République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. Mira de Béjaïa
Faculté des Sciences Exactes
Département d'informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master en informatique professionnel

Option : Génie Logiciel (GL)

Thème

Système de reconnaissance de la parole appliqué à la langue Tamazight

---

Encadré par :

Mme. Yaici Malika.
Mme. Aloui Soraya.

Réalisé par :

M. Ouldali Aomer Gaya

Devant le jury composé de :

Président : M. Akilal Abdellah
Examinateur : M. Ait Matene Zahir

2022-2023

Remerciments

Je tiens tout d'abord à remercier dieu pour toutes ses bénédictions.
Ensuite, je remercie mes parents pour tout ce qu'ils ont fait pour moi.
Puis je remercie également mes deux encadrantes, Madame Yaici Malika et Madame Aloui Soraya pour leur sérieux, leur disponibilité ainsi que pour tout le précieux temps qu'elles ont consacré pour l'aboutissement de ce projet.
Enfin, je remercie les membres du jury, Monsieur Akilal Abdellah et Monsieur Ait Matene Zahir pour avoir accepté de m'évaluer.

Table des matières

Liste des figures

List of Tables

Introduction Générale

La reconnaissance automatique de la parole est partout de nos jours. Que ce soit dans les ordinateurs, les téléphones, les voitures et même les maisons avec les assistants virtuels tels que Google Home ou Amazon Alexa.

Son utilité n'est plus à prouver, entre la transcription de la voix, comme pour les sous-titres générés automatiquement sur YouTube, les commandes vocales, pour contrôler les appareils électroniques à distance, sans oublier la traduction en temps réel, qui facilite grandement les voyages dans des pays étrangers. La reconnaissance vocale fait partie intégrante de nos vies.

L'intérêt pour une telle technologie s'est manifesté dès le début des années 1950, on cherchait déjà à extraire des caractéristiques pertinentes de données acoustiques. Plus tard, dans les années 1970, les technologies de l'intelligence artificielle ont été appliqués pour construire des systèmes qui comprennent la parole [3].

Cet intérêt a été motivé notamment par la perspective attirante de comprendre le mécanisme de perception des langages parlés, et ce, afin de mieux comprendre le cerveau humain. Autre que l'assistance médicale en général, un autre domaine d'application potentiel a motivé le développement d'une telle technologie, c'est le domaine industriel. On trouvait très intéressant le fait de pouvoir contrôler des machines à l'aide de commandes vocale.

La technologie utilisée aujourd'hui pour les systèmes ASR (Automatic Speech Recognition) a beaucoup évolué, avec l'explosion de la quantité de données générées par l'Homme, et la puissance de calcul des machines en constante évolution, des technologies telles que le Deep Learning sont appliquées à la parole, et de nouveaux modèles sont constamment inventés.

C'est donc d'autant plus étonnant qu'il n'y ait pas de système d'ASR (Automatic Speech Recognition) pour notre langue, Tamazight. Un tel système pourrait être très utiles pour déclencher diverses commandes vocales, notamment pour les personnes agées. C'est donc cela qui a motivé ce travail.

Notre mémoire est organisé comme suit :

Dans le premier chapitre, nous expliquerons les différents traitements électroniques des sons en général et de la voix en particulier.

Dans le deuxième chapitre, nous présenterons les différentes approches d'ASR, de quelques notions en rapport avec l'ASR et les différentes applications des systèmes ASR.

Dans le troisième chapitre, nous allons donner la réalisation de notre application.

Enfin, nous allons terminer notre mémoire par une conclusion et quelques perspectives.

Chapitre 1 : Traitement numérique du son

Introduction

La voix est le principal moyen de communication chez l'Homme. Pour lui, parler est aussi naturel qu'intuitif, ce qui n'est pas le cas pour les machines. Ces dernières ont quant à elles été conçues pour manipuler les nombres. Pour être manipulé par un ordinateur, le son doit d'abord être converti en signal numérique. Et pour être utilisé pour la reconnaissance de la parole, il doit en général subir une série de transformations afin d'en extraire les parties pertinentes. Dans ce chapitre, nous allons présenter les étapes du traitement de la voix, depuis sa conversion en signal numérique jusqu'à l'extraction de caractéristiques.

1.1 Numérisation du son

Le son est une onde mécanique qui se propage dans les solides et les fluides, comme l'air notamment. Cette onde est caractérisée par une amplitude et une fréquence qui varient continuellement. L'amplitude mesure la force du son, c'est-à-dire son volume. La fréquence, quant à elle mesure le ton (la hauteur) de ce dernier. L'unité de mesure du volume est le décibel, tandis que l'unité de mesure de la fréquence est le hertz ou le kilohertz (kHz) [4].
Récupéré par le microphone sous forme de signal électrique analogique, le son doit d'abord passer par un CAN (convertisseur analogique numérique) afin d'être converti à un format numérique et pouvoir être manipulé par un ordinateur [5].

1.1.1 Encodage des données audio

La méthode la plus courante pour la conversion d'un signal analogique en un signal numérique est la modulation PCM (Pulse Code Modulation). Le signal est échantillonné à une fréquence prédéterminée et chaque échantillon est représenté sous forme de code en binaire [6].
Le codage PCM commence donc par l'échantillonnage, suivi par la quantification et enfin le codage [7].

a - Échantillonnage

Selon [7] : "Echantillonner signifie prendre des valeurs instantanées du signal analogique à des intervalles de temps égaux". L'échantillonnage est donc la substitution d'un signal audio électrique analogique, par une suite d'échantillons ponctuels dont on ne retiendrait que la valeur numérique, voir figure 1.1. Le nombre d'échantillons que l'on retiendrait par seconde est appelé fréquence d'échantillonnage, elle est mesurée en kHz.

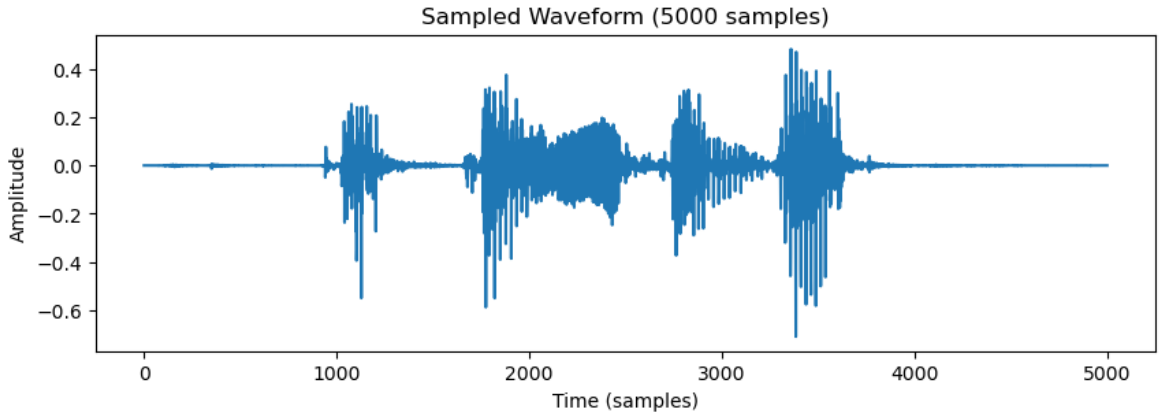

Figure 1.1: Forme d'onde échantillonnée.

b - Quantification

On divise les valeurs continues de l'amplitude en un nombre fini de plages, on attribue ensuite à chaque plage une valeur discrète. Puis, on attribue à chaque échantillon qui se situe dans une plage une valeur discrète selon la plage dans laquelle il se situe. Donc, tous les échantillons se situant dans le même intervalle d'amplitude reçoivent la même amplitude de sortie et cette dernière peut être stockée numériquement [7].

c - Encodage

Chaque échantillon est encodé sur un nombré préalablement fixé de bits appelé taille d'échantillon ou bien profondeur d'encodage.

1.1.2 Filtrage numérique d'un signal audio

Un filtre audio est un médium à travers lequel un signal audio passe et qui entraîne une modification quelconque de ce signal. Tous les filtres audio peuvent être simulés par un filtre numérique (qui lui opère sur un signal audio numérique) avec un certain degré de précision. Un filtre numérique peut aussi bien être sous forme logique (d'un bloc de code informatique), ou bien sous forme matérielle (une série de circuits interconnectés) [8].

a - Ré-échantillonnage

Les taux d'échantillonage utilisés diffèrent d'une application à une autre. Mais, pour connecter deux systèmes qui fonctionnent avec deux taux d'échantillonnage différents, il peut être nécessaire de faire une conversion, voir figure 1.2.

Lorsque la fréquence d'échantillonnage cible est plus grande que la fréquence d'échantillonnage source, l'opération s'appelle sur-échantillonnage (Upsampling en anglais). Après l'upsampling, un filtre anti-repliement (Anti-imaging en anglais) est appliqué, et ce, pour éviter la

distorsion, un effet audio indésirable. Lorsque la fréquence d'échantillonnage cible est moins grande que la fréquence d'échantillonnage source l'opération s'appelle sous-échantillonnage (Downsampling en anglais). Pour éviter la distorsion, un filtre anti-aliasing est appliqué après le downsampling [9].

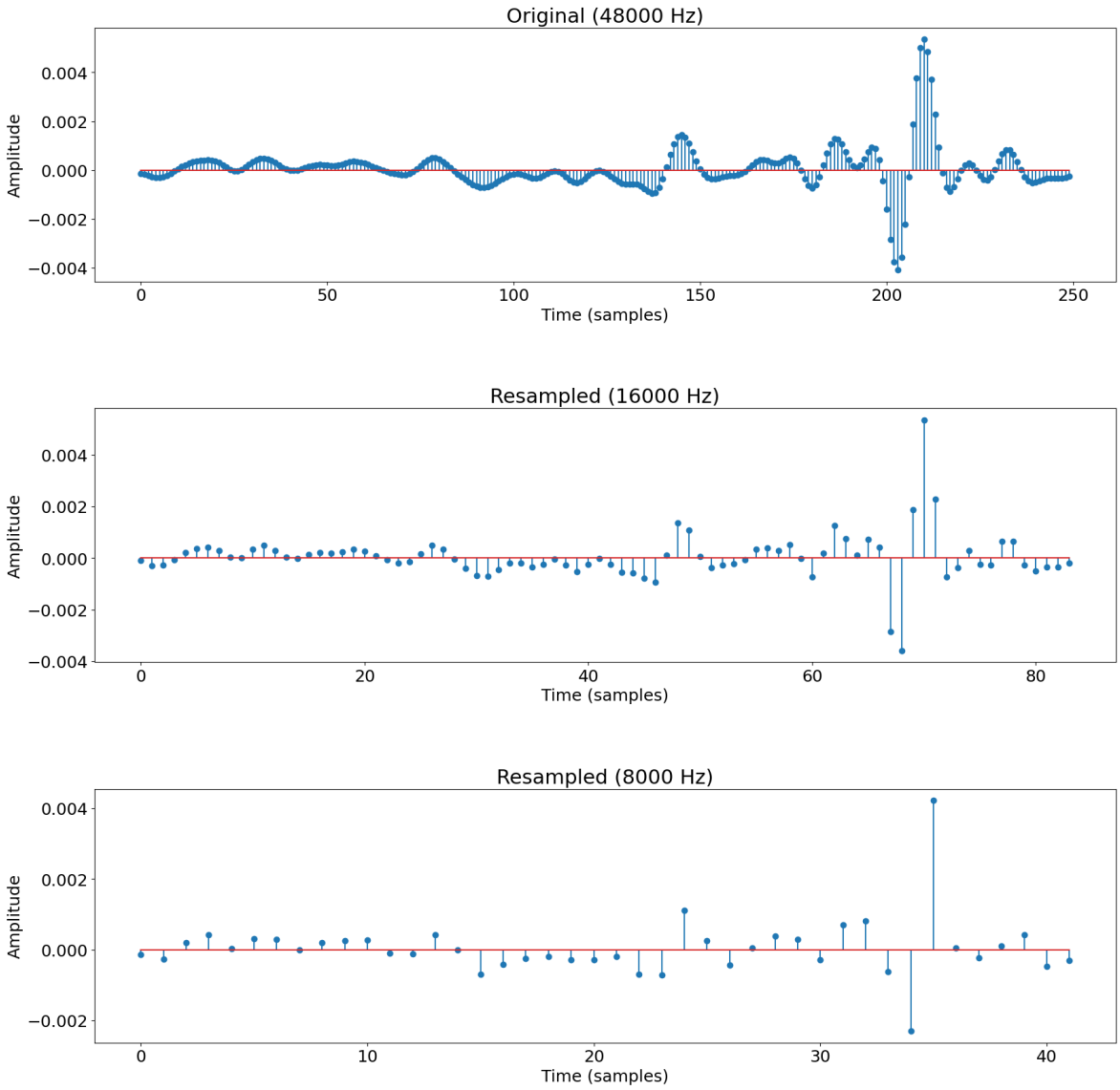

Figure 1.2: Forme d'onde échantillonnée en 48kHz, ré-échantillonnée en 16kHz et 8kHz.

1.1.3 Qualité du son numérisé

Plus la fréquence d'échantillonnage et la taille d'échantillon sont élevés, plus le son numérisé se rapprochera de la qualité du son original.
D'après le théorème de Shannon, pour avoir un signal numérique sans perte d'information, la fréquence d'échantillonnage doit être deux fois supérieure à la fréquence la plus élevée du signal analogique.

En pratique, la numérisation est une histoire de compromis. La perfection de la conversion du signal analogique en numérique est sacrifiée pour cause de contraintes

techniques et économiques. Par exemple, pour les CD audio (Compact disc audio), la norme d'échantillonnage est de 44.1 kHz avec des échantillons codés sur 16 bits, alors que l'association internationale des archives sonores et audiovisuelles (IASA) recommande au moins une fréquence de 48 kHz et une taille d'échantillon de 24 bits pour des données audio de qualité [10] [6].

1.1.4 Formats audio et compression

Selon l'application et le niveau de précision requis, plusieurs formats d'audio sont utilisés. D'abord, il y a les formats sans compression. Ces derniers sont généralement plus volumineux, mais garantissent un niveau maximal de qualité audio. Ensuite, il y a les formats compressés, qui offrent un compromis entre la qualité et la taille des données à stocker ou à transmettre [9].

a - Formats de fichiers audio

Les deux formats standards les plus courants sont le WAV (Waveform audio format) et le MP3 (MPEG-1 Audio Layer 3). Bien que la qualité de son du MP3 reste élevée, celui-ci contient 10 fois moins d'informations que le WAV grâce à un algorithme de compression spécial. Le nombre de bits par secondes que contient un fichier audio est appelé le débit binaire (bitrate en anglais), plus celui ci est élevé, meilleure sera la qualité et plus le fichier audio sera volumineux [4].
On distingue donc deux types de formats de fichiers audio, les formats compressés, comme le format MP3 et les formats non compressés comme le format WAV [9].

b - Compression de données audio

L'utilisation de l'audio dans divers domaines a engendré l'invention de diverses méthodes de compression de données audio, que ce soit pour le stockage ou la transmission de ces dernières. Certaines de ces méthodes sont même devenues des standards internationaux. L'encodage audio peut être divisé en deux catégories : l'encodage à perte (Lossy en anglais) et l'encodage sans perte (Lossless). L'objectif des deux méthodes est de réduire la taille des données [9].

Encodage Lossless Ce type d'encodage est basé sur des modèles statistiques. Au moment du décodage, le signal audio est reconstitué à l'identique par le décodeur audio. Il n'y a donc aucune perte de données au moment de la compression.

Encodage Lossy L'encodage à perte est basé sur des modèles psychoacoustiques de la perception humaine du son pour encoder le signal audio. Cette méthode vise à inclure uniquement les parties pertinentes du signal. Les échantillons du son original ne sont

pas reconstruits à l'identique. La méthode de compression lossy permet d'atteindre des taux de compression bien plus élevés que la méthode lossless.

La méthode lossy cause certains problèmes, mais les taux de compression élevés associés à cette méthode justifie ses nombreuses applications, dont la transmission.

1.2 Extraction de caractéristiques

Pour pouvoir reconnaitre la parole, les systèmes ASR utilisent en général des caractéristiques extraites à partir de l'audio. Pour ce faire, diverses méthodes existent, chacune avec ses avantages et ses lacunes, et chacune est plus utile dans un scénario plutôt que dans un autre. Parmi ces méthodes, on trouve le spectrogramme et les MFCCs (Mel-frequency cepstral c oefficients ou coefficients cepstraux en fréquence de Mel en français) entre autres.
Les avancées récentes ont tout de même permis de concevoir un modèle qui utilise l'audio brut, mais celui-ci nécessite une très grande quantité de données et beaucoup de puissance de calcul pour être entrainé.
Les caractéristiques extraites peuvent ensuite être utilisées pour entrainer un algorithme d'apprentissage automatique tel que les modèles de Markov cachés ou bien les réseaux de neurones profonds. Une fois le modèle entrainé, celui-ci peut transcrire du texte à partir des mêmes caractéristiques avec lesquelles il a été entrainé, mais cette fois extraites de données nouvelles jamais vues au cours de l'entrainement [11].

Dans les approches traditionnelles, il est commun d'utiliser les MFCCs pour entrainer les modèles. Mais, dans l'approche basée sur le deep learning il est commun de ne pas utiliser les caractéristiques conçues manuellement, car utiliser directement le spectrogramme donnerait des résultats supérieures. Cela s'explique par la capacité des réseaux de neurones profonds à extraire des caractéristiques qui ne peuvent pas être directement utilisées dans les systèmes traditionnels. En plus de ça, ils ont la capacité à réaliser des approximations de l'extraction de caractéristiques importantes, et de ce fait, il est possible de retirer certaines étapes de prétraitement habituellement effectuées dans les systèmes traditionnels sans compromettre la précision [12].

1.2.1 Transformée de Fourrier

La transformée de Fourrier est un concept mathématique qui permet à un signal de passer du domaine du temps au domaine de fréquences, tout en donnant la magnitude (l'intensité) de chaque fréquence, voir figure 1.3. La transformée de Fourrier inversée peut être utilisée pour synthétiser un signal transformé avec la transformée de Fourrier et retrouver le signal original.

La transformée de Fourrier rapide (Fast Fourrier Transformation notée FFT) est un algorithme qui permet d'appliquer la transformée de Fourrier discrète (Discrete Fourrier Transform, notée DFT) sur un signal numérique [13].

