Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Étude de la pertinence des évaluations des systèmes de reconnaissance automatique de la parole

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Ingénierie Multilingue

par

Lucie WARCKOL

Directeur de mémoire :

Frédérique Segond

Année universitaire 2019/2020

Remerciements

Je tiens à remercier particulièrement Mme Frédérique Segond pour m'avoir encadrée pour ce mémoire.

Je remercie également ma famille, mes parents et mes soeurs, qui m'ont soutenue et encouragée tout au long de ces mois de travail.

Je n'oublie pas Clémence, grâce à qui ces deux années d'études se sont déroulées dans la joie et le soutient mutuel.

Table des matières

LISTE DES FIGURES

RÉSUMÉ

La Reconnaissance Automatique de la Parole est devenue un outil incontournable dans les nouvelles technologies, et elle est de plus en plus utilisée dans des applications quotidiennes (automobile, domotique, aviation, administration, médical . . .). Ces applications étant de plus en plus pointues, spécifiques, exigeantes, les modèles de RAP doivent s'adapter pour accompagner cette évolution. Pour les rendre plus sophistiqués et plus robustes sont utilisées des méthodes d'évaluation qui doivent elles aussi s'adapter. Ce mémoire a pour objet d'étudier la pertinence de ces méthodes d'évaluation.

À travers l'état de l'art sont d'abord décrits les principaux modèles de RAP, les traditionnels et les modernes, ainsi que les principales méthodes d'évaluation correspondantes. Ensuite l'outil Kaldi a été testé sur deux corpus. Mais l'expérience a mis en évidence la complexité du processus, et ses résultats n'ont pas été probants.

En conclusion, les méthodes d'évaluation présentent encore des faiblesses (formatage des chiffres, entités nommées et ponctuation ...) face aux applications d'aujourd'hui, mais la démocratisation des processus et le développement des techniques d'optimisation contribuent à leur perfectionnement.

Mots-clés : reconnaissance automatique de la parole, méthodes d'évaluation des systèmes de RAP, Kaldi, WER

INTRODUCTION

Dans la vie quotidienne, de plus en plus d'outils numériques utilisent les transcriptions des systèmes de la Reconnaissance Automatique de la Parole (RAP) : sous-titrages TV, recherche vocale sur smartphones, GPS en voiture... L'utilisation de la voix permet d'économiser du temps et de gagner en productivité dans des situations oú les mains sont occupées ou plus lentes. La RAP présente de multiples avantages car la vue et les mains ne sont alors plus sollicitées pour l'utilisation des moyens de communications traditionnelles avec les machines (écran tactile, clavier, boutons...). De plus, les informations sont transmises plus rapidement par la voix que par la commande écrite ou le geste. Avec la voix tout est plus simple : le chirurgien en opération, le cuisinier dans sa recette, le conducteur au volant... Enfin, certains systèmes peuvent même réussir à comprendre mieux que l'oreille humaine ce qui est dit dans un environnement bruyant et dégradé.

Les modèles de la RAP sont de plus en plus précis quand ils sont évalués avec le *Word Error Rate* (WER) : un WER inférieur à 5% est atteint. Mais quand il s'agit d'utiliser les transcriptions produites pour des applications bien spécifiques, le taux d'erreurs augmente de façon significative. La raison : des erreurs de transcriptions gênantes subsistent. Des métriques adaptées aux différentes applications spécifiques permettraient de rendre les systèmes beaucoup plus robustes.

Comment adapter les méthodes générales d'évaluation des transcriptions produites par des systèmes de RAP à des applications quotidiennes particulières de plus en plus exigeantes ?

Pour le savoir, nous allons dans un premier temps dresser un état de l'art des systèmes de RAP, ensuite nous décrirons des expériences menées avec l'outil Kaldi, pour enfin discuter des résultats obtenus.

Avant d'aborder l'état de l'art, il semble important d'apporter les précisions suivantes :

— Le sujet traite de la reconnaissance de la parole, et non de la reconnaissance de la voix du locuteur ;
— Les termes techniques sont traduits en français et écrits « (en anglais en *italique*) » , et la liste de leurs abréviations se trouve en Annexe (Section A.1) ;
— Tous les modèles présentés dans ce mémoire sont décrits uniquement par rapport à la RAP (ces modèles sont utilisés dans tous les domaines du *Natural Language Processing* (NLP)) ;
— Il existe deux points de comparaison des modèles : ils sont comparés soit à une baseline, le modèle le plus basique à surpasser, soit aux modèles état de l'art, qui sont les modèles qui présentent les meilleurs résultats. Il existe deux types de modèles état de l'art : les modèles traditionnels qui utilisent des HMM, et

les modèles plus récents de types E2E ;
— Les définitions de certains termes sont insérés dans le texte entre {} quand ils sont mentionnés pour la première fois ;
— Les métriques d'évaluation font l'objet d'une partie spécifique (Chapitre 2) .

Première partie

Contexte général

CHAPITRE 1

ÉTAT DE L'ART

Sommaire

1.1 Introduction

Dans l'état de l'art sont décrits le système de la RAP traditionnelle, ensuite les systèmes modernes, enfin l'évaluation.

1.2 Les fondements de la RAP

Après avoir retracé les grands principes de la RAP, un petit rappel historique nous amènera à en décrire les modèles traditionnels.

1.2.1 Le Mécanisme de la RAP

Description de la RAP

Le principe de la RAP est de transcrire un énoncé oral. Dans le système de RAP, l'onde sonore produite par l'appareil vocal est transformée en caractéristiques (*features*). Ces dernières sont ensuite soumises à un algorithme de Machine Learning qui identifie les phonèmes et autres composants présents dans l'onde sonore. Enfin, on relie ces phonèmes pour former des mots, que l'on assemble ensuite pour

construire des phrases.
Le système de RAP prend en entrée un input sous la forme d'un fichier audio, et lui fait correspondre en sortie un output sous forme de transcription.
Il existe plusieurs types de paroles pour entraîner le système de RAP : plus la parole est difficile à percevoir, plus le modèle entrainé sera efficace.

- La parole lue : elle est peu coûteuse en production et facile à obtenir. Mais elle manque d'intonation, elle contient peu de variations dans la voix, et les locuteurs sont peu variés. On peut palier ce problème en enregistrant des lecteurs sous casque dont le son reproduit des bruits ambiants. On peut aussi superposer des bruits ambiants sur l'enregistrement ;
- La parole conversationnelle : il s'agit de l'enregistrement d'au moins deux personnes dans un environnement contrôlé qui discutent sur un sujet prédéterminé ;
- La parole spontanée : elle correspond à des enregistrements de locuteurs libres, sans guidage de mise en scène ni de sujet. C'est la parole la plus naturelle et la plus difficile à récolter, et donc la meilleure pour entraîner les modèles : intonations, vocabulaire, locuteurs, bruitages . . . ;
- La commande - contrôle : cet enregistrement permet de déclencher une action spécifique avec un seul mot ou expression, comme « ok google » de Google ou « alexa » de Amazon .

La RAP est un domaine complexe. Le langage formel, compris et utilisé par les machines, est très différent du langage naturel utilisé par les humains. Le langage formel est structuré par des règles très strictes non ambigues, alors que dans le langage naturel, des mots ou des phrases peuvent avoir plusieurs sens selon l'intonation du locuteur ou le contexte (https://fr.wikipedia.org/wiki/Reconnaissance_automatique_de_la_parole). Par exemple, la phrase « la petite brise la glace » peut signifier que la petite fille brise de la glace, ou que la brise du soir glace une personne qui a froid.

Les Applications de la RAP

La théorie de la RAP existe depuis des décennies, mais elle ne s'impose concrètement que depuis peu de temps dans notre quotidien grâce au Deep Learning qui a enfin rendu la RAP suffisamment précise pour être utile en dehors des environnements soigneusement contrôlés.
Les applications du système de la RAP sont multiples et variées, mais elles dépendent toutes de différents types de reconnaissance dont les principaux sont les suivants :

- La RAP en ligne [Jaitly et al., 2016], [Chiu and Raffel, 2017], [Raffel et al., 2017], [Moritz et al., 2020], [Moritz et al., 2019] : elle est utilisée par exemple pour le Call Center qui redirige la demande de l'interlocuteur téléphonique directement dans le bon service. On distingue la RAP en ligne de la RAP hors ligne. Dans la RAP en ligne, le mécanisme balaye tous les niveaux un par un, ligne par ligne, pour chaque élément de sortie, ce qui produit un affichage de sortie rapide. Dans la RAP hors ligne, le mécanisme regarde tous les niveaux pour chaque élément de sortie, ce qui engendre un lag plus important entre ce qui est dit en entrée et ce qui est produit en sortie ;
- La RAP continue [Jaitly et al., 2016], [Prabhavalkar et al., 2017], [Hori et al., 2017], [Chorowski et al., 2015], [Raffel and Ellis, 2015],

[Zeyer et al., 2018a], [Chiu et al., 2018], [Sak et al., 2017] : dictée vocale, recherche vocale, compte rendu de réunions et de cours magistraux... ;
— La RAP lue [Moritz et al., 2019], [Hori et al., 2017] : peut servir à entraîner des modèles basiques, quitte à bruiter ensuite la bande sonore ;
— La RAP de phonèmes [Jaitly et al., 2016], [Raffel et al., 2017], [Chorowski et al., 2015] : elle est utilisée dans les systèmes directement ;
— La RAP conversationnelle téléphonique [Moritz et al., 2019], [Hori et al., 2017] .

Les principales applications se retrouvent aussi bien dans la vie quotidienne que dans des domaines bien spécialisés. La dictée vocale, commercialisée à partir des 80's, est l'application la plus utilisée de la RAP. Elle permet de dicter oralement un texte retranscrit automatiquement par un processeur (https://www.authot.com/fr/2016/09/09/systeme-reconnaissance-de-la-parole/).

La RAP permet aussi de contrôler-commander un outil, de dicter des mots, de saisir des données. Elle peut être utilisée dans le domaine industriel (commande de machines ou de robots, préparation des commandes en logistique, contrôle qualité), automobile (GPS), avionique et domotique (température de la maison). Elle constitue également une aide intéressante pour la communication des personnes âgées ou handicapées (surdité) (https://www.archimag.com/vie-numerique/2019/02/06/reconnaissance-automatique-parole-commence-par-voix) ou les apprenants d'une langue. Elle peut être aussi utilisée dans tous les domaines administratifs (police, hôpital...) (https://www.archimag.com/demat-cloud/2018/03/05/rapports-administation-police-gendarmerie-efficacite) qui nécessitent une production énorme de documents individuels suivis et transmis dans de nombreux services différents, augmentant ainsi rapidité, productivité et efficacité . Sous titrage de vidéos, comptes rendus de réunions/consultations/cours magistraux , suivis hospitaliers et médicaux, prises de notes personnelles, retranscriptions de fichiers audio, enregistrements d'entretiens, envoie de SMS en respectant leur langage spécifique... (https://www.archimag.com/veille-documentation/2018/06/04/trois-applications-gratuites-retranscrire-automatiquement-fichiers) ceci en voiture ou dans tout autre milieu bruyant, enregistrements audio ou vidéos, courts ou longs, de bonnes ou mauvaises qualités sont autant d'exemples d'application de la RAP.

Les applications de la RAP sont donc bien diverses, et nécessitent chacune des systèmes dédiés. Par exemple la recherche par mots clés n'est pas comparable au sous titrage d'un enregistrement. De même, la transcription d'une séquence audio peut elle même refaire l'objet d'une réponse audio (*back end - front end*). Le type de RAP diffère selon la recherche du produit fini.

Même pendant la pandémie de COVID-19, la RAP peut être très utile pour éviter tout contact avec les surfaces, encore faut-il savoir bien gérer la voix alterée par le port du masque devant la bouche [Loukina et al., 2020].

Les Contraintes liées à la RAP

La grande difficulté de la RAP est d'engranger un maximum de données afin de balayer tous les cas de figures envisageables. Cette large diversité de données est nécessaire pour obliger le système de RAP à élargir son champ d'action et se renforcer. Le système d'évaluation doit alors améliorer sa qualité d'analyse pour évaluer correctement ce système de RAP encore plus robuste. Les contraintes inhérentes à la

diversité des données se situent aussi bien au niveau de l'input que de l'output.
La diversité des caractéristiques des enregistrements provient :

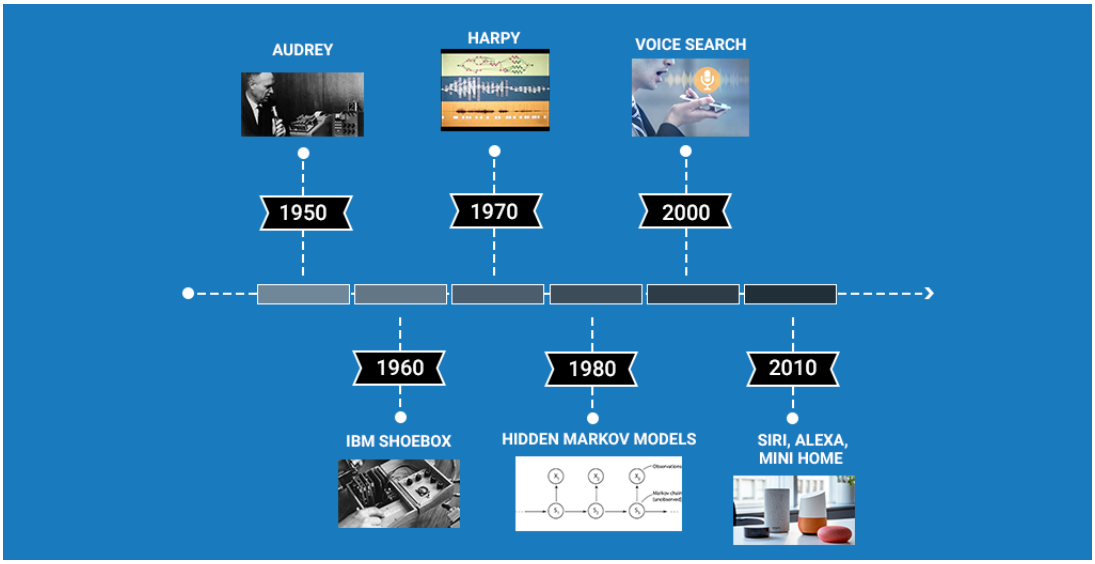
— du locuteur : sexe, age, handicap, taille, accent, personnalité... ;
— de la situation extérieure : bruit ambiant, locuteurs multiples, réverbération, échos... ;
— de l'action du locuteur : lecture, dialogue, conversation téléphonique, chant, commande vocale, recherche vocale, ordre donné... .

De plus, la diversité réside aussi plus précisément dans :

— la voix : vitesse de parole, prononciation, intonation, fréquence, puissance, effet Lombard (réaction du locuteur sur sa voix dans un milieu bruyant) [Vlaj and Kacic, 2011] ... ;
— la disfluence verbale : manifestations sonores indépendantes de la phrase, éléments non lexicaux, irrégularités verbales... (https://fr.wikipedia.org/wiki/Disfluence_verbale);
— les défauts de langage : bégaiements, tics de langage...(https://fr.wikipedia.org/wiki/B%C3%A9gaiement);
— le matériel : qualité du micro, format d'enregistrement du fichier audio. Les meilleurs formats audio sont FLAC ou WAV qui n'ont ni perte ni compression, même s'ils sont plus lourds à échanger et à stocker ;
— l'environnement sonore : environnement bruyant, endommagé et dégradé .

Enfin les propriétés linguistiques de la parole peuvent aussi poser problème, comme une écriture à deux prononciations et deux significations, comme est [ɛ]/[ɛst], fils [fis] / [fil], fier [fjɛʁ] / [fje], vis [vi] / [vis], éditions [editɔ̃] / [editjɔ̃], couvent [cuv(ə)] / [cuvã], content [cɔ̃tã] / [cɔ̃t(ə)] , convient [cɔ̃vi] / [cɔ̃vjɛ̃], parent [paʁã] / [paʁ(ə)], négligent [negliʒ(ə)] / [negliʒã], une écriture à une prononciation et deux significations, comme mousse [mus(ə)], bout [but(ə)], ainsi qu'une prononciation à plusieurs écritures comme [vɛʁ] vers, vert, verre, [dãs] danse, dense...
Les contraintes au niveau de l'output concernent le post traitement des transcriptions que l'on souhaite plus pertinentes, plus proches de l'écrit naturel et humain.
Le système spécialisé est évalué par un dispositif prévu pour un système plus généraliste. Les métriques d'évaluation ne sont donc pas assez pertinentes car elles ne dépassent pas le concept du WER alors qu'elles devraient être plus exigeantes. Une simple suite de mots sans ponctuation ni structure correspond à une évaluation WER. Dés lors que la transcription nécessite un formatage, comme la ponctuation, la syntaxe, les entités nommées, les séquences de chiffres, le WER n'est alors plus adapté. Par exemple, lors de la lecture de sous titres, le sens de la phrase dépend de sa ponctuation : son sens diffère si elle se termine par un point, un point d'exclamation ou un point d'interrogation. L'évaluation doit donc bien analyser la présence et la véracité de la ponctuation.
La pertinence de l'évaluation est d'autant plus importante qu'une mauvaise transcription peut provoquer un danger grave pour l'humain. Les conséquences d'une erreur sur le niveau sonore de la radio d'une voiture sont incomparables à une erreur de transcription de diagnostic médical qui peut se révéler fatal. L'évaluation devient alors la meilleure sécurité du système de la RAP.

