexions sémantiques. Étant donné un modèle statistique, des connexions sémantiques dont les relations sont pondérées, il est possible d'évaluer, en fonction de la distance sémantique, un coût pour chaque phrase ayant un terme de remplacement en sommant les poids du chemin emprunté dans le modèle. D'autres évaluations de coût sont toutefois possibles.

Dans la sous-section suivante, je donne un exemple simple d'application de cette méthode.

#### 8.2.2.3 Exemple

On suppose qu'un utilisateur produise le signal vocal S, qui est correctement transcrit par le système de RAP en "Où est-ce que je peux crécher pour pas cher?". Usuellement, cette transcription est adressée au système de CLN en tant que question Q pour générer une réponse R. Lorsque le système de CLN ne comporte pas le mot "crécher" dans son espace de vocabulaire associé à l'ontologie utilisée, dans le cas présent une ontologie hôtelière, il n'est pas capable de traiter la requête ou a un risque fort de formuler une réponse inadaptée. Pour autant, le vocabulaire associé à l'ontologie peut comporter les mots "dormir", "se restaurer", "loger" associés à la classe "service hôtelier".

A partir de la transcription résultant du système de RAP, le procédé proposé effectue les étapes suivantes :

1. On détermine l'ensemble I des mots de la transcription de signal vocal S n'appartenant pas au dit espace de vocabulaire B du système de CLN. Pour exemple, I = {"crécher"}.
2. On effectue un plongement lexical par composition pour les termes de I dans un espace de vocabulaire séparé V.
3. On effectue une projection des mots inconnus dans un espace commun (B $\cup$ V) et on sélectionne les N meilleurs candidats en considération de la distance sémantique et/ou de mesures de similarité.
4. En considération de critères de temps, de coût et de performance, le mot candidat est choisi. Selon l'urgence de la demande, le chemin à parcourir pour obtenir le terme choisi dans l'ontologie et le temps associé à la résolution de la requête, un autre terme peut être sélectionné. A noter que, bien que automatisée, cette sélection peut être contrôlée par l'utilisateur. Dans cet exemple, l'heure est tardive et l'utilisateur a un besoin pressant. Le mot "dormir" sera retenu pour sa proximité sémantique et son coût associé. Du fait de l'urgence de la demande, un poids plus important est mis sur des chemins dirigées vers des termes pouvant entraîner des actions ou scénarios supplémentaires. Ici, "se restaurer" et "loger" rajouteraient des critères de recherche et peuvent amener vers un scénario avec une série de questions-réponses.
5. On modifie la transcription par remplacement du mot inconnu du système de CLN par le mot sélectionné afin d'obtenir une transcription modifiée: "Où est-ce que je peux dormir pour pas cher ?".
6. La transcription modifiée est alors injectée dans le système de CLN afin de formuler une réponse adéquate à l'utilisateur.

Bien sûr, l'approche proposée n'est pas limitée à l'exemple qui vient d'être décrit et de nombreux aménagements peuvent être apportés sans sortir du cadre du brevet. De plus, les différentes caractéristiques, formes, variantes et modes de réalisation de l'invention peuvent être associés les unes avec les autres selon diverses combinaisons dans la mesure où ils ne sont pas incompatibles ou exclusifs les uns des autres.

## 8.3 Conclusion

La majorité des travaux de recherche effectués à Airudit au cours de ces dernières années pour la RAP s'est focalisée, outre les travaux et améliorations présentés précédemment, sur : 1) l'étude et la mise en place d'un système bout-en-bout en replacement des systèmes de RAP traditionnels actuellement utilisés dans nos projets, et 2) la réalisation industrielle du brevet décrit dans ce chapitre. La mise en production des systèmes industriels issus de ces travaux débutera au début de l'été 2021 afin de remplacer, nous l'espérons, notre système actuel au milieu de l'automne 2021.

Concernant les futurs travaux et les perspectives de travaux, nous référons le lecteur au chapitre suivant 9 qui donne une description détaillée des travaux futurs du côté académique et industriel.

# Chapter 9

# Conclusion, travaux en cours et perspectives

L'objectif principal de cette thèse était le développement d'un système de reconnaissance de parole en français et le traitement de différentes problématiques en rapport avec l'intégration du dit système dans un système de compréhension du langage parlé développé par la société Airudit. Ces problématiques sont, pour rappel, les suivantes :

1. Proposer un ou des système(s) de RAP temps-réel en français robuste(s) à différents environnements et locuteurs.
2. Proposer des approches permettant au système de RAP d'évoluer en considération de l'évolution naturelle du langage. Parallèlement, une attention particulière est donnée à l'étude du problème de mot hors-vocabulaire.
3. Proposer des approches permettant au système de RAP d'évoluer en considération de l'évolution du système de compréhension du langage naturel (CLN) développé séparement. Ces approches doivent considérer la gestion de la différence entre les vocabulaires du système de RAP et du système de CLN.
4. Réciproquement, renforcer le lien entre les deux systèmes présentés en incorporant des informations a priori dans le système de RAP et provenant du système de CLN.

Différents projets industriels ayant été effectués durant ma thèse, il était important de pouvoir proposer des systèmes de RAP pouvant être utilisés à court et long terme. Ainsi, différents systèmes ont été réalisés au cours de ma thèse, dont certains sont encore en cours d'industrialisation au sein d'Airudit. Pour des raisons de secret industriel, les détails concernant ces différents systèmes sont omis. Toutefois, et comme le lecteur doit s'en douter à ce moment du manuscrit, le choix a été fait d'utiliser à court terme des systèmes traditionnels entraînés avec la brique logicielle Kaldi, du fait de sa maturité et stabilité. Concernant la recherche et le développement à long terme, notre attention est portée sur les systèmes bout-en-bout, de par leur facilité d'adaptation à de nouvelles langues et à de nouveaux vocabulaires métiers. Ce choix s'est avéré être bénéfique au vu des retours de nos clients (Dassault, Naval, Airbus, EPS, etc) sur les systèmes actuellement en place chez eux mais aussi sur les démonstrations que j'ai pu effectuer avec nos *futurs* systèmes. Les systèmes développés se sont avérés particulièrement robustes pour différents environnements (p. ex., laboratoire, domotique ou encore militaire), type de parole (lue ou conversationnelle) ou encore locuteurs (p. ex., français avec différents accents).

Afin de répondre à la seconde problématique, différentes approches ont tout d'abord été étudiées afin de traiter le problème de mot hors-vocabulaire existant avec les systèmes de RAP traditionnels (voir introduction du chapitre 6). Toutefois, de par leur rapide popularisation au même moment, mon attention s'est rapidement portée sur les systèmes bout-en-bout qui, par nature, ne sont pas affectés par ce problème. Ainsi, il a été décidé d'étudier en profondeur l'utilisation de ces approches bout-en-bout pour la RAP en français. Pour cela, nous avons comparé différentes modélisations acoustiques et architectures ainsi que les unités orthographiques pouvant être utilisées en place et lieu des unités phonétiques. Bien que les premiers résultats ont démontré un écart de performance significatif entre les systèmes traditionnels et bout-en-bout (voir chapitre 6), cette étude fut ensuite approfondie afin d'évaluer plus précisément les erreurs formulées par les différents systèmes et définir les pistes d'améliorations possibles, que cela soit sur la modélisation acoustique ou les informations linguistiques supplémentaires à utiliser. Cette étude est reportée dans le chapitre 7.

Dans un souci de résoudre le troisième problème portant sur les différences de vocabulaires entre le système de RAP et de CLN, deux propositions à court et long terme ont été faites. Dans un premier temps, et afin de répondre aux contraintes industrielles, un ensemble de méthodes et outils ont été développés afin de pouvoir adapter facilement le système traditionnel de RAP actuel à de nouveaux projets, et cela en considération du vocabulaire métier cible et du vocabulaire connu par le système de CLN. Cela m'a permis de réduire considérablement le temps alloué à la définition des informations linguistiques et phonétiques pour de nouveaux systèmes traditionnels tout en conservant des performances, jugées satisfaisantes de par les évaluations comparatives internes effectuées, pour un ensemble de nouveaux domaines. Dans un second temps, j'ai proposé en collaboration avec mes collègues d'Airudit, de nouvelles méthodes visant à résoudre directement ce problème de vocabulaire sans avoir besoin d'intervention externe (p. ex., manuelle). Ces méthodes considèrent en outre l'utilisation a priori de connaissance issues des ontologies existantes dans le système de CLN, notre quatrième problème, afin de proposer des modes de fonctionnement en considération de différents critères. Ces méthodes ont donné lieu à un dépôt de brevet que je décris dans le chapitre 8.2 avec ces différents modes de fonctionnement. A ce jour, le système RAP-CLN joint utilisant les méthodes proposées est toutefois encore en cours de réalisation au sein d'Airudit. Un second brevet est aussi en préparation afin d'adapter notre proposition à nos futurs systèmes.

## 9.1 Travaux en cours et perspectives

Au moment de l'écriture de ce manuscrit, différents axes de recherche ont été initiés qui s'inscrivent dans la continuité directe des travaux présentés. Dans cette section, je donne un aperçu des axes étudiés ainsi qu'une perspective quant au développement d'un système de RAP de *nouvelle génération* au sein d'Airudit et des projets auxquels je participe.

### 9.1.1 Système de RAP bout-en-bout pour le français

Dans les chapitres 6 et 7, j'ai proposé respectivement une étude comparative des méthodes bout-en-bout pour la RAP en français et une analyse approfondie des erreurs formulées par les différentes approches bout-en-bout sur cette même langue. Depuis la réalisation de ces travaux, un nombre conséquent d'améliorations pour les modèles utilisés ont été proposées, et notamment pour le modèle Transducer dont je dirige le développement dans ESPnet. Ces améliorations comprenant de nouvelles architectures, comme par exemple

l'encodeur de type Conformer (voir Sec. 5.4.2), de nouveaux critères et planificateur d'entraînement [Vaswani et al., 2017, Popel and Bojar, 2018], et plus généralement des optimisations portant sur la qualité du modèle et la réduction de la mémoire nécessaire lors de l'entraînement.

De plus, et grâce notamment à la possibilité d'utiliser des ressources matérielles supérieures comparées aux travaux précédents, j'ai décidé de revenir sur la définition du modèle Transducer pour le français. Le tableau 9.1 décrit les performances actuelles, en terme de taux d'erreurs de mots (TEM), de ces nouveaux modèles comparés aux anciens modèles.

| Modèle | Unité | Modèle de langage | TEM |
|---|---|---|---|
| chain LF-MMI | phn | 3-gram word LM | 13.7 |
| RNN-Transducer | char | char LM + word LM | 18.1 |
| RNN-Transducer w/ att. | bpe | bpe LM | 17.4 |
| RNN-Transducer (new) | char | X | 15.7 |
| Conformer/RNN-Transducer (new) | char | X | 11.6 |

Table 9.1: Taux d'erreurs de mot (TEM, donné en %) pour différents modèles transducer sur le corpus d'évaluation d'ESTER2. Les anciens modèles sont aussi reportés.

Nous pouvons observer ici un gain de performance significatif entre l'ancien système basé RNN et le nouveau (-2.4% TEM). Bien qu'il soit difficile de comparer directement les deux modèles, de nombreux aspects de l'entraînement et du décodage de ce modèle ayant été modifiés au fil du temps, cette amélioration peut s'expliquer par 1) l'ajout de la technique d'augmentation de données *SpecAugment* [Park et al., 2019] et 2) l'augmentation du nombre de couches, d'unités ainsi que la taille du batch utilisé lors de l'entraînement.

Concernant le modèle utilisant un encodeur de type Conformer, les taux d'erreurs obtenus sont pour la première fois inférieurs à notre système traditionnel de référence (-2.1% WER) et donnent de nouvelles perspectives pour une utilisation industrielle. Toutefois, une étude approfondie est encore à effectuer, notamment sur l'adaptation à de nouvelles données et la comparaison avec les systèmes actuellement utilisés dans différents projets d'Airudit tel que le projet MMT (Voir chapitre 8.1.2). Dans le cadre de ce dernier, une étude est actuellement en cours.