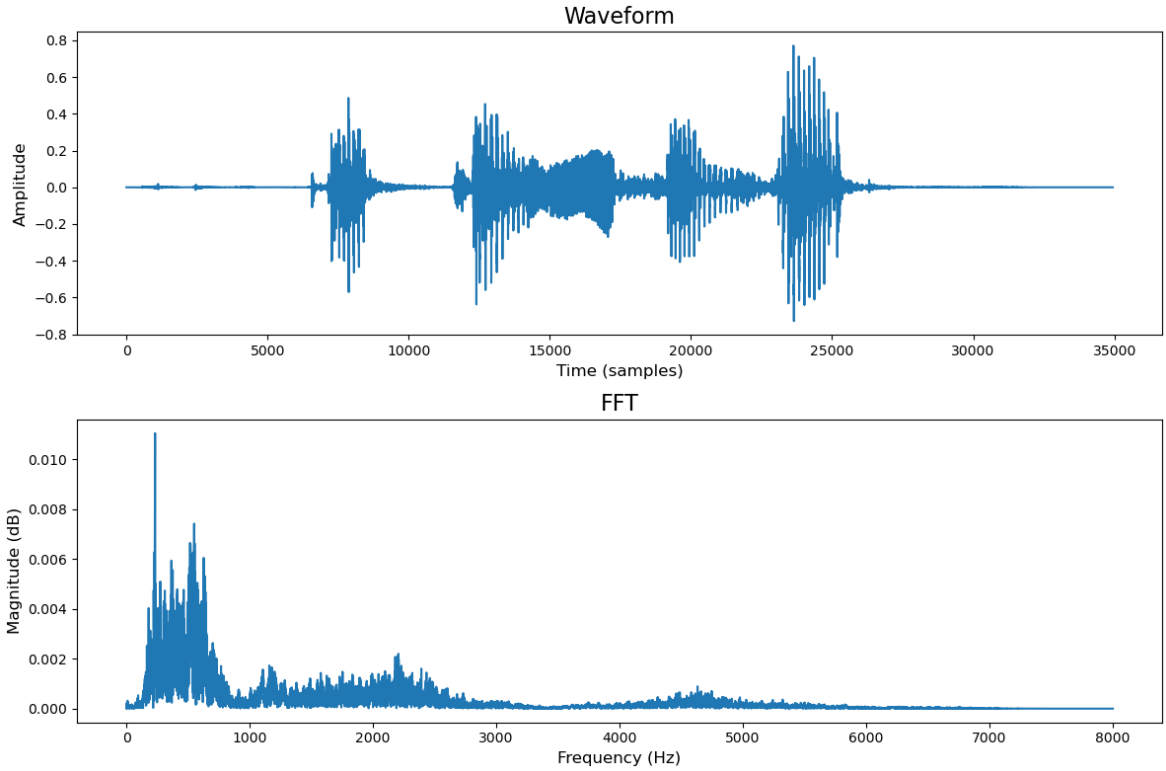

Figure 1.3: Forme d'onde et sa transformation avec FFT.

1.2.2 Forme d'onde brute

Cette approche vise à pré-entrainer un système ASR sur des audios bruts non étiquetés. Cette méthode est très utile, notamment pour les langues avec peu de ressources, Car les données non étiquetées sont plus facilement trouvables et en quantité supérieure par rapport aux données étiquetées.

Après le pré-entrainement, il est commun de faire un fine-tuning du modèle sur des données étiquetées [14].

1.2.3 Spectrogramme

Selon [15] : "Un spectrogramme est considéré comme une représentation très détaillée et précise de l'information audio. Un spectrogramme est une image où un axe représente le temps, l'autre axe représente la fréquence et la couleur de chaque point indique l'amplitude."

FFT permet au signal de passer au domaine de fréquences, mais cause la perte des informations concernant le temps [16]. Or, pour déterminer l'ordre des mots, un

système ASR a besoin de ces informations. C'est donc pour ça que les spectrogrammes sont utilisés.
Un spectrogramme est construit en découpant le signal en fenêtres, et pour chaque fenêtre, on calcule la DFT, voir figure 1.4. De ce fait, chaque fenêtre est composée de fréquences et l'ordre des fenêtres représente le temps.
Pour une tâche d'ASR typique, une taille de fenêtre de 20 à 30 millisecondes est recommandée pour ne pas perdre aucun phonème prononcé. Le chevauchement entre deux fenêtres peut varier, mais 50% est une valeur communément utilisée pour l'ASR.

Utiliser des spectrogrammes comme caractéristiques reviens à utiliser une matrice de deux dimensions, ce qui facilite la tâche de la reconnaissance de la parole comparé à l'utilisation de la forme d'onde brute [16].

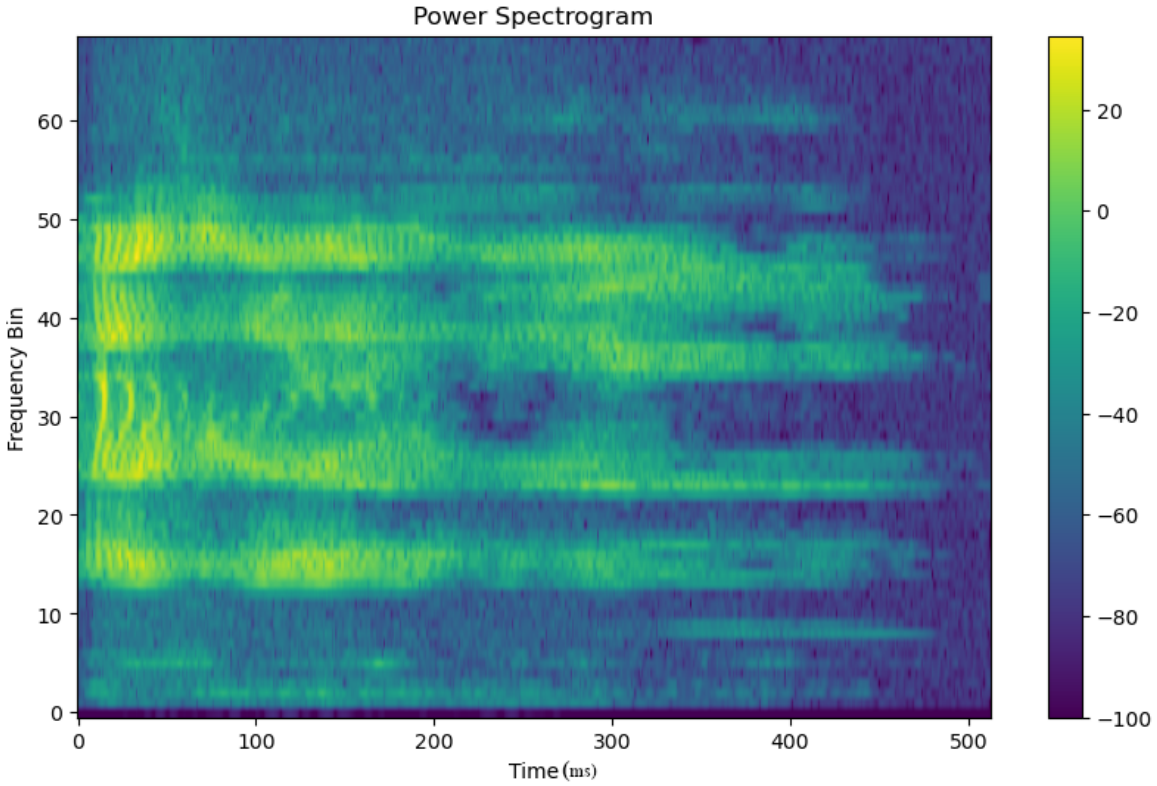

Figure 1.4: Spectrogramme de spectre de puissance.

1.2.4 Mel-Spectrogramme

Alors que le spectrogramme représente la fréquence en fonction du temps, le Mel-spectrogramme est basé sur l'échelle de Mel. Cette échelle se base sur la perception auditive humaine. En effet, l'Homme distingue mieux les basses fréquences entre elles comparé aux hautes fréquences. L'échelle de Mel espace donc les basses fréquences linéairement tandis qu'elle espace les hautes fréquences de façon logarithmique. Le Mel-spectrogramme représente le son en utilisant l'échelle de Mel en fonction de l'amplitude en décibels [17] [16].

1.2.5 MFCC

Les coefficients cepstraux en fréquence de Mel (MFCC) sont l'une des techniques d'extraction de caractéristiques les plus largement utilisées en traitement de la parole et en reconnaissance de la parole, ils sont basés sur le système de perception auditive humaine, tout comme le mel-spectrogramme. Cette méthode a l'avantage d'offrir une grande précison et une bonne discrimination ; elle permet de différencier des caractéristiques ou des modèles (patterns) similaires, mais distincts. Elle offre aussi une faible corrélation entre les caractéristiques ; les coefficients sont indépendants les uns des autres et peuvent de ce fait fournir des informations distinctes et complémentaires sur le signal. Par contre, ils sont imprécis dans le cas d'audio avec bruit [17] [18].

Les MFCCs sont surtout utilisés dans l'approche traditionnelle de l'ASR, car c'est une méthode d'extraction conçue manuellement et peut donc réduire les performances d'un modèle de Deep Learning [12].

1.3 Augmentation de données

Les techniques d'augmentation de données ; qui consistent à créer artificiellement de nouvelles données à partir des données disponibles en leurs appliquant toutes sortes d'effets, ont prouvé leur utilité. En effet elles permettent notamment aux langues avec très peu de ressources d'améliorer les taux d'erreur de leurs systèmes ASR considérablement.

Par exemple, une étude a été faite sur trois langues parlées en Inde disposant de peu de ressources ; 40 heures de données audio chacune seulement, l'application de diverses techniques d'augmentation de données a permis une amélioration de leurs systèmes ASR, avec des réductions de 17 à 33% du WER, ou taux d'erreur de mot (en anglais Word Error Rate), pour les trois langues [19].

1.3.1 SpecAugment

Alors que les autres techniques d'augmentation de données habituelles s'appliquent sur la forme d'onde brute, SpecAugment s'applique directement sur les caractéristiques utilisées pour alimenter les réseaux de neurones (typiquement le spectrogramme log-mel).

Cette technique d'augmentation permet aux systèmes ASR d'atteindre des WER plus petits sans quasiment aucun coût en termes de ressources et temps de calculs. Elle peut être appliquée en temps réel, c'est-à-dire au cours de l'entraînement et pas avant comme fait en général pour les autres techniques d'augmentation.

SpecAugment agit sur le spectrogramme log-mel comme si s'était une image en appliquant une combinaison de trois formes de déformations [20].

Time Warping(déformation temporelle en français)

La déformation temporelle (Time Warping) consiste à appliquer une distorsion aléatoire aux bandes de fréquences dans le spectrogramme. Cette technique permet d'étirer ou de contracter une partie du spectrogramme pour simuler des variations dans la vitesse de parole ou dans l'enregistrement. Le paramètre de déformation temporelle peut être ajusté pour contrôler le niveau de distorsion appliqué [20].

Frequency masking (masquage de fréquence)

Le masquage de fréquence (Frequency masking) consiste à masquer certaines bandes de fréquence aléatoirement dans le spectrogramme. Cette technique permet de simuler des situations où certaines fréquences sont manquantes. Les paramètres de masquage de fréquence peuvent être ajusté pour contrôler le nombre de bandes de fréquence masquées [20].

Time masking (masquage de temps)

Le masquage de temps (Time masking) consiste à masquer des trames consécutives aléatoirement dans le spectrogramme. Cette technique permet de simuler des situations où certaines parties de l'enregistrement sont manquantes ou inaudibles. Les paramètres de masquage de temps peuvent être ajusté pour contrôler le nombre de trames masquées [20].

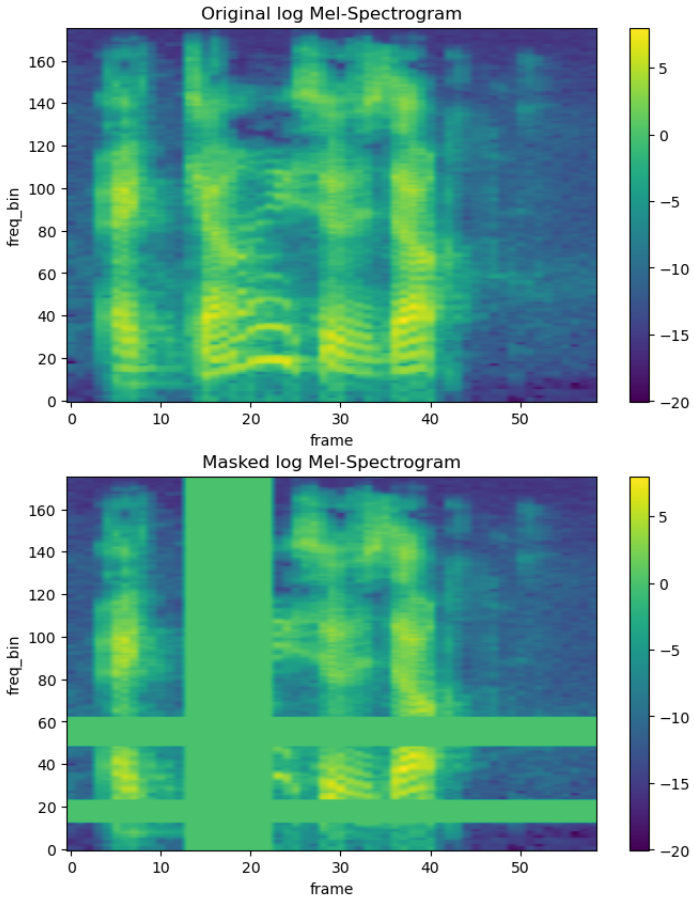

Figure 1.5: Spectrogramme log-mel original avec une version avec Time Masking et Frequency Masking.

Ces trois techniques sont appliquées en étant combinées entre elles, avec un nombre et une intensité aléatoire (avec un nombre maximal et une plage d'intensité préétablis) qui change à chaque époque de l'entraînement. Par exemple, dans la figure 1.5, sont appliqués deux masques de fréquences et un seul masque temporel. De ce fait, l'algorithme entrainé ne verra jamais (ou presque) deux fois le même spectrogramme, ce qui lui permettra d'extraire les caractéristiques pertinentes des spectrogrammes au lieu de se sur-ajuster (overfit en anglais) aux données d'entraînement ; l'overfitting est une situation dans laquelle le modèle mémorise les données d'entrainement au lieu de généraliser les patterns utiles, ce qui résulte par des performances médiocres sur des données nouvelles.
Cette technique permet d'entrainer des réseaux plus grands pendant plus longtemps sans risquer le sur-apprentissage [20].

a - Efficacité de SpecAugment

Selon [20], Cette technique a permis d'obtenir des performances jamais égalées auparavant sur l'ensemble de données LibriSpeech [21] ; un ensemble de données

souvent utilisé dans les études pour développer et tester de nouveaux algorithmes de traitement automatique de la parole, en améliorant la précédente meilleure performance sur ce même ensemble de données de 22% en termes de WER.

1.4 Conclusion

Nous avons expliqué dans ce chapitre les étapes que subit l'audio, qui est au début une onde, avant d'être utilisé par les machines sous forme numérique et subir divers traitements avant d'être utilisé pour construire un système ASR.
Dans le chapitre suivant, nous allons présenter les systèmes ASR ainsi que les différentes approches de ASR.

Chapitre 2 : Les systèmes ASR

Introduction

La Reconnaissance Automatique de la Parole (ASR pour Automatic speech recognition en anglais) est une technologie qui permet aux ordinateurs de transformer la parole en texte. Dans ce chapitre, nous présenterons les divers axes de classification des systèmes ASR. Ensuite, nous aborderons les méthodes les plus courantes pour évaluer les performances d'un système ASR, ainsi que les différents types d'encodage de texte pour les systèmes ASR. Nous parlerons également des différentes phases et composants d'un système ASR, ainsi que des approches possibles pour sa réalisation. Enfin, nous présenterons quelques domaines d'applications des systèmes ASR.
Les généralités concernant le Machine learning et le Deep learning sont détaillées dans l'annexe numéro 1.

2.1 Axes de classification des systèmes ASR

Les systèmes ASR peuvent être classifiés selon différents axes. Nous allons examiner quatre axes, à savoir le mode de discours, le mode de l'orateur, la taille du vocabulaire et le style de parole [22].

a - Mode de discours

Les systèmes ASR peuvent être classés selon le mode de discours, qui peut être soit isolé, soit continu.
Dans le mode de discours isolé, chaque mot doit être prononcé séparément, ce qui facilite la reconnaissance.
Pour ce qui est du mode de discours continu, les mots sont prononcés ensemble, les uns à la suite des autres sans interruption, ce qui rend la tâche de reconnaissance plus difficile, notamment parcequ'il est difficile de délimiter les mots entre eux.

Le mode de discours continu est plus adapté aux situations dans lesquelles les utilisateurs doivent pouvoir parler de manière naturelle, comme pour faire une transcription par exemple, tandis que le mode de discours isolé est plus adapté aux commandes individuelles.

b - Mode de l'orateur

Ce mode de classification divise les systèmes ASR en deux catégories, les systèmes dépendants de l'orateur et les systèmes indépendants de l'orateur.
Dans les systèmes ASR dépendants de l'orateur, l'utilisateur doit enregistrer un échantillon de chaque mot avec sa propre voix avant que ces mots ne puissent être reconnus par le

système.
Les systèmes indépendants de l'orateur en revanche peuvent reconnaitre la voix de n'importe quel locuteur.

Les systèmes dépendants de l'orateur sont dans certains cas plus précis, mais nécessitent un entrainement spécifique pour chaque utilisateur. Tandis que les systèmes indépendants de l'orateur sont généralement plus flexibles, mais peuvent être moins précis.

c - Taille du vocabulaire

La taille du vocabulaire est un autre critère important pour la classification des systèmes ASR.
Les systèmes avec un vocabulaire plus grand sont généralement plus précis, mais plus complexes à construire.
Les systèmes avec un vocabulaire plus petit sont plus simples à construire, mais peuvent ne pas être en mesure de reconnaître tous les mots prononcés par l'utilisateur.

Les systèmes avec un vocabulaire plus grand sont plus adaptés aux applications qui nécessitent la reconnaissance de mots rares ou spécialisés, tandis que les systèmes avec un vocabulaire plus petit sont plus adaptés aux applications qui nécessitent la reconnaissance des mots courants uniquement.

d - Style de parole

Le style de parole est un autre critère de classification des systèmes ASR.
Le style de parole peut être dicté ; l'utilisateur parle lentement et clairement. Il peut également être spontané ; l'utilisateur parle de manière plus naturelle et rapide.
Les systèmes de reconnaissance automatique de la parole qui sont conçus pour le style de parole dicté sont plus précis, car ils peuvent reconnaître les mots individuellement. Mais, ces systèmes ne sont pas bien adaptés à la parole spontanée, qui peut être plus difficile à reconnaître en raison des variations de la prononciation et de l'environnement sonore.

Les systèmes ASR de nos jours

Grâce aux récentes evolutions technologiques dans le domaine de la reconnaissance de la parole, les systèmes ASR modernes sont aujourd'hui très robustes. En effet, ils peuvent généralement fonctionner avec différents modes de parole, différents styles de parole, un vocabulaire large et sont indépendants du locuteur.

2.2 Évaluation des performances des systèmes ASR

Pour pouvoir mesurer les performances d'un système ASR, et donc avoir une estimation de sa précision sur des données nouvelles jamais vues au cours de l'entraînement, diverses unités de mesure de performance (Error metrics) existent. Parmi les plus communes, on trouve le WER (Word Error Rate, ou taux d'erreur de mots en français) et le CER (Char Error Rate, ou taux d'erreur de caractères en français) [22].

Word Error Rate (WER)

Cette unité de mesure est communément utilisée pour la mesure de performances de systèmes ASR. Elle indique simplement le poucentage de mots incorrectement prédits par le système ASR. Plus le WER est bas, meilleur sont les performances du système. Pour calculer ce taux d'erreur, trois genres d'erreurs sont pris en compte.