1.2.2 Un peu d'Histoire...



La première machine de la RAP date de 1952 et s'appelle *Audrey*. Ses inventeurs sont les entreprises Bell Labs. Audrey est un système construit avec des circuits électroniques analogiques qui ne reconnaît que les nombres de 1 à 9. A cette époque le consensus de la communauté scientifique est que le signal audio doit être divisé en petites unités phonétiques pour ensuite rassembler ces unités en mots. Mais les résultats ne sont pas probants. En raison des mauvais résultats obtenus, la RAP devient un sujet tabou délaissé.
Á la fin des 70's, l'équipe DARPA créé le système *HARPY* [Lowerre, 1977] qui utilise 15000 nœuds interconnectés. Chaque nœud rassemble tout le vocabulaire d'un domaine spécifique. Ils utilisent un algorithme de recherche par « force brute » (Brut Force Search Algorithm) qui permet de lier l'audio au bon nœud pour obtenir la bonne transcription. Les résultats sont bons, mais IBM invente ensuite un autre système qui utilise le modèle de Markov caché (HMM : *Hidden Markov Model*) . Ce modèle représente les énoncés comme des états et prédit statistiquement avec des probabilités ce qui est un mot sachant les phonèmes qui le composent. Par exemple, si un mot peut être prononcé différemment avec une durée variable, le modèle capture l'élasticité du mot en utilisant une approche probabilistique. Le HMM est resté maître de l'art dans la RAP pendant les 80's et 90's, toujours amélioré. Pendant ce temps, les premiers tests avec l'approche des réseaux de neurones artificiels ne sont pas encore probants.
Á partir des 80's et jusque dans les années 2010, Jeoffrey Hinton continue les tests sur les réseaux de neurones, et les résultats surpassent enfin tout ce qui existe alors. Cette grande amélioration est due à l'utilisation d'une plus grande quantité de données, ainsi qu'à l'accès à une plus grande puissance de calcul. Cette approche est celle des réseaux de neurones profonds (DNN : *Deep Neural Network*). Le Deep Learning (DL) est ce qu'utilisent dorénavant tous les assistants vocaux comme Siri et Google Now.

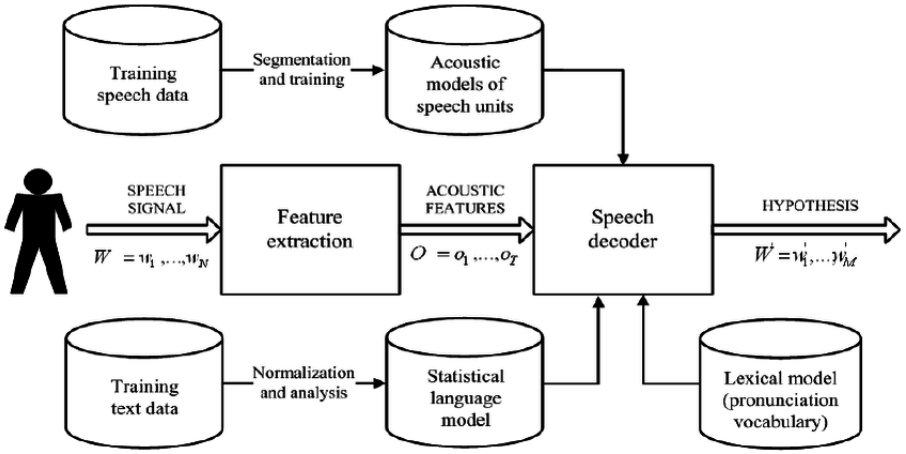
1.2.3 Les Modèles Traditionnels

Les anciens modèles traditionnels sont de type HMM-GMM (*Hidden Markov Model - Gaussian Mixture Model*). Les modèles hybrides plus récents sont en général

des HMM combinés avec des réseaux de neurones.
Les systèmes de RAP continue sont fondés sur une approche statistique proposée par [Jelinek, 1977] dont la formalisation est issue de la théorie de l'information.
Le but de ce système est de trouver la probabilité la plus forte de l'hypothèse $\hat{W}$ qui correspond à la probabilité la plus élevée d'une séquence de mots, sachant les observations acoustiques X, les modèles acoustiques, et le modèle de langage. C'est la formule de base des SRAP (Systèmes de Reconnaissance Automatique de la parole) (formule 1.1).
P(X | W) est la probabilité d'avoir X le signal de paroles sachant W la suite de mots prononcés (modèle acoustique). P(W) est la probabilité que la suite de mots générés soit valide (likelyhood) (modèle de langage), et P(X) la probabilité d'avoir le signal de parole acoustique X. P(X) est en général identique pour chaque suite de mots, et il est inutile pour déterminer la suite de mots la plus probable. P(X) sera donc ignorée.

$$\hat{W} = argmax P(W|X) = argmax \frac{P(X|W)P(W)}{P(X)} = argmax P(X|W)P(W) \tag{1.1}$$

Le HMM est un automate probabiliste contrôlé par deux processus stochastiques. Le premier est interne, caché, et se déplace d'état en état, et le deuxième génère les unités linguistiques correspondant à chaque état. La reconnaissance de la parole revient à choisir le HMM qui a la plus forte probabilité d'avoir émis le signal entrant. Une fonction de densité de probabilité est associée à chaque état du HMM, ce qui permet de relier un HMM aux vecteurs de paramètres acoustiques du signal de la parole. Les GMM sont utilisées pour approximer la densité de probabilité. Le but d'un HMM est de déterminer les paramètres optimaux selon le critère de vraisemblance maximale (*maximum likelyhood*). Ce qui signifie déterminer les paramètres maximisant la probabilité que ce qui a été observé soit généré par le modèle.
Les coefficients cepstraux de fréquence Mel (MFCC) sont une caractéristique largement utilisée dans les modèles de RAP traditionnels. Ils ont été introduits par [Davis and Mermelstein, 1980] dans les 1980's et sont toujours à la pointe de la technologie. Avant l'introduction des MFCC, les coefficients de prédictions linéaires (LPC) et les coefficients de prédictions cepstraux linéaires (LPCC) étaient les principales caractéristiques de la RAP, en particulier avec les classificateurs HMM.



Dans les modèles traditionnels, il existe plusieurs blocs (Traditional ASR pipeline) :

— Extraction des caractéristiques (*feature extraction*) : Ici on transforme le fichier audio en caractéristiques de la façon suivante : Le fichier audio, une onde sonore, est un vecteur 1D que l'on découpe en une multitude d'éléments.

On transforme chaque élément en un vecteur, qui représente la puissance de cet élément. Ces vecteurs tous rassemblés forment un spectrogramme dont les parties foncées correspondent aux parties parlées. Les éléments peuvent être adjacents ou se chevaucher ;

— Modèle acoustique (AM : *acoustic model*) : il apprend la relation entre la caractéristique du signal audio et ce que le locuteur dit. Il correspond au calcul de probabilité signe vocal – syllabe. A partir de la paramétrisation du signal et des caractéristiques des phones, il calcule leur statistique avec HMM et GMM, ce qui donne leur vraisemblance acoustique (*likelyhood*), leur probabilité d'observer le signal de parole X, étant donnée W, la suite de mots prononcés ;

— Modèle de prononciation : il associe à chaque mot une ou plusieurs séquences de phones correspondant à la ou les prononciations du mot ;

— Modèle de langage (LM : *language model*) : il représente la contrainte linguistique et correspond à la probabilité syllabe – mot . Entraîné sur du texte, il rassemble et comprend une grande partie des connaissances linguistiques écrites, et calcule ensuite les combinaisons et suites de mots les plus probables dans la langue à transcrire. Les probabilités des suites de mots sont calculées suivant l'approche des n-gram models qui sont simples et bien supportés (KenLM). Mais un mot écrit peut ne pas toujours correspondre à sa prononciation, le contexte est capital ;

— Décodeur (*decoder*) : il trouve la séquence de mots qui maximise la probabilité d'une séquence de mots particulière sachant le signal audio en optimisant le produit de la contribution du LM et du AM .

En 2012 est créé le Deep Belief Network (DBN) par [Mohamed et al., 2012] (*pre-training strategies*) (première génération de Deep Learning) qui devient le système état de l'art. En remplaçant le AM HMM-GMM, le système s'est amélioré de 5 à 10%. Le problème avec l'ancien modèle est que ses résultats n'évoluent plus malgré l'augmentation du nombre de données et de la puissance de calcul, alors qu'avec le Deep Learning, les résultats se sont améliorés de façon spectaculaire. C'est donc alors posée la question de pouvoir remplacer tous les modèles par des Deep Learning.

1.3 Les modèles récents : le système End to End

Le système *End to End* (E2E) est la nouvelle architecture de base qui consiste à avoir désormais un système à bloc unique sous la forme «input - bloc unique - output». Ce modèle générique concerne tous les domaines de l'apprentissage automatique, mais ici ne sont étudiés que les modèles appliqués au domaine spécifique de la RAP.

Ce modèle unique simple peut être entraîné from scratch, et opère directement sur les mots, les sous mots, ou les caractères/graphèmes. Ce modèle supprime le besoin d'un dictionnaire, ou d'un lexique de prononciation, et de toute modélisation explicite des phonèmes. Le réseau de neurones génère sa propre représentation des données sans devoir le forcer à utiliser la représentation des phonèmes. Il simplifie grandement le décodage. Le principe de ce système est d'entraîner un réseau de neurones tel qu'il suffit de donner en entrée ou input le fichier audio pour recevoir en sortie ou output directement la transcription sans avoir à créer des blocs comme dans le système traditionnel.

Cependant ne pas avoir des blocs écrits au préalable par l'humain peut aussi être un inconvénient : par manque de données, il peut s'avérer utile d'injecter des connais-

sances et des ressources extérieures supplémentaires pour aider le système. Typiquement le LM, qui est aussi un réseau de neurones qui apprend à partir d'un corpus de texte, apporte ses connaissances au modèle de RAP.

1.3.1 Réseaux de neurones et mécanisme d'Attention

Les réseaux de neurones récurrents

Le réseau de neurones est structuré en couches de nœuds connectés. Le lien entre un neurone et un autre possède un poids. Chaque neurone a un numéro unique (un biais). Ce biais est ajouté à la somme des inputs qui va rejoindre le neurone sur lequel on applique ensuite la fonction d'activation. Cette fonction détermine si un neurone sera activé ou non. Chaque neurone activé envoie des informations à la couche suivante jusqu'à l'avant dernière couche. Le neurone activé dans la dernière couche correspond à ce qu'on cherche au départ. Les poids et les biais s'ajustent en continu pour produire un réseau bien entraîné.
Il existe plusieurs types de réseaux de neurones qui ne sont pas tous utilisés pour les mêmes taches :

- Le format *one to one*, **un élément en entrée - un élément en sortie**, est utilisé par exemple pour la classification d'images : on donne en entrée une image, et le système donne en sortie une information unique (*label*). Par exemple, on donne une grande quantité d'images représentant un chien ou un chat en entrée, et le système renvoie en sortie le label « chien » ou « chat ».
- Le format *one to many* correspond à **un élément en entrée - plusieurs éléments en sortie**. Par exemple le sous titrage d'images, ou la description d'images : on donne une image en entrée, et le système donne en sortie une description de l'image sous forme de séquences de mots.
- Le format *many to one* correspond à **plusieurs éléments en entrée - un seul en sortie** : utilisé pour l'analyse des sentiments par exemple, on donne en entrée une séquence de mots, et en sortie il donne un mot pour définir si positif ou négatif, ou bien et mal : il donne la polarité de la phrase.
- Le format *many to many* correspond à **plusieurs éléments en entrée - plusieurs éléments en en sortie**. Il est utilisé notamment pour la RAP : une séquence de mots parlés est donnée en entrée et le système renvoie une séquence de mots écrits. Ce format est également connu sous le nom Séquence à Séquence (*Seq2Seq*).

Pour la RAP est utilisé essentiellement le réseau de neurones récurrents (RNN : *Recurrent Neural Network*). Dans ce type de réseau, le AM reprend toutes les propriétés physiques de la parole. On prend en entrée une onde sonore qui est une séquence temporelle naturellement récurrente. Puis le réseau de neurones traite des données séquentielles. Il est plus léger en termes de mémoire et de calcul , et correspond à un réseau de neurones récurrents. Pour chaque étape, le réseau de neurones sort une probabilité de chaque mot. Les mots aux probabilités les plus élevées sont regroupés pour former une transcription. Il est donc nécessaire ici d'utiliser beaucoup de données avec le plus de variations possibles.
En entrée l'onde sonore est convertie en spectrogramme à partir duquel sont extraites les caractéristiques. Le spectrogramme est découpé en fins segments qui sont convertis en vecteurs dont les coordonnées chiffrées correspondent aux informations du segment. Les informations sont alors vectorisées pour ne garder que les plus importantes et réduire ainsi le nombre de caractéristiques.

Voici quelques exemples de RNN reconnus dans la RAP.

Le *Neural Transducer* (NT) de [Jaitly et al., 2016] s'appuie sur le traitement traditionnel par bloc, avec une taille de fenêtre fixe et des intervalles, et produit des sorties de modèles d'Attention graduelle. Le modèle fait des prédictions progressives au fur et à mesure que les données arrivent. Le modèle est entraîné sur les données du TIMIT (*Texas Instrument Massachusetts Institut of Technology*), qui est un corpus de la parole transcrite phonémiquement et lexicalement. Le modèle est évalué sur la RAP en ligne et la reconnaissance de séquences longues. La RAP en ligne consiste à diminuer le lag entre l'input et l'output, visant à obtenir une transcription instantanée. Le modèle est composé de trois couches LSTM (*Long Short Term Memory*) unidirectionnelles, qui est l'encodeur, et d'un transducteur aussi composé de trois couches LSTM unidirectionnelles.
{Le LSTM est un type de réseau de neurones qui répond à la problématique de la mémoire : plus le mot est éloigné plus il faut de mémoire. Le LSTM permet d'augmenter cette mémoire. Sous réseau de RNN, il prend ensuite des décisions : à chaque étape, il décide de ce qu'il ne faut pas garder en mémoire, de ce qu'il faut changer, et de ce qu'il faut mettre dans chaque nœud. Le contenu des données de sortie est influencé non seulement par l'input mais aussi par tout l'historique des input à travers la boucle récurrente : la couche précédente est le premier paramètre de la couche suivante puisque la matrice est modifiée d'une couche à l'autre en prenant comme premier paramètre la couche précédente. Ce type de réseau retient l'information plus longtemps et obtient de meilleures performances. Le nombre de neurones existant par couche est déterminé. Par défaut, le réseau va faire de mauvaises prédictions ; par excès, il sera plus précis et donc risque de ne pas savoir généraliser (*overfitting*). Pour éviter cet *overfitting*, certains neurones sont désactivés de manière aléatoire pendant l'entraînement, pour l'obliger à trouver d'autres chemins entre les couches (*dropout*). Pour que le réseau se souvienne du début de la phrase quand il se trouve à la fin de la phrase, le système attribue au mot un vecteur qui est le mot précédent. En input est donné au réseau un vecteur et un mot. Il renvoie alors en sortie sa prédiction pour le mot suivant et une version modifiée mise à jour du vecteur. La boucle recommence alors en prenant en entrée le mot prédit à la sortie précédente, avec le vecteur mis à jour. Une fois le modèle entraîné, pour chaque mot donné au modèle, une partie du vecteur va contenir des informations grâce auxquelles il va pouvoir dire par exemple quel mot est le sujet de la phrase. Une autre partie du vecteur va garder en mémoire qu'il faut utiliser un autre mot pour cette phrase. Mais ce système fonctionne d'autant moins bien que la phrase est longue : l'information des premiers mots est effacée après un certain nombre de récurrences.}
Le LSTM est ici basé sur un mécanisme d'Attention. Les alignements produits par le modèle sont similaires aux alignements produits par un HMM-GMM. Plusieurs expériences ont été menées en modifiant l'architecture du modèle. Le meilleur résultat à la tache de reconnaissance de phonèmes est obtenu avec la combinaison d'un encodeur à trois couches et d'un transducteur à trois couches, avec 18% de PER (*Phoneme Error Rate*) (chapitre 2). Le modèle est précis en moyenne à 20% de PER, ce qui correspond aux performances des modèles état de l'art. Ce modèle présente l'inconvénient qu'il nécessite, pour obtenir une grande précision de reconnaissance, l'entraînement d'informations d'alignement d'un système RAP auxiliaire, et l'initialisation des paramètres d'un modèle pré-entraîné à séquences complètes.

Le *Deep Speech* de [Hannun et al., 2014] est un RNN composé de cinq couches cachées. Les trois premières ne sont pas récurrentes. La quatrième est une couche bidirectionnelle récurrente. La cinquième couche est non récurrente et prend en entrée la sortie de la couche précédente. La couche de sortie est une fonction Softmax standard. Elle donne les probabilités de caractères prédits pour chaque bande du spectrogramme sur une échelle de temps, ainsi que le caractère de l'alphabet. On applique alors la technique du *SpecAugment* [Park et al., 2019] {technique qui extrait des caractéristiques de façon aléatoire dans le temps et la fréquence. Ces extractions sont supprimées du spectrogramme, remplacées par une bande blanche. Le réseau de neurones est alors plus robuste car forcé à apprendre à faire de bonnes prédictions, avec des données imparfaites, ce qui le rend plus généralisé}. Ensuite est calculée la fonction de perte du *Connectionist Temporal Classification* (CTC) [Section 1.3.2] que l'on combine avec un LM 5-gram. Il est entraîné sur 220 M d'uttérances, soit un vocabulaire de 495 000 mots. Le *beam search* est appliqué pour améliorer la reconnaissance des caractères de façon plus précise.
{Le *beam search* permet de sélectionner à chaque étape les séquences les plus probables pour éviter une explosion exponentielle de combinaisons de mots. Par exemple, si le LM voit que le mot « read » existe dans la transcription, il va augmenter la probabilité des *beam search* qui contiennent le mot « read » comme reading au lieu de red, car cela a plus de sens. Ce processus va produire plus de transcriptions précises. Associer un LM au beam search permet d'injecter de l'information linguistique dans les sorties du AM, ce qui donne des transcriptions plus précises.}
Les corpus *Switch Board* (SwB) *Hub 5'00* et le *Fisher* (Fsh) sont utilisés comme données d'entrainement, ce qui correspond à 2300 heures de données totales. Malgré une quantité importante de données, le système est capable de faire un passage complet en seulement quelques heures. Le modèle est évalué en comparant un énoncé original à ce qui est produit en sortie, ce qui permet de calculer le WER. Les enregistrements audio sont pour la plupart sans bruits environnants. Pour y remédier et renforcer le modèle, une partie des données audio est modifiée en superposant des enregistrements de bruits environnants, ou en appliquant l'effet Lombard. Le modèle Deep Speech entraîné sur le corpus non modifié (propre) SwB et Fsh obtient 16% de WER quand il est testé sur le corpus complet Hub 5'00, comparé à la baseline qui obtient 18%. Le modèle est également évalué sur des données modifiées (ajout de bruitage) et obtient alors de 19% de WER . En comparaison sur les données bruitées, Apple Dictation obtient 43%, Bing Speech 36%, Google Api 30%, et WIT.AI 35%.