### 9.1.2 Fonctions de perte auxiliaires pour modèle Transducer

Dans un souci d'améliorer la qualité du modèle Transducer, je propose dans le cadre du projet ESPnet une nouvelle procédure d'entraînement augmenté. La structure du modèle est complétée par quatre couches de classification utilisées pour entraîner des tâches auxiliaires parallèlement à la fonction de perte RNN-T standard. L'architecture que je propose est représentée par la figure 9.1, où, pour plus de lisibilité, une seule couche d'encodeur intermédiaire est utilisée dans le calcul de $\mathcal{L}_{\text{aux-trans}}$ et $\mathcal{L}_{\text{symm-KL}}$.

Les cinq fonctions de perte peuvent être entraînées simultanément et optimisent conjointement la fonction de perte totale définie comme :

$$\begin{aligned}\mathcal{L}_{tot} = \lambda_{\text{trans}}\,\mathcal{L}_{\text{trans}} \;+\; \lambda_{\text{CTC}}\,\mathcal{L}_{\text{CTC}} \;+\; \lambda_{\text{aux-trans}}\,\mathcal{L}_{\text{aux-trans}} \\ +\; \lambda_{\text{symm-KL}}\mathcal{L}_{\text{symm-KL}} \;+\; \lambda_{\text{LM}}\,\mathcal{L}_{\text{LM}}\end{aligned} \tag{9.1}$$

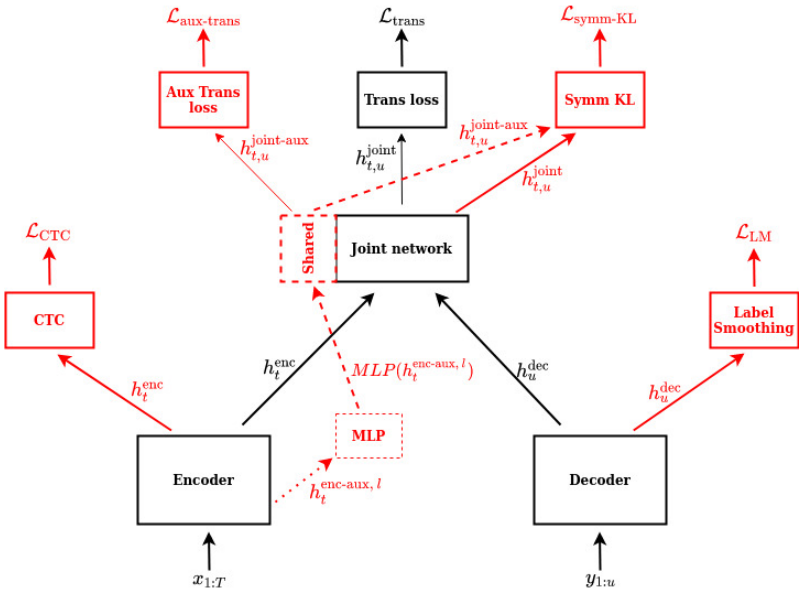



Figure 9.1: Architecture transducer avec tâches auxiliaires. En noir la structure de base. En rouge, les nouvelles fonctions de perte, connexions et modules intermédiaires.

Le rôle de ces fonctions dans le cadre de la modélisation acoustique via l'approche Transducer est le suivant.

#### 9.1.2.1 CTC loss ($\mathcal{L}_{\text{CTC}}$)

De manière similaire à *monotonic RNN-T* [Tripathi et al., 2019] qui limite le nombre d'étiquettes émises à chaque pas de temps à strictement un, j'explore l'utilisation d'une autre fonction de régularisation (faible) via la fonction auxiliaire CTC [Jeon and Kim, 2020]. En parallèle, des avantages ayant été reportés sur l'initialisation de l'encodeur du modèle Transducer avec un modèle pré-entraîné avec la CTC [Hu et al., 2020], j'entraîne ici conjointement les deux fonctions de perte de la même manière que ce qui est effectué pour le modèle CTC-Attention [Kim et al., 2017].

#### 9.1.2.2 Auxiliary transducer loss ($\mathcal{L}_{\text{aux-trans}}$)

Pour remédier au problème d'*underfitting* de l'encodeur, qui est dû au rôle majeur donné par le décodeur, j'incorpore la fonction de perte auxiliaire RNN-T proposée par Liu et al. [Liu et al., 2020] afin d'augmenter les gradients des signaux. Sur la base de leur proposition, les couches intermédiaires de l'encodeur sont connectées à la fonction de perte auxiliaire RNN-T au travers d'un réseau de type *Multilayered Perceptron* remplaçant la couche linéaire dans le réseau conjoint :

$$h_{t,u}^{\text{joint-aux}} = \text{JointAux}(\text{MLP}(h_t^{\text{enc-aux}}), h_u^{\text{dec}}) \tag{9.2}$$

#### 9.1.2.3 Auxiliary symmetric KL-divergence ($\mathcal{L}_{\text{symm-KL}}$)

Afin de pénaliser les gradients lors de la supervision avec le critère précédent, je considère également l'utilisation d'un critère basé sur la divergence de Kullback-Leibler, ici symétrique :

$$\mathcal{L}_{\text{symm-KL}} = \frac{1}{TLU} \sum_{t,l,u} I(l)(D_{\text{KL}}(\text{Softmax}(h_{t,u}^{\text{joint}}), h_{t,u}^{\text{joint-aux}}) + D_{\text{KL}}(\text{Softmax}(h_{t,u}^{\text{joint-aux}}), h_{t,u}^{\text{joint}})), \tag{9.3}$$

où $M = \text{Softmax}(\frac{1}{2}(h_{t,u}^{\text{joint}} + h_{t,u}^{\text{joint-aux}}))$, $L$ définit l'ensemble de couches intermédiaires de l'encodeur et $I(l)$ est un indicateur binaire désignant si une couche est utilisée pour la tâche auxiliaire.

#### 9.1.2.4 LM loss ($\mathcal{L}_{\text{LM}}$)

Pour améliorer et régulariser le décodeur jouant un rôle majeur dans la prédiction de $y_u$, j'ajoute enfin un classificateur auxiliaire similaire à celui utilisé pour les modèles de langage. La fonction de perte est basée sur le critère d'entropie croisée avec lissage d'étiquette [Szegedy et al., 2016]. Cet ajout a aussi été proposé dans [Jeon and Kim, 2020] afin d'optimiser le décodeur indépendamment.

#### 9.1.2.5 Résultats actuels

Au moment de l'écriture de ce manuscrit, une étude préliminaire sur l'entraînement augmenté proposé a été effectuée pour différentes architectures, sur la base du corpus Voxforge italien décrit dans le chapitre 4. Le choix d'utiliser ce corpus pour la phase initiale est motivé par le fait que 1) des points de références existent pour différentes architectures [Karita et al., 2019, Pengcheng et al., 2021], 2) celui-ci est de taille raisonnable ($\tilde{2}$0 heures) et permet donc d'entraîner rapidement de nombreux modèles avec différents paramètres pour notre étude et 3) cela permet, parallèlement, aux utilisateurs de reproduire facilement les résultats avec peu de ressources matérielles. Les résultats obtenus sur le corpus de développement et le corpus de test sont reportés à titre indicatif dans le tableau 9.2.

| **Model** | **Aux. task** | **Dev** | **Test** |
|---|---|---|---|
| **Attention** | | | |
| RNN [Karita et al., 2019] | | 12.9 | 12.6 |
| Conformer [Pengcheng et al., 2021] | | **8.7** | **8.2** |
| **Transducer (ours)** | | | |
| RNN-T | | 13.0 | 12.4 |
| RNN-T | $\mathcal{L}_{\text{CTC}}$ | 12.6 | 11.8 |
| RNN-T | $\mathcal{L}_{\text{CTC}} + \mathcal{L}_{\text{LM}}$ | 12.6 | 11.7 |
| RNN-T | $\mathcal{L}_{\text{CTC}} + \mathcal{L}_{\text{aux-trans}} + \mathcal{L}_{\text{LM}}$ | 12.5 | 11.6 |
| RNN-T | all | 12.2 | 11.6 |
| Conformer-T | | 9.6 | 8.7 |
| Conformer-T | $\mathcal{L}_{\text{CTC}} + \mathcal{L}_{\text{LM}}$ | **8.5** | **7.8** |
| Conformer-T | all | 8.4 | 7.7 |

Table 9.2: Taux d'erreur de caractères (en %) sur l'ensemble de développement et test du corpus Voxforge italien pour des modèles avec différentes tâches auxiliaires.

Une analyse concernant l'influence sur les espaces de recherche dans le cadre des stratégies de décodage pour le modèle Transducer est aussi considérée. L'utilisation de ces fonctions

auxiliaires pour l'entraînement du modèle de RAP en français est aussi évaluée.

### 9.1.3 Mode en ligne

Outre la notion de performance en terme de taux d'erreurs, différents facteurs s'avèrent cruciaux pour le déploiement d'un système de RAP dans un contexte industriel, tel que : le coût de calcul de la reconnaissance en continu, le temps de traitement, ou encore la latence par rapport à la fin de parole.

Bien que les modèles bout-en-bout développés lors de cette thèse soit compétitifs en terme de facteur temps-réel (voir section 5.4.4), notamment le modèle Transducer, ceux-ci n'ont pas la capacité d'effectuer un décodage *en ligne* actuellement. L'audio étant envoyé à la fin du tour de parole, une latence significative est à prévoir, remettant en perspective la compétitivé des algorithmes de recherche présentés mais aussi leur utilisation à des fins industrielles. Une attention particulière est donc actuellement donnée au développement d'architecture de type RNN [He et al., 2019] et Conformer [Chen et al., 2020] permettant d'effectuer ce type de décodage. Une partie de ces travaux seront rendus disponibles dans la boite à outils ESPnet, toutefois la version définitive qui sera utilisée au sein d'Airudit sera omise.

Parallèlement, des travaux concernant l'utilisation de la *quantization* (ou quantification en français) [Kwasniewska et al., 2019] sont considérés, où l'ensemble, ou une partie, des opérations du modèle sur les tenseurs sont effectués avec des entiers plutôt qu'avec des valeurs en virgule flottante. Cela permet ainsi d'avoir une représentation plus compacte du modèle, réduisant l'empreinte mémoire nécessaire, mais aussi l'utilisation d'opérations vectorisées à haute performance sur des plateformes à faible ressources.

# Bibliographie


[Abeillé and Barrier, 2004] Abeillé, A. and Barrier, N. (2004). Enriching a french treebank. In *Proceedings of the International Conference on Language Resources and Evaluation*.

[Al-Zubaide and Issa, 2011] Al-Zubaide, H. and Issa, A. A. (2011). Ontbot: Ontology based chatbot. In *Proceeding of the International Symposium on Innovations in Information and Communication Technology*, pages 7–12.

[Altinok, 2018] Altinok, D. (2018). An ontology-based dialogue management system for banking and finance dialogue systems. *arxiv:1804.04838 (arXiv preprint)*.

[Amodei et al., 2016] 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., M. Chrzanowski, A. C., Diamos, G., Ding, K., Du, N., Elsen, E., Engel, J., Fang, W., Fan, L., Fougner, C., Gao, L., Gong, C., Hannun, A., Han, T., Johannes, L. V., Jiang, B., Ju, C., Jun, B., LeGresley, P., Lin, L., Liu, J., Liu, Y., Li, W., Li, X., Ma, D., Narang, S., Ng, A., Ozair, S., Peng, Y., Prenger, R., Qian, S., Quan, Z., Raiman, J., Rao, V., Satheesh, S., Seetapun, D., Sengupta, S., Srinet, K., Sriram, A., Tang, H., Tang, L., Wang, C., Wang, J., Wang, K., Wang, Y., Wang, Z., Wang, Z., Wu, S., Wei, L., Xiao, B., Xie, W., Xie, Y., Yogatama, D., Yuan, B., Zhan, J., and Zhu, Z. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of International Conference on Machine Learning*, pages 172–182.