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \tag{1}$$

où S est le nombre de substitutions, D est le nombre de suppressions, I est le nombre d'insertions, C est le nombre de mots corrects et N est le nombre total de mots (c'est-à-dire S + D + C).
Exemple : le système donne en sortie "yugi ad iyi-d-terr" au lieu de "tugi ad iyi-d-terr", le WER de cette phrase est de 33.33%, c'est-à-dire qu'un mot sur trois a été incorrectement prédit.

Substitutions : Les mots qui apparessent à la place des mots qui devraient être prédits.
Exemple : le système donne en sortie "ssenset tafat" au lieu de "ssens tafat", Le mot "ssenset" est apparu à la place de "ssens", le WER est donc de 50% pour cette phrase.

Suppressions : Les mots qui devraient apparaître mais qui sont absents dans la prédiction.
Exemple : le système donne en sortie "yiwen n wass" au lieu de "yiwen n wass ad d-tass", les deux mots "ad" et "d-tass" sont manquants, le WER est donc de 40% pour cette phrase.

Insertions : Les mots qui sont en plus dans la prediction.
Exemple : le système donne en sortie "d acu ara nečč ass-a" au lieu de "d acu ara nečč", le mot "ass-a" est en plus, le WER est donc de 20% pour cette phrase.
Cette dernière regle fait que le WER peut dépasser les 100%, par exemple, si le système prédit "wagi d lḥal n udfel" au lieu de "wagi", le WER serait de 200% [23].

Char Error Rate (CER)

Cette unité de mesure est communément utilisée pour la mesure de performances de systèmes OCR (Optical Character Recognition) ou de reconnaissance de la parole. Elle indique simplement le pourcentage de caractères incorrectement prédits par le système. Plus le CER est bas, meilleures sont les performances du système [23].
Pour calculer ce taux d'erreur, trois genres d'erreurs sont pris en compte, tout comme le WER, mais cette fois par rapport aux caractères et non aux mots.

$$CER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \tag{2}$$

où S est le nombre de substitutions, D est le nombre de suppressions, I est le nombre d'insertions, C est le nombre de mots corrects et N est le nombre total de caractères.
Exemple : le système donne en sortie "tugiadd iyi-d-terr" au lieu de "tugi ad iyi-d-terr", le CER de cette phrase est de 14.29% [23].

2.3 Types d'encodage de texte pour les systèmes ASR

Le but des systèmes ASR est de produire une séquence de graphèmes, donc, du texte, à partir d'audio donné en entrée. Les systèmes traditionnels utilisent des phonèmes pour modéliser les unités acoustiques.
Mais les systèmes End-to-End (définis dans la séction 2.5.3) permettent d'atteindre des performances similaires aux systèmes traditionnels sans utiliser les phonèmes. Un graphème est une unité graphique qui représente la plus petite entité d'un système d'écriture, tandis qu'un phonème est une unité orale [24].

Pour segmenter le texte, différentes options sont disponibles : segmenter le texte en mots entier, en caractères, ou en sous-mots avec différentes options intermédiaires [25], voir tableau 2.1.

Type d'encodage	Texte encodé
graphèmes	ḥ | e | s | s | e | s
phonèmes	ḥ | ə | s | ə | s
Sous-mots exemple 1	ḥess | es
Sous-mots exemple 2	ḥe | sses

Tableau 2.1: Différents exemples d'encodages pour le mot "ḥesses".

Mots entiers

L'utilisation de mots entiers pour segmenter le texte donne de bons résultats si les données d'entrainement sont suffisantes. Mais, l'inconvénient de cette option est que le système ASR sera incapable de transcrire les mots OOV (out-of-vocabulary, ou mots hors vocabulaire en français). Les mots à reconnaître doivent impérativement figurer dans l'ensemble fini de mots composant les données d'entrainement [25].

Caractères

Aux débuts de l'approche End-to-End, on segmentait le texte en caractères. Mais cette approche nécessite l'utilisation d'un modèle de langue pour que les mots soient correctement orthographiés. Ce problème peut être résolu avec l'utilisation des sous-mots [25].

Sous-mots

Aujourd'hui, les sous-mots sont considérés comme étant la norme pour l'ASR End-to-End. En effet, ils permettent de dépasser la segmentation en caractères en termes de performances sur une large variété d'architectures de systèmes ASR.

Un des algorithmes utilisés pour la segmentation en sous-mots est Byte Pair Encoding (BPE). BPE est une méthode de compression de données qui permet de remplacer les séquences de caractères fréquentes par des sous-unités appelées tokens [25].

2.4 Phases et modules de l'ASR

La reconnaissance automatique de la parole implique deux phases : la phase d'entrainement et le phase de reconnaissance. Durant la phase d'entrainement, des fichiers audio

composée de parole connue sont enregistrés et pré-traités avant d'en extraire les caractéristiques. Ensuite, le modèle est créé, entrainé puis stocké. Pour la phase de reconnaissance, le modèle acoustique analyse les caractéristiques extraites de signaux audio qui lui sont inconnus avant de les convertir en texte [22].

2.4.1 Modules d'un système ASR

Les systèmes ASR peuvent être composés de plusieurs modules. Dans l'approche traditionnelle, chacun de ces composants doit être construit séparément, mais dans l'approche End-to-End, certains voir tous ces modules sont combinés en un seul module.

Modèle acoustique

Le modèle acoustique est le composant principal d'un système ASR, il sert à reconnaitre les phonèmes ou les graphèmes à partir de caractéristiques extraites d'audio inconnus. Lors de sa création, il est entrainé avec des enregistrements audio et leurs transcriptions [25].

Lexicon

Un lexicon (dictionnaire de prononciation en français), est un ensemble de correspondances entre une série de phonèmes et des mots. Le lexicon procure au modèle acoustique la décomposition des mots en phonèmes. Diverses prononciations des mots sont fournies afin de délimiter les mots valides pour la reconnaissance.

Modèle de langue

Un système ASR développe différentes hypothèses de séquences de sortie pour une seule séquence en entrée. C'est pourquoi il est commun d'utiliser un modèle de langue pour prédire la probabilité d'une séquence donnée de mots. Parmi les modèles utilisés, on trouve les modèles n-gram, qui, basé sur les n-1 mots, prédisent la probabilité du n-ième mot.

2.5 Approches de l'ASR

Il y a trois approches principales pour l'ASR : l'approche traditionnelle, hybride et End-to-End. L'approche traditionnelle utilise des modèles statistiques pour transcrire l'audio, tandis que l'approche hybride combine des modèles statistiques et les réseaux de neurones profonds, et enfin, l'approche End-to-End est basée sur les réseaux de neurones profonds.

2.5.1 Approche traditionnelle

L'approche traditionnelle (souvent appelée "conventional approch" en anglais) est une approche basée sur les statistiques. Étant donné un signal de parole, le but de ce système est de trouver la séquence de mots la plus probable.

Dans cette approche, il y a 3 composants principaux. Le premier étant un modèle acoustique, qui, produit des phonèmes à partir des caractéristiques extraites d'un signal audio. Ces phonèmes sont ensuite transformés en mots grâce à un lexicon, qui établis la correspondance entre les phonèmes et les mots. Enfin, le modèle de langue prédit la séquence de mots la plus probable, voir figure 2.1 [12].

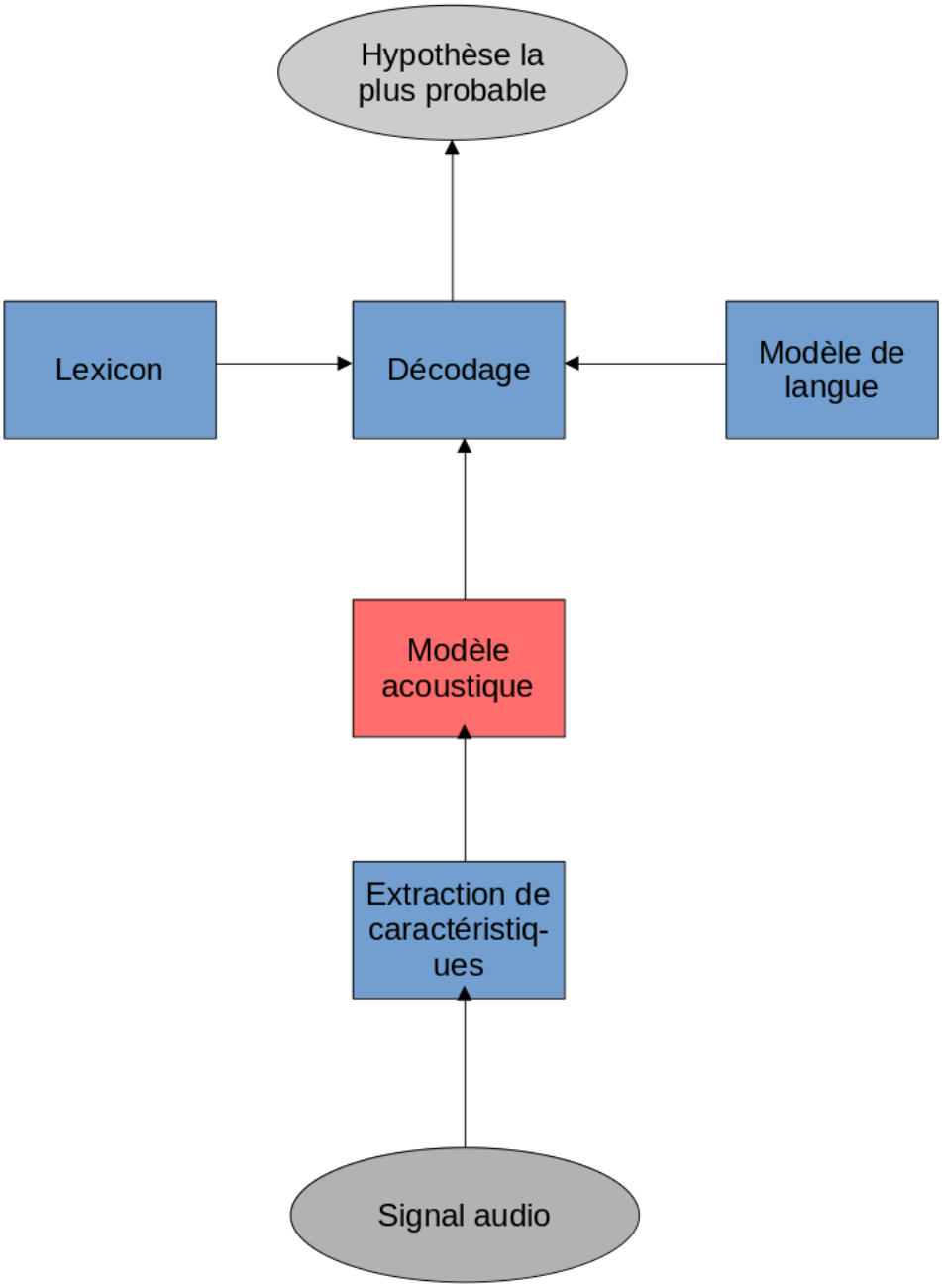

Figure 2.1: Diagramme représentant un système ASR traditionnel [1].

Dans l'approche traditionnelle, la modélisation acoustique se fait grâce à deux modèles probabilistes appelés HMM (Hidden Markov Models) et GMM (Gaussian Mixture Models). Le HMM est entrainé pour modéliser les caractéristiques acoustiques. Ensuite le GMM est entrainé pour modéliser les états du HMM et donc produire une suite de phonèmes [26].

2.5.2 Approche hybride DNN-HMM

Cette approche remplace le GMM par un modèle de réseau de neurones profond. Ces derniers ont la capacité de modéliser les représentations acoustiques de façon plus complexe grâce à plusieurs couches neuronales.

Cette approche donne de meilleurs résultats que l'approche HMM-GMM et a été pendant longtemps l'approche donnant les meilleurs résultats parmi toutes les approches. Mais, dans cette approche le rôle joué par le DNN (Deep neural network) est limité, Le HMM est toujours responsable de modéliser les caractéristiques acoustiques, contrairement à l'approche End-to-End qui est purement basée sur les DNNs [12].

2.5.3 Approche End-to-End

L'approche End-to-End, qui utilise les réseaux de neurones, a l'avantage de lier directement la séquence audio avec la transcription sans modélisations intermédiaires, alors que dans les systèmes traditionnels, chaque composant doit être entrainé séparément.

Cette approche, dite tout-en-un, est dotée d'une phase d'apprentissage plus simple et produit directement du texte et non des phonèmes. En plus de ça, puisque ce type de modèle produit des graphèmes, il peut prédire des mots jamais vus au cours de l'entrainement alors que les systèmes de l'approche traditionnelle souffre du problème des OOVs (mots hors vocabulaire) [12].

2.6 Approche End-to-End pour les systèmes ASR

Alors que les systèmes traditionnels d'ASR se composent d'algorithmes multiples et nécessitent des étapes de prétraitement conçues manuellement, "Baidu Research" [27] a proposé en 2014 un système End-to-End appelé Deep Speech qui remplace toutes ces étapes par le deep learning.

Une fois combiné à un modèle de langue, ce système surpasse les méthodes traditionnelles dans les tâches de reconnaissance difficiles, bien qu'il soit beaucoup plus simple à mettre en place.
En effet, avec assez de données d'entraînement, ce type de système permet d'apprendre à être robuste au bruit et à être indépendant de l'orateur.

Grâce au fait qu'il apprend directement des données, ce type de système excelle dans des situations où l'orateur varie et où l'audio est bruité, et ce, sans composant pour l'adaptation à l'orateur et sans filtrage de bruit.

Un système ASR End-to-End est donc un système qui regroupe de multiples modules en un seul réseau.
Son avantage est qu'il n'est plus nécessaire de concevoir divers modules pour représenter divers états intermédiaires, à la place, le système End-to-End fait correspondre directement

une séquence acoustique à une séquence de texte, et ce, sans nécessiter aucun traitement additionnel pour améliorer les performances de la reconnaissance [27].

Le sequence à séquence en général et le ASR End-to-End en particulier se fait à l'aide de modèles de Deep learning appelés Encodeur-Decodeurs.

2.6.1 Encoder-Decoder

Pendant longtemps, les séquences ont posé un challenge aux réseaux de neurones profonds. En effet, le fait que la longueur de la séquence de sortie était inconnue dans certains problèmes a empêché l'utilisation du deep learning dans des tâches telles que la traduction automatique par exemple.
Pour palier à ce problème, l'architecture de modèle Encoder-Decoder a été inventée. Encoder-Decoder est composé de deux parties : l'encodeur et le décodeur.
L'encodeur prend en entrée une séquence de taille quelconque et l'encode en un vecteur de taille fixe.
Le décodeur quant à lui prend en entrée la sortie de l'encodeur et la décode en séquence de sortie [28].

Transformer

Les réseaux de neurones récurrents (RNN) et leurs variantes ont longtemps été utilisées dans le type d'architectures Encoder-Decoder. Mais les RNN ont quelques inconvénients. En effet, du fait que les RNN traitent les entrées de façon séquentielle rend l'utilisation de la parallélisation pour l'entrainement impossible. De plus Les RNN utilisent beaucoup de mémoire, en particulier dans le cas des séquences longues, ce qui limite le traitement par lots (batching).
C'est pour ces raisons que Transformer, un modèle avec une architecture basée sur Encoder-Decoder, a été inventée par Google en 2017. Transformer peut être utilisé pour transformer une séquence d'entrée en une séquence de sortie sans l'utilisation de RNN en se basant uniquement sur les mécanismes de l'attention. L'attention est un mécanisme qui permet au modèle d'accorder plus d'importances aux éléments les plus pertinents d'une séquence, et ce, en attribuant un poids à chaque élément de la séquence [29].

2.6.2 Alignement souple

Pour pouvoir faire correspondre à un audio une séquence de labels non alignée, les modèles End-to-End utilisent l'alignement souple. L'alignement souple fait correspondre à chaque trame audio une distribution de probabilité associée à tous les états possibles, sans qu'il soit nécessaire d'établir une correspondance forcée ou explicite.

Parmi les catégories de modèles End-to-End (selon la méthode d'alignement souple utilisée), nous trouvons ceux basés sur CTC (Connectionist Temporal Classification) et les modèles RNN-transducer [26].

Connectionist Temporal Classification (CTC)

Étiqueter des séquences de données non segmentées, où des données bruitées en entrée sont annotées par des labels discrets tels que des mots ou des lettres, est un problème plutôt commun.
Beaucoup d'applications nécessitent donc la prédiction de séquences de labels à partir de données non segmentées, et, parmi elles, la reconnaissance de la parole.

CTC (Classification temporelle connectioniste) est donc une méthode qui permet d'entrainer un réseau de neurones à étiqueter des données non segmentées [30].

ASR End-to-End basé sur CTC

L'utilisation de CTC pour l'ASR règle deux problèmes. Le premier étant l'alignement des données. Le second étant que le modèle produit directement des séquences de labels au lieu des phonèmes, par exemple, qui sont souvent produits par les méthodes traditionnelles.

CTC est donc est une fonction de perte, qui résout le problème de l'alignement, tout en calculant la perte.

CTC se divise en deux sous-processus, le calcul de probabilité de chemin et l'agrégation de chemin. CTC introduit également un caractère vierge ("-"), ce caractère signifie : aucune sortie.

Dans le processus de calcul de probabilité de chemin, CTC prends en entrée une séquence de caractéristiques (la sortie du décodeur) et agis sur elle et la convertie en séquence de distribution de probabilités grâce à une fonction d'activation appelée softmax.
Cette distribution représente la probabilité de chaque élément du vocabulaire (ensemble des labels possibles) en plus de la probabilité du caractère vierge.
Une distribution de probabilités est donc calculée pour chaque séquence donnée en entrée au modèle.

Le processus suivant, l'agrégation de chemins est utilisé pour combiner certains chemins entre eux pour former une séquence de labels plus courte. En effet, la séquence de sortie dans une situation réelle est beaucoup plus courte que la séquence d'entrée.
Ce processus est constitué de deux opérations. La première étant de combiner les labels identiques consécutifs, par exemple : "a-z-u-u-l" ou "a-a-z-u-u-u-l" deviens "a-z-u-l". La deuxième étape est de supprimer toutes les occurrences du caractère vierge, puisque celui-ci signifie qu'il n'y a rien en sortie. Par exemple, "a-z-u-l" deviens "azul".

Du fait que CTC utilise ce processus, et donc, que de multiples chemins peuvent donner le même résultat, CTC ne nécessite pas que les entrées et les sorties soient alignées. Pour améliorer davantage la précision d'un système ASR basé sur CTC, il est possible d'utiliser un algorithme de recherche en faisceau (beam search) [26].

Beam search

CTC Beam search est un algorithme utilisé pour trouver la transcription la plus probable pour chaque entrée. Son but est de trouver la sortie la plus probable étant donné les sorties antérieures.

CTC beam search commence donc par l'initialisation d'un faisceau (beam) de séquences (un lot de séquences candidates) avec une séquence vide. Ensuite, pour chaque étape, les n séquences les plus probables sont considérés pour être étendues, avec n étant un hyperparamètre appelé taille de faisceau (beam size).
Pour chaque séquence dans le lot, l'algorithme calcule la probabilité de la séquence étendue avec chaque symbole possible à l'étape suivante. Les séquences étendues sont ensuite classées selon leur probabilité et les n séquences les plus probables sont sélectionnées pour la prochaine étape. Ce processus est répété jusqu'à ce que la séquence la plus probable soit trouvée [31].