Le *Very Deep Convolutional Neural Network* de [Zhang et al., 2017] a pour but d'ajouter plus de puissance expressive et une meilleure généralisation. Le modèle sans LM est testé sur le corpus *Wall Street Journal* (WSJ). Certaines techniques provenant du domaine de la vision sont utilisées ici. Par exemple le *Network in Network* (NIN) qui permet d'augmenter la profondeur du réseau et qui réduit le nombre total de paramètres. La technique de *Batch Normalization* (BN) normalise les entrées de chaque couche pour réduire le décalage des covariables internes. BN permet d'accélérer l'entraînement. *Residual Networks* (ResNets) permet d'entraîner un réseau très profond sans souffrir d'une mauvaise optimisation ou généralisation. Et enfin le *Convolutional LSTM* (ConvLSTM) permet de maintenir des représentations structurées dans les états et les outputs. Cette technique permet aussi d'utiliser plus de

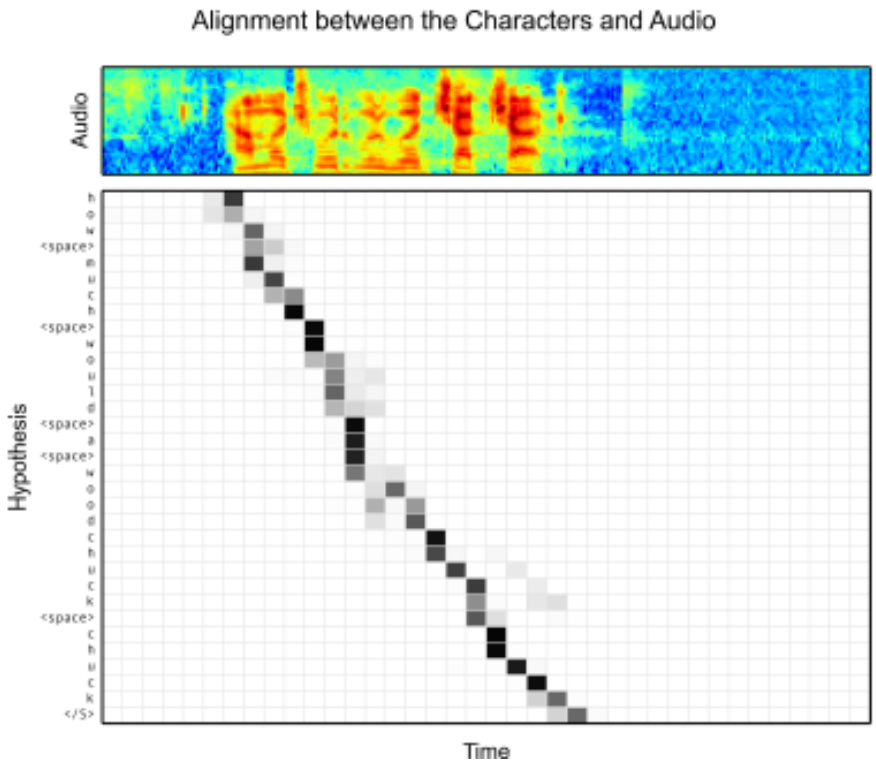


FIGURE 1.1 – Alignement entre le signal audio en entrée et les caractères en sortie [Chan et al., 2015]

puissance de calcul sans réduire le nombre de paramètres pour une meilleure généralisation. Le meilleur modèle testé est composé de deux couches convolutives de quatre blocs résiduels et d'un bloc NIN LSTM. Ce modèle obtient 10% de WER ce qui surpasse la baseline qui est à 14%, et le modèle état de l'art qui se situe à 18%.

Le mécanisme d'Attention

Les modèles de [Chorowski et al., 2015] basés sur Attention dans le domaine de la RAP se sont inspirés du *Attention Based Model* [Vaswani et al., 2017] appliqué à l'origine au domaine de la traduction automatique. L'une des premières expériences avec cet ancien modèle obtient de bons résultats (18% de PER) sur le corpus TIMIT mais présente des difficultés lorsqu'il est appliqué sur des uttérances plus longues que celles sur lesquelles il a été entraîné. Différents modèles ont été ensuite développés pour remédier à ce problème.

Description du mécanisme d'Attention

Dans une phrase, la couche d'Attention accède à tous les états précédents, les analyse, les pondère selon une mesure de pertinence pour calculer une probabilité et prendre une décision. Le mécanisme effectue l'alignement entre les trames acoustiques et les symboles reconnus (Figure 1.1). Il fait des prédictions basées sur toutes les prédictions précédentes. Il apporte des informations précises sur des tokens pertinents éloignés dans la phrase. Il répond ainsi au problème de mémoire du token éloigné et augmente la précision des prédictions.

Les différents types d'Attention

Il existe plusieurs types d'Attention ayant chacun une fonction particulière.

Le mécanisme de *Self Attention* [Povey et al., 2018] est l'Attention portée sur le mot lui-même en raison du poids plus important de la probabilité de se trouver lui même. Comme il faut annuler cet effet pour augmenter le poids des relations avec les autres mots, plusieurs vecteurs sont construits pour chaque mot. Ce processus est le *Multi Head Attention* (MHA). La moyenne de tous ces vecteurs est calculée pour obtenir le vecteur d'Attention final pour chaque mot.
Afin d'améliorer l'efficacité d'un modèle E2E de type *Transformer*, on a comparé les résultats de l'entraînement supervisé avec l'entraînement non supervisé en utilisant des algorithmes d'apprentissage de représentation qui soit quantifient explicitement les données audio, soit apprennent des représentations sans quantification. Les deux modèles de [Baevski and Mohamed, 2020] sont des versions modifiées du modèle *BERT* pour une tache de modélisation du langage masqué. Ils sont composés de 12 couches Transformer et 12 têtes d'Attention. Le modèle *vq-wav2vec* contient des encastrements à positions fixes à chaque bloc de MHA alors que le modèle *wav2vec* contient 128 encastrements positionnels relatifs à chaque bloc de MHA. Le décodeur *wav2letter++* est utilisé avec le LM 4-gram.
Tous les modèles sont pré-entraînés sur les 960 heures de données audio de l'ensemble d'entraînement du corpus *LibriSpeech*. Une mise au point est également effectuée sur l'ensemble d'entraînement supervisé à ressources limitées de 10 heures (24 locuteurs), 1 heure (24 locuteurs), et 1 minute (4 locuteurs). Les jeux d'entraînement sont échantillonnés à parts égales entre la partie propre et la partie bruyante du corpus. Ce travail s'inspire de la réduction de la dépendance à l'égard des données étiquetées à travers la découverte d'unités non supervisées, l'apprentissage par transfert multilingues et interlinguistiques dans des conditions de faibles ressources, et l'apprentissage semi-supervisé. Le modèle gardé après toutes les expériences est le *vq-wav2vec* avec un modèle BERT discret. Il obtient en 10 heures des résultats équivalents à d'autres modèles entraînés en 100 heures sur les mêmes données. La méthode la plus efficace consiste à apprendre d'abord un vocabulaire discret.
{Pour obtenir un vocabulaire discret, des étapes temporelles ne sont pas actives, elles sont cachées grâce à des tokens de probabilité très faible}.
Cette méthode est plus efficace que l'apprentissage direct à partir de données audio continues.

Le *Soft Attention* (SA) inspecte chaque entrée de la mémoire à chaque time step de sortie, ce qui permet au modèle de conditionner toute séquence d'entrée arbitraire. Mais il doit donc attendre que la séquence d'entrée soit traitée pour produire la sortie. Ceci représente un coût en temps et en mémoire et le rend inapplicable aux problèmes de transduction de séquences en temps réel.
Le SA est très long et coûteux en temps, et ne convient pas pour la RAP en ligne. Le *Hard Monotonic Attention* (HMA) [Raffel et al., 2017] passe d'un état à l'autre, de gauche à droite. A chaque étape de sortie, le décodeur inspecte la mémoire de gauche à droite (Figure 1.2). Il reprend le niveau suivant où il s'était arrêté au niveau précédent. Le changement de niveau s'effectue lorsque l'élément analysé est aligné sur l'élément de sortie. Deux modèles RNN avec HMA sont testés, l'un sur

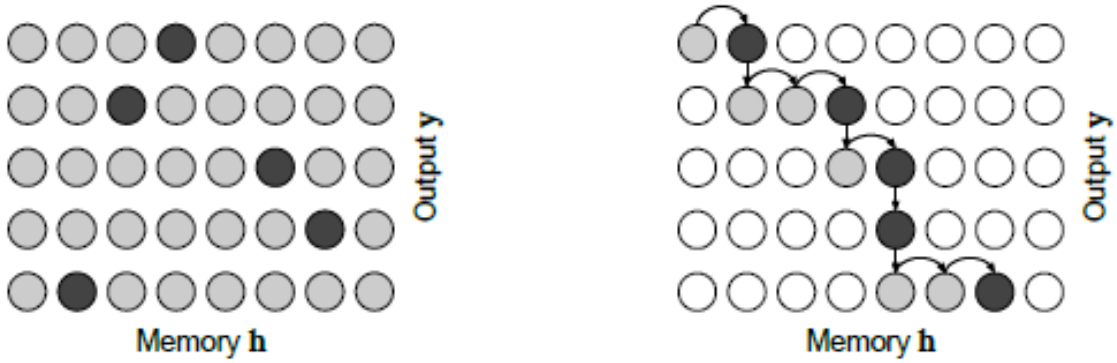


FIGURE 1.2 – Soft-Attention (gauche) - Hard Monotonic Attention (droit) [Raffel et al., 2017]

le corpus TIMIT et l'autre sur le corpus WSJ. L'architecture du modèle testé sur le corpus TIMIT est de type encodeur-décodeur.

L'encodeur est un RNN composé de trois couches LSTM unidirectionnelles, alternées avec des couches de réduction de temps. Le décodeur est un RNN composé d'une seule couche LSTM unidirectionnelle. En sortie, on obtient un des 60 phonèmes ou un token de début ou de fin de séquences. Le beam search est appliqué en plus du Softmax lors du test du modèle. La métrique d'évaluation est le PER. Le modèle testé utilisant le HMA obtient un PER de 20%, ce qui surpasse les performances des modèles utilisant BLSTM. La meilleure performance étant le modèle CTC qui obtient 19% de PER. La performance du modèle pour la RAP en ligne est équivalente aux performances des modèles pour la RAP hors ligne. L'architecture du modèle testé sur le corpus WSJ est de type encodeur-décodeur. L'encodeur est composé de huit couches dont deux convolutives, une LSTM unidirectionnelle convolutive et trois couches LSTM unidirectionnelles alternées avec des couches de projection linéaire. Le mécanisme d'Attention en ligne est ensuite appliqué sur la séquence de sortie de l'encodeur. Le résultat de cette opération est donné en entrée au décodeur constitué d'une couche LSTM unidirectionnelle suivie par une couche Softmax. Le décodeur produit en sortie un des 49 symboles (caractères alphanumériques, ponctuation, token de début ou de fin de séquences, inconnu, bruit ou délimiteur de mot). La métrique d'évaluation utilisée est le WER. Ce deuxième modèle testé obtient 17% de WER ce qui surpasse les performances des autres modèles pour la RAP en ligne et se rapproche des performances de modèles équivalents pour la RAP hors ligne.

Le *Monotonic Chunkwise Attention* (MoChA) [Chiu and Raffel, 2017] est une combinaison de *Soft Attention* et de *Hard Monotonic Attention*. MoChA vise à obtenir une transcription plus spontanée, plus rapide, sans décalage temporel. Il découpe les informations en chunk sur lesquels est appliqué l'Attention, permettant ainsi au décodeur de se référer aux états cachés dans l'encodeur qui sont utilisés comme mémoire. Ce processus permet au modèle d'obtenir une transcription plus spontanée et plus rapide. Le mécanisme MoChA utilise une couche supplémentaire par rapport aux deux autres types d'Attention. Elle permet de calculer une probabilité de sélections qui définit la longueur de la séquence de labels de sortie. Elle fournit également un alignement pour découper la séquence d'état de l'encodeur avant le mécanisme de Soft Attention. Les deux premières couches du système

sont des couches convolutives alternées avec un BN. La sortie de ces couches est ensuite donnée en entrée à une couche LSTM convolutive. Cette dernière couche est suivie d'une autre couche convolutive puis de trois couches LSTM unidirectionnelles. Ces dernières sont chacune alternées avec une couche dense. Le décodeur est une couche LSTM unidirectionnelle qui prend en entrée la sortie de la dernière couche de l'encodeur plus le vecteur de contexte du mécanisme d'Attention. Finalement, la couche de sortie Softmax prend en entrée la concaténation du vecteur de contexte du mécanisme d'Attention, et de l'état en cours du décodeur. Le décodeur produit alors en sortie une distribution sur une séquence de caractères. Les données d'entraînement proviennent du corpus WSJ composé d'enregistrements audio de lecteurs d'articles de journaux. La tache d'évaluation est la RAP en ligne. Le modèle utilisant MoChA obtient une moyenne de WER de 15%, dont le meilleur taux est 14%. Cette performance surpasse celle du modèle état de l'art.

Le modèle *Bidirectional Recurrent Neural Network* (BRNN) [Chorowski et al., 2015] utilise un mécanisme d'Attention hybride : *Content Based Attention* (CBA) et *Location Aware* (LA). Le CBA se focalise sur le mot en lui même avec son contenu, son sens, et le LA se focalise sur la position du mot dans la phrase. Le mécanisme d'Attention utilise l'alignement précédent pour sélectionner une liste courte d'éléments à partir de laquelle le CBA combiné au LA choisit les éléments pertinents de la liste sans confusion. Le meilleur modèle testé obtient 17% de PER par rapport à la baseline qui obtient 18% de PER. Le modèle état de l'art dans ce type d'architecture obtient aussi 17%.

Le modèle *Triggered Attention* (TA) [Moritz et al., 2019] est un classificateur basé sur CTC qui contrôle l'activation d'un réseau neuronal décodeur basé sur Attention. L'encodeur est composé d'un CNN suivi par un réseau BLSTM. Le modèle est construit avec deux décodeurs. L'un est un déclencheur basé sur un modèle CTC, l'autre est basé sur Attention. Ils prennent tous les deux en entrée l'output de l'encodeur. On utilise trois types d'Attention (Dot-Product, Content Based, Location Aware). Les deux premiers ne possèdent pas l'information de la distribution pondérée d'Attention de l'étape précédente. Le troisième type, lui, prend en compte la distribution pondérée d'Attention de l'étape précédente. Les modèles sont évalués sur trois corpus différents. Ces corpus sont de tailles différentes, allant de 90 à 960 heures d'audio, et de langues différentes (anglais et chinois mandarin). Les corpus sont le WSJ, LibriSpeech, et le corpus conversationnel téléphonique du HKUST. Le modèle entraîné sur le WSJ est évalué par rapport au *Character Error Rate* (CER) et au WER. Le modèle obtenant le CER le plus bas est le TALoc sans LM (Triggered Attention Location Aware Attention) avec 6% de CER. Ce même modèle avec un LM obtient 4% de CER ce qui est également le meilleur résultat parmi tous les modèles. Le même modèle sans LM obtient 16% de WER, et 7% avec LM. Le modèle obtenant la meilleure performance entraîné sur le HKUST est toujours le modèle TALoc sans LM avec 31%. Le meilleur modèle avec LM est le TACont (Content Based) qui obtient 30% au CER. Pour le corpus LibriSpeech, le meilleur modèle sans LM est le modèle TACont qui obtient 21% de CER et 19% de WER. Le meilleur modèle avec LM est le modèle TACont avec 19% de CER mais c'est le modèle Attention Loc qui obtient les meilleurs résultats avec 16% de WER. Ces résultats sont obtenus sur la partie bruitée du corpus LibriSpeech. Le système TA permet de réduire le délai de décodage de 80 ms.

Modèles Hybrides : RNN et Attention

Un modèle hybride [Hori et al., 2017] a été conçu pour la reconnaissance de langues telles que le japonais et le mandarin qui ne possèdent pas de mots ni de lettres mais des caractères typographiques. La métrique utilisée est donc le CER. Le décodeur basé sur Attention est entraîné avec un CTC dans un cadre d'apprentissage multitâches (MTL : *Multi Task Learning*). Ce décodeur est combiné à un *Recurrent Neural Network Langage Model* (RNN-LM). Le décodeur basé sur Attention et le CTC partagent un encodeur composé d'un *Deep Convolutional Neural Network* (DCNN) suivi par des couches BLSTM. Le modèle prend en entrée une séquence pour obtenir en sortie une séquence de labels. Les données proviennent de deux sources. D'une part le corpus *Corpus of Spontaneous Japanese* (CSJ) composé de speech data lu par un seul locuteur, comme par exemple des conférences universitaires, et des présentations simulées. Ce corpus compte 581 heures de données d'entraînement, et trois types de données de tests, où chaque test correspond à 5 heures. Ce corpus est utilisé pour la RAP lue. D'autre part le corpus HKUST (*Hong Kong University of Science and Technology*) MTS (*Mandarin Chinese conversational telephone speech recognition*) est composé de 167 heures de données d'entraînement, de 5 heures de données de développement, et de 5 heures de données de tests. Il est utilisé pour la RAP conversationnelle téléphonique. Pour les deux taches, reconnaissance du japonnais et celle du mandarin, on voit que le modèle basé sur Attention avec CTC obtient de meilleurs résultats que le modèle avec Attention utilisé en baseline. Pour la tache de la RAP lue, la meilleure architecture de modèle obtient entre 5% et 7% de CER. Pour la tache de la RAP conversationnelle téléphonique, la meilleure architecture obtient 28% de CER sans utiliser de ressources linguistiques, ce qui défie les systèmes de l'état de l'art . La baseline pour le mandarin est à 37%, et le meilleur modèle état de l'art est à 28%. Dans les systèmes traditionnels de reconnaissance du mandarin et du japonais, les ressources linguistiques sont des composants essentiels. Ce qui signifie que le modèle est bon s'il n'utilise pas ses ressources linguistiques.