[Anastasakos et al., 1996] Anastasakos, T., McDonough, J., Schwartz, R., and Makhoul, J. (1996). A compact model for speaker-adaptive training. In *Proceedings of International Conference on Spoken Language Processing*, pages 1137–1140.

[Andrusenko et al., 2020a] Andrusenko, A., Laptev, A., and Medennikov, I. (2020a). Exploration of end-to-end asr for openstt - Russian open speech-to-text dataset. In *Proceedings of the International Conference on Speech and Computer*, pages 35–44.

[Andrusenko et al., 2020b] Andrusenko, A., Laptev, A., and Medennikov, I. (2020b). Towards a competitive end-to-end speech recognition for chime-6 dinner party transcription. *arxiv:2004.10799 (arXv preprint)*.

[Arp et al., 2015] Arp, R., Smith, B., and Spear, A. D. (2015). *Building ontologies with basic formal ontology*. MIT Press.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arxiv:1409.0473 (arXiv preprint)*.

[Bahdanau et al., 2016] Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In

*Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4945–4949.

[Bahl et al., 1986] Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 49–52.

[Bahl et al., 1983] Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *Transactions on Pattern Analysis and Machine Intelligence*, pages 179–190.

[Baker, 1973] Baker, J. K. (1973). Machine-aided labeling of connected speech. *Working papers in speech recognition - II, Carnegie Mellon University.*

[Baker, 1975] Baker, J. K. (1975). The dragon system - An overview. *Transactions on Acoustics, Speech, Signal Processing*, pages 24–29.

[Battenberg et al., 2017] Battenberg, E., Chen, J., Child, R., Coates, A., Li, Y. G. Y., Liu, H., and Zhu, Z. (2017). Exploring neural transducers for end-to-end speech recognition. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 206–213.

[Baum and Eagon, 1967] Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *the Bulletin of the American Mathematical Society*, pages 360–363.

[Baum and Petrie, 1966] Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *the Annals of Mathematical Statistics*, pages 1554–1563.

[Baum et al., 1970] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *the Annals of Mathematical Statistics*, pages 164–171.

[Baum and Sell, 1968] Baum, L. E. and Sell, G. R. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, pages 211–227.

[Bellegarda and Nahamoo, 1990] Bellegarda, J. R. and Nahamoo, D. (1990). Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 2033–2045.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic programming.* Princeton University Press.

[Bello et al., 2018] Bello, A., Melgar, A., and Pizarro, D. (2018). Experti: A knowledge based system for intelligent service desks using free text. In *Proceeding of the International Conference on Information Theoretic Security*, pages 397–406.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Transactions on Neural Networks*, pages 157–166.

[Biing-Hwang, 1984] Biing-Hwang, J. (1984). On hidden markov models and dynamic time warping for speech recognition - An unified view. *AT&T Bell Laboratories Technical Journal*, pages 1213–1244.

[Biing-Hwang, 1985] Biing-Hwang, J. (1985). Maximum likelihood estimation for mixture multivariate stochastic observations of markov chains. *AT&T Bell Laboratories Technical Journal*, pages 1235–1250.

[Bilmes, 1997] Bilmes, J. (1997). A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov model. *Technical Report, University of Berkeley.*

[Bisani and Ney, 2005] Bisani, M. and Ney, H. (2005). Open vocabulary speech recognition with flat hybrid models. In *Proceedings of the Conference of the International Speech Communication Association*, pages 725–728.

[Boyer and Rouas, 2019] Boyer, F. and Rouas, J.-L. (2019). End-to-end speech recognition: A review for the French language. *arXiv:1910.08502 (arXiv preprint.*

[Béchet and Favre, 2014] Béchet, F. and Favre, B. (2014). Détection et caractérisation d'erreurs dans des transcriptions automatiques pour des systèmes de traduction parole-parole. In *Proceedings of Journées d'Etudes sur la Parole.*

[Bérard et al., 2018] Bérard, A., Besacier, L., Kocabiyikoglu, A. C., and Pietquin, O. (2018). End-to-end automatic speech translation of audiobooks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 6224–6228.

[Bérard et al., 2016] Bérard, A., Pietquin, O., Besacier, L., and Servan, C. (2016). Listen and translate: A proof of concept for end-to-end speech-to-text translation. In *Proceedings of NIPS Workshop on end-to-end learning for speech and audio processing.*

[Chan et al., 2016] Chan, W., Jaitlyn, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4960–4964.

[Chen and Goodman, 1998] Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, pages 359–394.

[Chen et al., 2020] Chen, X., Wu, Y., Wang, Z., Liu, S., and Li, J. (2020). Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. *arxiv:2010.11395 (arXiv preprint).*

[Chiu and Raffel, 2018] Chiu, C. and Raffel, C. (2018). Monotonic chunkwise attention. In *Proceedings of the International Conference on Learning Representations.*

[Cho et al., 2014] Cho, K., Merriënboer, B. V., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arxiv:10.3115 (arXv preprint).*

[Chorowski et al., 2014] Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: First results. In *in Proceedings of Neural Information Processing Systems: Workshop Deep Learning and Representation Learning Workshop.*

[Chorowski et al., 2015] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 577–585.

[Chorowski and Jaitly, 2017] Chorowski, J. and Jaitly, N. (2017). Towards better decoding and language model integration in sequence to sequence models. In *Proceedings of the Conference of the International Speech Communication Association*, pages 523–527.

[Cimiano et al., 2014] Cimiano, P., Unger, C., and McCrae, J. P. (2014). Ontology-based interpretation of natural language. In *Synthesis Lectures on Human Language Technologies*, page 178.

[Claeskens and Hjort, 2008] Claeskens, G. and Hjort, N. (2008). *Model selection and model averaging*. Cambridge University Press.

[Coral et al., 2010] Coral, C., Ruiz, F., and Piattini, M. (2010). *Ontologies for software engineering and software technology*. Springer.

[Dai et al., 2019] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

[Davis and Mermelstein, 1980] Davis, B. and Mermelstein, P. (1980). Comparison of parametric rrepresentations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 357–366.

[Davis et al., 1952] Davis, K., Biddulph, R., and Balashek, S. (1952). Automatic recognition of spoken digits. *the Journal of the Acoustic Society of America*, pages 637–642.

[Descout et al., 1986] Descout, R., Serignat, J. F., and Cervantes, O. (1986). BDSONS: Une base de données des sons du français. In *Proceedings of the International Congress on Acoustic*, pages 4–7.

[Devlin and M.-W. Chang, 2019] Devlin, J. and M.-W. Chang, K. Lee, K. T. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

[Dong et al., 2018] Dong, L., Xu, S., and Xu, B. (2018). Speech transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 5884–5888.

[Elfwing et al., 2018] Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Network*, pages 3–11.

[Esteve et al., 2010] Esteve, Y., Bazillon, T., Antoine, J. Y., Béchet, F., and Farinas, J. (2010). The epac corpus: Manual and automatic annotations of conversational speech in french broadcast news. In *Proceedings of the International Conference on Language Resources and Evaluation*.

[Fant, 1970] Fant, G. (1970). *Acoustic theory of speech production*. Walter de Gruyter.

[Ferrer-i-Cancho and Vole, 2002] Ferrer-i-Cancho, R. and Vole, R. V. (2002). Zipf's law and random texts. *Advances in Complex Systems*, pages 1–6.

[Furui, 1986] Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. In *Transactions on Acoustics, Speech and Signal Processing*, pages 52–59.

[Gales, 1998] Gales, M. (1998). Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech & Language*, pages 75–98.

[Gales, 2001] Gales, M. (2001). Factored semi-tied covariance matrices. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 779–785.

[Gales and Steve, 2008] Gales, M. and Steve, Y. (2008). Application of hidden markov models in speech recognition. *Foundations and Trends R in Signal Processing*, pages 195–304.

[Galliano et al., 2009] Galliano, S., Gravier, G., and Chaubard, L. (2009). The ester 2 evaluation campaign for the rich transcription of french radio broadcasts. In *Proceedings of the Conference of the International Speech Communication Association*, pages 2583–2586.

[Gauvain et al., 1994] Gauvain, J. L., Lamel, L., Adda, G., and Adda-Decker, M. (1994). The LIMSI continuous speech dictation systemt. In *Proceedings of the Workshop on Human Language Technology.*

[Giraud, 2015] Giraud, C. (2015). *Introduction to high-dimensional statistics.* Chapman & Hall.

[Graves, 2012a] Graves, A. (2012a). Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 37–45. Springer.

[Graves, 2012b] Graves, A. (2012b). Sequence transduction with recurrent neural networks. *arXiv:1211.3711 (arXiv preprint).*

[Graves, 2013] Graves, A. (2013). Generating sequences with recurrent neural networks. *arxiv:1308.0850 (arXiv preprint).*

[Graves et al., 2006] Graves, A., Fernandez, S., Gomez, F., and Schidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of International Conference on Machine Learning*, pages 369–376.

[Graves et al., 2013] Graves, A., Mohamed, A. R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 6645–6649.

[Gravier et al., 2004] Gravier, G., Bonastre, J.-F., Geoffrois, E., Galliano, S., McTait, K., and Choukri, K. (2004). The ester evaluation campaign for the rich transcription of french broadcast news. In *Proceedings of the International Conference on Language Resources and Evaluation.*

[Gravier et al., 2006] Gravier, G., Bonastre, J.-F., Geoffrois, E., Galliano, S., McTait, K., and Choukri, K. (2006). Corpus description of the ester evaluation campaign for the rich transcription of french broadcast news. In *Proceedings of the International Conference on Language Resources and Evaluation.*

[Gulati et al., 2020] Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition. In *Proceedings of the Conference of the International Speech Communication Association.*

[Gupta and P. Kenny, 2014] Gupta, V. and P. Kenny, P. Ouellet, T. S. (2014). I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 6334–6338.

[Hadian et al., 2018] Hadian, H., Sameti, H., Povey, D., and Khudanpur, S. (2018). End-to-end speech recognition using lattice-free mmi. In *Proceedings of the Conference of the International Speech Communication Association*, pages 12–16.

[Hariharan et al., 2001] Hariharan, M., Chee, L. S., Ai, O. C., and Yaacob, S. (2001). Classification of speech dysfluencies using lpc based parametrization techniques. *Journal of Medical Systems*, pages 1821–1830.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[He et al., 2019] He, Y., Sainath, T. N., Prabhavalkar, R., McGraw, I., Alvarez, R., Zhao, D., Rybach, D., Kannan, A., Wu, Y., Pang, R., Liang, Q., Bhatia, D., Shangguan, Y., Li, B., Pundak, G., Sim, K. C., Bagdy, T., Chang, S.-Y., Rao, K., and Gruenstein, A. (2019). Streaming end-to-end speech recognition for mobile devices. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 6381–6385.

[Hermansky et al., 2000] Hermansky, H., Ellis, D. P., and Sharma, S. (2000). Tandem connectionist feature extraction for conventional HMM systems. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 1635–1638.

[Hinton et al., 2012a] Hinton, G., Deng, L., Yu, D., Dahl, G., Abdel-rahman, M., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, pages 6645–6649.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep beliefs nets. *Neural Computation*, pages 1527–1554.

[Hinton et al., 2012b] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arxiv:1207.0580 (arXiv preprint).*

[Hochreiter, 1991] Hochreiter, S. (1991). Investigations on dynamic neural networks. *Diploma thesis.*

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, pages 1735–1780.

[Hori et al., 2017a] Hori, T., Watanabe, S., and Hershey, J. R. (2017a). Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 287–293.

[Hori et al., 2017b] Hori, T., Watanabe, S., Zhang, Y., and Chan, W. (2017b). Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. In *Proceedings of the Conference of the International Speech Communication Association*, pages 949–954.