Il est possible d'ajouter un modèle de langue au niveau des labels (token-level language model) pour estimer la probabilité du n-ième label étant donné les n-1 derniers labels.

Il est également possible de sauvegarder les séquences les plus probables pour ensuite utiliser un modèle de langue pour sélectionner la séquence la plus probable.

CTC et les modèles de langue

CTC considère chaque label dans la séquence de sortie comme étant indépendant des autres labels. De ce fait, CTC ne peux pas modéliser les langues.
C'est pour cela qu'il est commun d'ajouter un modèle de langue à CTC. Cet ajout peut apporter une amélioration considérable des performances du système ASR.

Deux méthodes existent pour utiliser un modèle de langue avec CTC : la méthode second-pass (aussi appelée re-score) et la méthode first-pass.
Dans la méthode second-pass, un algorithme de recherche en faisceau est utilisé pour obtenir un certain nombre de candidats basé sur la distribution de probabilités calculée par CTC. Ensuite, le modèle de langue est utilisé pour sélectionner le candidat le plus probable et déterminer le résultat le plus optimal de la reconnaissance.
La méthode first-pass quant à elle incorpore directement le modèle de langue dans la recherche en faisceau. Cette méthode donne en général de meilleur résulalts et est plus

largement utilisée [26].

RNN-Transducer

RNN-Transducer est une architecture de réseaux de neurones qui utilise les réseaux de neurones récurrents pour des tâches où des séquences en entrée sont transduites (transformées) en séquences en sortie. Parmi ces tâches, on trouve la traduction automatique, la synthèse vocale ou encore la reconnaissance automatique de la parole [32].

ASR End-to-End basé sur RNN-Transducer

RNN-Transducer est, tout comme CTC une méthode d'alignement souple utilisée dans les systèmes ASR End-to-End. La pricipale différence entre les deux est que RNN-Transducer est une architecture de modèle, alors que CTC est une fonction de perte (loss function).
RNN-Transducer présente beaucoup de similitudes avec CTC. Ils visent tous les deux à résoudre le problème de l'alignement dans l'ASR, ils introduisent tous les deux un caractère vierge et ils calculent tous les deux les probabilités de chaque chemin possible avant de faire l'agrégation des chemins pour obtenir la séquence de labels. Cependant, RNN-Transducer présente certains avantages par rapport à CTC. D'abord, CTC considère les sorties comme étant indépendantes les unes des autres. De ce fait, CTC ne peux pas modéliser les langues, contrairement au RNN-Transducers. De plus, la sortie doit être plus courte que l'entrée pour que CTC puisse les faire correspondre. Ce qui rend CTC inutiles dans les cas où la sortie serait plus longue que l'entrée
Cependant, RNN-Transducer présente également certains inconvénients par rapport à CTC. En effet, les modèles RNN-Transducer sont plus complexes en termes de ressources de calculs et sont en général plus difficiles à entrainer [26].

Architecture de RNN-Transducer

L'architecture de RNN-Transducer consiste en trois sous réseaux : un réseau de transcription, un réseau de prédiction et un réseau conjoint.

Le réseau de transcription est un encodeur, il joue le rôle d'un modèle acoustique. Il fait correspondre à une séquence en entrée une séquence de caractéristiques, et donc, modélise les dépendances entre les entrées acoustiques.

Le réseau de prédiction joue le rôle d'un modèle de langue. Il s'agit d'un réseau de neurones récurrent qui modélise les dépendances entre les labels de sortie.

Enfin, le réseau conjoint est responsable de l'alignement entre la séquence en entrée et la séquence en sortie. Il utilise les sorties des réseaux de transcription et de prédiction

pour produire la distribution de probabilité des labels.

RNN-Transducer permet donc de générer une séquence de n'importe quelle longueur, et donc fonctionne même dans les cas où la séquence de sortie est plus longue que la séquence en entrée.
La nature du réseau de prédiction, qui est un RNN, fait que chaque sortie est basée sur les sorties précédentes. De ce fait, il peut modéliser les langues.
Pour finir, le RNN-Transducer permet donc d'entrainer un modèle acoustique et un modèle de langue conjointement puisque le réseau conjoint combine les sorties des deux autres réseaux pour produire le résultat final [26].

2.7 Quelques domaines d'application de ASR

La reconnaissance de la parole offre des avantages qui ne sont pas des moindres. En effet, pouvoir contrôler une machine avec les mains et les yeux libres rend possible l'assistance par des machines dans des situations où il serait difficile d les utiliser, voir impossible dans certains cas. Ce qui a valu à l'ASR des applications dans des domaines divers , allant du domaine médical aux assistants personnels [3].

Assistance médicale

Parmi les utilités de ASR dans le domaine médical, on trouve la rédaction de rapport, la saisie de données et le contrôle par la voix de certains outils dans des situations où les médecins n'auraient pas les mains libres. La parole peut être utilisée par exemple en entrée pour contrôler des instruments pendant une intervention chirurgicale. Les systèmes ASR destinés au domaine médical se doivent donc d'être précis, efficaces et fiables.

Une autre application de ce type de système dans ce domaine est la détection des troubles de la parole. Certains outils basés sur l'ASR permettent aux pathologistes de la parole de détecter certaines maladies avec vitesse et à moindre coût.

La robotique industrielle

Avec la hausse constante du besoin pour plus de produits, avec des meilleurs prix et une meilleure qualité, l'utilisation de la robotique dans le domaine industriel, pour assurer l'autonomie des systèmes, augmente d'année en année.

La reconnaissance de la parole est implémentée avec succès et est largement utilisée et acceptée dans de nombreuses industries sur les robots industriels. Par exemple, l'ASR peut être utilisé pour entrer des commandes et naviguer dans les ordinateurs personnels.
D'autre part, des ordinateurs avec entrée vocale sont utilisés dans de nombreuses

applications d'inspection industrielle où les yeux et les mains sont occupée par d'autres tâches.

Automatisation

Les applications de l'ASR dans l'automatisation peuvent être constatées dans plusieurs domaines.

Des systèmes ASR sont installés dans diverses voitures et permettent une conduite plus sûre en offrant des commandes contrôlées par la voix, et donc, en minimisant le besoin de les déclencher manuellement, et donc, occuper les mains et les yeux du conducteur.

Ce type de système permet également aux personnes souffrant de divers handicaps de contrôler divers appareils électriques. Par exemple, il existe des fauteuils électriques pouvant être contrôlés par la voix.

Technologies de l'information et électronique grand public

Les progrès dans la puissance de calcul ont permis une large diffusion des systèmes ASR. En effet, ceux-ci sont aujourd'hui disponibles sur des laptops, des téléphones mobiles ou encore des systèmes de navigation.

Le support de la reconnaissance de la parole permet l'accès à internet aux vielles personnes ou aux personnes souffrant de handicaps, grâce aux assistants conversationnels. Ce type d'assistant permet de contrôler un ordinateur et donc d'accéder au web grâce à des commandes vocale et grâce à la transcription vocale.

Les moteurs de recherche permettent aussi d'interagir avec eux grâce à la voix. En effet, Google, Microsoft et Apple ont démontré l'utilisation de l'ASR dans leurs moteurs de recherche, en particulier sur les téléphones mobiles.

2.8 Systèmes ASR Commerciaux et APIs

Dans cette section, nous allons présenter quelques systèmes ASR commerciaux ainsi que des API (Application Programming Interface) pouvant être utilisées dans des applications tierces. Ces outils et services ont été développés par des entreprises spécialisées dans la reconnaissance automatique de la parole et offrent des fonctionnalités avancées pour la conversion précise de la parole en texte et sont largement utilisées, par les entreprises notamment. Parmi eux, nous pouvons citer les systèmes suivants :

Google Voice Search et Google Speech to Text

Google Voice Search est un service proposé par Google, et il est largement utilisé grâce à son intégration dans tous les téléphones Android. Il fait partie intégrante

de l'écosystème Google et est directement intégré à des applications telles que Google Traduction, ce qui facilite la traduction de texte sans avoir à le saisir manuellement. De plus, il permet de tenir des conversations en temps réel avec des personnes parlant une langue étrangère grâce à la traduction automatique [33] . Google propose également une API appelée Google Speech to Text qui permet aux développeurs d'inclure facilement une fonctionnalité d'ASR dans leurs propres applications. Cette API offre une flexibilité et une facilité d'utilisation pour intégrer les fonctionnalités de reconnaissance de la parole de Google dans diverses applications et services. L'API Google Speech to Text prend en charge une grande variété de langues. Cependant, cette API ne supporte pas la langue Tamazight [34].

Microsoft Azure Speech to Text

Microsoft propose une API de reconnaissance de la parole puissante et flexible qui permet aux développeurs d'intégrer des fonctionnalités de transcription vocale dans leurs propres applications. Cette API, disponible sur la plateforme Azure de Microsoft, offre des capacités de conversion précise et rapide de la parole en texte. Elle prend en charge différents formats audio, y compris les flux en direct, les fichiers audio et les enregistrements téléphoniques. Ce service propose également d'entrainer un modèle personnalisé à partir de son propre ensemble de données, ce qui pourrait servir pour créer un système ASR spécialisé pour un certain vocabulaire spécifique ou bien pour une langue pour laquelle un modèle pré-entrainé ne serait pas disponible. Cette API pourrait donc servir à la création de notre application puisqu'elle permet l'entrainement de modèles avec n'importe quel ensemble de données [35].

2.9 Systèmes ASR open-source

Dans cette section, nous allons présenter quelques systèmes ASR open-source. Du fait que le code source de ces applications peut être consulté librement, il est possible de déterminer avec précision l'approche utilisée pour la construction de ces systèmes, et donc, cela peut potentiellement nous guider dans la réalisation de notre projet. Parmi eux, nous pouvons citer les systèmes suivants :

DeepSpeech

DeepSpeech est un système ASR developpé par Mozilla, basé sur l'architecture de modèle Deep Speech. Le modèle DeepSpeech est un réseau de neurones récurrents (RNN) qui a été spécialement conçu pour la reconnaissance vocale. Il utilise les LSTM (Long Short-Term Memory), qui sont une variante des RNN permettant de traiter efficacement les séquences temporelles. DeepSpeech peut être utilisé pour transcrire

du texte avec un modèle pré-entraîné, mais il permet également d'entraîner un modèle sur son propre ensemble de données [27].

Whisper

Whisper, un système ASR developpé par la société OpenAI, est un système qui utilise l'approche End-to-End, un encodeur-décodeur Transformer plus précisément. Ce système a été entraîné sur une vaste quantité de données comprenant plus de 680 000 heures d'audio, ce qui lui permet d'atteindre une précision exceptionnelle, même pour des fichiers audio de qualité médiocre ou avec un fort bruit de fond, et ce, dans plusieurs langues. Malgré sa haute précision, le fonctionnement de ce système est relativement simple. Le flux audio d'entrée est divisé en blocs de 30 secondes, puis transformé en spectrogramme log-Mel avant d'être transmis à l'encodeur. Un décodeur est ensuite formé pour prédire la séquence de labels correspondante à l'entrée.
En plus de la transcription, ce modèle est aussi capable d'acomplir de tâches telles que l'identification de la langue, les horodatages au niveau des phrases, la transcription multilingue de la parole et la traduction de la parole vers l'anglais [36] [37].

2.10 Quelques toolkits pour l'ASR

Dans cette section nous allons présenter quelques toolkits destinés à l'ASR, ces derniers offrent habituellement un ensemble d'outils pouvant gérer tous les aspects pour la création de systèmes ASR.

Kaldi

Kaldi est un toolkit open-source écrit en C++ qui est largement utilisé pour la recherche et le développement de systèmes ASR avancés. Il offre une API de bas niveau, ce qui permet aux utilisateurs de créer des systèmes ASR très spécifiques en fonction de leurs besoins. Kaldi est reconnu pour sa puissance et sa flexibilité, mais il peut également être considéré comme complexe et difficile à utiliser pour les utilisateurs novices. Il prend en charge toutes les tâches nécessaires à la création d'un système ASR, depuis l'extraction des caractéristiques acoustiques jusqu'à la formation du modèle final. Cependant, pour faciliter son utilisation, une librairie Python appelée pykaldi a été développée, offrant une interface plus conviviale pour interagir avec Kaldi et simplifier son utilisation [38].

CMUSphinx

CMUSphinx est un autre toolkit d'ASR open-source. Il se distingue par sa convivialité, ce qui le rend particulièrement adapté aux débutants en ASR. Il offre des APIs dans

plusieurs langages de programmation tels que C, C++, C#, Python, Ruby, Java et JavaScript, permettant ainsi aux utilisateurs de choisir le langage qui convient le mieux à leurs compétences et à leur environnement de développement. CMUSphinx est activement maintenu et prend en charge tous les aspects de la création de systèmes ASR, allant de la préparation des données à l'entraînement des modèles en passant par l'évaluation et la transcription. Sa simplicité d'utilisation et sa polyvalence en font une option attrayante pour les développeurs qui souhaitent se lancer dans la reconnaissance automatique de la parole sans avoir à se plonger dans des détails techniques complexes [39].

2.11 Ressources en rapport avec notre projet

Nous présenterons dans cette section diverses ressources très utiles pour la réalisation de notre projet, ainsi, nous présenterons l'ensemble de données Common Voice qui est très utile pour la création d'un système ASR, ainsi que divers recueils de textes utiles pour l'entraînement d'un modèle de langue pour la langue Tamazight.

Common Voice

Common Voice est une initiative de Mozilla qui s'engage à rendre les ensembles de données vocales pour les systèmes de reconnaissance automatique de la parole (ASR) gratuits et accessibles à tous. Cette plateforme collaborative permet aux individus de contribuer en enregistrant leur voix ou en validant les enregistrements soumis par d'autres participants. Elle offre une diversité de Data sets dans pas moins de 104 langues, comprenant de courts enregistrements vocaux et leurs transcriptions correspondantes. Pour la langue Tamazight, un ensemble de données de plus de 550 heures est disponible. Ces Data sets sont publiés sous la licence Creative Commons CC0, garantissant leur libre utilisation et partage. Common Voice permet ainsi de développer des modèles de reconnaissance vocale plus précis et adaptés à différentes langues, y compris Tamazight, tout en encourageant la participation de la communauté pour enrichir ces ensembles de données et améliorer la performance des systèmes ASR [40].

Tatoeba

Tatoeba est une collection de phrases dans 420 langues différentes, incluant Tamazight. Cette base de données linguistique contient un total de 1,2 million de phrases écrites en Tamazight, qui s'avèrent extrêmement utiles pour l'entraînement de modèles de langue. De plus, elle comprend plus de 47.000 phrases avec des enregistrements audio correspondants, offrant ainsi des ressources à la fois écrites et orales pour améliorer la

reconnaissance vocale. Ces ressources sont également disponibles sous diverses licences Creative Commons, garantissant leur accessibilité et leur partage libre [41].

Wikipédia

Le projet sur la reconnaissance automatique de la parole (ASR) peut grandement bénéficier de l'utilisation de Wikipédia comme ressource linguistique pour la langue Tamazight. En tant qu'encyclopédie libre en ligne, Wikipédia propose une collection immense de plus de 6500 articles rédigés en Tamazight. Ces articles sur des sujets divers peuvent servir à alimenter un modèle de langue [42].

Kabyletexts

Le référentiel GitHub "Kabyletexts" est un grand recueil de textes utiles pour les créations de modèles de langue. Ce référentiel contient une vaste collection de textes en Tamazight, couvrant divers sujets et domaines, tels que la littérature, la poésie, l'histoire, et bien plus encore. Ce référentiel est donc utile pour fournir des ressources supplémentaires pour entrainer un modèle de langue [43].

2.12 Projets en rapport avec l'ASR pour la langue Tamazight

Dans cette section, nous présenterons les projets en rapport avec l'ASR pour la langue Tamazight que nous avons pu trouver. Cela pourrait s'avérer utile pour nous orienter vers la démarche à suivre pour la réalisation de notre projet.

Okwugbé

Une librairie python open-source spécialement conçue pour les langues africaines, dont Tamazight. Elle vise à faciliter le développement de systèmes ASR pour ces langues le maximum possible. Elle peut cependant être utilisée avec n'importe quelle langue. Mais, une telle librairie n'offre pas le même niveau de contrôle et de personnalisation qu'un framework dédié à l'ASR [44].

ASR Deep Speech kabyle

Ce projet open-source met en œuvre un système ASR fonctionnel pour la langue kabyle en utilisant le toolkit Mozzila DeepSpeech. Basé sur le modèle Deep Speech, ce toolkit offre une base solide pour la reconnaissance vocale. En exploitant le modèle DeepSpeech combiné avec l'ensemble de données Common Voice de Mozilla, le projet parvient à entraîner un système ASR spécifique à la langue Tamazight [45].

Kabyle_xlsr et Stt_kab_conformer_transducer_large

Kabyle_xlsr est un projet open-source qui utilise un modèle appelé "Wav2Vec" pré-entrainé sur plus de 436 mille heures de données, ce modèle, après avoir subis un fine-tuning sur l'ensemble de données Common Voice pourrait permetre d'obtenir un système ASR avec un taux d'erreur relativement bas [46].
Pour ce qui est de Stt_kab_conformer_transducer_large, c'est aussi un projet open-source, mais qui est réalisé en faisant le fine-tuning d'un modèle différent, en l'occurrence Conformer-Transducer [47].

2.13 Conclusion

La technologie de Reconnaissance Automatique de la Parole (ASR) a fait de grands progrès ces dernières années et est utilisée dans une variété d'applications, allant de l'assistance médicale aux assistants virtuels. Ce chapitre a fourni un aperçu des différents axes de classification, des méthodes d'évaluation des performances, des types d'encodage de texte, des phases et composants d'un système ASR, des approches possibles pour sa conceptionn, des domaines d'applications, des applications et toolkits, des ressouces ainsi que projets en rapport avec l'ASR. Il est clair que l'ASR continuera à jouer un rôle important dans de nombreux domaines et qu'il reste encore beaucoup à découvrir et à améliorer dans cette technologie en constante évolution. Dans le chapitre suivant, nous allons réaliser notre application de ASR en langue Tamazight.

Chapitre 3 : Réalisation de notre application "Mmeslay"

Introduction

Dans ce chapitre, nous allons détailler les étapes qui nous ont permis de réaliser notre projet. Nous allons tout d'abord présenter la problématique, pour détailler les raisons qui nous ont poussé à faire ce travail.
Nous allons ensuite expliquer la solution que nous avions envisagé pour résoudre cette problématique, en détaillant l'architecture de notre application et du modèle que nous avons utilisé.
Puis, nous allons présenter les outils qui nous ont permis de construire notre application.
Enfin nous allons présenter notre solution à la problématique et nous allons expliquer les étapes qui nous ont permis de réaliser notre application.

3.1 Problématique

Du fait des nombreux domaines d'application des systèmes ASR, l'utilité de ce genre de système n'est plus à prouver. Pour notre langue, Tamazight, Nous n'avons trouvé aucune application d'ASR, seulement quelques modèles entrainés sur Common Voice.

Ce type de systèmes pourrait s'avérer très utile, pour par exemple aider les personnes agées. En effet, alors que les personnes des autres tranches d'ages maîtrisent typiquement l'anglais ou le français, les vielles personnes en Kabylie ne parlent en général que la langue Tamazight, or, ces personnes sont parmi celles ayant le plus besoin de ce genre de système, du fait de leur analphabétisme ou tout simplement de leurs difficultés à utiliser de simples interfaces graphiques.