A partir d'un modèle E2E hybride RNN avec Attention, différents éléments ont été modifiés afin d'en améliorer les performances. Le modèle [Zeyer et al., 2018b] est pré-entraîné avec un facteur de réduction du temps et un encodeur BLSTM alterné avec des couches de MaxPooling. Cette opération réduit le volume des données traitées par l'encodeur en n'en gardant que les informations les plus importantes. L'encodage des données se fait par *Byte Pair Encoding* (BPE) pour créer des unités de sous-mots. Le décodage avec le beam search passe par ces unités BPE puis sélectionne la meilleure hypothèse. Le décodeur est un LSTM qui rassemble les sous unités de mots en mots. Lors du beam search, un LM-LSTM est utilisé avec une couche d'entrée de projection et deux autres couches avec descente stochastique standard et découpage du *global gradient*. L'entraînement du LM est basé sur le comptage des n-gram (3-, 4- ou 5-gram). Les performances du modèle sont comparées aux performances d'autres modèles qui sont le modèle hybride HMM, le modèle CTC et le modèle avec Attention. Les données utilisées proviennent du corpus *LibriSpeech* qui compte 1000 heures de lecture audio. Une partie des données est également utilisée pour entraîner le modèle E2E. Le modèle étudié est comparé à des modèles hybrides et CTC. L'utilisation ou non d'un LM est prise en compte dans la comparaison, ainsi que l'unité sur laquelle on doit appliquer un label (CDP :

Context Dependant Phonem, caractère ou BPE). Le modèle E2E avec Attention obtient sur les données d'entraînement et de tests des résultats bien meilleurs que ceux obtenus par les autres modèles. Le modèle E2E est également entraîné sur les données du corpus *SwitchBoard*. L'utilisation ou non d'un LM est prise en compte dans la comparaison, ainsi que l'unité sur laquelle on doit appliquer un label (CDP, caractère, mot ou BPE). Le système E2E avec Attention obtient 11% de WER comparé à 8% de WER pour le système hybride HMM sur les données *SwitchBoard*. Le système obtient également un WER équivalent à celui obtenu par un modèle CTC état de l'art sur les données du corpus difficile *Call Home*. Cette nouvelle méthode utilisée pour pré-entraîner l'encodeur a permis d'améliorer les performances du modèle E2E avec Attention.

Le modèle *Listen Attend and Spell* (LAS) [Chan et al., 2015] est un réseau de neurones basé sur Attention qui construit sa propre représentation des données, il n'est pas obligé d'utiliser la représentation traditionnelle des phonèmes. *Listen* est un encodeur sous la forme d'un AM qui transforme le signal audio en représentation de haut niveau. Il présente une structure pyramidale utilisant un BLSTM avec une première couche BLSTM et trois couches pyramidales BLSTM. La partie *Attend and Spell* représente le décodeur de caractères basé sur Attention qui prend en entrée la représentation de haut niveau et qui produit une distribution de probabilités de séquences de caractères facilitée par un vocabulaire fini. Grâce au mécanisme de CBA, l'alignement entre les caractères prédits et l'audio est visible. Le décodeur est composé d'un transducteur de deux couches LSTM basées sur Attention. A chaque étape de sortie, le transducteur produit une distribution de probabilités du prochain caractère sachant tous les caractères vus jusque là sur laquelle un algorithme beam search est appliqué. Les données d'entraînement proviennent de *Google Voice Search Traffic*. Les données d'entraînement correspondent à 2000 heures d'audio, soit environ 3 M d'uttérances. Les données d'entraînement et de test sont divisées en jeux de données bruitées et non bruitées. Les performances du modèle sont comparées à un modèle hybride état de l'art pour la tache de recherche vocale. Le modèle état de l'art obtient 8% de WER sur le jeu de données propres, et 9% de WER sur le jeu de données bruitées. La meilleure configuration du modèle LAS utilisant un LM et une astuce d'échantillonnage obtient 10% de WER sur les données propres, et 12% de WER sur les données bruitées.

Les approches RAP E2E les plus populaires et les plus réussies sont basées sur le CTC [Graves et al., 2006], le RNN-T [Graves, 2012], ou les architectures encodeur-décodeur basées sur Attention [Chorowski et al., 2015]. Les systèmes RNN-T permettent d'obtenir des performances de pointe pour les applications en ligne. Les architectures encodeur-décodeur basés sur Attention sont cependant les systèmes de RAP E2E les plus performants [Vaswani et al., 2017]. Mais ils sont difficilement applicables au *streaming* ce qui freine son utilisation. Pour surmonter cette limite, différentes méthodes de Streaming qui génèrent un output directement après chaque mot prononcé, sont proposées avec des systèmes basés sur Attention, comme le NT [Jaitly et al., 2016], le MoChA [Chiu and Raffel, 2017], et le TA [Moritz et al., 2019]. Une autre méthode pour le streaming est un modèle E2E de type *Transformer* [Moritz et al., 2020]. L'encodeur de l'architecture *Transformer* est composé de deux couches *Convolution Neural Network* (CNN) et de plusieurs couches de *Self Attention*. Les couches de *Self Attention* sont composées de couches

MHA et de deux réseaux de neurones *feed-forward*, chacun séparé par une fonction d'activation. Pour contrôler le lag de l'encodeur, le contexte futur de la séquence d'entrée est limité à une taille fixe, appelée *Restricted Self Attention*, ou *Time Restricted Self Attention*. Cette technique a été appliquée pour la première fois sur un modèle hybride. Le processus force ensuite un alignement entre la séquence de l'encodeur et celle étiquetée avec le CTC. L'utilisation d'un RNN LM composé de quatre couches LSTM améliore la reconnaissance des tokens. Le corpus d'entraînement est *LibriSpeech*. Le modèle obtient 3% de WER sur les données de test propres, et 7% de WER sur les données de test bruitées. Ce sont les meilleurs résultats obtenus sur *LibriSpeech* pour un système de RAP en streaming.

Le modèle LAS [Chan et al., 2015] état de l'art a fait l'objet de modifications visant à améliorer sa structure et ses paramètres d'optimisation [Chiu et al., 2018]. Le modèle LAS est un réseau neuronal unique qui comprend un encodeur semblable à un AM traditionnel. *Attend* agit comme un modèle d'alignement. *Spell* est un décodeur semblable à un LM dans un système traditionnel. La structure du modèle est améliorée en utilisant des *Word Piece Models* (WPM) qui correspondent à des unités de sortie plus longues, ce qui permet d'obtenir un décodeur LM plus robuste. Le modèle mémorise aussi la prononciation des mots fréquents et les unités plus longues nécessitent alors moins d'étapes de décodage. La taille des WPM va de graphèmes aux mots entiers ce qui évite le problème du *Out Of Vocabulary* (OOV). Le modèle LAS utilise également le mécanisme MHA. Chaque tête génère une distribution d'Attention différente, ce qui permet au décodeur d'apprendre plus facilement à retrouver une information de l'encodeur. Dans l'architecture classique à une seule tête, le modèle s'appuie davantage sur l'encodeur pour fournir des signaux plus clairs sur les uttérances afin que le décodeur puisse capter l'information qui contient l'Attention. Le MHA présente l'avantage de mieux distinguer la parole du bruit quand la représentation encodée provient d'un enregistrement audio endommagé et dégradé.
Les paramètres d'optimisation du modèle sont améliorés en utilisant le *Minimum Word Error Rate* (MWER) dont le but est de minimiser le nombre d'erreurs par mot. Le décodeur est entraîné par le *Scheduled Sampling* (SS). Ce processus prélève des échantillons à partir de la distribution de probabilités de la prédiction précédente. Il utilise ensuite le token produit comme token d'entrée de la suite des opérations, pour prédire le label suivant. Le processus SS est une amélioration du mécanisme conventionnel *Teacher Forcing*.
{ Le principe du *Teacher Forcing* est de donner en entrée le label correct en tant que prédiction précédente. Il utilise l'output qui devrait être produit plutôt que l'output de l'état précédent.}
Le mécanisme SS utilise des probabilités et choisit lui même le token suivant. L'entraînement synchrone et asynchrone [Dean et al., 2012] est un autre paramètre d'optimisation amélioré. La réplique d'un modèle est la copie de ce modèle. Le mécanisme standard est l'entraînement synchrone avec un seul modèle, alors que l'entraînement asynchrone travaille sur plusieurs répliques d'un modèle. L'entraînement synchrone permet de fournir une meilleure qualité de modèles mais nécessite plus d'efforts afin de stabiliser l'entraînement du réseau de neurones. Il utilise l'accélération de la vitesse d'apprentissage, il fait tout l'entraînement d'un seul coup, alors que l'entraînement asynchrone utilise une montée en puissance graduelle. Le Labor Smoothing est un autre paramètre d'optimisation. C'est un mécanisme de régulation qui empêche le modèle de faire des prédictions trop confiantes, il le rend plus adapté. Le modèle

s'entraîne sur 12 500 heures d'audio, soit environ 15 M d'uttérances anonymisées et retranscrites à la main. Ces données proviennent de *Google Voice Search Traffic*. Le *Multicondition Training Data* (MTD) est ensuite appliqué sur une partie des données pour les complexifier. Les taches d'évaluation sont la recherche vocale et la dictée vocale. Concernant la tache de recherche vocale, le modèle final obtient 5% de WER, comparé au modèle hybride conventionnel état de l'art qui obtient 6%. Ces deux modèles sont comparés pour la tache de dictée vocale. Le modèle LAS proposé obtient 4% de WER, comparé au modèle hybride qui obtient 5%. Ces résultats sont corrects, mais le décodage d'un label nécessite l'encodage complet de l'uttérance, ce qui limite le système LAS ici proposé. Une amélioration du modèle serait alors d'utiliser un *Streaming Attention Based Model* [Jaitly et al., 2016], [Moritz et al., 2020].

1.3.2 Le Connectionist Temporal Classification

Le CTC [Prabhavalkar et al., 2017], [Hori et al., 2017], [Graves et al., 2006] est un algorithme qui permet de résoudre le problème d'alignement entre l'input et l'output qui sont respectivement un signal audio et une transcription. Cet algorithme est très important dans la fiabilisation d'un modèle.
Le problème d'alignement est que la longueur de l'input audio peut dépasser la longueur de l'output écrit. Un signal audio d'une certaine longueur, par exemple le mot « navire », est transformé en spectrogramme découpé ensuite en phonèmes par un RNN. Ce RNN produit une sortie qui est une distribution de probabilités sur tous les éléments du vocabulaire. Un son allongé peut être transcrit avec autant de phonèmes que sa longueur. La transcription du phonème correspondant est multipliée tant que dure la prononciation de ce phonème. Ce qui donne par exemple [navi : ːʁ] au lieu de [naviʁ].
Le vocabulaire est composé de l'ensemble des labels qui représentent les lettres de l'alphabet qui composent le mot. Pour transformer le mot et obtenir la transcription finale correcte, le CTC utilise, en plus de ce vocabulaire, un blank « <b> ». Ce symbole est placé avant les caractères censés être en double dans le mot pour garantir qu'ils ne restent que deux lettres ensuite. Sinon le caractère multiplié sans la présence du blank sera d'office réduit à un seul. Dans l'exemple, pas de symbole blank devant les [i] annulera les [i] supplémentaires. Le CTC décide de mettre le blank ou pas. Ensuite, selon les probabilités attribuées aux éléments de la séquence de sortie, plusieurs séquences probables contenant le blank ou non sont proposées. Le CTC enlève tous les caractères multipliés qui ne sont pas précédés d'un blank puis il retire les blanks et donne en sortie les différentes transcriptions les plus probables. Le CTC est calculé en faisant la somme des probabilités de tous les alignements possibles pour un input, le logarithme calcule ensuite la fonction de perte. Le CTC augmente donc la sécurité et la fiabilité des combinaisons de lettres.
Le CTC peut être utilisé comme base pour entraîner d'autres modèles comme le RNN-T [Graves, 2012], [Prabhavalkar et al., 2017], un modèle basé sur Attention [Prabhavalkar et al., 2017], et un RNNT avec Attention [Prabhavalkar et al., 2017].
Pour le RNN-T, la base du CTC est associée à un modèle de langage récurrent séparé qui prend en entrée la prédiction de l'étiquette du graphème précédent et qui calcule un vecteur de sortie dépendant de toute la séquence d'étiquette. La sortie de l'encodeur et celle du modèle de langage sont transmises à un réseau commun qui calcule les logits de sortie pour chaque entrée dans la séquence d'encodage ainsi que l'étiquette du réseau de prédiction. Le modèle avec Attention est un encodeur qui as-

socie l'acoustique de l'entrée sous la forme d'une représentation de haut niveau. Un décodeur basé sur Attention prédit le prochain symbole de sortie d'après la séquence entière précédente prédite. Il utilise également un jeu de labels augmenté par deux symboles : « sos » *Start Of Sentence* et « eos » *End Of Sentence*. Le RNN-T avec Attention contient une structure modifiée. Le réseau de prédiction est remplacé par un décodeur basé sur Attention et utilise seulement le jeu de labels augmenté avec « sos » car le processus de décodage se termine quand toutes les fenêtres d'encodage ont été traitées.
L'encodeur du modèle CTC est composé de cinq couches BLSTM. Les poids de cet encodeur sont utilisés pour initialiser les encodeurs des autres modèles Seq2Seq. Tous les encodeurs des modèles ont la même taille et la même configuration, mais leurs paramètres diffèrent après l'entraînement des modèles. Les modèles sont entraînés sur 12 500 h d'audio, ce qui correspond à environ 15 M d'uttérances. Ces données sont extraites de la recherche vocale pour *Google Voice Search Traffic*, et elles sont anonymisées et transcrites à la main (par l'humain). Pour rendre le système plus performant et plus robuste à tout ce qui est bruitage sont générées des MTR.
La tache de reconnaissance vocale utilisée est *Large Vocabulary Continuous Speech Recognition* (LVCSR) : la dictée présente de bons résultats, la recherche vocale présente des résultats plutôt mauvais (comparé à une baseline). Les ensembles d'entraînement sont les suivants : un ensemble de 13 000 uttérances, soit environ 124 000 mots, utilisé pour la tache de dictée vocale, un ensemble de 12 900 uttérances soit 63 000 mots utilisés pour la tache de recherche vocale, et un ensemble de 12 600 uttérances soit 75 000 mots qui correspondent à des instances de temps, de dates, ... où la transcription attendue correspond au domaine écrit des chiffres, pour lesquels il existe des conventions écrites. Par exemple à l'oral « twelve fifty one p m » doit être reconnu comme « 12 :51 p.m. ». En plus de ces trois ensembles, sont crées deux ensembles supplémentaires de données bruitées artificielles qui augmentent la difficulté de l'entraînement et la robustesse du système.
La baseline est un système de RAP , *le Context Dependant Phonem* (CDP), évalué sur une tache de sous titrage d'une video youtube. Elle obtient en dictée vocale propre 5% de WER, en recherche vocale propre 8%, en dictée vocale bruitée 6%, et en reconnaissance des chiffres 11%. Le modèle CTC est évalué sur le jeu de test propre du domaine de la dictée vocale et sur les données de test propre du domaine de la recherche vocale. On obtient en dictée vocale 39% de WER, et 53% de WER en recherche vocale. Le RNN-T obtient 6% de WER pour l'ensemble des données propres sur la tache de dictée vocale, et 12% pour le jeu de test propre du domaine de la recherche vocale. Il obtient 8% de WER pour l'ensemble des données bruitées pour la tache de dictée vocale, et 22% pour le jeu de test bruité du domaine de la recherche vocale. Sur la reconnaissance de chiffres, il obtient 9%. Le modèle avec Attention obtient 6% de WER pour les données propres sur la dictée vocale, et 11% sur la recherche vocale. Il obtient 8% de WER sur les données bruitées sur la dictée vocale et 20% sur la recherche vocale. Sur la reconnaissance de chiffres, il obtient 8%. Le RNN-T avec Attention obtient 6% de WER pour les données propres sur la dictée vocale, et 12% sur la recherche vocale. Il obtient 8% de WER sur les données bruitées sur la dictée vocale et 21% sur la recherche vocale. Sur la reconnaissance de chiffres, il obtient 9%. Il est intéressant de noter que RNN-T ainsi que les systèmes basés sur Attention présentent des performances nettement supérieures à la baseline concernant le test de reconnaissance des chiffres. Ce dernier est particulièrement difficile à réaliser car la correspondance entre la forme orale et la forme écrite dépend souvent du contexte.

Lorsque toutes les approches Seq2Seq sont comparées, et formées avec des données et une initialisation identiques, nous constatons que le modèle basé sur Attention avec deux couches de décodeur est le système le plus performant, réalisant des améliorations significatives par rapport aux autres modèles Seq2Seq sur les ensembles de recherche vocale et de test numérique.
Certaines erreurs commises par les modèles sont dues à l'absence de larges LM. Ceux utilisés par les systèmes de RAP traditionnels sont entraînés sur de grosses quantités de données textuelles plus faciles à obtenir que les transcriptions de données audio. Par exemple, dans la phrase « who wrote tortoise and the hare » (qui a écrit le lièvre et la tortue), une erreur de confusion de mots entre « hare » (lièvre) et «hair » (cheveux) a été commise par les modèles Seq2Seq testés. Cette erreur n'aurait pas eu lieu dans un système de RAP traditionnel car le LM aurait permis de prendre la bonne décision sur le bon mot dans ce contexte précis, ici « hare ». Une autre source d'erreurs des systèmes Seq2Seq est la reconnaissance des entités nommées. Ces erreurs sont très fréquentes dans les recherches vocales. Par exemple, les modèles ont reconnu « Cuca Lake » au lieu du nom correcte du lac des États-Unis, écrit « Keuka Lake ». Ces exemples montrent qu'il est nécessaire d'utiliser des ressources linguistiques extérieures (LM, etc...) pour que les modèles Seq2Seq arrivent au niveau des performances des systèmes état de l'art.