[Hu and Zahorian, 2010] Hu, H. and Zahorian, S. A. (2010). Dimensionality reduction methods for HMM phonetic recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4854–4857.

[Hu et al., 2020] Hu, H., Zhao, R., Li, J., Lu, L., and Gong, Y. (2020). Exploring pre-training with alignments for RNN-transducer based end-to-end speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 7079–7083.

[Huang et al., 2001] Huang, X., Acero, A., Hon, H.-W., and Reddy, R. (2001). *Spoken Language Processing: A guide to theory, algorithm, and system development*. Prentice Hall PTR.

[Huang et al., 1993] Huang, X., Alleva, F., Hon, H., Hwang, M., Lee, K., and Rosenfeld, R. (1993). The sphinx-ii speech recognition system: An overview. In *Proceedings of the Workshop on Human Language Technology*, pages 137–148.

[Huang et al., 2014] Huang, X., Baker, J., and Reddy, R. (2014). A historical perspective of speech recognition. *Communications of the ACM*, pages 94–103.

[Huang et al., 1996] Huang, X., Hwang, M.-Y., Jiang, L., and Mahajan, M. (1996). Deleted interpolation and density sharing for continuous hidden markov models. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 885–888.

[Huang, 1992] Huang, X. D. (1992). Phoneme classification using semicontinuous hidden markov models. *IEEE Transactions on Signal Processing*, pages 1062–1067.

[Inaguma et al., 2019] Inaguma, H., Duh, K., Kawahara, T., and Watanabe, S. (2019). Multilingual end-to-end speech translation. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 570–577.

[Jaitly et al., 2015] Jaitly, N., Le, Q. V., Vinyals, O., Sutskever, I., and Bengio, S. (2015). A neural transducer. *arxiv:1511.04868 (arXv preprint)*.

[Jelinek, 1976] Jelinek, F. (1976). Continuous speech recognition by statistical methods. *IEEE Proceedings*, pages 532–556.

[Jeon and Kim, 2020] Jeon, J.-J. and Kim, E. (2020). Multitask learning and joint optimization for transformer-RNN-transducer speech recognition. *arxiv:2011.00771 (arXv preprint)*.

[Juang, 1984] Juang, B.-H. (1984). On the hidden markov model and dynamic time warping for speech recognition — a unified view. *AT & T Bell Laboratories Technical Journal*, pages 1213–1243.

[Juang and Rabiner, 2005] Juang, B. H. and Rabiner, L. R. (2005). Automatic speech recognition – A brief history of the technology development. In *Encyclopedia of Language and Linguistics*. Elsevier.

[Kannan et al., 2018] Kannan, A., Wu, Y., Nguyen, P., Sainath, T. N., Chen, Z., and Prabhavalkar, R. (2018). An analysis of incorporating an external language model into a sequence-to-sequence model. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 5824–5828.

[Karita et al., 2019] Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., Watanabe, S., Yoshimura, T., and Zhang, W. (2019). A comparative study on transformer vs RNN in speech applications. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 449–456.

[Katz, 1987] Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Transactions on Acoustics, Speech, Signal Processing*, pages 400–401.

[Kim and Stern, 2016] Kim, C. and Stern, R. M. (2016). Power-normalized cepstral coefficients (PNCC) for robust speech recognition. *IEEE Transactions on audio, speech and language processing*, pages 1315–1329.

[Kim and Yoonhan, 2020] Kim, J. and Yoonhan, L. (2020). Accelerating rnn transducer inference via one-step constrained beam search. *arxiv:2011.01576 (arXv preprint).*

[Kim et al., 2019] Kim, K., Lee, K., Gowda, D. N., Park, J., Kim, S., Jin, S., Lee, Y.-Y., Yeo, J., Kim, D., Jung, S., Lee, J., Han, M., and Kim, C. (2019). Attention based on-device streaming speech recognition with large speech corpus. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 956–963.

[Kim et al., 2017] Kim, S., Hori, T., and Watanabe, S. (2017). Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4835–4839.

[Kim and Seltzer, 2018] Kim, S. and Seltzer, M. (2018). Towards language-universal end-to-end speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4914–4918.

[Ko et al., 2012] Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. (2012). Audio augmentation for speech recognition. In *Proceedings of the Conference of the International Speech Communication Association*, pages 2345–2349.

[Koehn, 2005] Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation Summit*, pages 79–86.

[Konishi and Kitagawa, 2008] Konishi, S. and Kitagawa, G. (2008). *Information criteria and statistical modeling.* Springer.

[Kozielski et al., 2013] Kozielski, M., Rybach, D., Hahn, S., Schlüter, R., and Ney, H. (2013). Open vocabulary handwriting recognition using combined word-level and character-level language models. In *Proceedings of the Conference of the Statistical and Perceptual Audition Workshops and the Speech Communication with Adaptive Learning Consortium*, pages 8257–8261.

[Kudo, 2018] Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 66–75.

[Kudo and Richardson, 2018] Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*, pages 66–71.

[Kunze et al., 2017] Kunze, J., Kirsch, L., Kurenkov, I., Krug, A., Johannsmeier, J., and Stober, S. (2017). Transfer learning for speech recognition on a budget. In *Proceedings of Workshop on Representation Learning for NLP*.

[Kwasniewska et al., 2019] Kwasniewska, A., Szankin, M., Ozga, M., Wolfe, J., Das, A., Zajac, A., Ruminski, J., and Rad, P. (2019). Deep learning optimization for edge devices: Analysis of training quantization parameters. In *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society*, pages 96–101.

[Lamel et al., 1991] Lamel, L. F., Gauvain, J. L., and Eskenazi, M. (1991). Bref, a large vocabulary spoken corpus for french. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 505–508.

[Lee, 2010] Lee, K.-F. (2010). Context-independent phonetic hidden markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 599–609.

[Li et al., 2019] Li, J., Zhao, R., Hu, H., and Gong, Y. (2019). Improving rnn transducer modeling for end-to-end speech recognition. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 114–121.

[Ling et al., 2015] Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., and Luís, T. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

[Liu et al., 2020] Liu, C., Zhang, F., Le, D., Kim, S., Saraf, Y., and Zweig, G. (2020). Improving RNN transducer based ASR with auxiliary tasks. *arxiv:2011.03109 (arXiv preprint)*.

[Lu et al., 2020] Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., and Liu, T.-Y. (2020). Understanding and improving transformer from a multi-particle dynamic system point of view. In *Proceedings of the International Conference on Learning Representations*.

[Lugosch et al., 2019] Lugosch, L., Ravanelli, M., Ignoto, P., Tomar, V. S., and Bengio, Y. (2019). Speech model pre-training for end-to-end spoken language understanding. In *Proceedings of the Conference of the International Speech Communication Association*, pages 814–818.

[Luong and Vu, 2016] Luong, H.-T. and Vu, H.-Q. (2016). A non-expert kaldi recipe for vietnamese speech recognition system. In *Proceedings of the International Workshop on Worldwide Language Service Infrastructure*, pages 51–55.

[M. A. Basha Shaik et al., 2012] M. A. Basha Shaik, D. R., Hahn, S., Schlüter, R., and Ney, H. (2012). Hierarchical hybrid language models for open vocabulary continuous speech recognition using wfst. In *Proceedings of the Conference of the Statistical and Perceptual Audition Workshops and the Speech Communication with Adaptive Learning Consortium*, pages 46–51.

[Manohar et al., 2018] Manohar, V., Hadian, H., Povey, D., and Khudanpur, S. (2018). Semi-supervised training of acoustic models using lattice-free mmi. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4844–4848.

[McCrae et al., 2011] McCrae, J. P., Spohr, D., and Cimiano, P. (2011). Linking lexical resources and ontologies on the semantic web with lemon. In *Proceedings of the Extended Semantic Web Conference*, pages 245–259.

[Mermelstein, 1976] Mermelstein, P. (1976). Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, pages 374—-388.

[Messina and Kermovant, 2014] Messina, R. and Kermovant, C. (2014). Over-generative finite state transducer n-gram for out-of-vocabulary word recognition. In *Proceedings of the International Workshop on Document Analysis Systems*, pages 212–216.

[Messner et al., 2018] Messner, E., Zöhrer, M., and Pernkopf, F. (2018). Heart sound ssgmentation - An event detection approach using deep recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, pages 1–1.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*.

[Mikolov et al., 2011] Mikolov, T., Kombrink, S., Burget, L., Černocký, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 5528–5531.

[Mikolov and Zweig, 2012] Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. In *Proceedings of IEEE Spoken Language Technology Workshop*, pages 234–239.

[Mohamed et al., 2009] Mohamed, A., Dahl, G., and Hinton, G. (2009). Deep belief networks for phone recognition. In *Proceedings of the International Conference on Neural Information Processing Systems*, page 39.

[Mohri et al., 1996] Mohri, M., Pereira, F., and Riley, M. (1996). Weighted automata in text and speech processing. In *Proceedings of the European Conference on Artificial Intelligence*.

[Mohri et al., 2000] Mohri, M., Pereira, F., and Riley, M. (2000). The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, pages 17–32.

[Mohri et al., 2002] Mohri, M., Pereira, F. C., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. In *Computer Speech and Language*, pages 69–88.

[Ney et al., 1994] Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, pages 1–38.

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, pages 1345–1359.

[Park et al., 2019] Park, D. S., Chan, W., Zhang, Y., Cheng, C.-C., ad E. D. Cubuk, B. Z., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. In *Proceedings of the Conference of the International Speech Communication Association*, pages 2613–2617.

[Peddinti et al., 2015] Peddinti, V., Povey, D., and Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In *Proceedings of the Conference of the International Speech Communication Association*, pages 3214–3218.

[Pengcheng et al., 2021] Pengcheng, G., Boyer, F., Xuankai, C., Hayashi, T., Higuchi, Y., Inaguma, H., Kamo, N., Chenda, L., Garcia-Romero, D., Shi, J., Shi, J., Watanabe, S., Wei, K., Zhang, W., and Zhang, Y. (2021). Recent developments on espnet toolkit boosted by conformer. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

[Pierce, 1969] Pierce, J. R. (1969). Whither speech recognition? *The Journal of the Acoustic Society of America*, pages 1049–1051.

[Ping et al., 2019] Ping, W., Peng, K., and Chen, J. (2019). ClariNet: Parallel wave generation in end-to-end text-to-speech. *arxiv:1807.07281 (arXiv preprint)*.

[Popel and Bojar, 2018] Popel, M. and Bojar, O. (2018). Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, pages 43–70.

[Povey et al., 2010] Povey, D., Burget, L., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Glembek, O., Goel, N., Karafiat, M., Rastrow, A., Rose, R. C., and Schwarz, P. (2010). Subspace gaussian mixture models for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 4330–4333.

[Povey et al., 2011] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., and Silvosky, J. (2011). The kaldi speech recognition toolkit. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 1–4.

[Povey et al., 2018] Povey, D., Hadian, H., Gharemani, P., Li, K., and Khudanpur, S. (2018). A time-restricted self-attention layer for asr. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 5874–(8878.

[Povey et al., 2016] Povey, D., Peddinti, V., Galvez, D., Gharemani, P., Manohar, V., Na, X., and Khudanpur, S. (2016). Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Proceedings of the Conference of the International Speech Communication Association*, pages 2751–2755.

[Prabhavalkar et al., 2017] Prabhavalkar, R., Rao, K., Sainath, T., Li, B., Johnson, L., and Jaitly, N. (2017). A comparison of sequence-to-sequence models for speech recognition. In *Proceedings of the Conference of the International Speech Communication Association*, pages 939–944.

[Rabiner and Juang, 1986] Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16.

[Rabiner, 1991] Rabiner, L. R. (1991). A tutorial on hidden markov models and selected applications in speech recognition. *IEEE Proceedings*, pages 82–97.

[Ramachandran et al., 2018] Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for activation functions. *arxiv:1710.05941 (arXiv preprint)*.

[Rao et al., 2017] Rao, K., Sak, H., and Prabhavalkar, R. (2017). Exploring architectures, data and units for streaming end-to-end speech recognition with RNN transducer. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 193–199.