Un système ASR en langue Tamazight pourrait donc grandement bénéficier à ce genre de personnes, pour par exemple construire des applications pour déclencher diverses commandes à l'aide d'un smartphone.

Ce travail a donc pour but de réaliser une application Android appelée "Mmeslay", qui permet de transcrire la parole en langue Tamazight en texte.

3.2 Solution envisagée

Afin de réaliser notre système ASR pour la langue Tamazight, nous avons envisagé d'entrainer le modèle Squeezeformer [2] avec l'ensemble de données Common Voice, plus particulièrement le sous-ensemble en langue Tamazight (Kabyle).

Cet ensemble de données contient des pairs de phrases avec leurs transcriptions correspondantes, ce qui est parfait pour entrainer un modèle ASR End-To-End, tel que Squeezeformer (détaillé dans la section suivante) par exemple.

Common Voice est costruit en demandant à des volontaires de lire des phraseset le sous-ensemble pour la langue Tamazight est composé de fichiers audio enregsistrés par plus de 1400 personnes différents. De ce fait, un modèle entrainé sur cet ensemble de données sera un modèle dictationnel, et la variété des locuteurs devrait permettre au modèle d'être indépendant de l'orateur.

En plus de ça, nous envisageons d'entrainer un modèle de langue sur les recueils de phrases et textes présentés dans le premier chapitre, et pour ce, nous envisageons d'utiliser le programme KenLM [48] qui permet d'entrainer un modèle de langue n-gram. Cette librairie est très adaptée pour ce genre de tâches du fait de sa grande efficacité en termes de temps de calculs, elle est d'ailleurs largement utilisée pour ce genre de tâches.

3.3 Architecture de notre application

Notre système ASR se compose d'un modèle End-To-End et d'un modèle de langue n-gram, Après avoir été entrainés, ces deux modèles peuvent être expolités pour transcrire la parole en Tamazight récupérée à partir d'un enregistrement audio fait à l'aide du front-end qui s'executera sur un téléphone Android, voir figure 3.1.

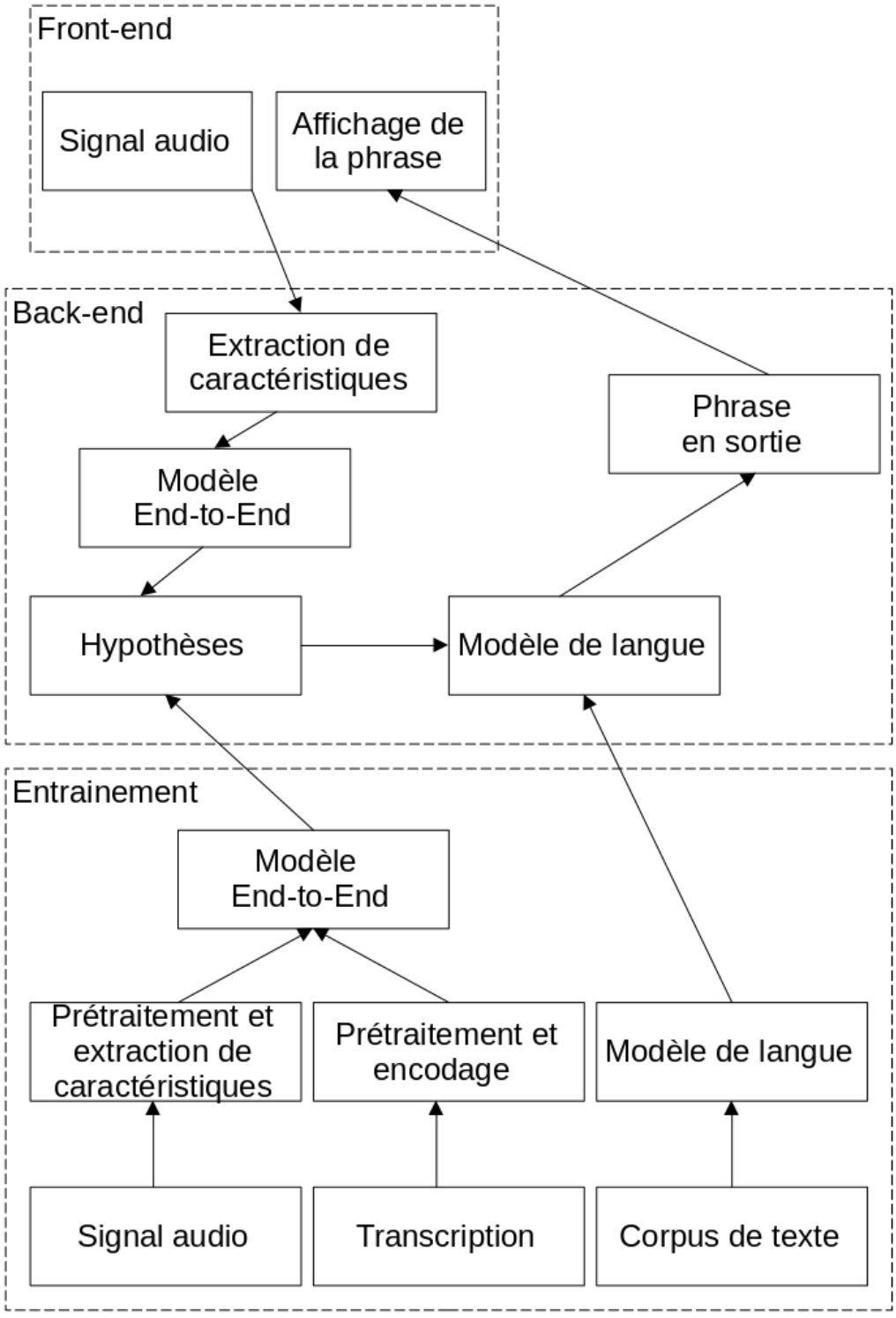

Figure 3.1: Block diagram de notre système ASR.

3.3.1 Le modèle Squeezeformer

Le modèle Squeezeformer est un modèle d'ASR End-to-End composé d'un encodeur et d'un décodeur, voir partie gauche de la figure 3.2.

Avant d'arriver à l'encodeur, les entrées passent d'abord par un module de sous-échantillonage, qui sert à rendre la reconnaissance plus robuste aux variations dans leurs positions [49],

Ensuite, viens la partie encodeur, composée de N-1 blocs "Squeezeformer", suivis de N blocs "Squeezeformer" avec connection résiduelle et enfin un seul bloc "Squeezeformer" (sans connection résidulle), N étant déterminé par le nombre total d'étages ; les blocs avec connection résiduelle prennent en entrée la sortie de l'étage précédent aditionnée à l'entrée de l'étage précédant, ce qui facilite l'entrainement des modèles plus profonds [50].

Le bloc "Squeezeformer" de ce modèle est composé de deux sous-structures : la première est basée sur l'attention, la seconde est convolutionelle. La première sert à extraire les caractéristiques plus globales ; les relations entre les élements plus éloignées de la séquence, tandis que la seconde sert à extraire les caractéristiques locales ; entre les éléments plus rapprochés de la séquence. Dans chacune des sous-structures, un étage de réseau de neurones à propagation avant est ajouté après le module principal. Une connection résiduelle est ajoutée à chaque étage. Un module de normalisation (Post-Layer Normalization) est également ajouté après chaque étage. Voir partie droite de la figure 3.2.

Le decodeur qu'utilise Squeezeformer est un décodeur CTC, son fonctionnement a été decrit en détail dans le chapitre précédant.

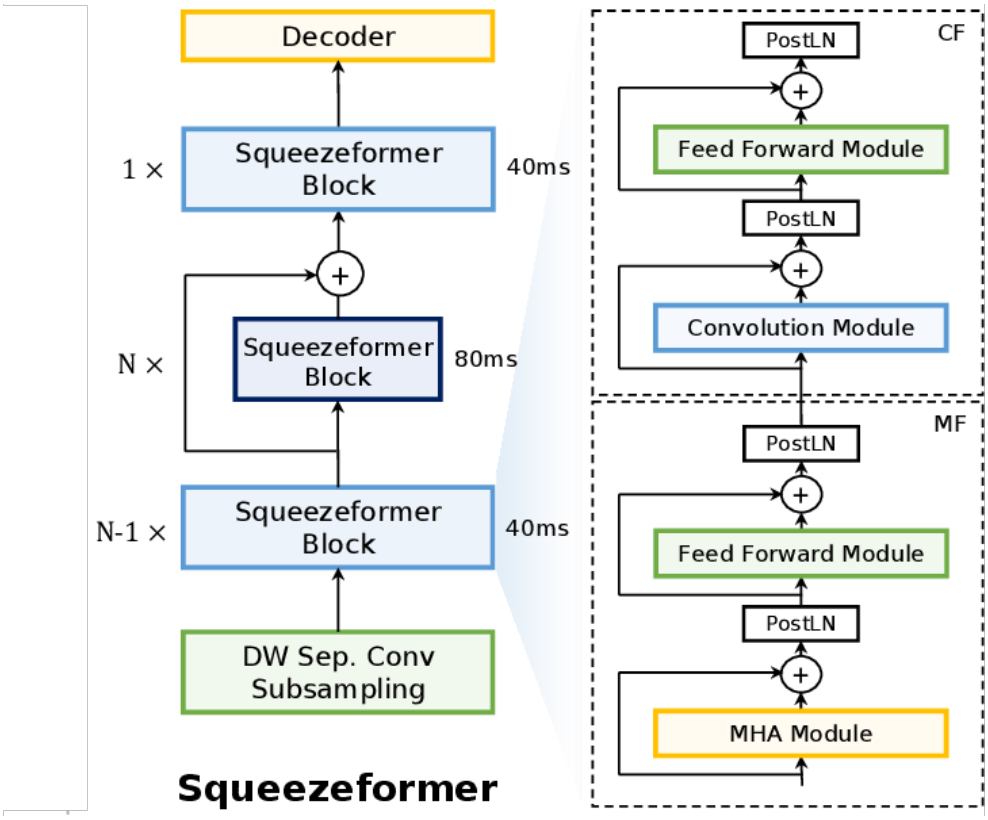

Figure 3.2: Architecture du modèle Squeezeformer [2].

Ce modèle a l'avantage d'offrir des performances excellentes en plus d'être très efficace en termes de coûts de calculs. Squeezeformer se décline en 6 variantes selons sa taille, les variantes les plus petites en termes de nombre total de paramètres sont plus efficace en termes de coûts de calculs, mais sont légèrement moins perfomantes.

Pour notre projet, nous avons choisi Squeezeformer-XS (la variante la plus petite), étant donné nos ressources de calculs limitées. Cette variante compte 9 Millions de paramètres et est composée de 16 étages [2].

3.4 Outils utilisés

3.4.1 Pandas

Pandas est une librairie python open-source qui permet la manipulation des données sous divers formats, comme les fichiers CSV (Comma-separated values, valeurs séparées par virgule en français) ou TSV (Tab-separated values, valeurs séparées par tabulation). Elle permet de charger un fichier parmi de nombreux formats supportés et les convertir en un format appelé DataFrame. Ce format permet d'acceder aux données désirées à l'aide de diverses requêtes, de réaliser diverses opérations sur les données pour obtenir des statistiques par exemple ou encore d'appliquer une fonction quelconque pour traiter les données. [51]

3.4.2 Pytorch

Pytorch est un framework de Deep Learning open-source. Il offre de nombreux outils pour entrainer des modèles de Deep Learning en plus de supporter l'accéleration par GPU(Graphics processing unit). Il offre des outils pour divers domaines d'applications du Deep Learning, tel que la vision artificielle, le traitement de langage naturel, ou encore l'audio et le traitment de signal. Il offre entre autres des moyens pour implementer des modèles facilement, des modèles prêts à l'emploi, des optimiseurs, des planificateurs ...
Pytorch dispose également d'un "Data Loader", qui permet de charger les données au fur et à mesure de l'entrainement pour éviter la saturation de la mémoire en cas d'ensemble de données de taille supérieure à la mémoire disponible. Ça permet aussi d'appliquer divers prétraitements aux données pendant l'entrainement [52].

3.4.3 Lightning

"Pytorch Lightning" est une librairie open-source qui permet de faciliter l'enterainement de modèles implémentés avec pytorch. Cette librairie permet notamment de paralléliser l'entraînement pour utiliser plusieurs machines facilement et donc accélerer le processus.
De plus, cette librairie offres diverses "callbacks", qui sont des fonctionnalités supplementaires pour l'entrainement. Par exemple "ModelCheckpoint" permet de sauvegarder l'avancée de l'entraînement sous forme d'un point de controle à chaque fin d'époque pour continuer l'entraînement en cas d'interruption accidentelle [53].

3.4.4 Tensorboard

Tensorboard est un outil permettant de suivre l'avancée de l'entrainement en traçant automatiquement des graphes en fonction des valeurs suivies au cours de l'entraînement. Il permet de savoir si l'entraînement se passe comme prévu et de détecter un potentiel overfitting [54].

3.4.5 torchmetrics

Torchmetrics est une librairie open-source qui implemente diverses unités d'Evaluation des performances pour les modèles d'apprentissage automatique. Elle permet d'avoir une API standardisée, ce qui permet d'éviter d'utiliser différentes implementation de la même unité et donc d'accroitre la reproductibilité.
Elle implémente notamment le WER et le CER, unités utilisées pour évaluer les performances de notre modèle [55].

3.4.6 KenLM

KenLM est un programme qui permet d'entrainer facilement un modèle de langue n-gram à partir d'un corpus de texte. Il consomme très peu de ressources matérielles et est très rapide, il est donc très adapté pour être utilisé pour améliorer les résultats d'un système ASR. Il offre aussi une API python, ce qui permet de l'utiliser facilement dans un projet utilisant ce langage de programmation [**?**].

3.4.7 Nemo

Nemo est un framework qui permet d'entraîner divers modèles de Deep Learning End-To-End. Il offre divers outils, notamment pour l'ASR, comme l'implementation de beaucoup de modèles populaires, tel que Squeezeformer par exemple, ou encore des outils pour le prétraitement ou l'augmentation de données.
Il propose également des modèles pré-entrainés prêts à l'emploi [56].

3.4.8 SentencePiece

SentencePiece est une librairie qui permet d'entrainer un modèle supérvisé pour encoder le texte en labels. Elle est largement utilisée en traitement de langage naturel ou encore en ASR. En plus de permettre un encodage en graphèmes basique, elle implémente les modèles "BPE" et "Unigram" pour ségmenter le texte en sous-mots. L'un de ces deux modèles peux être facilement entrainé avec un fichier contenant du texte brut. [57]

3.4.9 Flask

Flask est un framework qui permet de réaliser des applications web en python. C'est un framework minimaliste, il inclut les fonctionnalités de base pour ce genre d'application, mais diverses extensions existent pour permettre si besoin d'avoir accès à d'autres fonctionnalités [58].

3.4.10 Conda

Conda est un géstionnaire de paquets open-source pour python. Il permet de créer des environnements virtuel et y installer les dépendances de son projet sans avoir à les installer dans son système au risque d'avoir des conflits entre les différents paquets.
De plus, Conda permet de créer divers environnements avoir des versions de python différentes si besoin, ce qui pourrait s'avérer difficle voir impossible à faire sans ce type d'outil sous Linux [59].

3.4.11 Flutter

Flutter est un framework qui permet de développer des applications en langage Dart pour diverses plateformes avec le même code. En effet, la même application peut être compliée pour focntionner sur Android, IOS, Linux, Windows, MacOS et même dans un navigateur web [60].

3.4.12 Record

Record est une librairie Dart qui permet d'enregistrer de l'audio à partir d'un microphone dans divers taux d'échantillonage selon le besoin et de le sauvegarder dans divers formats [61].

3.4.13 Visual Studio Code - Open Source

Code - OSS est un éditeur de texte open-source qui permet de développer et de déboguer des applications facilement. Il offre diverses fonctionnalités pour faciliter le développement telles que la completion de code, la coloration syntaxique et des débogueurs pour divers languages de programmations. Cet éditeur est très extensible et permet le support de Python et Dart/Flutter simplement en installent leurs extensions officielles [62].

3.5 Réalisation de l'application "Mmeslay"

Dans cette section, nous allons détailler les étapes pour la réalisation de notre application. Nous allons d'abord expliquer les étapes de prétraitement de notre ensemble de données, puis les détails de l'entraînement du modèle, et enfin, L'exploitation de notre système pour transformer la parole en transcription.

3.5.1 Pré-traitement

L'ensembe de données Common Voice est fournis avec un fichier TSV, qui contient diverses informations concernant l'ensemble de données. Ce fichier TSV contient les noms des fichiers audio avec leurs transcriptions correspondantes, mais aussi certaines métadonnées qui donnent des informations sur la personne ayant fait l'enregistrement. Ces dernières ne sont pas utiles pour l'entraînement de notre modèle et sont donc retirées du fichier TSV. Les transcriptions des enregistrements audio de l'ensemble de données Common Voice doivent être pré-traitées avant de servir à l'entraînement de notre modèle. Le fichier TSV est lu et est manipulé grâce à la librairie Pandas.

a - Texte

Dans les transcriptions d'origines, certaines phrases contiennent des majuscules, et de la ponctutation. Or, notre but est d'entrainer un modèle pour produire du texte en minuscule et sans ponctuation.
Par exemple, la phrase "Jeddi-m mazal-it yehlek?" est tranformée en "jeddi-m mazal-it yehlek".

En plus d'être tranformées pour être intégralement en minuscule, tous les caractères spéciaux, à l'exception du '-' qui fait partie de la grammaire de la langue Tamazight, sont retirés. De plus, l'ensemble de données contient certains échantillons avec des transcriptions invalides, soit parcequ'elles contiennent des lettres qui ne font pas partie de la langue Tamazight, soit à cause d'une erreur syntaxique. C'est donc pour ça que nous avons utilisé une expression regulière pour filtrer les échantillons avec des transcriptions erronées.
Par exemple, cette phrase "yefka-as -ent-id akken yella" sera retirée à cause de l'éspace avant un '-'.
Les transcriptions contenant les lettres 'p' et 'o' sont gardées malgré le fait qu'elles n'existent pas dans l'alphabet de la langue Tamazight, car il y a de nombreux noms propres qui contiennent cette lettre.

Pour alimenter le modèle, la transcription de chaque fichier audio doit être encodée en un vecteur de nombres entiers. Pour ce faire, la librairie SentencePiece est utilisée. Pour entrainer un tokenizer, nous nous sommes servi de l'ensemble des phrases uniques composant les sous ensemble Common Voice utilisé pour l'entrainement (en excluant celles apparessant dans les sous-ensembles de validation et de test). Nous avons choisi une taille de vocabulaire de 128, une valeur souvent utilisée pour les modèles basés sur CTC.
Par exemple, la phrase "qrib d lawan ad nebdu" sera encodée en [3, 41, 80, 26, 16, 43, 2, 35, 29, 23, 28, 69, 12, 7], voir tableau 3.1.
Alors que toutes les étapes de prétraitement du texte ont été fait à l'avance, l'encodage des phrases en jetons se fait à la volée.