1.3.3 Modèles de Langage récents

Le LM détermine la phrase la plus probable, les successions possibles de mots, en construisant une distribution de probabilités sur les séquences de mots. Il est utilisé pour réévaluer les probabilités selon le contexte des phrases. Au LM peut s'ajouter un algorithme de réévaluation supplémentaire, le beam search.
La plupart des modèles traditionnels ont une mémoire courte (LSTM, n-gram). Ils ne mémorisent facilement que jusqu'à trois mots en arrière. Au delà, la dépense en temps et en puissance de calcul est exponentielle. Pouvoir traiter les *Long Term Dependancies* (LTD) représente un progrès notable pour les LM. Le modèle de langue est presque nécessaire pour la plupart des taches de NLP : génération de texte (par exemple la suggestion de mots sur les téléphones portables), résumé de texte, traduction, chat bot, questions-réponses. Le LM permet de déterminer le sens d'un mot en fonction de son contexte, il évite l'ambiguïté des homonymes.
Dans les modèles de langage les plus performants se distingue le modèle *Generative Pretrained Transformer* (GPT). C'est un système assez simple entraîné sur un énorme corpus de textes non structurés. Il est entraîné et testé seulement comme un modèle de langue.
Le GPT-2 [Radford et al., 2019] est un LM plus puissant que le précédent. Il contient jusqu'à 1.5 G de paramètres et peut donc être utilisé sur un seul ordinateur. Étant donnée sa puissance, ce LM peut aussi être testé sur d'autres taches de NLP pour lesquelles il n'a pas été prévu à l'origine. Comparé aux autres systèmes traditionnels qui stagnent à partir d'une certaine grande quantité de données, le GPT-2 n'a pour le moment pas encore atteint ses limites.
Le GPT-3 [Brown et al., 2020] contient plus de 175 G de paramètres. Un cluster d'ordinateurs est nécessaire pour le faire fonctionner. Il n'a pas non plus encore atteint ses limites. Un humain a 50% de chances de ne pas différencier un article écrit par GPT-3 d'un article écrit par un humain.

1.4 Conclusion

Nous avons présenté les fondements de la RAP ainsi que les modèles récents utilisés dans ce domaine. Tous ces modèles font l'objet d'évaluations que nous allons décrire au chapitre suivant.

CHAPITRE 2

MÉTHODES

Sommaire

2.1 Introduction

Pour évaluer un système, la métrique évalue les erreurs concernant l'ajout, la modification ou la suppression d'une donnée. Plus la métrique (calcul qui évalue le modèle) et la tache d'évaluation (ce pour quoi est fait le modèle) sont précises, plus le système évalué sera performant. Leur précision est donc déterminante dans la fiabilité des applications dédiées finales. Les difficultés les plus connues sont actuellement la reconnaissance des entités nommées et le formatage des séquences de chiffres.
Certaines métriques d'évaluation plus générales s'appliquent à tous les modèles, d'autres plus spécifiques ne s'appliquent que pour des taches bien précises. Mais auparavant il est nécessaire de formater les transcriptions.

2.2 Formatage des Transcriptions

Pour affiner une évaluation, le formatage des transcriptions permet de diriger l'évaluation selon l'application prévue. Par exemple, une transcription prévue pour un sous-titrage fera l'objet d'une attention particulière sur les entités nommées (NE : *Named Entities*), une autre pour un agenda sera sur les dates, une autre pour une

alarme sur les heures. Ce formatage s'effectue soit en modifiant directement le système de RAP, soit en post-traitement, une fois la transcription obtenue.

2.2.1 Modification du système de RAP

Une méthode consiste en un traitement qui aide à la reconnaissance des chiffres en contexte [Haynor and Aleksic, 2020], comme la reconnaissance des entités en réponse aux invites de dialogue des assistants virtuels des systèmes interactifs de réponse vocale. Les données générées par les grammaires numériques au format FST sont intégrées dans le système de RAP avant le Beam Search car elles vont biaiser le système et le forcer à reconnaître des entités numériques. Les *Finite State Transducer* (FST) permettent d'introduire du contexte dans le système. Après intégration dans le système, ces données numériques boostent les probabilités d'hypothèses obtenues grâce au Beam Search.
Les jeux de données utilisés sont les suivants, suivis d'exemples :

- Time pour des heures : "6 :30" (alarme) ;
- Percentage pour des pourcentages : "68%" ;
- Four digits pour des nombres à 4 chiffres : "1996" (année) ;
- Ten Digits pour des nombres à 10 chiffres : "0123456789" (numéro de téléphone) .

Tous ces jeux de données contiennent uniquement des chiffres sans autres mots prononcés en plus qui correspondent à une réponse d'un utilisateur à qui l'assistant vocal fait une demande.

- Unprompted Ten Digits pour des nombres à 10 chiffres en contexte comme par exemple "entrez 0123456789" ;
- Anti-context pour des uttérances qui sont hors sujet des entités numériques.

Le biais contextuel corrige bien plusieurs types d'erreurs, comme le mauvais nombre de chiffres dans les séquences à plusieurs chiffres, ou une prononciation similaire d'un mot à une séquence de chiffres.
Le *Multi Tasking* (MT) [Ghannay et al., 2018] est une méthode qui entraîne un système de RAP pour accomplir successivement les tâches de RAP et de reconnaissance des entités nommées (NER : *Named Entity Recognition*) par exemple. Le modèle est d'abord entraîné pour la tâche de RAP sans prendre en compte les tags qui annotent les NE. En fin de processus, la couche Softmax est réinitialisée pour les prendre en compte. Le modèle est ensuite réentrainé sur la tâche de NER avec seulement des données annotées.
Une autre méthode de formatage est d'extraire les NE [Ghannay et al., 2018] à partir de l'input en utilisant un système de RAP sans en modifier l'architecture, afin de savoir si le système est capable de trouver des informations sémantiques lui permettant de reconnaître des NE. En modifiant la séquence de caractères normalement produite en output en ajoutant des tags pour délimiter les NE, le modèle prédit un caractère à chaque output time step. Cela permet également de prédire les tags, et la fonction de perte CTC prend en compte les tags pendant l'entraînement. Le système obtient de moins bons résultats pour reconnaître des tags NE mais est plus performants pour extraire les valeurs NE.
La reconnaissance de séquences numériques reste une problématique importante dans la RAP car le système ne peut assimiler l'infinité des combinaisons possibles de chiffres [Peyser et al., 2019] . Dans la chaîne de traitement, plusieurs éléments ont pour objet de réduire la difficulté liée à la *Data Sparsity* et à l'OOV.

Le *Text To Speech* (TTS) génère des données artificielles supplémentaires d'entraînement qui représentent des séquences numériques plus compliquées et plus réalistes à partir d'uttérances de *Google Voice Search Traffic*, combiné avec une grammaire WFST.
Le *Written Domain Neural Correction* (WDNC) corrige des hypothèses du RNN-T grâce à un post traitement qui utilise un modèle Seq2Seq basé sur Attention. Il prend en entrée les hypothèses du RNN-T et produit en sortie les transcriptions corrigées. Le mécanisme d'Attention et le décodeur sont appliqués uniquement sur des morceaux de textes pertinents, ce qui réduit les coûts et augmente l'exactitude.
Le *FST Denorming* permet un alignement forcé des phonèmes du domaine parlé et du domaine écrit.
Le *Neural Denorming* est une adaptation du modèle WDNC qui prend en entrée des données du domaine parlé et produit des outputs du domaine écrit.
Les résultats des modèles de domaine écrit se caractérisent par une forte baisse de la qualité à mesure que la longueur numérique augmente. Le TTS et le WDNC permettent de dépasser les résultats de la baseline sans pour autant dépasser ceux sur le jeu de données non-numérique. Reste la difficulté à formater les données numériques et la confusion de certaines parties numériques qui sonnent comme d'autres mots. Le formatage des modèles de domaine parlé échoue mais l'utilisation du *Neural Denorming* améliore grandement les résultats. Les différentes approches combinées donnent les meilleurs résultats, en particulier sur le jeu de données avec les séquences numériques les plus longues, ce qui permet de résoudre le problème de *Data Sparsity* et d'OOV.

2.2.2 Post-traitement sur les transcriptions

Une méthode de post-traitement s'appuie sur des grammaires qui sont écrites par des humains (*Hand Written Grammars*). Elles génèrent toutes les formes écrites possibles et plausibles. Elles se définissent sous la forme de FST [Shugrina, 2010]. Le score de probabilités est ensuite calculé sur ces formes générées par les grammaires avec un LM. Les séquences de mots ainsi pondérées par le LM permettent alors d'utiliser un *Weighted Finite State Transducer* (WFST), qui attribue un poids supplémentaire sur les transitions. Le LM produit alors des séquences de mots qui ont du sens, et une transcription correcte et une bonne orthographe (majuscules, ponctuation et séquences numériques) grâce aux grammaires. Ce qui produit le bon label des mots en output.
Le *Language Model Adaptation* (LMA) [Mdhaffar et al., 2019] est une autre méthode de post-traitement qui consiste à nourrir un LM de vocabulaire spécifique à un sujet. Il permet de minimiser les erreurs dues à des mots rares ou techniques non reconnus par un LM traditionnel. Le LMA est la synthèse d'un LM traditionnel et d'un LM entraîné sur des données internet. Il est réalisé sur le corpus PASTEL, un ensemble de données en français composé d'enregistrements de conférences, de chapitrages manuels, de transcriptions manuelles et de diapositives.
Une autre méthode consiste à annoter automatiquement des transcriptions utilisées pour entraîner un système RAP et à les réinjecter dans les données d'entraînement du système RAP pour extraire les NE [Ghannay et al., 2018] à partir de l'input.
Le *Inverse Text Normalization* (ITN) [Pusateri et al., 2017] est le processus qui consiste à convertir la transcription produite par un système de RAP en une forme écrite adaptée à l'utilisateur final.

Une approche hybride utilise des grammaires écrites par des humains (Hand Crafted Grammars) et un LM BLSTM statistique à classes de n-gram pour normaliser les transcriptions produites par les systèmes. Pour effectuer la transformation de la forme parlée à la forme écrite, un processus en trois étapes est mis en place. La forme parlée est la transcription de l'oral à l'écrit, la forme écrite est le format final attendu. Dans la première étape, un label est attribué à chaque token sous la forme parlée. Dans la deuxième étape, les modifications spécifiées par le label sont appliquées sur chaque token, et les résultats sont regroupés. Pour la troisième étape, les grammaires sont appliquées sur les tokens labellisés à l'étape deux. Par exemple, sur la forme parlée « twenty » est appliquée le label « cardinal decade ». Les grammaires modifient cette forme parlée qui devient alors la forme écrite attendue correcte « 20 ».
Un modèle entraîné sur toutes les caractéristiques obtient de très bonnes performances sur 5 M d'uttérances. Les résultats obtenus pour des jeux de tests sur des catégories particulières, par exemple le temps, les adresses, les dates, la monnaie, sont également très bons sur 5 M d'uttérances. L'ITN peut donc s'avérer utile pour la dictée, les sous-titrages, la prise de rendez-vous, ainsi que pour la recherche vocale qui peut bénéficier de mots clés correctement formatés.

2.3 Les Métriques générales

Selon la donnée évaluée, phonème, caractère ou mot, s'applique une évaluation particulière.

2.3.1 Phonem Error Rate

Il calcule le taux d'erreur concernant les phonèmes.

$$PER = \frac{(I + S + D)}{N} \tag{2.1}$$

avec *Insertion*, *Substitution*, *Delition* et *Total Number of Phonems* dans la référence. Reconnaître un phonème est difficile. Il est très court (le temps de prononciation est très rapide) et aléatoire (un locuteur prononce rarement deux fois un phonème de la même façon). Mais la reconnaissance de phonèmes est finie car le nombre de phonèmes est exhaustif (Alphabet Phonétique International).
Le PER ne reconnaît pas les bruits. Par exemple, dans le sous titrage de videos, tv, séries, films, il faudrait pouvoir ajouter des tokens dans la transcription finale pour préciser un bruitage comme des rires, une porte qui claque ou de la musique...

2.3.2 Character Error Rate

Il concerne les langues à caractères écrits , comme les langues asiatiques japonais, mandarin, coréen... Il calcule le taux d'erreur concernant les caractères.

$$CER = \frac{(I + S + D)}{N} \tag{2.2}$$

avec *Insertion*, *Substitution*, *Delition* et *Total Number of Characters* dans la référence.

2.3.3 Word Error Rate

Cette métrique consiste à compter les erreurs selon les types prédéfinis d'insertion, de suppression et de substitution de mots dérivés par un alignement de [Levenshtein, 1965] entre les transcriptions manuelles (référence) et automatiques (hypothèse).

$$WER = \frac{(I + S + D)}{N} \tag{2.3}$$

avec *Insertion*, *Substitution*, *Delition* et *Total Number of Words* dans la référence.
Le WER compte dans la transcription le nombre de mots qui sont modifiés, ajoutés ou supprimés par rapport au texte de référence, ce qui comprend les mots vides comme les disfluences verbales et les défauts de langage. Hors calculer ce taux d'erreur sans ces mots vides serait une amélioration de l'évaluation car elle ne représenterait alors plus que les mots réellement erronés. La reconnaissance exhaustive des mots est impossible car il existe une infinité de possibilités.

2.3.4 Sentence Error Rate

Le SER focalise sur une fraction des phrases contenant une ou plusieurs erreurs. Il est plus pertinent dans le domaine des tâches de reconnaissance d'entités numériques, car les séquences numériques seules hors contexte peuvent être considérées comme erronées plus facilement. Replacée dans son contexte, la séquence numérique erronée est plus visible par l'évaluation.
L'application du SER permet de montrer des performances améliorées du système grâce à la prise en compte du contexte.
Il améliore la qualité du système RAP sur tous les tests set avec une réduction du WER allant de 12% à 59% et une réduction du SER allant de 25% à 55%. Le test set anti-contexte montre une légère augmentation des WER et SER allant de 0% à 2%.

2.4 Les Métriques spécifiques

2.4.1 Numéric Entity Error Rate

Le NEER évalue la performance du système à correctement transcrire les nombres [Shugrina, 2010]. Trois types d'erreurs concernent les nombres et le NEER :

- Mauvais formatage (*Incorrect Formating*) : Le système formate incorrectement une entité correctement formatée dans la référence : on se retrouve vers 5-6 heures devient 56 heures ;
- Sur-formatage (*Over Formating*) : Le système formate une entité qui n'en a pas besoin : être sur son « trente-et-un » est écrit « 31 » ;
- Sous-formatage (*Under Formating*) : Le système ne formate pas une entité qui devrait l'être : « one fourty » écrit 1 40 au lieu de « 1 :40 pm » .

Les résultats du NEER se répartissent en quatre groupe : le NEER Global ; le I-FR qui est le NEER causé par le incorrect-formating ; le O-FR causé par le over-formating ; le U-FR causé par le under-formating. En dehors des erreurs d'espacement qui se révèlent nombreuses, 84% des entités sont parfaitement formatées. Cette approche est donc performante dans la tache de formatage des nombres.

2.4.2 Précision, Rappel, et F-mesure

La précision (formule 2.4), le rappel (formule 2.5) et la F-mesure (formule 2.6) [Shugrina, 2010], [Mdhaffar et al., 2019], [Makhoul et al., 2000], sont des scores qui évaluent l'annotation faite par le système.
On distingue :

— Les vrais positifs : ce qui est annoté comme correct par le système et qui est vrai ;
— Les faux positifs : ce qui est annoté comme correct par le système mais qui est faux ;
— Les faux négatifs : ce qui est annoté comme incorrect par le système mais qui est faux .

$$\text{Précision} = \frac{\text{vrais positifs}}{\text{vrais positifs + faux positifs}} \tag{2.4}$$

$$\text{Rappel} = \frac{\text{vrais positifs}}{\text{vrais positifs + faux négatifs}} \tag{2.5}$$

$$\text{F-mesure} = \frac{(1+\beta^2) \times \text{ précision } \times \text{ rappel}}{\beta^2 \times \text{ précision } + \text{ rappel}} \tag{2.6}$$

Les F-mesures pour les majuscules et le point sont plutôt bonnes, mais pour la virgule elle est très mauvaise : L'annotation des majuscules et des points obtient une F-mesure de 0.6, ce qui est considéré comme correct. La virgule obtient 0.3, ce qui est considéré comme mauvais.
Ce processus d'évaluation est pertinent pour l'annotation, mais dans un autre contexte, il diminue l'importance des erreurs de suppressions et d'insertions.

2.4.3 IWER

La métrique *Individual Word Error Rate* (IWER) [Mdhaffar et al., 2019] est inspirée du WER qui ne s'avère pas pertinent pour évaluer dans un domaine spécifique des transcriptions utilisées pour une tâche de recherche de documents. Le WER ne permet pas de faire la différence entre des mots généraux et des mots techniques, il considère tous les mots, alors que le IWER prend en compte l'importance des insertions, suppressions et substitutions (formule 2.7).
Le IWER attribue 1 ou 0 en comparant l'hypothèse et la référence : 1 signifie que le mot de référence a été supprimé ou substitué, 0 qu'il n'y a pas de changement. Dans le cas d'erreurs d'insertion, deux mots de référence adjacents peuvent être responsables. Sans savoir lequel, une responsabilité partielle égale est appliquée aux deux mots adjacents.

$$IWER(w_i) = del_i + sub_i + \alpha.ins_i \tag{2.7}$$

Même si le IWER est appliqué par nature sur un LM en visant des mots spécifiques à un domaine particulier, il fait l'objet d'une généralisation afin d'être appliqué sur tout un corpus qui contient des mots spécifiques sur des sujets différents, pour obtenir un score global sur un corpus de transcriptions multi-domaines.