[Rose et al., 2010] Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: Applications and theory*, pages 1–20.

[Sak et al., 2014] Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of the Conference of the International Speech Communication Association*, pages 338–342.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 43–49.

[Salakhutdinov and Hinton, 2009] Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 448–455.

[Saon et al., 2000] Saon, G., Padmanabhan, M., Gopinath, R., and Chen, S. (2000). Maximum likelihood discriminant feature spaces. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 1129–1132.

[Saon et al., 2020] Saon, G., Tüske, Z., and Audhkhasi, K. (2020). Alignment-length synchronous decoding for rnn transducer. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 7804–7808.

[Schneider et al., 2019] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. In *Proceedings of the Conference of the International Speech Communication Association*, pages 3465–3469.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, pages 2673–2681.

[Serdyuk et al., 2018] Serdyuk, D., Wang, Y., Fuegen, C., Kumar, A., Liu, B., and Bengio, Y. (2018). Towards end-to-end spoken language understanding. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 5754–5758.

[Shewalkar, 2019] Shewalkar, A. (2019). Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. *Journal of Artificial Intelligence and Soft Computing Research*, pages 235–245.

[Simonnet et al., 2018] Simonnet, E., Ghannay, S., Camelin, N., and Estève, Y. (2018). Simulating asr errors for training slu systems. In *Proceedings of the International Conference on Language Resources and Evaluation*.

[Simonnet et al., 2017] Simonnet, E., Ghannay, S., Y., N. C., Estève, and Mori, R. D. (2017). Asr error management for improving spoken language understanding. In *Proceedings of the Conference of the International Speech Communication Association*, pages 3329–3333.

[Stolcke, 2002] Stolcke, A. (2002). Srilm - An extensible language modeling toolkit. In *Proceedings of the Conference of the International Speech Communication Association*, pages 901–904.

[Stolcke et al., 2011] Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). SRILM at sixteen: Update and outlook. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*.

[Su and Kuo, 2019] Su, Y. and Kuo, C.-C. J. (2019). On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*, pages 151–161.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the International Conference on Neural Information Processing Systems*.

[Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Re-thinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

[Tomashenko et al., 2019] Tomashenko, N., Caubrière, A., Estève, Y., Laurent, A., and Morin, E. (2019). Recent advances in end-to-end spoken language understanding. In *Proceedings of the International Conference on Statistical Language and Speech Processing*, pages 44–55.

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180.

[Tripathi et al., 2019] Tripathi, A., Lu, H., Sak, H., and Soltau, H. (2019). Monotonic recurrent neural network transducer and decoding strategies. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, pages 944–948.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 5998–6008.

[Vergara-Martinez and Swaab, 2012] Vergara-Martinez, M. and Swaab, T. Y. (2012). Orthographic neighborhood effects as a function of word frequency: An event-related potential study. In *Psychophysiology*, pages 1277–1289.

[Viterbi, 1967] Viterbi, A. (1967). Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, pages 260–269.

[Voxforge.org, 2019] Voxforge.org (2019). Voxforge (italian). `http://www.voxforge.org/`. Last version: 10/25/2019.

[Wagner and Fischer, 1976] Wagner, R. A. and Fischer, M. J. (1976). The string-to-string correction problem. *Journal of the ACM*, pages 168–173.

[Waibel et al., 1989] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 328–339.

[Wang et al., 2019] Wang, Y., Fan, X., Chen, I.-F., Liu, Y., Chen, T., and Hoffmeister, B. (2019). End-to-end anchored speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 7090–7094.

[Wang et al., 2017] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R., Jaitly, N., Yang, Z., Xiao, Y., Ying, C., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. (2017). Tacotron: Towards end-to-end speech synthesis. In

*Proceedings of the Conference of the International Speech Communication Association*, pages 4006–4010.

[Watanabe et al., 2021] Watanabe, S., Boyer, F., Chang, X., Guo, P., Hayashi, T., Higuchi, Y., Hori, T., Huang, W.-C., Inaguma, H., Kamo, N., Karita, S., Li, C., Shi, J., Subramanian, A. S., and Zhang, W. (2021). The 2020 espnet update: New features, broadened applications, performance improvements, and future plans. In *Proceedings of the IEEE Data Science & Learning Workshop*.

[Watanabe et al., 2018a] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unni, Y., and Renduchintala, A. (2018a). Espnet: End-to-end speech processing toolkit. In *Proceedings of the Conference of the International Speech Communication Association*, pages 2207–2211.

[Watanabe et al., 2018b] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Yalta, N., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., and Ochiai, T. (2018b). Espnet: End-to-end speech processing toolkit. In *Proceedings of the Conference of the International Speech Communication Association*, pages 2207–2211.

[Welch, 2003] Welch, L. R. (2003). Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*.

[Weng et al., 2019] Weng, C., Yu, C., Cui, J., Zhang, C., and Yu, D. (2019). Minimum bayes risk training of rnn-transducer for end-to-end speech recognition. *arxiv:1911.12487 (arXiv preprint)*.

[Williams and Zipser, 1995] Williams, R. and Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. *In back-propagation: Theory, architectures and applications*, pages 433–486.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arxiv:1609.08144 (arXiv preprint)*.

[Xu et al., 2018] Xu, H., Gao, T., Wang, Y., Li, K., Goel, N., and Khudanpur, S. (2018). A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 5929–5933.

[Yeh et al., 2019] Yeh, C.-F., Mahadeokar, J., Mahadeokar, K., Wang, Y., Le, D., Jain, M., Schubert, K., Fuegen, C., and Seltzer, M. L. (2019). Transformer-transducer: End-to-end speech recognition with self-attention. *arxiv:1910.12977 (arXiv preprint)*.

[Young et al., 1994] Young, S., Odell, J., and Woodland, P. (1994). Tree-based state tying for high accuracy modelling. In *Proceedings of the Workshop on Human Language Technology*.

[Young and Woodland, 1994] Young, S. and Woodland, P. (1994). State clustering in HMM-based continuous speech recognition. *Computer Speech and Language*, pages 369–394.

# Bibliographie personnelle


[Boyer et al., 2020] Boyer, F., Couraud, M., Lebas, P., and Sottiau, A. (2020). Procédé et dispositif d'obtention d'un réponse à partir d'une question orale posée à une interface homme-machine. *WIPO:WO2020260797*.

[Boyer and Rouas, 2019] Boyer, F. and Rouas, J.-L. (2019). End-to-end speech recognition: A review for the French language. *arXiv:1910.08502 (arXiv preprint)*.

[Boyer and Rouas, 2020] Boyer, F. and Rouas, J.-L. (2020). Evaluation of end-to-end ASR errors for the french language. *Unpublished (01/08/2021)*.

[Boyer et al., 2021] Boyer, F., Shinohara, Y., Ishii, T., Inaguma, H., and Watanabe, S. (2021). A study of transducer based end-to-end ASR with ESPnet: Architecture, auxiliary loss and decoding strategies. *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*.

[Martin et al., 2021a] Martin, V. P., Rouas, J.-L., Boyer, F., and Philip, P. (2021a). Automatic speech recognition systems errors for accident-prone sleepiness detection through voice. In *Proceedings of the European Signal Processing Conference*.

[Martin et al., 2021b] Martin, V. P., Rouas, J.-L., Boyer, F., and Philip, P. (2021b). Automatic speech recognition systems errors for objective sleepiness detection through voice. In *Proceedings of the Conference of the International Speech Communication Association*.

[Pengcheng et al., 2021] Pengcheng, G., Boyer, F., Xuankai, C., Hayashi, T., Higuchi, Y., Inaguma, H., Kamo, N., Chenda, L., Garcia-Romero, D., Shi, J., Shi, J., Watanabe, S., Wei, K., Zhang, W., and Zhang, Y. (2021). Recent developments on espnet toolkit boosted by conformer. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

[Watanabe et al., 2021] Watanabe, S., Boyer, F., Chang, X., Guo, P., Hayashi, T., Higuchi, Y., Hori, T., Huang, W.-C., Inaguma, H., Kamo, N., Karita, S., Li, C., Shi, J., Subramanian, A. S., and Zhang, W. (2021). The 2020 espnet update: New features, broadened applications, performance improvements, and future plans. In *Proceedings of the IEEE Data Science & Learning Workshop*.

# Annexe 1 : Brevet

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
30 décembre 2020 (30.12.2020)

WIPO | PCT

(10) Numéro de publication internationale
**WO 2020/260797 A1**

(51) Classification internationale des brevets :
*G06F 40/30* (2020.01) *G06F 16/332* (2019.01)

(21) Numéro de la demande internationale :
PCT/FR2020/051042

(22) Date de dépôt international :
17 juin 2020 (17.06.2020)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
1906997 27 juin 2019 (27.06.2019) FR

(71) Déposant : **EA4T** [FR/FR] ; ENSEIRB MATMECA, 1 Avenue du Docteur Albert Schweitzer, 33403 TALENCE CEDEX (FR).

(72) Inventeurs : **LEBAS, Philippe** ; 27 Hameau de Noailles, 33400 TALENCE (FR). **SOTTIAU, Antoine** ; 68 rue Lamartine, Apt 10, 33400 TALENCE (FR). **COURAUD, Mathilde** ; 3 rue Lafitte, Apt 10 - Résidence Hélios, 33400 TALENCE (FR). **BOYER, Florian** ; 1 rue Edison, 33400 TALENCE (FR).

(74) Mandataire : **BREESE, Pierre** ; IP TRUST, 2 rue de Clichy, 75009 Paris (FR).

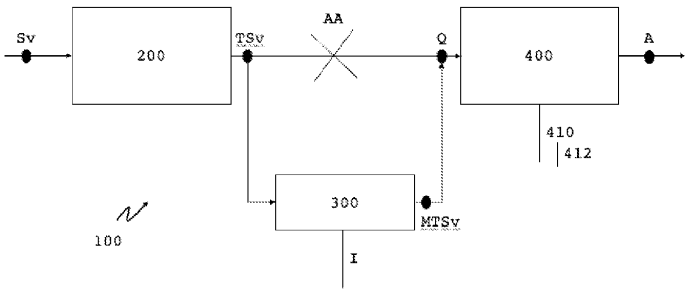
(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM,

WO 2020/260797 A1

(54) **Title:** METHOD AND DEVICE FOR OBTAINING A RESPONSE TO AN ORAL QUESTION ASKED OF A HUMAN-MACHINE INTERFACE

(54) **Titre :** PROCEDE ET DISPOSITIF D'OBTENTION D'UNE REPONSE A PARTIR D'UNE QUESTION ORALE POSEE A UNE INTERFACE HOMME-MACHINE

Fig. 1




(57) **Abstract:** The invention provides a method for modifying a voice-signal transcription TSv generated by an automatic speech recognition engine 200 from a voice signal Sv, prior to a generation of a response A by a semantic engine 400 equipped with at least one ontology comprising a vocabulary space, from an injection of a question Q, comprising the following steps: determining the set of all of the words of said transcription not belonging to said space, and for each word of said determined set: a step of word-embedding of the word in the space; determining semantic distances between the embedding and each of the words of said space; a step of selecting at least one word of said space; modifying M the transcription by replacing the word of the transcription with at least one selected word in order to generate a modified signal transcription MTSv.

(57) **Abrégé :** L'invention propose un procédé de modification d'une transcription de signal vocal TSv générée par un moteur de reconnaissance automatique de la parole 200 à partir d'un signal vocal Sv, préalablement à une génération d'une réponse A par un moteur sémantique 400, doté d'au moins une ontologie comportant un espace de vocabulaire, à partir d'une injection d'une question Q, comportant les étapes suivantes : détermination de l'ensemble des mots de ladite transcription n'appartenant pas audit espace, pour chaque mot dudit ensemble déterminé : une étape de plongement lexical du mot dans l'espace; une détermination de distances sémantiques entre le plongement et chacun des mots dudit espace; une étape de sélection d'au moins un mot dudit espace; une modification M de la transcription par remplacement du mot de la transcription par l'au moins un mot sélectionné pour générer une transcription de signal modifiée MTSv.