Token	Index
<unk>	0
_	1
a	2
▁	3
t	4
▁t	5
s	6
u	7
m	8
i	9
k	10
en	11
d	12
▁i	13
n	14
-	15
▁d	16
ɣ	17
l	18
r	19
g	20
eɣ	21
em	22
▁ad	23
f	24
e	25
b	26
▁a	27
▁n	28
an	29
c	30
z	31
-t	32
er	33
el	34
w	35
y	36
-d	37
▁u	38
▁ur	39
▁s	40
q	41

Token	Index
ḥ	42
▁l	43
-i	44
ɛ	45
la	46
in	47
▁ay	48
-a	49
ed	50
▁ara	51
▁li	52
▁y	53
x	54
ḍ	55
▁ye	56
ra	57
▁yi	58
ed	59
al	60
▁ta	61
ḥ	62
ar	63
-nni	64
ent	65
▁deg	66
et	67
ma	68
eb	69
ṛ	70
ek	71
es	72
▁w	73
ẓ	74
▁as	75
j	76
ef	77
mi	78
-ine	79
ṅ	80
▁ti	81
wen	82
▁am	83

Token	Index
wi	84
wa	85
ay	86
na	87
iɣ	88
-s	89
▁akk	90
ur	91
▁wa	92
-is	93
▁m	94
-as	95
ṭ	96
-iyi	97
ess	98
id	99
▁ɣer	100
▁ma	101
ṭ-id	102
ez	103
▁ɣef	104
eɛ	105
ṣ	106
ir	107
ǧ	108
▁d-t	109
-iw	110
ew	111
▁fell-	112
eḥ	113
▁acu	114
▁seg	115
eq	116
wal	117
-nwen	118
▁d-y	119
▁yid-	120
eṛ	121
▁tett	122
-nsen	123
eqq	124
o	125
p	126
č	127

Tableau 3.1: Tableau représentant les tokens utilisés pour encoder le texte.

b - Audio

Les fichiers audio utilisés pour l'entraînement ont été ré-échantillonnés en 16Khz, et ce, avant l'entraînement, dans le but d'accélerer ce dernier. En effet, ajouter cette opération à la boucle d'entraînement ajouterais des calculs supplementaires et en conséquence ralentirais considérablement ce processus. Le reste des opérations de prétraitement se font à la volée.

Les échantillons audio sont ensuite transformés en mel-spectrogrammes avec une taille de fenêtre de 25 ms et un chevauchement de 40%.
SpecAugment (avec 2 masques de fréquence et 5 masques temporels) est appliqué aux spectrogrammes qui servent pour l'entraînement uniquement, les chiffres précédants sont recommendés par [63].

3.5.2 Entrainement

a - Partitionnement des données

Avant de servir pour l'entraînement, l'ensemble de données est divisé en trois sous-ensembles : d'entraînement, de validation et de test. L'ensemble de données est composé de 600000 échantillons environ, dont 180000 pharases uniques. Avant d'être partitionné, nous avons regroupé chaque phrase avec tous les audios qui la représentent. Nous avons ensuite divisé l'ensemble de données en 80% pour l'entrainement, 10% pour la validation et 10% pour le test.

La plupart des phrases sont représentées par plusieurs fichiers audios, enregistrés par des personnes différentes. Pour utiliser toutes les données sans risquer que le système ne se sur-ajuste aux phrases représentées par le plus de fichiers audio, un des fichiers représentant chaque phrase est sélectionné aléatoirement pour chaque époque de l'entrainement. Par contre, une seule et même version par phrase est utilisée pour les ensembles de validation et de test.

b - Execution de l'entraînement

Nous avons entrainé le modèle Squeezeformer implémenté dans le framework Nemo. La partie encodeur a été initialisée avec un point de controle du modèle entrainé pour l'anglais sur l'ensemble LibriSpeech [21], car selon [40] cela permet d'attendre un WER plus petit. Nous avons utilisé l'optimiseur RAdam [64], qui a pour avantage de ne pas nécessiter de warmup (échauffement) ; qui consiste en l'augmentation du taux d'aprantissage progressivement au début de l'entrainement, RAdam permet donc de tester les différents hyperparamétres (taille de lot, taux d'apprentissage et intensité de SpecAugment entre autres) plus rapidement. De plus RAdam est moins sensible au taux d'apprentissage, ce qui réduit le temps nécessaire pour régler cet hyperparamétre [65].
L'entrainement s'est fait avec une taille de lot de 64 pour 121 époques, avant d'être changée pour 256 pour le reste de l'entrainement, car selon [66], augmenter la taille de lot a des effets similaires que la réduction du taux d'apprentissage et permet d'atteindre une perte de validation plus basse. Après avoir testé plusieurs valeurs, nous avons trouvé que le meilleur taux d'apprentissage était de 2×10^{-4}.

Après chaque époque, l'ensemble de validation est utilisé pour vérifier l'avancée de l'entrainement. La perte de validation, le WER et le CER de validation sont calculés à chaque fin d'époque, voir figures 3.3 et 3.4

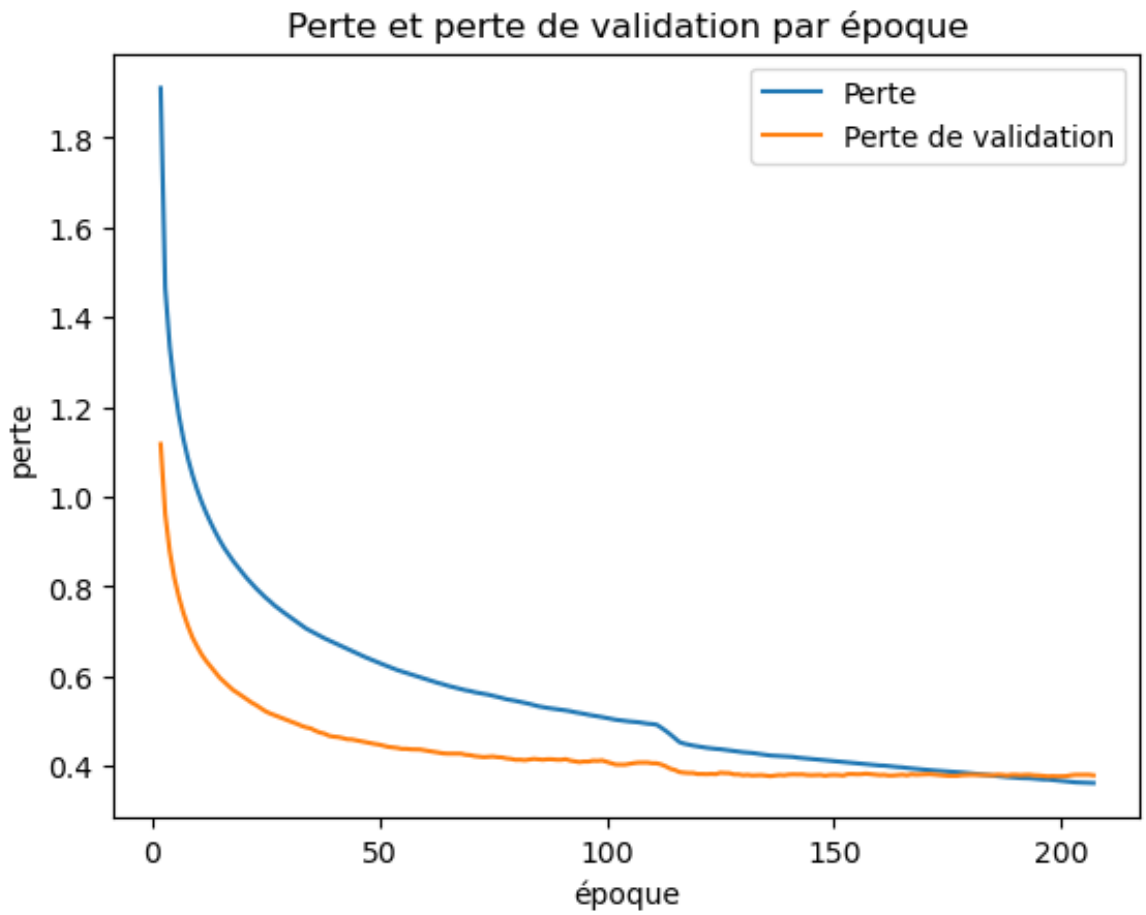

Figure 3.3: Perte d'entraînement et de validation en fonction de l'époque.

L'entrainement a été arrété après 211 époques, car la perte sur l'ensemble de validation a commencé à stagner tandis que la perte sur l'ensemble d'entraînement a continué à diminuer, et ce, pour éviter le sur-ajustement. Le point de contrôle du modèle avec la meilleure perte sur l'ensemble de validation est gardé. Réduire le taux d'apprentissage ou augmenter la taille de lot n'améliore pas davantage les résultats même après de nombreuses époques supplementaires.

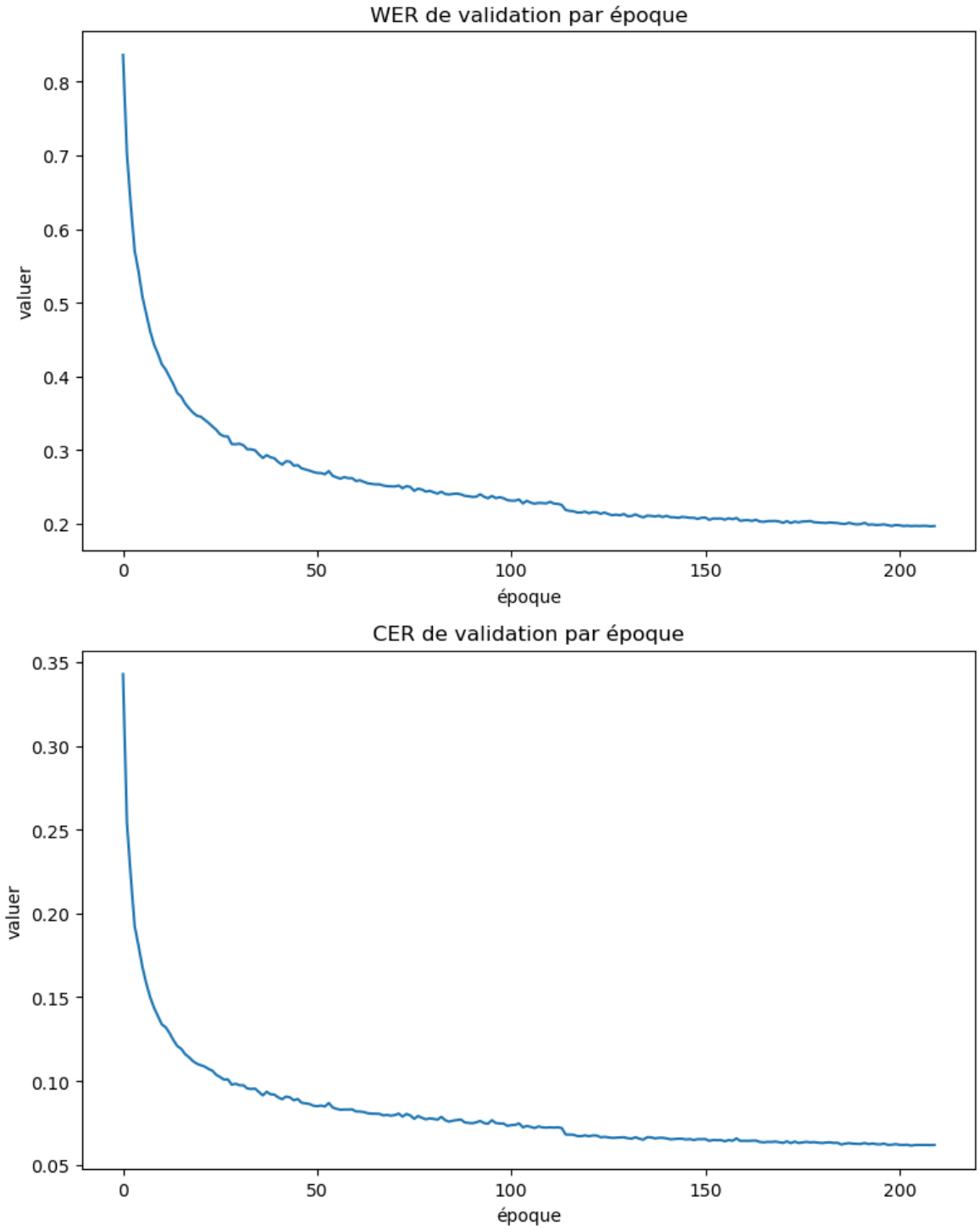

Figure 3.4: WER et CER sur l'ensemble de validation en fonction de l'époque.

c - Evaluation des performances

Modèle de langue

Pour améliorer les performances de notre système nous avons entrainé un modèle de langue grâce au programme KenLM [48]. Pour ce faire nous avons utilisé les recueils de

texte que nous avons trouvés (voir le chapitre précédant), nous les avons prétraités de la même manière que pour les transcriptions de l'ensemble Common Voice, mais nous n'avons gardé que les phrases contenant des mots se répétant au moins une fois de plus quelque part dans le corpus de texte, et ce, pour éviter que des phrases contenant des mots inexistants ou des fautes d'ortographe ne figurent dans notre corpus.
Pour entrainer note modèle de langue, ségmenter les phrases en labels est nécessaire. Pour ce faire nous avons entrainé un tokenizer sur le corpus de texte en utilisant la librairie SentencePiece, nous avons utilisé une taille de vocabulaire de 5220 (le maximum possible avec l'algorithme "unigram" pour ce corpus). Utiliser cette méthode pour ségmenter les phrases permet de réduire les risques que le modèle de langue rencontre des OOVs. La ségmentation en mots poserait un problème pour la langue Tamazight, notamment à cause de l'utilisation des tirets ("-"). Par exemple, si le mot "idrimen-iw" apparait dans le corpus, mais pas le mot "idrimen-nwen", ce dernier serait un OOV.
Nous avons entrainé deux variantes de modèles n-gram : un modèle trigram et un modèle 6-gram.

Nous avons évalué les performances de notre système sur les deux ensembles de validation et de test, et ce, avec plusieurs configurations pour le décodage. Nous avons testé le décodeur CTC par défaut, mais aussi le décodeur Beam Search (avec 128 faisceaux), d'abord sans, puis avec un modèle de langue (trigram et 6-gram), voir le tableau 3.2.

	Sans BS	BS sans LM	BS et LM trigram	BS et LM 6-gram
Validation WER (%)	19.69	19.47	18.81	18.71
Validation CER (%)	6.18	6.11	6.01	5.99
Test WER (%)	19.54	19.31	17.99	17.94
Test CER (%)	6.00	6.23	6.01	6.00

BS : Beam Search , LM : Modèle de langue (Language Model)

Tableau 3.2: Evaluation des performances de notre système ASR.

Les performances du modèle sur l'ensemble de test sont meilleures que celles sur l'ensemble de validation, ce qui indique une bonne généralisation du système. Améliorer davantage les performances du système requiert une quantité supérieure de données ou de plus grandes avancées technologiques de systèmes ASR ou des techniques d'augmentation

de données pour mieux tirer parti des ensembles de données relativement petits.

3.5.3 Frontend

Le front end permet d'exploiter le modèle, en enregistrant sa parole en langue Tamazight, l'application Android envoi un fichier audio vers un point de terminaison HTTP (Hypertext Transfer Protocol) d'une API écrite en python grâce au framework Flask. Ce fichier audio, enregistré dans un fichier temporaire sur le serveur est chargé et le modèle est utilisé pour prédire la transcription correspondante. Ensuite, la transcription est envoyée sous forme de réponse HTTP et est affichée sur le téléphone Android.

3.6 Conclusion

Ce chapitre a détaillé les étapes qui ont permis la réalisation de notre projet. Nous avons présenté la problématique et les motivations qui nous ont poussés à entreprendre ce travail. Ensuite, nous avons expliqué la solution envisagée, en détaillant l'architecture de notre application et le modèle utilisé. Nous avons également présenté les outils utilisés pour construire notre application. Enfin, nous avons exposé les étapes de la réalisation de notre application.
Nous avons réussi à réaliser une application qui permet de transcrire la parole récupérée par un smartphone Android en texte en langue Tamazight.
Malgré le WER relativement élevé de notre application, l'augmentation constante du nombre d'heures de parole en Tamazight enregistrées dans Common Voice, et la rapidité des avancées technologiques en matière de systèmes ASR, nous ésperons que notre langue puisse bénéficier de systèmes ASR aussi précis que les autres langues dans un futur assez proche.

Conclusion Générale et perspectives

Notre projet consiste en une application Android fonctionnelle dotée d'une fonctionnalité d'enregistrement vocal et de transcription en Tamazight en utilisant un modèle End-to-End. Cette application permet aux utilisateurs d'enregister des données audio via leur appareil Android et de les convertir en texte en utilisant un modèle End-to-End combiné à un modèle de langue n-gram.

Notre application présente plusieurs avantages. L'utilsation du modèle Squeezeformer-XS permet d'avoir une application à la complexité en termes de temps et ressources de calculs relativement réduits, ce qui réduirait les frais d'hébergement, en plus de garantir des latences très réduites. De plus, elle est assez précise pour servir de base pour une application de déclenchement de commandes, surtout si ces commandes sont prédéfinies et que le décodeur CTC est restreint à un nombre fini de transcriptions possibles.

Cependant, notre modèle étant entrainé sur un ensemble de données publique, de taille relativement petite qui plus est, il n'est pas étonnant qu'il peine à transcrire des mots de certains domaines spécialisés ou encore des noms propres qui ne figurent pas dans les phrases de l'ensemble de données. Il pourrait également échouer à transcrire correctement les accents peu ou pas représentés.

Le taux d'erreur de notre application est significativement plus élevé que celui que l'on peut trouver pour des langues plus courantes telles que l'anglais. Cet écart peut s'expliquer par la disponibilité limitée de données pour notre langue, ce qui entraîne des difficultés pour l'entraînement des modèles de ASR. Cela empecherais d'incorporer notre application dans un traduteur par exemple.

Il est encourageant de noter que l'ensemble de données Common Voice continue de croître et de se développer. Nous ésperons qu'à l'avenir, grâce à une plus grande quantité de données disponibles, les systèmes ASR pour notre langue deviendront aussi précis que ceux des langues plus largement répandues.
De plus, les avancées technologies en matière de modèles d'ASR et des techniques d'augmentation de données pourraient très certainement permettre à l'avenir pour des langues dotées de peu de ressources de mieux exploiter les données disponibles et donc de réduire l'écart dans les performances des systèmes ASR.

IL est également possible d'ajouter à notre application une fonctionnalité de collecte de données dans le but d'améliorer les performances de notre système à mesure que notre application est utilisée.

Nous ésperons que notre application puisse représenter une première étape importante vers le développement de solutions de reconnaissance automatique de la parole en Tamazight. Nous espérons que nos efforts, combinés à l'expansion des ressources, contribueront à améliorer les systèmes ASR pour notre langue dans un avenir proche.

Bibliographie

[1] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. Tan, ``Deep spiking neural networks for large vocabulary automatic speech recognition,'' *Frontiers in Neuroscience*, vol. 14, 03 2020.

[2] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer, ``Squeezeformer: An efficient transformer for automatic speech recognition,'' *arXiv preprint arXiv:2206.00888*, 2022.

[3] J. Vajpai and A. Bora, ``Industrial applications of automatic speech recognition systems,'' *International Journal of Engineering Research and Applications*, vol. 6, pp. 88--95, Mar. 2016.