2.4.4 La vitesse de réponse d'un système QR

Pour évaluer la pertinence du travail de formatage (majuscules, ponctuation et chiffres) effectué sur les transcriptions , on compare la vitesse de réponse d'un système de questions-réponses QR [Shugrina, 2010] en utilisant des transcriptions formatées, des transcriptions non formatées et des transcriptions manuelles (produites par un humain) . La vitesse de réponse du système QR pour les données formatées est équivalente à celle des données des transcriptions manuelles. Les transcriptions non formatées sont plus difficiles à lire par le système de QR. Ce qui montre que formater les transcriptions pour qu'elles soient plus lisibles a un impact positif sur leur exploitation en aval.

2.5 Conclusion

Dans les chapitres précédents, nous avons vu que les métriques d'évaluation ne sont pas toujours complètement adaptées aux applications actuelles qui utilisent des transcriptions.

Or pour les développer, il faudrait que toute personne (chercheur, étudiant, entreprise, ...) puisse accéder à ces modèles, ce qui est compromis par le fait que les systèmes de RAP décrits ci dessus sont des modèles issus de la recherche, pointus, scientifiques, confidentiels, protégés par des licences, et donc difficiles d'accès pour tout public. Les métriques d'évaluations correspondantes sont tout aussi restreintes d'accès.

Il semble donc compliqué de tester et diversifier les applications dans ce périmètre protégé, indépendamment de la volonté de renforcer et d'affiner les évaluations. Ces systèmes ne peuvent être ni reproduits, ni évalués, ni analysés facilement. Alors qu'en développer les applications renforcerait grandement leur efficacité.

Deuxième partie

Expérimentations

CHAPITRE 3

OUTILS

Sommaire

3.1 Introduction

Pour contourner cette “privatisation forcée” étudiée dans l’état de l’art, et rendre public des modèles de RAP, d’autres modèles “open source” existent qui sont plus accessibles : Codage vulgarisé, accès aux données facilité, protections levées, puissance informatique moindre rendent ces systèmes plus souples et plus ouverts. Ils permettent de partager l’expérience, de communiquer par des articles, de développer les applications, tout en appliquant des méthodes d’évaluations connues comme le WER ou des post-traitements. L’idée est de présenter aux systèmes de RAP le plus de situations publiques possibles : une seule phrase prononcée par chaque personne dans le monde entier pourrait enrichir les données du système, et donc le fiabiliser, avant de devoir en modifier les évaluations. Ces outils proposent des architectures complètes ou des composants (AM, LM, etc...) d’un modèle qui peuvent être entraînés ou utilisés déjà pré-entrainés. Ils permettent à tout individu d’obtenir des transcriptions à une échelle particulière avec un modèle pré-entraîné, plus facile à exploiter et à évaluer, pour des applications choisies elles aussi particulières.

Nous allons décrire maintenant les outils les plus connus dans le domaine de la RAP ainsi que celui utilisé pour les expériences.

3.2 Le système de RAP de la RWTH

A la fin des années 2000, l’université Rheinisch-Westfälische Technische Hochschule (RWTH) d’Aix-la-Chapelle (Aachen, Allemagne) a mis sur son site internet son outil open-source [Rybach et al., 2009]. Elle l’a ensuite rendu disponible sur Github (https://github.com/rwth-i6/rasr) avec les différents projets sur lesquels tra-

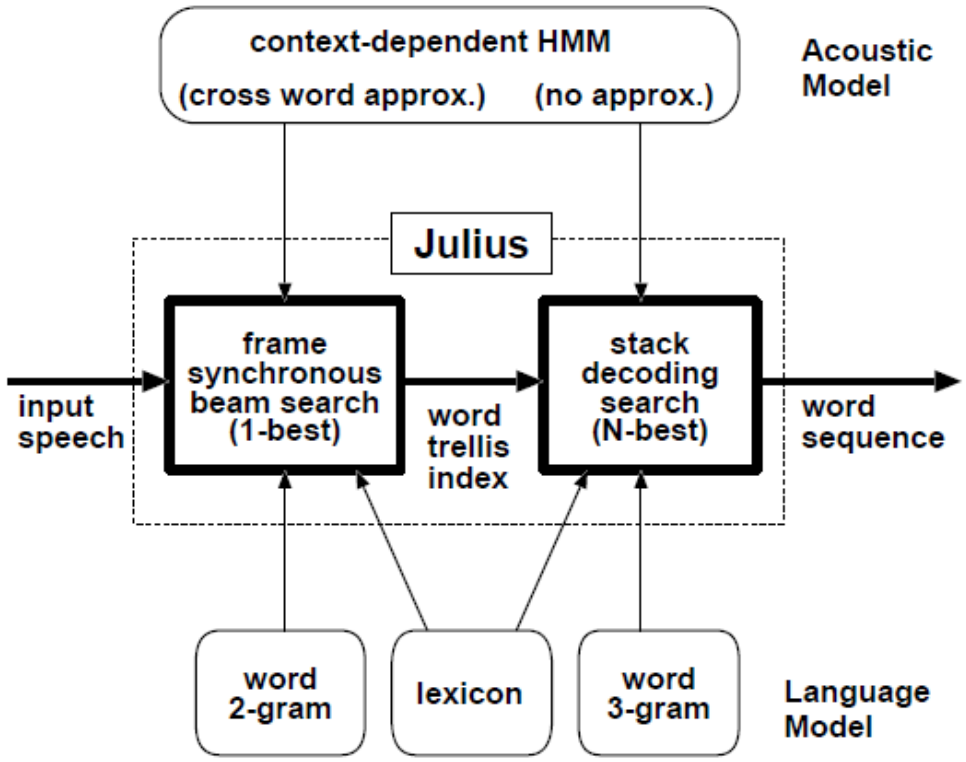


FIGURE 3.1 – Schéma de l'architecture de Julius [Lee et al., 2001]

vaillent les chercheurs de l'Université. Il est possible notamment d'y trouver des modèles à entraîner ou pré-entrainés pour la RAP, écrits en C++. L'idée est de donner un accès public aux différents blocs qui composent un système de RAP traditionnel ou à un modèle complet, tout en ayant la possibilité de modifier, personnaliser et améliorer ces blocs. Ce système est encore en activité et maintenu à la pointe de la technologie.

3.3 Julius

Julius (http://julius.osdn.jp/en_index.php) est un outil développé en C à la fin des 90's par deux universités japonaises, l'Institut des Sciences et Technologies de Nara, et le Département d'Informatique de l'Université de Kyoto. Il est entraîné sur un large vocabulaire (LVCSR) [Lee et al., 2001]. C'est un décodeur basé sur un *HMM context-dependent* et des *3-gram* (Figure 3.1).

L'écriture des grammaires s'inspire du format HTK. Cet outil (https://github.com/takeyamajp/julius) présente l'avantage de pouvoir être combiné à d'autres blocs pour former un système de RAP complet. Sa performance est la transcription quasi spontanée. Il est un des précurseurs des systèmes de streaming. Il obtient aussi de bons résultats pour une tâche de dictée vocale en japonais, combinaison rare dans la littérature.Mais il n'est plus mis à jour depuis 2015.

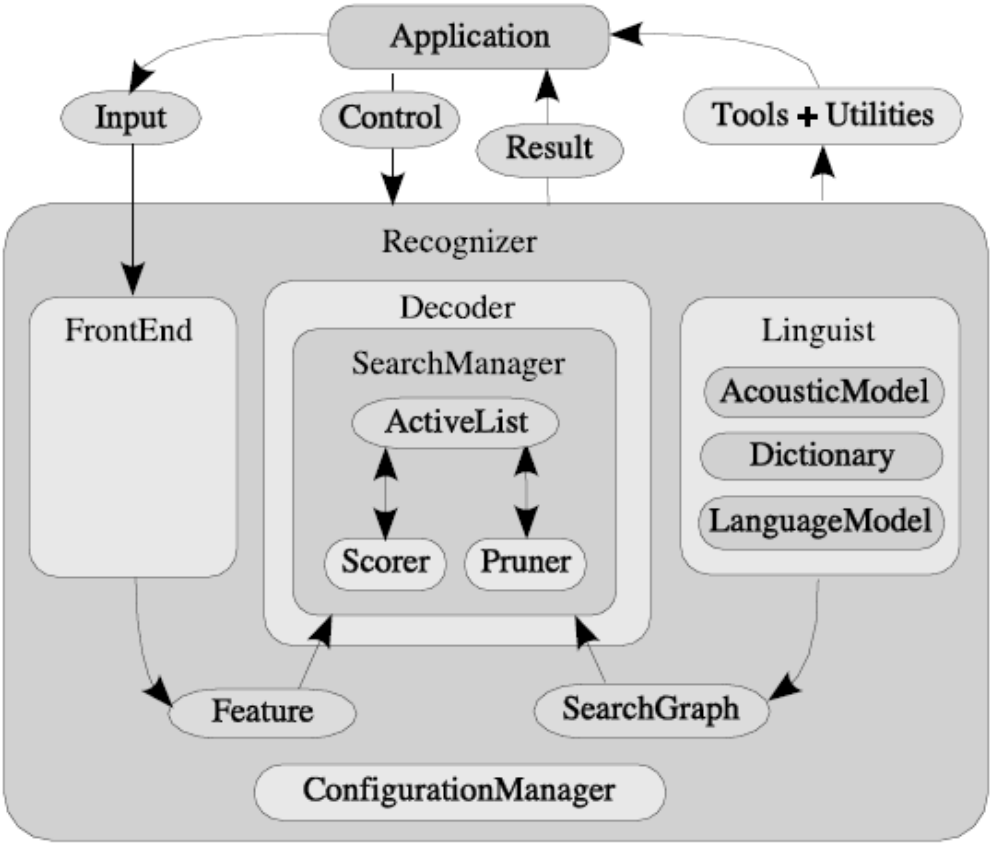


FIGURE 3.2 – Schéma de l'architecture de Sphinx [Walker et al., 2004]

3.4 Sphinx-4

Sun Mycrosystems a développé, en partenariat avec l'université Carnegie Mellon et l'entreprise Mitsubishi Electric Reasearch Labs, l'outil *Sphinx-4* (https://sourceforge.net/projects/cmusphinx/), écrit en Java. Il explore différentes architectures de HMM [Walker et al., 2004] (Figure 3.2).

L'outil a évolué en s'adaptant aux changements dans le domaine de la RAP et en proposant des modèles état de l'art. Il collabore également sur des projets utilisant d'autres outils tels que *Kaldi* (https://github.com/alphacep/vosk-android-demo) ou crée des applications telles que *Pocketvox* pour Linux. Cet outil est toujours maintenu à la pointe de la technologie et se retrouve dans de nombreuses applications commerciales et appareils électroniques d'usage quotidien.

3.5 HTK

L'outil *Hidden Markov Model Toolkit* (HTK) a été développé en C à la fin des 90's dans les laboratoires de l'Université de Cambridge par [Young et al., 2002]. Bien que l'outil soit en open-source, Microsoft détient le copyright de HTK. Comme son nom l'indique, HTK permet de construire et de manipuler des HMM, et il est principalement utilisé pour la RAP (Figure 3.3).

Il a également été utilisé dans les domaines de recherche de la synthèse de la parole, la reconnaissance de caractères et le séquençage ADN. Certains composants

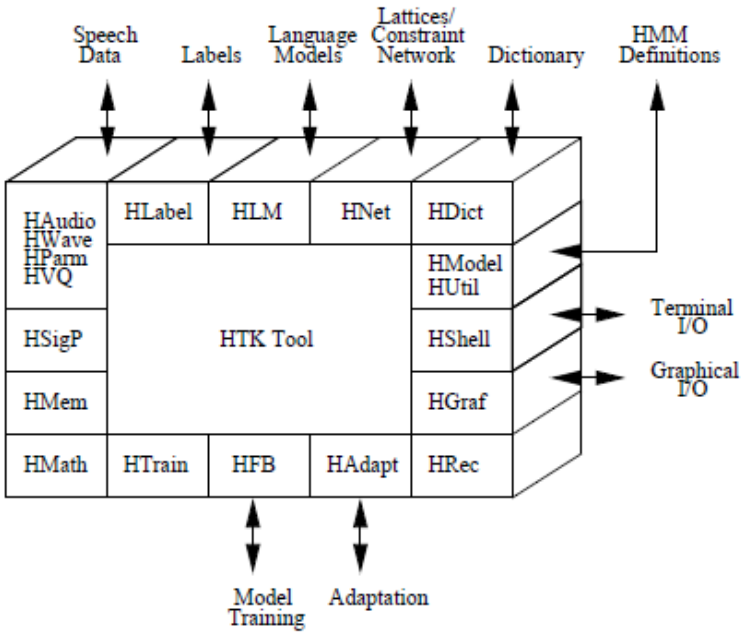


FIGURE 3.3 – Schéma de l'architecture de HTK [Young et al., 2002]

de l'outil comme les grammaires ont inspiré d'autres outils. La dernière mise à jour de l'outil date de 2016.

Il existe d'autres outils open-source développés par des entreprises telles que *Facebook* (*wav2letter* https://github.com/facebookresearch/wav2letter) ou *Mozilla* (*DeepSpeech* https://github.com/mozilla/DeepSpeech) qui deviennent des acteurs importants dans le domaine de la RAP.

Parmi les principaux modèles open-sources connus dans le domaine de la RAP, Kaldi se distingue par son accessibilité. Kaldi est plus utilisé que les autres outils.

3.6 Kaldi

Le nom *Kaldi* provient du légendaire berger éthiopien Kaldi, qui aurait découvert la plante de café. Le système a été développé en 2009 à l'université Johns Hopkins en collaboration principalement avec Microsoft, l'Université de la Sarre (Saarland, Allemagne), et le Centre de Recherche Informatique de Montréal (Canada) [Povey et al., 2011]. Écrit en C++, *Kaldi* est basé sur des FST. Il est moderne, flexible, facile à comprendre et à modifier, et donc à enrichir. Comparé aux autres outils, *Kaldi* propose une licence plus ouverte. Les différents outils proposés par *Kaldi* sont modulables et peuvent se combiner de différentes façons (Figure 3.4).

Kaldi met à la disposition des utilisateurs un grand nombre de modèles entraînés sur une grande variété de corpus (WSJ, LibriSpeech, CSJ,...) parus dans la littérature scientifique. Cependant les corpus ne sont pas fournis pour des raisons techniques (trop grand volume de données) et de droit d'accès (certains sont payants ou ne sont pas open-source). En fonction de l'application souhaitée, l'utilisateur choisit d'entraîner un modèle proposé par *Kaldi*, ou d'utiliser un modèle pré-entrainé,

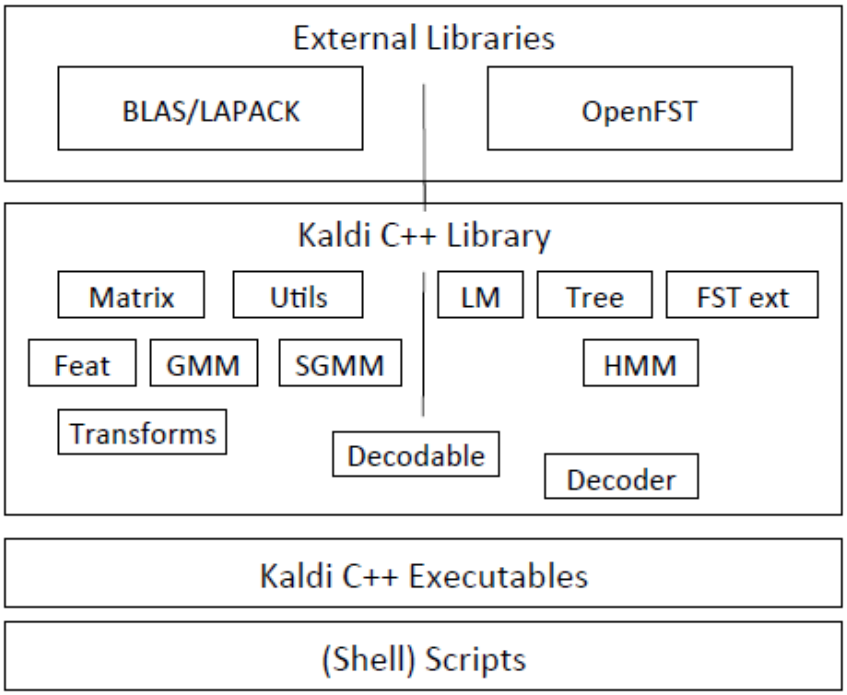


FIGURE 3.4 – Schéma de l'architecture de Kaldi [Povey et al., 2011]

chacun étant entraîné sur un corpus spécifique pour un type de parole particulier.

Il utilise également des modèles pré-entraînés sur de petits jeux de données. Ceci représente un avantage certain car les données orales en grande quantité sont rares, et les utilisateurs peuvent accéder aux technologies de la RAP sans grande puissance de calcul ni temps nécessaire pour entraîner des modèles.

3.7 Conclusion

Suite à la description de tous ces outils open-source, notre choix s'est porté sur l'outil Kaldi pour mener les experimentations.

CHAPITRE 4

CORPUS

Sommaire

Les scripts présentés dans cette section sont tous disponibles en annexe (Section A.2).

4.1 Installation

Pour installer *Kaldi* (https://kaldi-asr.org/doc/tutorial_setup.html) et le tester (https://kaldi-asr.org/doc/kaldi_for_dummies.html), nous avons utilisé deux jeux de données : un jeu de chiffres et un jeu de phrases, tous deux complémentaires dans leur utilisation et leurs applications, car utiliser des jeux de données similaires ne présente pas beaucoup d'intérêt dans ce mémoire.

Le *Free Spoken Digit Dataset* (FSDD) (https://github.com/Jakobovski/free-spoken-digit-dataset), ensemble de fichiers audio provenant de six locuteurs différents prononçant les chiffres de 0 à 9 en anglais. Chaque chiffre est prononcé cinquante fois par un même locuteur. Les fichiers audio sont au format WAV.

Le *Tatoeba* (https://tatoeba.org/fra/). Le site regroupe une grande collection de phrases (8 930 102) et leur traduction dans 387 langues (anglais, français, allemand, néerlandais,...). Le jeu de données *Tatoeba* utilisé ici regroupe des enregistrements de phrases en anglais prononcées par six locuteurs différents.