*[Suite sur la page suivante]*

PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

**(84) États désignés** *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Publiée:**

— *avec rapport de recherche internationale (Art. 21(3))*

# Description

PROCEDE ET DISPOSITIF D'OBTENTION D'UNE REPONSE A PARTIR D'UNE QUESTION ORALE POSEE A UNE INTERFACE HOMME-MACHINE.

## Domaine technique

La présente invention se rapporte à un procédé d'obtention d'une réponse à partir d'une question orale posée à une interface homme-machine. L'invention concerne également un dispositif configuré pour obtenir une telle réponse

## État de la technique antérieure

La reconnaissance automatique de la parole est une technique informatique qui permet d'analyser la voix humaine captée au moyen d'un microphone pour la transcrire sous la forme d'un texte exploitable par une machine. La reconnaissance de la parole, ainsi que la synthèse de la parole, l'identification du locuteur ou la vérification du locuteur, font partie des techniques de traitement de la parole. Ces techniques permettent notamment de réaliser des interfaces homme-machine IHM où une partie de l'interaction se fait à la voix. Ces interfaces peuvent être désignées par le terme d'interfaces vocales. La reconnaissance automatique de la parole est un domaine complexe, car il existe une différence importante entre le langage formel, qui est compris et utilisé par les machines, et le langage naturel, que les humains utilisent. Le langage formel est structuré par des règles syntaxiques strictes et sans ambigüité. À l'inverse, dans le langage naturel, des mots ou des phrases peuvent avoir plusieurs sens selon l'intonation de l'énonciateur ou le contexte par exemple.

On connaît des moteurs de reconnaissance automatique de parole, ci-après désignés par moteurs ASR, pour l'anglais *Automatic Speech Recognition*. Un signal de parole est injecté à un moteur de reconnaissance automatique de la parole. Dans le formalisme RAP (ASR en anglais), le découpage fonctionnel est le suivant :

- un traitement acoustique (*front-end* en anglais) permet d'extraire d'un signal vocal, aussi désigné sous le nom de signal de parole, une image acoustique compacte sous forme de vecteurs acoustiques correspondant à des tranches de 20 à 30 ms de signal avec un pas de 10 ms. Le signal est numérisé et paramétré par une technique d'analyse fréquentielle utilisant la transformée de Fourier (par exemple MFCC, pour l'anglais *Mel-Frequency Cepstral Coefficients*).

- un apprentissage automatique réalise une association entre les segments élémentaires de la parole et des éléments lexicaux. Cette association fait appel à une modélisation statistique entre autres par modèles de Markov cachés (HMM, pour l'anglais *Hidden Markov Models*) et/ou par réseaux de neurones artificiels (ANN, pour l'anglais *Artificial Neural Networks*).
- un décodage en concaténant les modèles élémentaires précédemment appris reconstitue le discours le plus probable. Il s'agit donc d'une correspondance de motif (*pattern matching*) temporelle, réalisée souvent par l'algorithme de déformation temporelle dynamique (DTW, pour l'anglais *dynamic time warping*).

Ainsi, un moteur ASR est ainsi doté d'un espace de vocabulaire, qui comporte tout le vocabulaire admis par ledit moteur ASR, c'est-à-dire les l'ensemble des éléments lexicaux précédemment désignés.

On connaît par ailleurs des moteurs d'analyse sémantique, ci-après désignés par moteurs MS. De tels moteurs sémantiques sont également désignés par le terme NLU system, pour l'anglais *Natural-Langage Understanding system*.

Les moteurs d'analyse sémantique ont pour objectif de générer une réponse suite à une question posée. A cet effet, les moteurs d'analyse sémantique comportent une ou plusieurs ontologies, et déterminent, lorsqu'une question est posée, l'ontologie à mettre en œuvre pour apporter une réponse à la question posée.

En informatique et en science de l'information, une ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms, ou les éléments d'un domaine de connaissances. L'ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Une ontologie est employée pour raisonner à propos des objets du domaine concerné. Les concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques ou des relations de subsomption.

Une ontologie permet de modéliser un ensemble de connaissances dans un domaine donné, qui peut être réel ou imaginaire. Les ontologies sont employées comme une forme de représentation de la connaissance d'un domaine ou d'une certaine partie d'un domaine, dans des domaines variés tels que ceux de l'intelligence artificielle, du Web sémantique, du génie logiciel, de l'informatique biomédicale ou encore de l'architecture de l'information.

Les ontologies décrivent généralement :

- classes : ensembles, ou types d'objets;

- individus : occurrence des classes, connaissances, objets ;
- attributs : propriétés, fonctionnalités, caractéristiques ou paramètres que les objets peuvent posséder et partager ;
- relations : les liens que les objets peuvent avoir entre eux ;
- événements : changements subis par des attributs ou des relations ;
- métaclasse (web sémantique) : des collections de classes qui partagent certaines caractéristiques.

Ainsi, un moteur MS, munie d'une ontologie, est doté d'un espace de vocabulaire associé à l'ontologie, qui comporte tout le vocabulaire admis par ledit moteur MS, c'est-à-dire l'ensemble des individus précédemment désignés de l'ontologie retenue par le moteur MS.

La problématique résolue par la présente invention s'inscrit dans la problématique générale d'apporter une réponse par un moteur MS à une question orale posée et numérisées sous forme de signal vocal.

Dans la présente description, une ontologie vise un ensemble de données techniques muni de classes formant les sommets d'un graphe et de relations formant les arrêtes entre les sommets du graphe, l'ontologie étant implémentée sous forme numérique. L'ensemble de données peut en outre comporter notamment des métadonnées, des indices syntaxiques, linguistiques.

L'ontologie peut être stockée numériquement sous forme de fichiers informatiques, par exemple au format XML.

Dans l'état de l'art antérieur, les procédés visant à répondre à la question générale comportent une étape de génération par un moteur ASR d'une transcription de signal vocal à partir du signal vocal puis une génération d'une réponse par un moteur MS à partir de la question formée par le signal vocal transcrit.

Chacun des moteurs ASR et MS étant dotés d'un vocabulaire indépendant de l'autre, on comprend que la chaîne de traitement mise en place ne peut fonctionner que pour le lexique commun aux deux vocabulaires.

Une solution immédiate qui peut être envisagée est de compléter chacun des vocabulaires des moteurs par le lexique connu seulement de l'autre des deux moteurs.

Toutefois, une telle solution est difficile à mettre en œuvre, car elle nécessite d'augmenter chacun des vocabulaires dès lors qu'un nouveau terme apparaît.

## Exposé de l'invention

Un but de l'invention est notamment de remédier à tout ou partie des inconvénients précités.

Selon un premier aspect de l'invention, il est proposé un procédé de génération d'une réponse à un signal vocal, de préférence mettant uniquement en œuvre des moyens techniques, à partir d'une transcription de signal vocal générée par un moteur de reconnaissance automatique de la parole à partir dudit signal vocal, comportant une génération de ladite réponse par un moteur sémantique à partir d'une question, le moteur sémantique étant doté d'au moins une ontologie comportant un espace de vocabulaire, la question étant générée à partir d'un prétraitement de la transcription du signal vocal, le prétraitement comportant, pour chacun des mots de ladite transcription du signal vocal n'appartenant pas audit espace de vocabulaire, une étape de plongement lexical dudit mot dans ledit espace de vocabulaire, une détermination d'une mesure de similarité entre ledit plongement lexical dudit mot et chacun des mots dudit espace de vocabulaire, une étape de sélection d'au moins un mot dudit espace de vocabulaire, le remplacement dudit mot de ladite transcription par ledit au moins un mot sélectionné pour former ladite question.

Le plongement lexical du mot n'appartenant pas audit espace de vocabulaire est un plongement lexical à la volée par composition. Un article de référence, incorporé à la présente demande est « Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation », de Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, Tiago Luís , page 1520-153 dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Le procédé de modification peut comporter une étape intermédiaire entre l'étape de plongement lexical du mot n'appartenant pas audit espace de vocabulaire et l'étape de sélection, ladite étape intermédiaire comportant une modélisation statistique des connexions sémantiques faites entre les termes utilisateurs et les concepts définis dans le moteur sémantique en se basant sur un corpus de requêtes utilisateurs. Dans ce cas les connexions sémantiques correspondent aux relations sémantiques entre les classes et individus de l'ontologie. L'utilisation d'un corpus de requête utilisateur permet à la fois de générer de nouvelles relations, mais également de pondérer ces relations. Le poids assigné à chaque relation peut être estimé par une méthode de remise, en se servant de la fréquence relative de chaque concept par exemple. De ce fait une relation rarement représentée dans les requêtes utilisateurs sera ainsi faiblement pondérée, et inversement. De préférence, la méthode de remise et d'interpolation entre les poids est déterminée de sorte qu'une relation avec une pondération très faible puisse être considérée.

Une pondération peut être obtenue par un modèle probabiliste simple. Si on considère pour la phase d'entraînement un ensemble de questions que des utilisateurs ont déjà posés à un système de dialogue homme machine, en utilisant un modèle de type word2vec, la pondération peut

tenir compte de la moyenne des vecteurs de mots pour chaque phrase afin de regrouper les phrases similaires. De cette manière, un modèle probabiliste peut être défini en calculant la probabilité d'occurrence de chaque phrase.

L'étape de sélection d'au moins un mot dudit espace de vocabulaire peut comporter un algorithme de sélection prenant en entrée les N-mots connus du moteur sémantique les plus proches sémantiquement du mot inconnu et proposant un ou des termes de remplacement étant donné plusieurs critères pilotables par l'utilisateur, comme par exemple la probabilité de la demande/question en considération de l'expérience utilisateur, et le coût estimé pour la résolution de la question/demande en considération des connexions sémantiques. Étant donné un modèle statistique des connexions sémantiques dont les relations sont pondérées, il est possible d'évaluer, en se basant sur la distance sémantique, un coût pour chaque phrase ayant un terme de remplacement en sommant les poids du chemin emprunté dans le modèle. D'autres évaluations de coût sont possibles.

Selon un deuxième aspect de l'invention, il est proposé un produit programme d'ordinateur, téléchargeable depuis un réseau de communication et/ou stocké sur un support lisible par ordinateur et/ou exécutable par un microprocesseur, et chargeable dans une mémoire interne d'une unité de calcul, caractérisé en ce qu'il comporte des instructions de code de programme qui, lorsqu'elles sont exécutées par l'unité de calcul, mettent en œuvre les étapes du procédé selon le premier aspect de l'invention, ou l'un ou plusieurs de ses perfectionnements.

Selon un troisième aspect de l'invention, il est proposé un procédé de génération d'une réponse à un signal vocal, comportant une étape de génération d'une transcription de signal vocal par un moteur de reconnaissance automatique de la parole à partir dudit signal vocal, une génération de ladite réponse par un moteur sémantique à partir d'une injection d'une question, le moteur sémantique étant doté d'au moins une ontologie comportant un espace de vocabulaire, le procédé comportant en outre une modification de ladite transcription de signal vocal selon le premier aspect de l'invention, ou l'un ou plusieurs de ses perfectionnements, pour obtenir une transcription de signal modifiée, ladite question injectée dans ledit moteur sémantique étant la transcription modifiée.

Selon un quatrième aspect de l'invention, il est proposé un produit programme d'ordinateur, téléchargeable depuis un réseau de communication et/ou stocké sur un support lisible par ordinateur et/ou exécutable par un microprocesseur, et chargeable dans une mémoire interne d'une unité de calcul, caractérisé en ce qu'il comporte des instructions de code de programme qui, lorsqu'elles sont exécutées par l'unité de calcul, mettent en œuvre les étapes du procédé de génération d'une réponse à un signal selon le troisième aspect de l'invention, ou l'un ou plusieurs de ses perfectionnements.