[4] N. D. Mirzahmedova, ``Working with digital information on a computer,'' *World Bulletin of Social Sciences*, vol. 6, pp. 88--89, Jan. 2022. "https://scholarexpress.net/index.php/wbss/article/view/496".

[5] A. Ambardar, *Analog and digital signal processing*. Cengage Learning, 2nd ed., 2000.

[6] B. Teston, ``A la poursuite du signal de parole: suite et fin,'' in *Journées d'Étude sur la Parole (JEP)*, (Avignon, France), June 2008. "https://hal.archives-ouvertes.fr/hal-00308393".

[7] P. D. F. S. Mamedov, ``Pulse-code modulation technique for digital telephone,'' *Proc. Int. Switching Symp.*, 1985.

[8] J. O. Smith III, *Introduction to Digital Filters with Audio Applications*. W3K Publishing, September 2007.

[9] U. Zölzer, *Digital Audio Signal Processing*. Hamburg, Germany: John Wiley & Sons, Ltd, 2 ed., 2008.

[10] M. Jacobson, ``Numériser l'oral. l'ifan face à la virtualisation de son patrimoine - enjeux et perspectives,'' in *L'IFAN face à la virtualisation de son patrimoine - enjeux et perspectives*, 2007. "https://halshs.archives-ouvertes.fr/halshs-00353968".

[11] M. Hira, ``Audio feature extraction tutorial,'' 2021. "https://pytorch.org/audio/main/tutorials/audio_feature_extractions_-tutorial.html".

[12] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology, Springer London, 2014.

[13] K. Chaudhary, ``Understanding audio data, fourier transform, fft and spectrogram features for a speech recognition system,'' *Towards Data Science-Medium*, 2020.

[14] P. Vieting, C. L"uscher, W. Michel, R. Schl"uter, and H. Ney, ``On architectures and training for raw waveform feature extraction in asr,'' *arXiv preprint arXiv:2104.04298*, 2021.

[15] Y. Zeng, H. Mao, D. Peng, and Z. Yi, ``Spectrogram based multi-task audio classification,'' *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3705--3722, 2019. http://dx.doi.org/10.1007/s11042-017-5539-3.

[16] K. Doshi, ``Audio deep learning made simple - part 2: Why mel spectrograms perform better,'' *Towards Data Science*, 2021.

[17] P. K. Kurzekar, R. R. Deshmukh, V. B. Waghmare, and P. P. Shrishrimal, ``A comparative study of feature extraction techniques for speech recognition system,'' *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, pp. 18113--18119, December 2014.

[18] M. Labied and A. Belangour, ``Automatic speech recognition features extraction techniques: A multi-criteria comparison,'' *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 8, 2021.

[19] R. Damania, ``Data augmentation for automatic speech recognition for low resource languages,'' 2021. "https://scholarworks.rit.edu/theses/10968".

[20] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, ``Specaugment: A simple data augmentation method for automatic speech recognition,'' in *Interspeech*, 2019.

[21] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, ``Librispeech: an asr corpus based on public domain audio books,'' in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206--5210, IEEE, 2015.

[22] S. J. Arora and R. P. Singh, ``Automatic speech recognition: A review,'' *International Journal of Computer Applications*, vol. 60, no. 9, pp. 1--5, 2012.

[23] P. Lightning, ``torchmetrics documentation,'' 2021. Consulté le 25 Avril 2023, "https://torchmetrics.readthedocs.io/en/stable/".

[24] Graphème, "https://fr.wikipedia.org/wiki/Graph%C3%A8me" , Consulté le 12 Juin 2023.

[25] V. Papadourakis, M. Muller, J. Liu, A. Mouchtaris, and M. Omologo, ``Phonetically induced subwords for end-to-end speech recognition,'' *Alexa Machine Learning, Amazon, USA*, 2020.

[26] D. Wang, X. Wang, and S. Lv, ``An overview of end-to-end automatic speech recognition,'' *Review*, vol. 1, no. 1, pp. 1--11, 2019.

[27] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, ``Deep speech: Scaling up end-to-end speech recognition,'' 2014. https://arxiv.org/abs/1412.5567.

[28] I. Sutskever, O. Vinyals, and Q. V. Le, ``Sequence to sequence learning with neural networks,'' *Advances in neural information processing systems*, vol. 27, pp. 3104--3112, 2014.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, ``Attention is all you need,'' Advances in Neural Information Processing Systems, pp. 5998--6008, 2017.

[30] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, ``Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,'' *Proceedings of the 23rd international conference on Machine learning*, pp. 369--376, 2006.

[31] A. Graves, S. Fernández, and F. Gomez, ``Towards end-to-end speech recognition with recurrent neural networks,'' *International Conference on Machine Learning*, 2014.

[32] A. Graves, ``Sequence transduction with recurrent neural networks,'' *arXiv preprint arXiv:1211.3711*, pp. 1--9, 2012. https://arxiv.org/pdf/1211.3711.

[33] Google Voice Search, "https://en.wikipedia.org/wiki/Google_Voice_Search", Consulté le 22 Mai 2023.

[34] Google Speech to Text Documentation, "https://cloud.google.com/speech-to-text/docs", Consulté le 22 Mai 2023.

[35] Microsoft Azure Speech to Text Documentation, "https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/index-speech-to-text", Consulté le 22 Mai 2023.

[36] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, ``Robust speech recognition via large-scale weak supervision,'' *arXiv preprint arXiv:2212.04356*, 2022.

[37] DeepSpeech Documentation, "https://deepspeech.readthedocs.io/en/r0.9", Consulté le 22 Mai 2023.

[38] Kaldi Documentation, "https://kaldi-asr.org/doc/about.html", Consulté le 22 Mai 2023.

[39] CMUSphinx Documentation, "https://cmusphinx.github.io/", Consulté le 22 Mai 2023.

[40] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, ``Common voice: A massively-multilingual speech corpus,'' *arXiv preprint arXiv:1912.06670*, 2019.

[41] Tatoeba, "https://tatoeba.org/en/", Consulté le 22 Mai 2023.

[42] Wikipedia, "https://kab.wikipedia.org", Consulté le 22 Mai 2023.

[43] Kabyletexts, "https://github.com/MohammedBelkacem/Kabyletexts", Consulté le 22 Mai 2023.

[44] B. F. Dossou and C. C. Emezue, ``Okwugbé: End-to-end speech recognition for fon and igbo,'' *arXiv preprint arXiv:2103.07762*, 2021.

[45] ASR Deep Speech kabyle, "https://github.com/asafu-art/deepspeech-kabyle", Consulté le 22 Mai 2023.

[46] Kabyle_xlsr, "https://huggingface.co/Akashpb13/Kabyle_xlsr", Consulté le 22 Mai 2023.

[47] Stt_kab_conformer_transducer_large, "https://catalog.ngc.nvidia.com/orgs nvidia/teams/nemo/models/stt_kab_conformer_transducer_large", Consulté le 22 Mai 2023.

[48] K. Heafield, ``Kenlm: Faster and smaller language model queries,'' in *Proceedings of the sixth workshop on statistical machine translation*, pp. 187--197, 2011.

[49] You should understand sub-sampling layers within deep learning, https://towardsdatascience.com/you-should-understand-sub-sampling-layers-within-deep-learning-b51016acd551, Consulté le 29 Mai 2023.

[50] What is residual connection, "https://towardsdatascience.com/what-is-residual-connection-efb07cab0d55", Consulté le 29 Mai 2023.

[51] T. pandas development team, ``pandas-dev/pandas: Pandas,'' Feb. 2020. https://doi.org/10.5281/zenodo.3509134.

[52] Y.-Y. Y. et al., ``Torchaudio: Building blocks for audio and speech processing,'' *arXiv preprint arXiv:2110.15018*, 2021.

[53] W. Falcon and The PyTorch Lightning team, ``PyTorch Lightning,'' Mar. 2019. "https://github.com/Lightning-AI/lightning".

[54] M. A. et al., ``Tensorflow: Large-scale machine learning on heterogeneous distributed systems,'' *CoRR*, vol. abs/1603.04467, 2016.

[55] N. S. Detlefsen, J. Borovec, J. Schock, A. Harsh, T. Koker, L. D. Liello, D. Stancl, C. Quan, M. Grechkin, and W. Falcon, ``Torchmetrics - measuring reproducibility in pytorch.'' Journal of Open Source Software, 2022.

[56] E. e. a. Harper, ``Nemo a toolkit for conversational ai and large language models.''

[57] T. Kudo and J. Richardson, ``Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,'' *CoRR*, vol. abs/1808.06226, 2018.

[58] Flask Documentation, "https://flask.palletsprojects.com/en/2.3.x/", Consulté le 29 Mai 2023.

[59] Conda Documentation, "https://docs.conda.io/en/latest/", Consulté le 29 Mai 2023.

[60] Flutter Documentation, "https://docs.flutter.dev/", Consulté le 29 Mai 2023.

[61] record, "https:pub.devpackagesrecord" , Consulté le 12 Juin 2023.

[62] Visual Studio Code - Open Source, "https://github.com/microsoft/vscode".

[63] tao toolkit docs, "https:docs.nvidia.comtaotao-toolkittextasrspeech_-recognition_with_conformer.html" , Consulté le 12 Juin 2023.

[64] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, ``On the variance of the adaptive learning rate and beyond,'' *arXiv preprint arXiv:1908.03265*, 2019.

[65] radam, "https://paperswithcode.com/method/radam", Consulté le 30 Mai 2023.

[66] S. L. Smith, P. Kindermans, and Q. V. Le, ``Don't decay the learning rate, increase the batch size,'' *CoRR*, vol. abs/1711.00489, 2017.

Annexe 1 : Machine learning et Deep learning

Introduction

L'apprentissage automatique, ou le Machine Learning (ML), est une discipline qui permet aux machines d'apprendre à partir des données. Au cours des dernières années, la quantité de données générées par l'homme a considérablement augmenté, et les techniques de ML sont devenues indispensables pour extraire des informations significatives à partir de ces vastes ensembles de données. Le ML permet de découvrir des modèles cachés et des relations complexes dans les données, ouvrant ainsi la voie à des prédictions précises et à des prises de décision basées sur des connaissances. Dans ce chapitre, nous allons présenter le machine learning en général puis le deep learning en particulier [1].

A1.1 Machine Learning (ML) [2]

Le Machine Learning (apprentissage automatique en français) est un outil qui permet de transformer des informations en connaissances. Avec la grande quantité de données générées par l'Homme au cours des dernières années, les techniques de ML sont utilisées pour extraire des modèles (patterns) chachés à partir de données qui seraient autrement inutiles. Les connaissances extraites peuvent ensuite servir à faire des prédictions sur des données nouvelles. Dans cette section nous allons commencer par definir quelques termes en rapport avec le ML, puis nous allons présenter le procédé du ML et enfin nous allons décrire les principales approches du ML.

A1.1.1 Terminologie

Ensemble de données

Un ensemble de données (Dataset en anglais) est une collection de données contenant les caractéristiques importantes pour la résolution d'un problème donné.

Caractéristiques

Les caractéristiques (features en anglais) sont les parties pertinentes des données, elles sont utilisées en entrée pour alimenter un algorithme de ML.

Modèle

Un modèle est une repréntation d'un phénomène, il est produit en entrainant un algorithme de ML sur un ensemble de données.

Généralisation

La généralisation est la capacité du modèle à s'adapter à de nouvelles données (non vues au cours de l'entraînement) et à modéliser la relation réelle entre les entrées et les sorties. Maximiser la généralisation est une partie cruciale du ML.

Fonction de perte

La fonction de perte est une fonction qui calcule la distance entre la sortie du modèle et la prédiction attendue. Par exemple, pour une tâche de regression, l'erreur quadratique moyenne peut être utilisée pour calculer la moyenne des distances entre les valeurs prédites et les valeurs réelles.

Overfitting et underfitting

L'overfitting est une situation ou le modèle est sur-ajusté ou trop complexe pour la quantité de données d'enterainement, et est donc incapable de s'adapter à de nouvelles données. Cette situation est aussi appelée "haute variance", car un changement minime dans les données d'entrée entraine un changement majeur dans la sortie du modèle. L'overfitting peut être reconnu quand la perte d'entraînement est basse alors que la perte de validation et de test sont élevées.

L'underfitting est quant à lui le cas inverse, le modèle est trop simple pour les données utilisées. Le modèle donne en sortie des prédictions similaires pour des entrées similaires, mais les prédicitons sont éronnées. Cette situation est aussi appelée "haut biais". L'underfitting peut être reconnu quand les pertes d'entraînement, de validation et de test sont toutes élevées.

Un bon résultat serait donc un modèle avec une variance et un biais tous les deux bas.

A1.1.2 Procédé du ML

D'abord, il faut collecter les données qui vont servir à entrainer le modèle. Ces données doivent ensuite être prétraitées pour être dans un format adéquat. Les données d'entrée doivent généralement subir une extraction de caractéri- stiques. Il faut également diviser les données en sous-ensembles d'entraînement, de test et de valdiation. Ensuite, l'algorithme de ML doit être entrainé pour apprendre les patterns pertinents à partir des données d'enterainement, les performances du modèle sur l'ensemble de

validation doivent être suivies tout au long de l'enterainement pour éviter l'overfitting. Puis, le modèle est testé sur l'ensemble de test pour éstimer ses performances sur des données nouvelles. Enfin, il est également possible de faire un fine-tuning (réglage fin) du modèle pour maximiser ses performances, qui consiste à ajuster les poids et les paramètres d'un modèle pré-entraîné afin de l'adapter à une tâche ou à un domaine spécifique.

A1.1.3 Approches de ML

a - Apprentissage supervisé

Ce type d'apprentissage a pour but d'apprendre à faire correspondres des entrées à des sorties.

L'apprentissage supervisé s'appelle ainsi, car l'entrainement se fait sur des données étiquetées (labelled data en anglais) ; des pairs d'entrées/sorties sont utilisée au cours de l'entraînement pour que le modèle puisse apprendre la correspondance entre les deux.

La sortie du modèle de ML peut être une catégorie parmi un nombre fini de catégories, auquel cas la tâche s'appelle la classification, comme elle pourrait être une valeur continue (un nombre), et dans ce cas, la tâche s'appelle la regression.

b - Apprentissage non supervisé

Dans ce type d'apprentissage, seul les données d'entrée sont fournies au modèle de ML, qui doit découvrir des patterns dans les données sans que celles-ci ne soient étiquetées.

Parmi les tâches de ML non supervisé, on trouve le clustering, qui vise à regrouper des échantillons avec des caractéristiques semblables dans un même sous-groupe, avec un nombre total de sous groupes prédéfini.

c - Apprentissage semi-supervisé

Ce type d'apprentissage est un mix des deux types précé- dant. Une faible quantité de données étiquetées jointe à une grande quantité de données non étiquetées sont utilisée pour entrainer le modèle. Ce type d'enterainement est utile lorsqu'il est difficile de récolter une quantité suffisante de données étiquetées pour faire un apprentissage supervisé.

d - Apprentissage par renforcement

Dans ce type d'apprentissage, le modèle, ici appelé agent, utilise des récomp- enses pour apprendre. Après une action, l'algorithme reçoit un retour, qui peut être soit

positif, soit négatif et change son comportement pour essayer de maximiser le nombre de recompenses reçues.

A1.2 Réseaux de neurones et Deep Learning

Le deep learning est une branche du ML qui permet d'accomplir des tâches telles que la reconnaissance d'image, la traduction automatique ou encore la reconnaissance de la parole. Dans cette section, nous allons commencer par introduire quelques termes en rapport avec le Deep learning. Ensuite nous allons présenter les réseaux de neurones les plus communs. Enfin, nous allons parler des algorithmes d'optimisation et des techniques de régularisation. [3]

A1.2.1 Terminologie

Perceptron

Un perceptron est une procédure qui traite des données. Il prend en entrée des données, il multiplie chaque entrée par un poid (qui est propre à chaque entrée), somme le tout et y ajoute une constante appelée biais. Ensuite, il applique une fonction appelée fonction d'activation qui diffère selon la tâche, et enfin, il donne le resultat en sortie [3].

Réseau de neurones

Un réseau de neurones est constitué de plusieurs neurones (perceptrons) regroupés en couches. La première couche est appelée couche d'entrées, la dernière est appelée couche de sorties, le reste des chouches intermediaires sont appelées couches cachées [3].

Deep learning et réseau de neurones profond

Le deep learning (apprentissage profond) est une sous-catégorie du Machine learning qui utilise les réseaux de neurones profonds. Un réseau de neurones profond (deep neural network ou DNN) est un réseau de neurones avec plusieurs couches cachées. L'utilisation d'un DNN permet l'accomplissement de tâches plus complexes qu'avec une seule couche cachée [1].

Paramètres

Les paramètres sont les différents poids et biais du réseau de neurones à optimiser durant l'entr- ainement. Ces derniers peuvent être initialisés de façon aléatoire ou à partir d'un modèle avec la même architecture entrainé pour une tâche similaire (ce procédé est appelé apprentissage par transfert, ou transfer learning en anglais) [3].

Taux d'apprentissage

Le taux d'apprentissage (Learning rate en anglais) est une valeur qui influence la covnergence d'un modèle. Si cette valeur est trop élevée, le modèle pourrait ne pas converger. Si elle est trop petite l'entraînement pourrait durer trop longtemps [3].

Planificateur de taux d'apprentissage

Parfois dans le deep learning, il peut être bénéfique pour l'entrainement de réduire le taux d'apprentissage au fur et à mesure que l'entraînement avance. C'est pour ça que sont utilisés les planificateurs de taux d'apprentissage (learning rate schedulers) [4].

Epoque

Une époque (epoch en anglais) est une itération sur l'ensemble de données tout entier. Durant une époque, le modèle est entrainé sur chacun des échanti- llons de l'ensemble de données une seule fois [5].

A1.2.2 Quelques types de réseaux de neurones

Dans cette partie nous allons présenter les types des réseaux de neurones les plus communs : les réseaux de neurones à propagation avant, les réseaux de neurones récurrents et les réseaux de neurones convolutifs [6].

a - Réseaux de neurones à propagation avant

Dans ce type de réseau de neurones, chaque neurone d'un étage est connecté à tous les neurones de l'étage suivant. Les réseaux de neurones à propagation avant (feed-forward neural networks) sont plus performants qu'un perceptron seul, et ils sont utilisés dans diverses tâches.

b - Réseaux de neurones récurrents

Les réseaux de neurones récurrents (Reccurent neural networks ou RNNs) sont un type de réseaux de neurones utilisés pour les séquences. Ils produisent des sorties en prenant compte des entrées précédentes. Ils sont utilisés dans des tâches telles que le traitement de signal ou encore dans la traduction automatique. D'autres variantes de RNNs plus sophistiquées, telles que les LSTMs (Long short-term memory) existent et sont plus adaptées aux séquences d'entrée plus longues.

c - Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (Convolutional neural networks ou CNNs) agissent sur les matrices en appliquant des filtres permettant de réduire la taille des entrées et en extraire les caractéristiques pertinentes. Les CNNs sont largement utilisées notamment dans la vision artificielle.