4.2 Pré-traitements

Le jeu de données FSDD étant déjà prêt à l'emploi, il ne nécessite pas de pré-traitement à effectuer sur les données audio.

La quantité de données du Tatoeba étant beaucoup trop importante pour l'ordinateur utilisé, elle a nécessité quelques pré-traitements :

— Nous avons écrit le script « reduce_csv.py » pour limiter le nombre d'uttérances par locuteur à cinquante uttérances, excepté pour le locuteur « Delian » qui ne compte que 6 uttérances ;
— Nous avons choisi de diviser le jeu de données FSDD en un jeu d'entraînement (george, jackson, nicolas, theo, yweweler) et un jeu de test (lucas) ;
— Nous avons choisi de diviser le jeu de données Tatoeba en un jeu d'entraînement (CK, Delian, rhys_mcg et Susan1430) et un jeu de test (BE et pencil) ;
— Les enregistrements audios devant être au format WAV, nous avons écrit le script « mp3_to_wav.sh » qui convertit les fichiers au format MP3 en format WAV grâce à l'outil sox .

4.3 Structure

La structure des dossiers de travail se trouve en annexe : Figures A.1, A.2 et A.3.

4.4 Préparation des données

Nous avons préparé les données comme décrit sur le site de Kaldi (https://kaldi-asr.org/doc/data_prep.html) :

— A l'aide du script « csv_to_text.py » que nous avons écrit, nous avons créé deux fichiers "text" contenant l'identifiant du locuteur, l'identifiant de l'uttérances et la transcription de l'uttérance sous la forme <spkID>_<uttID><text>. Chacun est placé dans le dossier « train » ou le dossier « test » selon les informations qu'ils contiennent ;
— Encore à l'aide du script « csv_to_text.py », nous avons créé un fichier « corpus.txt » contenant toutes les transcriptions des uttérances ;
— Nous avons écrit le script « no_duplicates.sh » qui contient des commandes permettant d'extraire tous les mots des uttérances, de les capitaliser et de les trier pour en supprimer les doublons. Il génère un fichier « words.txt » ;
— Nous avons utilisé l'outil en ligne *Legios* (http://www.speech.cs.cmu.edu/tools/lextool.html) qui permet d'obtenir la transcription phonétique des mots. Le fichier obtenu est enregistré sous le nom « lexicon.txt » ;
— Nous avons écrit le script « lex_to_phones.py » qui permet d'extraire tous les phonèmes vers le fichier « nonsilence_phones.txt » ;
— Nous avons écrit le script « get_filepath.py » qui permet de créer les fichiers « wav.scp » et « utt2spk ». Le fichier « wav.scp » contient l'identifiant du locuteur, l'identifiant de l'uttérance et le chemin pour accéder au fichier audio sous la forme <spkID>_<uttID><path/to/audio/file.wav>. Le fichier « utt2spk » contient l'identifiant du locuteur, l'identifiant de l'uttérance et l'identifiant du locuteur sous la forme <spkID>_<uttID><spkID> .

4.5 Lancement

Après avoir préparé les données, comme indiqué sur le site de Kaldi, nous avons utilisé les scripts suivants :

— les fichiers « mfcc.conf » et « decode.config » sont des fichiers de configuration ;
— le fichier « cmd.sh » permet au système d'utiliser le fichier run.sh plutôt que le CPU en ligne ;

— le fichier « path.sh » définit la racine du dossier Kaldi et le chemin vers les outils utilisés ;
— le fichier « run.sh » représente la pipeline et lance les différentes étapes vers le WER .

CHAPITRE 5

RÉSULTATS

Sommaire

5.1 Introduction

Nous allons présenter le protocole d'experimentation que nous avons choisi de suivre puis nous détaillerons les résultats constatés.

5.2 Protocole

Le protocole d'expérimentation suppose que l'on utilise un système de RAP avec lequel on teste différents jeux de données destinés à des applications différentes : les chiffres pour la reconnaissance numérique (alarme, calendrier, numéro de téléphone, code postal...), et les phrases pour le sous titrage, la dictée vocale ou la recherche d'informations à partir de mots clés par exemple.Une fois obtenus les transcriptions, le WER et la liste des erreurs, on analyse ces résultats pour ensuite appliquer les différentes méthodes d'évaluations (post traitement ou métriques) citées à la fin du chapitre 2 afin de définir la pertinence de ces méthodes d'évaluation.

Dans notre expérience, le protocole a été compliqué à suivre.

5.3 Résultats

Suite au lancement sur les données de FSDD, les résultats WER n'ont pas été très bons car ils sont arrivés à 50% de WER. Ce résultat est normal : les enregistrements des fichiers audio sont très courts, ce qui représente un obstacle pour le système de RAP car reconnaître un seul chiffre dit une seule fois sans contexte audible est compliqué. Mais nous n'avons pu prolonger l'expérience car nous n'avons pu obtenir les transcriptions ni la liste des erreurs.

Le lancement sur les données du Tatoeba n'a pas donné de résultats car il n'a pu se dérouler correctement. Les difficultés rencontrées se situent au niveau du

déroulement du lancement de Kaldi, et non au niveau de l'application des méthodes d'évaluation. Ces obstacles provenant d'un manque de connaissance de ce système peuvent se résumer selon les points suivants :

La documentation officielle de Kaldi est essentiellement orientée vers l'entraînement des modèles qui est impossible à gérer par manque de temps et de puissance de calcul dans le cadre de ce mémoire.

Suite à l'installation de Kaldi, les fichiers audio d'input doivent être au format WAV, alors que les appareils d'enregistrement créent en général des fichiers au format MP3. La conversion d'un ensemble conséquent de fichiers audio augmente alors le temps de préparation des données.

Le lancement de Kaldi a demandé plusieurs jours à l'ordinateur. De plus, par manque d'explications sur les détails de ce qui est attendu par Kaldi, nous avons dû lancer plusieurs fois l'outil en rectifiant les erreurs affichées dans le terminal au fur et à mesure, ce qui a ralenti énormément le protocole d'expérimentation.

Les erreurs affichées par Kaldi sur le terminal sont les suivantes :

- La fréquence des enregistrements doit être de 6kHz exactement : nous avons dû adapter la fréquence des enregistrements donnés en entrée à Kaldi en ajoutant des options dans le fichier de configuration « mfcc.conf » ;
- La transcription phonétique obtenue avec l'outil Legios est en majuscule : nous avons dû convertir tous les mots du fichier "words.txt" également en majuscules. Les commandes se trouvent dans le fichier « no_duplicates.sh » ;
- Il est possible que certains mots dans le fichier "lexicon.txt" soient en double : nous avons dû supprimer ces doublons. Les commandes se trouvent dans le fichier « no_duplicates.sh » ;
- Il manquait le fichier « feats.scp » qui normalement est créé au même moment que l'extraction des MFCC. Ce problème ne s'est pas posé lors du lancement du jeu de données FSDD .

5.4 Conclusion

Suite à tous ces obstacles rencontrés, nous pouvons en déduire que Kaldi n'est pas un outil si accessible, en tout cas pour un utilisateur novice.

L'utilisation des open sources suppose une préparation en amont et une connaissance certaine si on veut pouvoir étudier les modèles et les méthodes d'évaluations pour les rendre plus accessibles : formation, temps, puissance des machines sont nécessaires pour mener à bien les recherches.

Conclusion générale

N'ayant pu faire aboutir l'expérience, il nous semble difficile d'étudier les méthodes d'évaluations de la RAP pour les rendre plus pertinentes en fonction des applications demandées.

Cependant, notre étude nous a permis de constater que les méthodes d'évaluation générales dans le domaine de la RAP ont atteint leurs limites, avec le WER qui s'améliore mais sans générer pour autant d'amélioration des modèles selon les applications particulières. Une nouvelle approche des méthodes d'évaluation est nécessaire pour prendre en compte les nouvelles exigences de ces applications. Majuscules, ponctuation, formatage des chiffres, tics de langage ... sont encore des difficultés à surmonter dans les méthodes d'évaluation. Même si des optimisations spécifiques sont appliquées sur les méthodes d'évaluations, elles restent marginales comparées à l'utilisation des métriques générales et classiques. Une solution serait alors de développer ces processus d'optimisation tout en en facilitant l'accès au grand public.

BIBLIOGRAPHIE

[Baevski and Mohamed, 2020] Baevski, A. and Mohamed, A. (2020). Effectiveness of self-supervised pre-training for asr. pages 7694–7698. – Cité page 24.

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., and Amodei, D. (2020). Language models are few-shot learners. – Cité page 32.

[Chan et al., 2015] Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2015). Listen, attend and spell. – Cité pages 6, 23, 28 et 29.

[Chiu and Raffel, 2017] Chiu, C.-C. and Raffel, C. (2017). Monotonic chunkwise attention. – Cité pages 14, 25 et 28.

[Chiu et al., 2018] Chiu, C.-C., Sainath, T., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R., Rao, K., Gonina, E., Jaitly, N., Li, B., Chorowski, J., and Bacchiani, M. (2018). State-of-the-art speech recognition with sequence-to-sequence models. pages 4774–4778. – Cité pages 15 et 29.

[Chorowski et al., 2015] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. – Cité pages 14, 15, 23, 26 et 28.

[Davis and Mermelstein, 1980] Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28:57–366. – Cité page 18.

[Dean et al., 2012] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., Mao, M., Ranzato, A., Senior, A., Tucker, P., Yang, K., and Ng, A. (2012). Large scale distributed deep networks. *Advances in neural information processing systems*. – Cité page 29.

[Ghannay et al., 2018] Ghannay, S., Caubriere, A., Estève, Y., Camelin, N., Simonnet, E., Laurent, A., and Morin, E. (2018). End-to-end named entity and semantic concept extraction from speech. pages 692–699. – Cité pages 36 et 37.

[Graves, 2012] Graves, A. (2012). Sequence transduction with recurrent neural networks. – Cité pages 28 et 30.

[Graves et al., 2006] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, 2006:369–376. – Cité pages 28 et 30.

[Hannun et al., 2014] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. (2014). Deepspeech: Scaling up end-to-end speech recognition. – Cité page 22.

[Haynor and Aleksic, 2020] Haynor, B. and Aleksic, P. (2020). Incorporating written domain numeric grammars into end-to-end contextual speech recognition systems for improved recognition of numeric sequences. pages 7809–7813. – Cité page 36.

[Hori et al., 2017] Hori, T., Watanabe, S., Zhang, Y., and Chan, W. (2017). Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. pages 949–953. – Cité pages 14, 15, 27 et 30.

[Jaitly et al., 2016] Jaitly, N., Sussillo, D., Le, Q., Vinyals, O., Sutskever, I., and Bengio, S. (2016). A neural transducer. – Cité pages 14, 15, 21, 28 et 30.

[Jelinek, 1977] Jelinek, F. (1977). Continuous speech recognition. *ACM SIGART Bulletin*, pages 33–34. – Cité page 18.

[Lee et al., 2001] Lee, A., Kawahara, T., and Shikano, K. (2001). Julius—an open source real-time large vocabulary recognition engine. *Proceedings of European Conference on Speech Communication and Technology*, 3:1691–1694. – Cité pages 6 et 46.

[Levenshtein, 1965] Levenshtein, V. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710. – Cité page 39.

[Loukina et al., 2020] Loukina, A., Evanini, K., Mulholland, M., Blood, I., and Zechner, K. (2020). Do face masks introduce bias in speech technologies? the case of automated scoring of speaking proficiency. – Cité page 15.

[Lowerre, 1977] Lowerre, B. (1977). Dynamic speaker adaptation in the harpy speech recognition system. pages 788 – 790. – Cité page 17.

[Makhoul et al., 2000] Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. (2000). Performance measures for information extraction. *Proceedings of DARPA Broadcast News Workshop*. – Cité page 40.

[Mdhaffar et al., 2019] Mdhaffar, S., Estève, Y., Hernandez, N., Laurent, A., Dufour, R., and Quiniou, S. (2019). Qualitative evaluation of asr adaptation in a lecture context: Application to the pastel corpus. pages 569–573. – Cité pages 37 et 40.

[Mohamed et al., 2012] Mohamed, A.-r., Dahl, G., and Hinton, G. (2012). Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20:14 – 22. – Cité page 19.

[Moritz et al., 2019] Moritz, N., Hori, T., and Le Roux, J. (2019). Triggered attention for end-to-end speech recognition. pages 5666–5670. – Cité pages 14, 15, 26 et 28.

[Moritz et al., 2020] Moritz, N., Hori, T., and Le Roux, J. (2020). Streaming automatic speech recognition with the transformer model. pages 6074–6078. – Cité pages 14, 28 et 30.

[Park et al., 2019] Park, D., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E., and Le, Q. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. pages 2613–2617. – Cité page 22.

[Peyser et al., 2019] Peyser, C., Zhang, H., Sainath, T., and Wu, Z. (2019). Improving performance of end-to-end asr on numeric sequences. pages 2185–2189. – Cité page 36.

[Povey et al., 2011] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., and Vesel, K. (2011). The kaldi speech recognition toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. – Cité pages 6, 48 et 49.

[Povey et al., 2018] Povey, D., Hadian, H., Ghahremani, P., Li, K., and Khudanpur, S. (2018). A time-restricted self-attention layer for asr. pages 5874–5878. – Cité page 24.

[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. pages 939–943. – Cité pages 14 et 30.

[Pusateri et al., 2017] Pusateri, E., Ambati, B., Brooks, E., Platek, O., McAllaster, D., and Nagesha, V. (2017). A mostly data-driven approach to inverse text normalization. pages 2784–2788. – Cité page 37.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. – Cité page 32.

[Raffel and Ellis, 2015] Raffel, C. and Ellis, D. (2015). Feed-forward networks with attention can solve some long-term memory problems. – Cité page 14.

[Raffel et al., 2017] Raffel, C., Thng, L., Liu, P., Weiss, R., and Eck, D. (2017). Online and linear-time attention by enforcing monotonic alignments. – Cité pages 6, 14, 15, 24 et 25.

[Rybach et al., 2009] Rybach, D., Gollan, C., Heigold, G., Hoffmeister, B., Lööf, J., Schlüter, R., and Ney, H. (2009). The rwth aachen university open source speech recognition system. pages 2111–2114. – Cité page 45.

[Sak et al., 2017] Sak, H., Shannon, M., Rao, K., and Beaufays, F. (2017). Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. pages 1298–1302. – Cité page 15.

[Shugrina, 2010] Shugrina, M. (2010). Formatting time-aligned asr transcripts for readability. pages 198–206. – Cité pages 37, 39, 40 et 41.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. – Cité pages 23 et 28.

[Vlaj and Kacic, 2011] Vlaj, D. and Kacic, Z. (2011). The influence of lombard effect on speech recognition. – Cité page 16.

[Walker et al., 2004] Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P., and Wölfel, J. (2004). Sphinx-4: A flexible open source framework for speech recognition. *Sun Microsystems*. – Cité pages 6 et 47.

[Young et al., 2002] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. (2002). *The HTK book*. – Cité pages 6, 47 et 48.

[Zeyer et al., 2018a] Zeyer, A., Irie, K., Schlüter, R., and Ney, H. (2018a). Improved training of end-to-end attention models for speech recognition. pages 7–11. – Cité page 15.

[Zeyer et al., 2018b] Zeyer, A., Irie, K., Schlüter, R., and Ney, H. (2018b). Improved training of end-to-end attention models for speech recognition. pages 7–11. – Cité page 27.

[Zhang et al., 2017] Zhang, Y., Chan, W., and Jaitly, N. (2017). Very deep convolutional networks for end-to-end speech recognition. pages 4845–4849. – Cité page 22.