Selon un cinquième aspect de l'invention, il est proposé un dispositif de modification d'une transcription de signal vocal générée par un moteur de reconnaissance automatique de la parole à partir d'un signal vocal, préalablement à une génération d'une réponse par un moteur

sémantique à partir d'une injection d'une question, le moteur sémantique étant doté d'au moins une ontologie comportant un espace de vocabulaire Vms, le dispositif de modification étant configuré pour :

- déterminer l'ensemble des mots de ladite transcription de signal vocal n'appartenant pas audit espace de vocabulaire, ce qui peut s'écrire :
- pour chaque mot dudit ensemble déterminé :

i.effectuer un plongement lexical dudit mot dans ledit espace de vocabulaire,

ii. déterminer des distances sémantiques entre ledit plongement lexical dudit mot et chacun des mots dudit espace de vocabulaire,

iii. sélectionner au moins un mot dudit espace de vocabulaire,

iv. modifier ladite transcription par remplacement dudit mot de ladite transcription par ledit au moins un mot sélectionné pour générer une transcription de signal modifiée.

Selon un sixième aspect de l'invention, il est proposé un système de génération d'une réponse à un signal vocal comportant :

- un moteur de reconnaissance automatique de la parole configuré pour générer à une transcription de signal vocal à partir dudit signal vocal,
- un moteur sémantique configuré pour générer ladite réponse à partir en réponse à une injection d'une question, le moteur sémantique étant doté d'au moins une ontologie comportant un espace de vocabulaire,
- un dispositif conforme au cinquième aspect de l'invention, ou l'un ou plusieurs de ses perfectionnements, configuré pour générer une modification de ladite transcription de signal et injecter la transcription modifiée dans ledit moteur sémantique.

## Brève description des dessins

D'autres avantages et particularités de l'invention apparaîtront à la lecture de la description détaillée de mises en œuvre et de modes de réalisation nullement limitatifs, au regard de dessins annexés sur lesquels :

- La figure 1 représente schématiquement un système de génération d'une réponse à un signal vocal selon l'invention.
- La figure 2 représente schématiquement un procédé de génération d'une réponse à un signal vocal selon l'invention.

## Description détaillée

Les modes de réalisation décrits ci-après n'étant nullement limitatifs, on pourra notamment considérer des variantes de l'invention ne comprenant qu'une sélection de caractéristiques décrites, par la suite isolées des autres caractéristiques décrites, si cette sélection de

caractéristiques est suffisante pour conférer un avantage technique ou pour différencier l'invention par rapport à l'état de la technique antérieure. Cette sélection comprend au moins une caractéristique, de préférence fonctionnelle sans détails structurels, ou avec seulement une partie des détails structurels si cette partie uniquement est suffisante pour conférer un avantage technique ou pour différencier l'invention par rapport à l'état de la technique antérieure.

Sur les figures, un élément apparaissant sur plusieurs figures conserve la même référence.

Un mode de réalisation d'un système 100 ainsi qu'un mode de réalisation d'un procédé M de génération d'une réponse A à un signal vocal Sv sont décrits en même temps ci-après, en référence aux figures.

La figure 1 illustre un mode de réalisation d'un système 100 de génération de la réponse A au signal vocal Sv.

Le système 100 comporte un moteur de reconnaissance automatique de la parole 200 configuré pour générer, au cours d'une étape E1, une transcription de signal vocal Tsv à partir dudit signal vocal Sv.

On peut par exemple utiliser la boîte à outils Kaldi ou la collection de logiciels et outils CMU Sphinx pour construire, ou interagir avec, le moteur de reconnaissance de parole 200. Il est aussi possible, dans le cas de la réalisation d'un moteur de reconnaissance de parole dit bout-en-bout, d'utiliser les librairies informatiques d'apprentissage profond TensorFlow ou PyTorch ou de s'appuyer sur une boîte à outils tel que ESPNET proposant des méthodes d'entrainement pour différents modèles de ce type.

Le système comporte également un moteur sémantique 400 configuré pour générer, au cours d'une étape E2, la réponse A en réponse à une injection de question Q.

Le moteur sémantique 400 est doté d'au moins une ontologie Oms, référencée 410, comportant un espace de vocabulaire Vms, référencé 412.

On peut par exemple créer un système composé de : une ontologie composée avec Protégé, comprenant des connaissances à restituer ; une pile de traitement du langage naturel basé sur les outils Stanford NLP et un système d'apprentissage machine tel que BERT entrainé à associer une séquence de mots d'entrée aux connaissances possibles présentes dans l'ontologie.

Selon l'art antérieur, la transcription de signal vocal TSv est utilisé en tant que question Q pour être injectée dans le moteur sémantique 400 et obtenir la réponse A. Cette étape est modélisée par la flèche barrée AA.

A la différence de l'art antérieur, le système 100 comporte en outre un dispositif 300 de modification, au cours d'une étape M, de la transcription de signal vocal TSv, fonctionnellement interposé entre le moteur de reconnaissance automatique de la parole 200 et le moteur sémantique 400.

Plus précisément, le dispositif 300 reçoit en entrée la transcription de signal vocal TSv, générée par le moteur de reconnaissance automatique de la parole 200 à partir d'un signal vocal Sv.

Préalablement à la génération de la réponse A par le moteur sémantique 400 à partir de l'injection de la question Q, le moteur sémantique étant doté de l'ontologie 410 comportant l'espace de vocabulaire Vms, le dispositif de modification est configuré pour :

- déterminer l'ensemble I des mots de transcription de signal vocal TSv n'appartenant pas à l'espace de vocabulaire 412, ce qui peut se noter

$$I = \{x \in TSv, x \notin Vms\}$$

- pour chaque mot dudit ensemble déterminé :
  - effectuer un plongement lexical dudit mot dans ledit espace de vocabulaire,
  - déterminer des distances sémantiques entre ledit plongement lexical dudit mot et chacun des mots dudit espace de vocabulaire plongé dans ledit espace de vocabulaire,
  - sélectionner au moins un mot dudit espace de vocabulaire, cette sélection étant réalisée à partir des distances sémantiques déterminées,
- modifier ladite transcription TSv par remplacement dudit mot de ladite transcription par ledit au moins un mot sélectionné, pour former la transcription modifiée MTSv.

Un plongement d'un mot dans un espace de vocabulaire est une technique bien connue, aussi appelée *word embedding* en anglais.

Le *word embedding*, aussi appelé plongement de mot, ou encore plongement lexical, est une méthode d'apprentissage d'une représentation de mots utilisée notamment en traitement automatique des langues.

Cette technique permet de représenter chaque mot d'un dictionnaire par un vecteur de nombres réels correspondant. Ceci facilite l'analyse sémantique des mots. On distingue deux approches pour encoder le contexte (le voisinage) d'un mot. Les approches à base de fréquences dénombrent les mots cooccurrents avec un mot donné, puis distillent cette information pour créer des vecteurs denses et de petite dimension. Un exemple est l'analyse sémantique latente. Les plongements lexicaux constituent une seconde approche par laquelle on cherche à prédire un mot donné à l'aide de son contexte ou vice-versa. C'est ce processus qui engendre les

plongements lexicaux. Une implémentation populaire du calcul des plongements de mots est fournie par Tomas Mikolov dans son programme word2vec.

L'idée générale est de projeter un ensemble de mots d'un vocabulaire de taille V dans un espace vectoriel où les vecteurs de ces mots ont une taille N relativement petite. De plus, la représentation vectorielle de chaque mot de V doit être déterminée de sorte que les mots aux représentations voisines apparaissent dans des contextes similaires. L'approche du chercheur Thomas Mikolov s'appuie sur des réseaux de neurones artificiels pour construire ces vecteurs. Ces modèles sont entraînés sur des corpus très volumineux.

On peut par exemple utiliser le groupe de modèles word2vec ou GloVE, implémentables dans une librairie informatique d'apprentissage profond tel que TensorFlow ou PyTorch, pour réaliser tous les plongements lexicaux de l'invention. Dans le cadre de la réalisation des plongements lexicaux par composition, il est possible d'utiliser la bibliothèque Java Neural Network (JNN) mis en place par les auteurs ou de s'appuyer sur les librairies informatiques d'apprentissage profond précédentes pour une réalisation personnalisée. En particulier, le dispositif de modification de la transcription de signal vocal TSv générée par le moteur de reconnaissance automatique de la parole 200 à partir du signal vocal Sv peut être configuré pour déterminer, lors d'une étape initiale, les plongements lexicaux dans l'espace de vocabulaire Vms de chacun des mots de l'espace de vocabulaire Vms, par exemple en appelant autant de fois que nécessaire la commande word2vec.

**Exemple**

On support qu'un utilisateur produise le signal vocal Sv « famille des canidés », qui est correctement transcrit par le moteur de reconnaissance automatique de la parole 200 en une transcription de signal vocal TSv.

Usuellement, le signal TSv est adressé au moteur sémantique 400 en tant que question Q.

Lorsque le moteur sémantique 400 ne comporte pas le mot « canidés » dans son espace de vocabulaire Vms associé à l'ontologie utilisée, dans le cas présent une ontologie animalière, il n'est pas capable de traiter les mots.

Pour autant, le vocabulaire Vms associé à l'ontologie peut comporter les mots « Canidae », associés à d'autres mots, que sont les genres « Atelocynus Cabrera », « Canis Linnaeus », « Cerdocyon Hamilton-Smith », « Chrysocyon », etc.

Aussi, lorsqu'un utilisateur produit la question Q sous forme de transcription de signal vocal TSv « quels sont les genres de la famille des canidés », le moteur sémantique 400 ne peut pas apporter de réponse adéquate, alors même que les réponses sont connues.

Le procédé selon la présente invention comporte les étapes suivantes, lors de l'étape M :

- lors d'une étape initiale, on calcule, pour chacun des mots de l'espace de vocabulaire Vms, la résultante de la plongée lexicale du mot dans l'espace de vocabulaire, c'est-à-dire de chacun des mots « Canidae », mais aussi « Atelocynus Cabrera », « Canis Linnaeus », « Cerdocyon Hamilton-Smith », « Chrysocyon », etc.,
- on détermine l'ensemble I des mots de la transcription de signal vocal n'appartenant pas audit espace de vocabulaire : I = « canidés »,
- on plonge lexicalement par composition le mot « canidé » dans l'espace de vocabulaire,
- on détermine les distances sémantiques et/ou autres mesures de similarité entre ledit plongement lexical dudit mot et chacun des mots dudit espace de vocabulaire,
- on détermine un mot dudit espace de vocabulaire le plus proche, ici le mot « canidae »,
- on modifie la transcription par remplacement dudit mot de ladite transcription par ledit au moins un mot pour obtenir une transcription de signal modifiée MTSv : « quels sont les genres de la famille des canidae »

La transcription du signal modifiée est alors injectée en tant que question Q dans le moteur sémantique 400 et la réponse A adéquate est obtenue.

**Exemple**

On suppose qu'un utilisateur produise le signal vocal Sv « Où est-ce que je peux crécher pour pas cher ? », qui est correctement transcrit par le moteur de reconnaissance automatique de la parole 200 en une transcription de signal vocal TSv.

Usuellement, le signal TSv est adressé au moteur sémantique 400 en tant que question Q.

Lorsque le moteur sémantique 400 ne comporte pas le mot « crécher » dans son espace de vocabulaire Vms associé à l'ontologie utilisée, dans le cas présent une ontologie hôtelière, il n'est pas capable de traiter la requête.

Pour autant, le vocabulaire Vms associé à l'ontologie peut comporter les mots « dormir », « se restaurer », « loger » associés à la classe « service hôtelier ».

Aussi, lorsqu'un utilisateur produit la question Q associé au signal vocal Sv, le moteur sémantique 400 ne peut pas apporter de réponse adéquate, alors même que les réponses sont connues.

Le procédé selon la présente invention comporte les étapes suivantes, lors de l'étape M :

- Lors d'une étape initiale, on calcule, pour chacun des mots connus de l'espace de vocabulaire Vms, la résultante de la plongée lexicale du mot dans l'espace de vocabulaire, c'est-à-dire de chacun des mots « dormir », « se restaurer », « loger », etc.,
- On détermine l'ensemble I des mots de la transcription de signal vocal n'appartenant pas audit espace de vocabulaire : I = « crécher »,
- On effectue un plongement lexical par composition pour le mot « crécher » dans un espace de vocabulaire nommé Vunk,
- On effectue une projection des mots inconnus dans un espace commun (Vms ∪ Vunk) et on sélectionne les N-meilleurs candidats en considération de la distance sémantique et/ou de mesures de similarité,
- En considération de critères (temps, coût, performance) définis par l'utilisateur, le mot candidat est choisi. Dans cet exemple, le mot « dormir » sera retenu,
- On modifie la transcription par remplacement dudit mot de ladite transcription par ledit au moins un mot pour obtenir une transcription de signal modifiée MTSv : « Où est-ce que je peux dormir pour pas cher ? »

La transcription du signal modifiée est alors injectée en tant que question Q dans le moteur sémantique 400 et la réponse A adéquate est obtenue.

Bien sûr, l'invention n'est pas limitée aux exemples qui viennent d'être décrits et de nombreux aménagements peuvent être apportés à ces exemples sans sortir du cadre de l'invention. De plus, les différentes caractéristiques, formes, variantes et modes de réalisation de l'invention peuvent être associés les uns avec les autres selon diverses combinaisons dans la mesure où ils ne sont pas incompatibles ou exclusifs les uns des autres.

## Revendications

1. Procédé de modification (M) d'une transcription de signal vocal (TSv) générée par un moteur de reconnaissance automatique de la parole (200) à partir d'un signal vocal (Sv), préalablement à une génération d'une réponse (A) par un moteur sémantique (400) à partir d'une injection d'une question (Q), le moteur sémantique étant doté d'au moins une ontologie comportant un espace de vocabulaire, le procédé de modification comportant les étapes suivantes :
   - une détermination de l'ensemble des mots de ladite transcription de signal vocal n'appartenant pas audit espace de vocabulaire,
   - pour chaque mot dudit ensemble déterminé :
     i. une étape de plongement lexical dudit mot dans ledit espace de vocabulaire,
     ii. une détermination de distances entre ledit plongement lexical dudit mot et chacun des mots dudit espace de vocabulaire,
     iii. une étape de sélection d'au moins un mot dudit espace de vocabulaire selon les distances déterminées
     iv. une modification (M) de ladite transcription par remplacement dudit mot de ladite transcription par ledit au moins un mot sélectionné pour générer une transcription de signal modifiée (MTSv).

2. Produit programme d'ordinateur, téléchargeable depuis un réseau de communication et/ou stocké sur un support lisible par ordinateur et/ou exécutable par un microprocesseur, et chargeable dans une mémoire interne d'une unité de calcul, caractérisé en ce qu'il comporte des instructions de code de programme qui, lorsqu'elles sont exécutées par l'unité de calcul, mettent en œuvre les étapes du procédé de modification selon la revendication précédente.

3. Procédé (E) de génération d'une réponse (A) à un signal vocal (102), comportant une étape de génération (E1) d'une transcription de signal vocal (TSv) par un moteur de reconnaissance automatique (200) de la parole à partir dudit signal vocal, une génération de ladite réponse par un moteur sémantique (400) à partir d'une injection d'une

question (Q), le moteur sémantique étant doté d'au moins une ontologie comportant un espace de vocabulaire, le procédé comportant en outre une modification (M) de ladite transcription de signal vocal selon la revendication 1 pour obtenir une transcription de signal modifiée (MTSv), ladite question injectée dans ledit moteur sémantique étant la transcription modifiée.

4. Produit programme d'ordinateur, téléchargeable depuis un réseau de communication et/ou stocké sur un support lisible par ordinateur et/ou exécutable par un microprocesseur, et chargeable dans une mémoire interne d'une unité de calcul, caractérisé en ce qu'il comporte des instructions de code de programme qui, lorsqu'elles sont exécutées par l'unité de calcul, mettent en œuvre les étapes du procédé de génération d'une réponse à un signal selon la revendication précédente.

5. Dispositif (300) de modification d'une transcription de signal vocal (TSv) générée par un moteur de reconnaissance automatique de la parole (200) à partir d'un signal vocal (Sv), préalablement à une génération d'une réponse (R) par un moteur sémantique (400) à partir d'une injection d'une question (Q), le moteur sémantique étant doté d'au moins une ontologie (410) comportant un espace de vocabulaire Vms (412), le dispositif de modification étant configuré pour :
   - déterminer l'ensemble (I) des mots de ladite transcription de signal vocal n'appartenant pas audit espace de vocabulaire, ce qui peut s'écrire :
   - pour chaque mot dudit ensemble déterminé :
     i. effectuer un plongement lexical dudit mot dans ledit espace de vocabulaire,
     ii. déterminer des distances sémantiques entre ledit plongement lexical dudit mot et chacun des mots dudit espace de vocabulaire,
     iii. sélectionner au moins un mot dudit espace de vocabulaire,
     iv. modifier ladite transcription par remplacement dudit mot de ladite transcription par ledit au moins un mot sélectionné pour générer une transcription de signal modifiée (MTSv).

6. Système (100) de génération d'une réponse (A) à un signal vocal (Sv) comportant :
   - un moteur de reconnaissance automatique de la parole (200) configuré pour générer à une transcription de signal vocal (Tsv) à partir dudit signal vocal,

- un moteur sémantique (400) configuré pour générer ladite réponse à partir en réponse à une injection d'une question (Q), le moteur sémantique étant doté d'au moins une ontologie (410) comportant un espace de vocabulaire (412),
- un dispositif (300) conforme à la revendication précédente, configuré pour générer une modification (M) de ladite transcription de signal et injecter la transcription modifiée (MTSv) dans ledit moteur sémantique.

Fig. 1

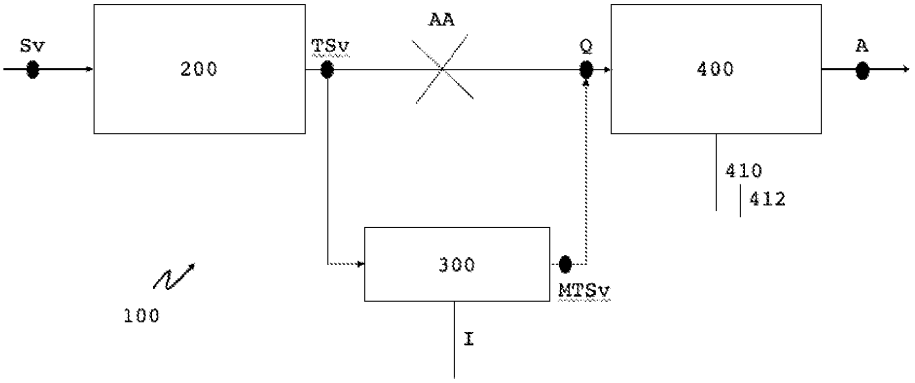
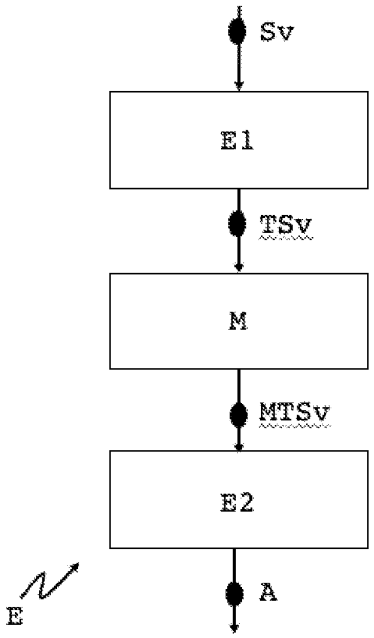



Fig. 2



# INTERNATIONAL SEARCH REPORT

International application No.
**PCT/FR2020/051042**

**A. CLASSIFICATION OF SUBJECT MATTER**

***G06F 40/30***(2020.01)i; ***G06F 16/332***(2019.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, COMPENDEX, INSPEC, IBM-TDB, WPI Data

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | US 2017185375 A1 (MARTEL MATHIEU JEAN [FR] ET AL) 29 June 2017 (2017-06-29)<br>paragraph [0027] - paragraph [0037]; figures 1, 2a<br>paragraph [0193] - paragraph [0222]; figures 7a,7b, 7c | 1-6 |
| A | VINDULA JAYAWARDANA ET AL. "Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population"<br>*INTERNATIONAL JOURNAL ON ADVANCES IN ICT FOR EMERGING REGIONS (ICTER)*,<br>Vol. 11, No. 1, 09 August 2018 (2018-08-09), page 1<br>DOI: 10.4038/icter.v11i1.7191<br>ISSN: 1800-4156, XP055674955<br>the whole document | 1-6 |

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| **29 July 2020** | **06 August 2020** |

| Name and mailing address of the ISA/EP | Authorized officer |
|---|---|
| **European Patent Office**<br>**p.b. 5818, Patentlaan 2, 2280 HV Rijswijk**<br>**Netherlands**<br>Telephone No. **(+31-70)340-2040**<br>Facsimile No. **(+31-70)340-3016** | **Lechenne, Laurence**<br>Telephone No. |

**INTERNATIONAL SEARCH REPORT**
**Information on patent family members**

International application No.
**PCT/FR2020/051042**

| Patent document cited in search report | Publication date (day/month/year) | Patent family member(s) | Publication date (day/month/year) |
|---|---|---|---|
| US 2017185375 A1 | 29 June 2017 | CN 108292203 A | 17 July 2018 |
| | | EP 3365767 A1 | 29 August 2018 |
| | | US 2017185375 A1 | 29 June 2017 |
| | | US 2019220245 A1 | 18 July 2019 |
| | | WO 2017112003 A1 | 29 June 2017 |

# RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2020/051042

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06F40/30 G06F16/332
ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)
EPO-Internal, COMPENDEX, INSPEC, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERES COMME PERTINENTS

| Catégorie* | Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents | no. des revendications visées |
|---|---|---|
| X | US 2017/185375 A1 (MARTEL MATHIEU JEAN [FR] ET AL) 29 juin 2017 (2017-06-29) alinéa [0027] - alinéa [0037]; figures 1, 2a alinéa [0193] - alinéa [0222]; figures 7a,7b, 7c | 1-6 |
| | ----- | |
| A | VINDULA JAYAWARDANA ET AL: "Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population", INTERNATIONAL JOURNAL ON ADVANCES IN ICT FOR EMERGING REGIONS (ICTER), vol. 11, no. 1, 9 août 2018 (2018-08-09), page 1, XP055674955, ISSN: 1800-4156, DOI: 10.4038/icter.v11i1.7191 le document en entier | 1-6 |
| | ----- | |

☐ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

| Date à laquelle la recherche internationale a été effectivement achevée | Date d'expédition du présent rapport de recherche internationale |
|---|---|
| 29 juillet 2020 | 06/08/2020 |

| Nom et adresse postale de l'administration chargée de la recherche internationale | Fonctionnaire autorisé |
|---|---|
| Office Européen des Brevets, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040,<br>Fax: (+31-70) 340-3016 | Lechenne, Laurence |



Formulaire PCT/ISA/210 (deuxième feuille) (avril 2005)

## RAPPORT DE RECHERCHE INTERNATIONALE

**Renseignements relatifs aux membres de familles de brevets**

**Demande internationale n°**
PCT/FR2020/051042

| Document brevet cité au rapport de recherche | Date de publication | Membre(s) de la famille de brevet(s) | Date de publication |
|---|---|---|---|
| US 2017185375 A1 | 29-06-2017 | CN 108292203 A | 17-07-2018 |
| | | EP 3365767 A1 | 29-08-2018 |
| | | US 2017185375 A1 | 29-06-2017 |
| | | US 2019220245 A1 | 18-07-2019 |
| | | WO 2017112003 A1 | 29-06-2017 |