ANNEXES

Sommaire

A.1 Liste des abréviations

AM	Acoustic Model
ASR	Automatic Speech Recognition
BLSTM	Bidirectional LSTM
BN	Batch Normalization
BPE	Byte Pair Encoding
BLSTM	Bidirectional LSTM
BRNN	Bidirectional RNN
CBA	Content Based Attention
CDP	Context Dependant Phonem
CER	Character Error Rate
CNN	Convolution Neural Network
ConvLSTM	Convolutional LSTM
CSJ	Corpus of Spontaneous Japanese
CTC	Connectionist Temporal Classification
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
E2E	End To End
EOS	End Of Sentence
FLAC	Free Lossless Audio Codec
FSDD	Free Spoken Digit Dataset
Fsh	Fisher
FST	Finite State Transducer
GMM	Gaussian Mixture Model
GPT	Generative Pretrained Transformer
HKUST	Hong Kong University of Science and Technology
HMA	Hard Monotonic Attention
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
ITN	Inverse Text Normalization
IWER	Individual Word Error Rate

LA	Location Aware
LAS	Listen Attend and Spell
LM	Language Model
LMA	Language Model Adaptation
LPC	Linear Prediction Coefficient
LPCC	Linear Prediction Cepstral Coefficient
LSTM	Long Short Term Memory
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Frequency Cepstral Coefficient
MHA	Multi Head Attention
ML	Machine Learning
MoChA	Monotonic Chunkwise Attention
MT	Multi Tasking
MTL	Multi Task Learning
MTR	Multicondition Training Data
MTS	Mandarin (Chinese Conversational) Telephone Speech (Recognition)
MWER	Minimum Word Error Rate
NE	Named Entities
NEER	Numéric Entity Error Rate
NER	Named Entity Recognition
NIN	Network In Network
NLP	Natural Language Processing
NT	Neural Transducer
OOV	Out Of Vocabulary
PER	Phonem Error Rate
QR	Question&Réponse
RAP	Reconnaissance Automatique de la Parole
ResNets	Residual Networks
RNN	Recurrent Neural Network
RNN-T	RNN Transducer
RWTH	Rheinisch-Westfälische Technische Hochschule
SA	Soft Attention
Seq2Seq	Sequence To Sequence
SER	Sentence Error Rate
SOS	Start Of Sentence
SRAP	Système de Reconnaissance Automatique de la Parole
SS	Scheduled Sampling
SwB	Switch Board
TA	Triggered Attention
TACont	TA Content Based
TALoc	TA Location Aware Attention
TIMIT	Texas Instrument Massachusetts Institute of Technology
TTS	Text To Speech
WAV	Waveform Audio File Format
WDNC	Written Domain Neural Correction
WER	Word Error Rate
WFST	Weighted FST
WPM	Word Piece Models
WSJ	Wall Street Journal

A.2 Scripts

A.2.1 Pré-traitements

csv_to_text.py

```
import csv

punct=[".", "!", "?", ","]

with open("transcriptions.txt", "r") as fileIn,
    open("./data/train/text","w") as fileOutTrain,
    open("./data/test/text","w") as fileOutTest,
    open("./data/corpus.txt","w") as fileOut2:
        csv_file = csv.reader(fileIn, delimiter="\t")
        next(csv_file)
        for line in csv_file:
                utt_id = line[0]
                spk = line[1]
                txt = line[2]
                for words in txt:
                        if words in punct:
                                txt = txt.replace(words, " "+words+" ")
                if spk == "BE" or spk == "pencil":
                        fileOutTest.write(spk+"_"+utt_id+"\t"+txt+"\n")
                if spk == "CK" or spk == "Delian" or spk == "rhys_mcg"
                    or spk == "Susan1430":
                        fileOutTrain.write(spk+"_"+utt_id+"\t"+txt+"\n")
                fileOut2.write(txt+"\n")
```

get_filepath.py

```
from pathlib import Path

directory = Path("./audio")

with open("./data/train/wav.scp", "w") as WavTrain, open
    ("./data/train/utt2spk","w") as uttTrain,
    open("./data/test/wav.scp", "w") as WavTest, open
    ("./data/test/utt2spk","w") as uttTest:
        for child in directory.iterdir():
                # print(child) test/train
                for subchild in child.iterdir():
                        # print(subchild)
                           CK/Delian/rhys_mcg/Susan1430/BE/pencil
                        for sub in subchild.iterdir():
                                # print(sub)
                                if "BE" in subchild.name or "pencil"
                                    in subchild.name:
                                        # print(subchild.name)
                                        WavTest.write(str(subchild.name)+"_"
                                        +str(sub.name)
                                        .split(".")[0]+"\t"+str(sub)+"\n")
                                        uttTest.write(str(subchild.name)+"_"
                                        +str(sub.name)
```

```
                                        .split(".")[0]+"\t"
                                        +str(subchild.name)+"\n")
                                if subchild.name == "CK" or
                                    subchild.name == "Delian" or
                                    subchild.name == "rhhys_mcg" or
                                    subchild.name == "Susan1430":
                                        WavTrain.write(str(subchild.name)+"_"
                                        +str(sub.name).split(".")[0]+"\t"
                                        +str(sub)+"\n")
                                        uttTrain.write(str(subchild.name)+"_"
                                        +str(sub.name).split(".")[0]+"\t"
                                        +str(subchild.name)+"\n")
```

lex_to_phones.py

```
with open("./data/local/dict/lexicon.txt", "r") as lex,
    open("./data/local/dict/nonsilence_phones.txt", "w") as NSphones:
        lop = list()
        for line in lex:
                phones = line.split("\t")[1]
                list_of_phones = phones.split(" ")
                for p in list_of_phones:
                        if p == "\n":
                                pass

                        if "\n" in p:
                                p = p.strip()
                                lop.append(p)
                        else:
                                lop.append(p)
        final_list = set(lop)
        for ele in final_list:
                NSphones.write(ele+"\n")
```

mp3_to_wav.sh

```
#!/bin/bash

for i in *.mp3
do
    sox "$i" "$(basename -s .mp3 "$i").wav"
        rm "$i"
done
```

no_duplicates.sh

```
cut -d ' ' -f 2- data/train/text | sed 's/ /\n/g' | sort -u |tr a-z
    A-Z > words.txt
cut -d ' ' -f 2- data/test/text | sed 's/ /\n/g' | sort -u |tr a-z A-Z
    >> words.txt
sort -u words.txt > words1.txt
```

reduce_csv.py

```
with open("sentences_with_audio.csv", "r") as fileIn, open
    ("transcriptions.txt", "w") as fileOut:
        i=0
        for line in fileIn:
                utt, spk, txt = line.split("\t")
                if spk == "CK":
                        fileOut.write(utt+"\t"+spk+"\t"+txt)
                        i+=1
                        if i == 51:
                                break
                if spk == "pencil":
                        fileOut.write(utt+"\t"+spk+"\t"+txt)
                        i+=1
                        if i == 51:
                                break
                if spk == "BE":
                        fileOut.write(utt+"\t"+spk+"\t"+txt)
                        i+=1
                        if i == 51:
                                break
                if spk == "Susan1430":
                        fileOut.write(utt+"\t"+spk+"\t"+txt)
                        i+=1
                        if i == 51:
                                break
                if spk == "rhys_mcg":
                        fileOut.write(utt+"\t"+spk+"\t"+txt)
                        i+=1
                        if i == 51:
                                break
                if spk == "Delian":
                        fileOut.write(utt+"\t"+spk+"\t"+txt)
```

reduce_audio.py

```
import pathlib
import os

directory = pathlib.Path("./audio")

with open("./data/train/wav.scp", "r") as WavTrain,
    open("./data/test/wav.scp", "r") as WavTest:
        i=0
        trainWav = WavTrain.readlines()
        # print(trainWav)
        testWav = WavTrain.readlines()
        for child in directory.iterdir():
                # print(child) test/train
                for subchild in child.iterdir():
                        # print(subchild)
                           CK/Delian/rhys_mcg/Susan1430/BE/pencil
                        for sub in subchild.iterdir():
                                # print(sub.name) uttID.wav
```

```
                    if sub.name not in trainWav or
                        sub.name not in testWav:
                            sub.unlink()
```

A.2.2 Lancement

Les scripts montrés ci-dessous proviennent du tutoriel de Kaldi (https://kaldi-asr.org/doc/kaldi_for_dummies.html)

cmd.sh

```
# Setting local system jobs (local CPU - no external clusters)
export train_cmd=run.pl
export decode_cmd=run.pl
```

path.sh

```
# Defining Kaldi root directory
export KALDI_ROOT=`pwd`/../..
# Setting paths to useful tools
export
    PATH=$PWD/utils/:$KALDI_ROOT/src/bin:$KALDI_ROOT/tools/openfst/bin:$KALDI_ROOT/
src/fstbin/:$KALDI_ROOT/src/gmmbin/:$KALDI_ROOT/src/featbin/:$KALDI_ROOT/src/lmbin/
:$KALDI_ROOT/src/sgmm2bin/:$KALDI_ROOT/src/fgmmbin/:$KALDI_ROOT/src/latbin/
:$PWD:$PATH
# Defining audio data directory (modify it for your installation
    directory!)
export DATA_ROOT="/home/kitaro/kaldi/egs/digits/digits_audio"
# Enable SRILM
. $KALDI_ROOT/tools/env.sh
# Variable needed for proper data sorting
export LC_ALL=C
```

run.sh

```
#!/bin/bash
. ./path.sh || exit 1
. ./cmd.sh || exit 1
nj=1       # number of parallel jobs - 1 is perfect for such a small
    dataset
lm_order=1 # language model order (n-gram quantity) - 1 is enough for
    digits grammar
# Safety mechanism (possible running this script with modified
    arguments)
. utils/parse_options.sh || exit 1
[[ $# -ge 1 ]] && { echo "Wrong arguments!"; exit 1; }
# Removing previously created data (from last run.sh execution)
rm -rf exp mfcc data/train/spk2utt data/train/cmvn.scp
    data/train/feats.scp data/train/split1 data/test/spk2utt
    data/test/cmvn.scp data/test/feats.scp data/test/split1
    data/local/lang data/lang data/local/tmp
    data/local/dict/lexiconp.txt
echo
```

```
echo "===== PREPARING ACOUSTIC DATA ====="
echo
# Needs to be prepared by hand (or using self written scripts):
#
# spk2gender  [<speaker-id> <gender>]
# wav.scp     [<uterranceID> <full_path_to_audio_file>]
# text        [<uterranceID> <text_transcription>]
# utt2spk     [<uterranceID> <speakerID>]
# corpus.txt  [<text_transcription>]
# Making spk2utt files
utils/utt2spk_to_spk2utt.pl data/train/utt2spk > data/train/spk2utt
utils/utt2spk_to_spk2utt.pl data/test/utt2spk > data/test/spk2utt
echo
echo "===== FEATURES EXTRACTION ====="
echo
# Making feats.scp files
mfccdir=mfcc
# Uncomment and modify arguments in scripts below if you have any
    problems with data sorting
utils/validate_data_dir.sh data/train     # script for checking
    prepared data - here: for data/train directory
utils/fix_data_dir.sh data/train          # tool for data proper
    sorting if needed - here: for data/train directory
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/train
    exp/make_mfcc/train $mfccdir
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/test
    exp/make_mfcc/test $mfccdir
# Making cmvn.scp files
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test $mfccdir
echo
echo "===== PREPARING LANGUAGE DATA ====="
echo
# Needs to be prepared by hand (or using self written scripts):
#
# lexicon.txt           [<word> <phone 1> <phone 2> ...]
# nonsilence_phones.txt [<phone>]
# silence_phones.txt    [<phone>]
# optional_silence.txt  [<phone>]
# Preparing language data
utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang data/lang
echo
echo "===== LANGUAGE MODEL CREATION ====="
echo "===== MAKING lm.arpa ====="
echo
loc=`which ngram-count`;
if [ -z $loc ]; then
        if uname -a | grep 64 >/dev/null; then
                sdir=$KALDI_ROOT/tools/srilm/bin/i686-m64
        else
                        sdir=$KALDI_ROOT/tools/srilm/bin/i686
        fi
        if [ -f $sdir/ngram-count ]; then
                        echo "Using SRILM language modelling tool from
                            $sdir"
                        export PATH=$PATH:$sdir
```

```
        else
                        echo "SRILM toolkit is probably not installed.
                                Instructions: tools/install_srilm.sh"
                        exit 1
        fi
fi
local=data/local
mkdir $local/tmp
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt
    -wbdiscount -text $local/corpus.txt -lm $local/tmp/lm.arpa
echo
echo "===== MAKING G.fst ====="
echo
lang=data/lang
arpa2fst --disambig-symbol=#0 --read-symbol-table=$lang/words.txt
    $local/tmp/lm.arpa $lang/G.fst
echo
echo "===== MONO TRAINING ====="
echo
steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train data/lang
    exp/mono  || exit 1
echo
echo "===== MONO DECODING ====="
echo
utils/mkgraph.sh --mono data/lang exp/mono exp/mono/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd
    "$decode_cmd" exp/mono/graph data/test exp/mono/decode
echo
echo "===== MONO ALIGNMENT ====="
echo
steps/align_si.sh --nj $nj --cmd "$train_cmd" data/train data/lang
    exp/mono exp/mono_ali || exit 1
echo
echo "===== TRI1 (first triphone pass) TRAINING ====="
echo
steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train
    data/lang exp/mono_ali exp/tri1 || exit 1
echo
echo "===== TRI1 (first triphone pass) DECODING ====="
echo
utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd
    "$decode_cmd" exp/tri1/graph data/test exp/tri1/decode
echo
echo "===== run.sh script is finished ====="
echo
```

A.3 Structure des dossiers de travail

A.3.1 FSDD

Cette section montre la structure du dossier digits contenant le corpus FSDD: Figures A.1 et A.2.

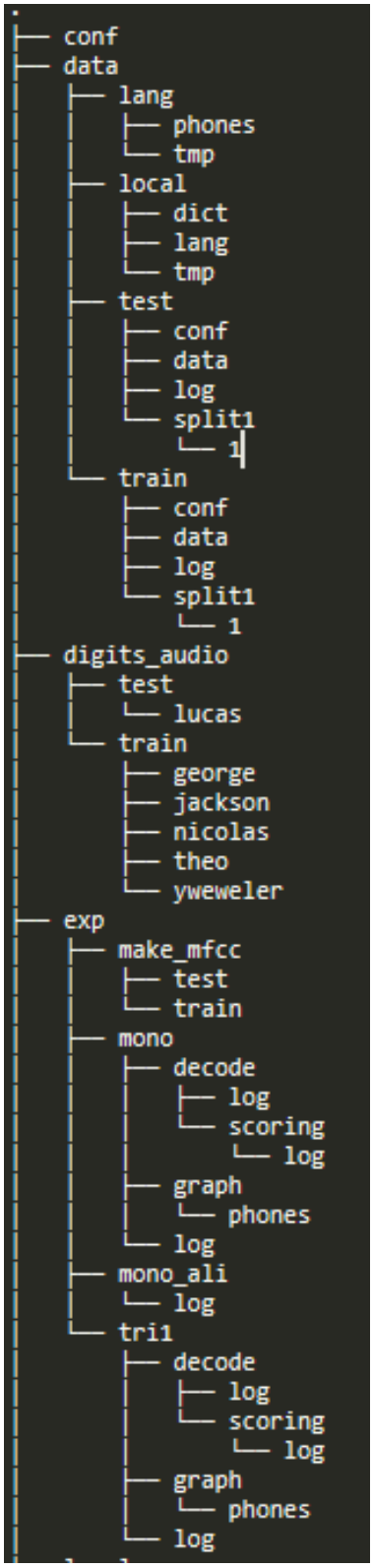


Figure A.1 – Structure du dossier digitis contenant le corpus FSDD

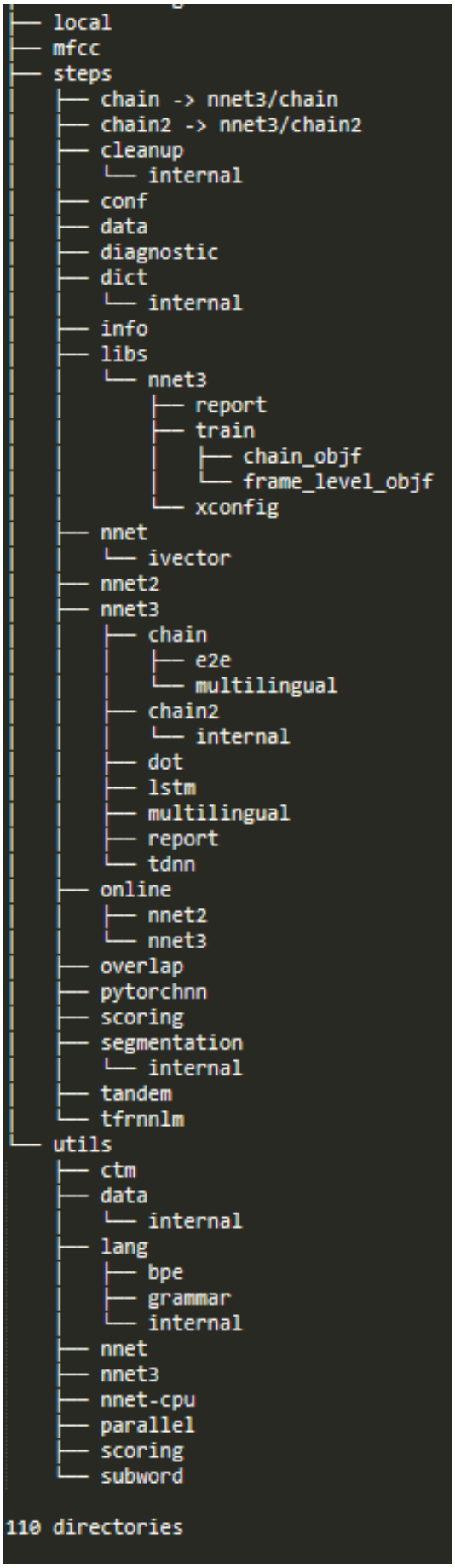


Figure A.2 – Structure du dossier digitis contenant le corpus FSDD

A.3.2 Tatoeba

Cette section montre la structure du dossier tatoeba contenant le corpus Tatoeba: Figure A.3.

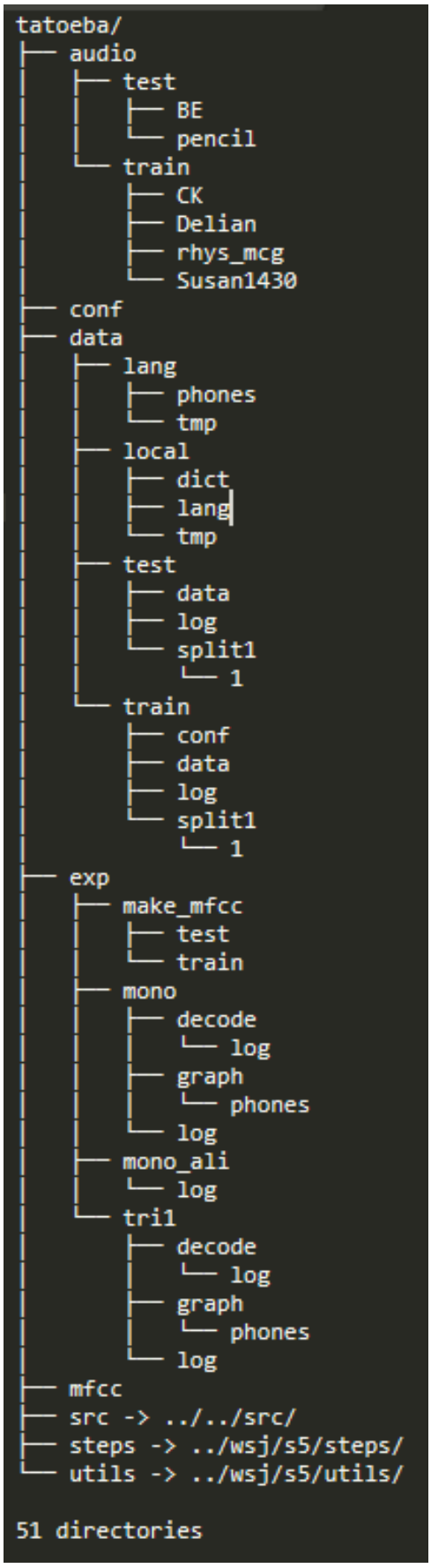


Figure A.3 – Structure du dossier tatoeba contenant le corpus Tatoeba