A1.2.3 Algorithmes d'optimisation

Les algorithmes d'optimisation sont utlisés pour optimiser les paramètres d'un réseau de neurones. Gradient descent est le plus basique de ces algorithmes, et il se décline en trois variantes : Batch Gradient Descent, Stochastic Gradient Descent et Mini Batch Gradient Descent. Dans cette section, nous allons présenter ces trois variantes, ainsi que les autres algorithmes qui s'en dérive.

a - Gradient descent

Le but de gradient descent (déscente du gradient) est de minimiser la fonction de perte. Pour cela, cet algorithme calcul le gradient (dérivée par rapport à chaque poid) de la fonction de perte au point actuel, qu'il multiplie par le Learning rate pour trouver le prochain pas (mise à jour du modèle) [7].

b - Batch Gradient Descent

Dans Batch Gradient Descent, toutes les données d'entraînement sont prises en compte à chaque mise à jour des poids du modèle. La moyenne des gradients de tous les échantillons de l'ensemble de données est utilisée pour mettre à jour les paramètres du modèle. De ce fait, il y a une seule mise à jour pour chaque époque.
Le problème avec cette variante est que si l'ensemble de données est trop grand, calculer les gradients de touts les échantillons pour une seule étape (mise à jour du modèle) s'averre inefficace [8].

c - Stochastic Gradient Descent (SGD)

Dans cette variante, un seul échantillon sélectionné aléatoirement est utilisé à chaque étape. De ce fait, cette méthode est plus efficace et permet une convergence plus rapide lorsque l'ensemble de données est très grand. Mais, du fait de l'utilisation d'un seul exemple à chaque étape, la fonction de perte va fluctuer tout au long de l'entrainement, ce qui empeche l'algorithme de trouver la valeur minimale de la fonction de perte.
SGD peut donc permettre d'atteindre des résultats satisfaisants en moins de temps sans qu'ils soient toujours optimaux [8].

d - Mini Batch Gradient Descent

Cette variante combine les avantages des deux variantes précédentes. En effet, elle permet une convergence rapide pour les grands ensembles de données tout en ayant un entrainement sans fluctuations. Dans cette variante, des mini-batch (lots de quelques exemples) d'une certaine taille fixe (par exemple 32, 64 ou 128) sont formés. Pour chaque étape, un batch est séléctionné et la moyenne des gradients pour ce batch est calculée et est utilisée pour mettre à jour les paramètre du modèle [8].

d - Autres algorithmes d'optimisation

En plus des variantes de Gradient Descent, d'autres algorithmes d'optimi- sation existent. L'un des algorithmes les plus utilisés aujourd'hui est Adam (Adaptive Moment Estimation), qui permet d'entrainer un réseau de neurones de manière plus efficace et en moins de temps par rapport aux différentes variantes de gradient descent. En pratique, chaque algorithme d'optimisation a ses propres avantages et inconveniants et l'algorithme optimal à utiliser varie selon le problème, le modèle, l'ensemble de donées, etc [9].

A1.2.4 Quelques techniques de régularisation

Les techniques de régularisation sont des techniques utilisées en deep learning pour éviter l'overfitting, et ce, en réduisant la complexité du modèle. Parmi elle, on trouve la régularisation L2 et le dropout [10].

L2 Regularization (weight decay)

Cette technique consiste à ajouter un terme à la fonction de perte pour encourager des poids plus petits, et de ce fait, éviter l'overfitting.

Dropout

Cette technique consiste à désactiver un pourcentage des connexions entre les neurones aléatoirement à chaque itération, et ce, dans le but de rendre plus robuste et d'améliorer la généralisation du modèle.

Bibliographie

[1] Deep learning, "https://en.wikipedia.org/wiki/Deep_learning" , Consulté le 6 Juin 2023.

[2] Machine Learning | An Introduction, "https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0", Consulté le 6 Juin 2023.

[3] Introducing Deep Learning and Neural Networks — Deep Learning for Rookies (1)," https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883" , Consulté le 6 Juin 2023.

[4] Learning Rate Scheduler, "https://towardsdatascience.com/learning-rate-scheduler-d8a55747dd90" , Consulté le 6 Juin 2023.

[5] Machine learning glossary and terms, " https:deepai.orgmachine-learning-glossary-and-termsepoch" , Consulté le 6 Juin 2023.

[6] 6 Types of Neural Networks Every Data Scientist Must Know," https://towardsdatascience.com/6-types-of-neural-networks-every-data-scientist-must-know-9c0d920e7fce" , Consulté le 6 Juin 2023.

[7] Gradient Descent Algorithm — a deep dive, "https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21" , Consulté le 6 Juin 2023.

[8] Batch, Mini Batch & Stochastic Gradient Descent, "https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a" , Consulté le 6 Juin 2023.

[9] Various Optimization Algorithms For Training Neural Network, "https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6" , Consulté le 6 Juin 2023.

[10] Types of Regularization in Machine Learning, "https://towardsdatascience.com/types-of-regularization-in-machine-learning-eb5ce5f9bf50", Consulté le 6 Juin 2023.

Annexe 2 : Les caractéristiques de la langue Tamazight [1]

Introduction

Le berbère ou tamazight[1], langue commune, réalisée de différentes façons selon les régions, en Afrique du Nord et au Sahel, existe bien en tant qu'unité structurale profonde (grammaire et fonds lexical communs), mais il n'est pas une langue véhiculaire (ou un outil de communication) et encore moins un standard. Il ne s'exprime et ne se pratique qu'à travers les dialectes qui le composent (chaoui, chleuh, kabyle, rifain, touareg, etc.).

Le berbère est généralement présenté comme un ensemble de dialectes (ou variétés régionales) distincts, à frontières plus ou moins hermétiques, qui sont, le plus souvent, étudiés de façon clivée, voire exclusive. Les pays où le berbère est reconnu comme langue nationale sont : (Mali et Niger, en 1966 ; Algérie en 2001) ou officielle (Maroc, en 2011 ; Algérie, en 2016).

En effet, toute langue doit avoir ou bien choisir ses propres alphabets qui vont être uniques. Dans le cas du berbère, l'utilisation de graphies[2] différentes (arabe, latine, tifinagh) constitue aussi un obstacle à la normalisation et à la diffusion de cette langue.

A2.1 Particularité de la langue Tamazight

Actuellement, on utilise trois alphabets (arabe, latin, tifinagh) pour transcrire le berbère, mais l'usage se fait dans des proportions différentes selon les pays ou même les régions. En Algérie, on emploie concomitamment les trois avec une nette prédominance du latin, d'où dans le cadre de notre projet, on va se focaliser sur l'alphabet latin.

L'alphabet du tamazight contient 23 lettres latines standards et 11 lettres supplémentaires. Sa forme a été fixée par le linguiste Mouloud Mammeri dans les années 1960. Elle est composée de 33 lettres ; 29 consonnes et 4 voyelles longues (a, e, i, u).

[1]Tamazight est le nom originel de la variété berbère parlée au Maroc central (Moyen Atlas) mais aussi le terme générique désignant la " langue berbère".

[2]Graphies c'est toute représentation écrite d'un mot ou d'une lettre.

Les lettres de tamazight avec leurs prononciations en arabe																
A	**b**	**c**	**č**	**d**	**ḍ**	**e**	**ɛ**	**f**	***g***	**ǧ**	**ɣ**	**h**	**ḥ**	**I**	**j**	**K**
ء	ب	ش	تش	د	ذ	ء	ع	اف	*ga*	ج	غ	ه	ح	أي	جو	ك
L	**m**	**n**	**q**	**r**	**ṛ**	**s**	**ṣ**	**t/tt**	**ṭ**	**u**	**V**	**w**	**X**	**Y**	**z**	**ẓ**
ال	ام	ان	ق	ر	رر	س	ص	ت/ث	ط	u	V	و	خ	ي	ز	زز ا

Tableau A2.1: Les lettres de la langue amazighe.

A2.2 Les catégories du mot

Il existe deux catégories principales de mots en tamazight : les noms et les verbes. Les autres mots sont les adjectifs, les adverbes, les pronoms et autres morphèmes [3]. Tamazight, à l'instar des autres langues, a fait des emprunts auprès des langues étrangères avec lesquelles il a été en contact (latin, grec, arabe, turc, français, etc.). Exemples : Le mot akamyun "camion". Le verbe urar "jouer" est concurrencé par le verbe arabe lɛeb.

A2.2.1 Le nom

Il est composé d'un radical accompagné d'éléments indicatifs (genre, nombre, affixes dérivationnels, etc.).

***argaz*« homme »**	
a	***rgaz***
⇩	⇩
Indique que le nom est masculin	Le radical

Tableau A2.2: Les parties du mot "argaz".

En tamazight le nom varie en genre, en nombre et en état (libre/d'annexion).

a - Genre (masculin ou féminin)

Le nom masculin commence généralement par la voyelle initiale : a, i ou u. On forme généralement le féminin préfixant et en suffixant le masculin par t devant le nom masculin et un autre à la fin de mot.

[3]morphèmes : c'est l'unité minimale porteuse de sens, isole par la segmentation d'un mot, le plus souvent dépourvu d'autonomie linguistique

Nom berbère masculin	Signification en français	Féminin de nom berbère
Igider	Aigle	Tigidert
Agujil	Orphelin	Tagujilt
Uccen	Chacal	Tuccent

Tableau A2.3: Quelques exemples de mots convertis du masculin au féminin.

Exceptions sur les noms

Les noms de parenté ne comportent pas de marque de genre en particulier comme Egma (mon frère).

Le nom de nombre se comporte comme adjectifs tel que :

Berbère	Français	Berbère	Français
Sin	Deux .M	**Snat**	Deux. F
Amezwaru	Premier	**Tamezwarut**	Première
Imezwura	Premiers	**Timezwura**	Premières
wis sin	Le deuxième	**Tissnat**	La deuxième

Tableau A2.4: Quelques exemples de noms des nombres convertis du masculin au féminin.

- Des noms qui n'ont pas de voyelle initiale comme : seksu (couscous).
- Des noms féminins qui n'ont pas de masculin comme : tala (fontaine).
- Des noms masculins qui n'ont pas de féminin comme : adfel (neige).

b - Nombre (singulier et pluriel)

Pour décrire le pluriel des noms singuliers de tamazight, on change le a du singulier à i au pluriel tandis que les voyelles initiales i et u restent inchangées.

Pour décrire le pluriel des noms féminins singuliers qui se terminent par i et it on change seulement leurs terminaisons en a, Et ceux qui terminent par a on change leur terminaison en iwin ou wa. On distingue trois types de pluriel : externe, interne et mixte.

Dans le pluriel externe,on ajoute un suffixe au masculin (en, an, awen, ayen, iwen, etc.), ainsi qu'au féminin (in, atin, awin, etc.), sans modifier le radical.

Amɣar	**Vieux**	*Imɣaren*
Inzer	**Nez**	*Inzeren*
Tamɣart	**Vielle**	*Timɣarin*

Tableau A2.5: Exemples du pluriel externe.

Dans le pluriel interne, le radical est modifié par alternance vocalique [4] ou consonantique [5] :

○ transformation d'une voyelle (ou plus) en une autre.

○ disparition d'une voyelle.

○ apparition éventuelle d'une semi-consonne.

Azrar	**Chaîne**	*Izurar*
Amcic	Chat	*Imcac*
Igider	Aigle	*Igudar*
Awtul	Lièvre	*Iwtal*

Tableau A2.6: Exemples de pluriel interne.

Le pluriel mixte est la combinaison des pluriels externe et interne. Le radical est modifié par alternance vocalique ou consonantique et un des suffixes est ajouté. Ce pluriel est caractérisé par la tension d'une consonne du radical.

Abrid	**Chemin**	*Iberdan*
Tamurt	Pays	*Timura*
Adlis	Livre	*Idlisan*
Tawwurt	Porte	*Tiwura*

Tableau A2.7: Exemples de pluriel mixte.

c - État libre/d'annexion

Le nom possède deux états : un état libre et un état d'annexion.

[4] alternance vocalique appelé aussi gradation vocalique ou ablaut désigne un système de gradations des timbres vocaliques en indo-européen qui a encore des effets dans la langues indo-européennes modernes

[5] Alternance consonantique désigne les modifications qui subissent les consonnes dans certaines langues.

Etats de noms	Des catégories des états de noms	*Des exemples*
Libre	Isole	*uccen* « chacal » *argaz*« homme»
	En début de phrase	*Argaz iruḥ ɣer ssuq* L'homme est allé au marché
Annexion **C'est l'action d'un substantif** **Sur un autre, s'exprime de manière suivante :**	Quand il est précédé d'une préposition	*Allen* ***n*** *wuccen* « *Les yeux* ***de*** *chacal* »
	Quand il suit un verbe	*Iruḥ* ***wargaz*** *ɣer ssuq* L'homme est allé au marché

Tableau A2.8: États de nom et leurs catégories.

Les conditions d'apparition du nom à l'état libre ou à l'état d'annexion dans l'énoncé ainsi que les règles de modification de sa première syllabe sont nombreuses et variées.

A2.2.2 L'adjectif

L'adjectif est identifié par sa morphologie et sa syntaxe, et il varie en genre et en nombre. Par exemple le mot rouge "azgaɣ"

En genre	azga***ɣ***	tazga***ɣet***	Quelque chose qui est rouge
En nombre	azga***ɣ***	**Izga***ɣen*	Quelques trucs qui sont rouges
En état d'annexion	azga***ɣ***	**Uzwi***ɣ*/*comme* **Uzwi***ɣ udem*	Son visage est toujours rouge

Tableau A2.9: Un exemple illustrant les variations de l'adjectif.

A2.2.3 L'adverbe

Il existe un ensemble de (noms, prépositions et autres déterminants invariables) qui peuvent avoir un emploi adverbial.

aṭas	⇨	**Beaucoup**	drus	⇨	**Peu**
azekka	⇨	Demain	**tameddit**	⇨	Après-midi

Tableau 12 : Formes figées, composées du nom et/ou d'éléments grammaticaux.

afella	⇨	**Le dessus**	***sufella***	⇨	**Au/par-dessus**
ass	⇨	Jour	*assa*	⇨	Aujourd'hui

Tableau A2.10: Déterminants invariables.

A2.2.4 Le verbe

La forme verbale est composée d'une désinence [6] personnelle et thème verbal qui se compose lui-même d'une racine significative et d'un schème indiquant l'aspect.

La forme verbale *ččiɣ* « j'ai mangé »		
Thème verbal «*ćći*»		Désinences personnelles :*ɣ* qui représente la 1ère personne du singulier.
***čč (est la racine* du verbe *ečč*« manger»)**	*i (qui* exprime le passé composé)	

Tableau A2.11: Un exemple illustrant les parties du verbe "ččiɣ".

A2.3 La structure morphologique d'un mot Tamazight

Le modèle sémitique de formation du mot (racine + schème) s'applique donc tel quel en berbère et il n'est pas sûr que l'analyse du système morphologique berbère selon ce modèle soit totalement conséquente.

- racine est une suite ordonnée de consonnes radicales.
- schème est constitué de divers éléments indicatifs.

Tamazight est une langue générative, les noms et les verbes sont dérivés d'une racine. Nous pouvons engendrer plusieurs mots différents à l'aide d'un schème et à partir d'une même racine. Il existe différents procédés de formation lexicale : la dérivation (la plus courante et la plus productive), la composition et l'expressivité.

[6] Désinence : élément variable qui s'ajoute au radical d'un mot pour marquer la personne, le nombre, la voix, le mode, le temps d'un verbe.

A2.3.1 Dérivation

Il existe deux procédés principaux de dérivation lexicale : la dérivation verbale et la dérivation nominale.

La dérivation verbale se fait en préfixant la base par un morphème dérivationnel (s–, m–, n–, t– ou une de leurs combinaisons).

***awal* « mot »**	⟹	**Base nominale**	⟹	***Siwel* « parler »**
***aker* « voler »**	⟹	Base verbale	⟹	*Ittwaker* « être volé»

Tableau A2.12: Exemples de dérivation verbale.

La dérivation nominale sert à créer des mots tels que le nom verbal, le nom, le nom d'animé, le nom d'instrument, etc., par préfixation d'un morphème (m–, s–, ms–, etc.)

***abrid* «chemin »**	⟹	***amsebrid* « passant »**	⟹	**Nom d'animé**
***aḍen* « être malade »**	⟹	*amaḍun* « malade »	⟹	Nom patient

Tableau A2.13: Exemples de dérivation nominale.

A2.3.2 Composition

Consiste à la formation d'une nouvelle unité lexicale à partir de deux unités lexicales autonomes, elle se fait généralement par la juxtaposition de mots ou sous forme de [7].

[7]Synthème est une concaténation des monèmes (unité de langage de la plus petite taille possible) qui peuvent être analyses en plusieurs unités significatives.

Juxtaposition	**Nom-nom**	**Le nom est «*ifer*»**	***iferzizwi***	**Mélisse**
			ifer-tizizwit	Feuille-abeille
	Verbe-nom	Le verbe est «**Magar**»	**Magritij [19]** magar + iṭij »	Tournesol
		(Se rencontrer)	(rencontrer + soleil)	
			Mager aman Amagraman "va à la rencontre de l'eau" Magar Aman	Inule visqueuse
Synthème	*tawetẓit n uḍar*			Cheville

Tableau A2.14: La juxtaposition et le synthème de mot.

A2.3.3 Expressivité

C'est le fait de rendre les choses (abstraites ou concrètes) plus expressives en les nommant de manière suggestive par des effets lexicaux (mouvement, bruit, son, musicalité, etc.) En ce qui concerne le lexique, l'expressivité est un moyen morphologique qui permet d'attribuer un trait particulier à un mot en lui modifiant la forme et en ajoutant à son sens, une valeur augmentative, diminutive, affective, méliorative, péjorative, etc.

- Affixer un ou deux phonèmes (b, c, f, h, ḥ, ɛ, x, hn, xn, etc.) au mot	aḍad **« Doigt »**		tadaḍet **« Petit doigt ».**
	Inzer « Nez »		*Funzer* « Saigner du nez ».
- Utiliser l'onomatopée	*biṭbiṭ*		Vanneau
- Passer du féminin au masculin et inversement :	*Afus* « Main »	*Tafust* « Petitemain »	Pour Le diminutif
	Argaz« homme »	*Targazt*« femmelette »	Pour le diminutif/dépréciatif
- Dupliquer /redoubler une consonne ou une syllabe :	**Ferfer**« voleter »		

Tableau A2.15: Les opérations qui rendent le mot expressif.

Phonèmes c'est le son de prononciation (sonore) des alphabets berbères.

Onomatopée (mot féminin issu du grec) est une catégorie d'interjections émises pour simuler un bruit particulier associe à un être, un animal ou un objet.

Conclusion

Dans ce chapitre, nous avons évoqué les spécificités majeures de la langue Tamazight et qui peuvent expliquer la différence au niveau du traitement automatique entre Tamazight et les autres langues

Bibliographie

[1] H. C. S. Bessaoudi, « Conception et réalisation d'un traducteur tamazight–français sous Android », Mémoire de master, Université A. Mira de Béjaïa, 2020.

Résumé

Les systèmes de reconnaissance de la parole sont très utiles pour la vie quotidienne. Notre langue, Tamazight, n'est pas dotée d'un tel système, et c'est pourquoi nous avons réalisé ce travail.
Nous avons entrainé un modèle de Deep learning sur des paires d'enregistrements audio et leurs transcriptions.
Le modèle est ensuite exploité grâce à un backend qui permet de générer un trancription à partir d'un fichier audio.
Enfin une application Android réalisée avec Flutter permet aux utilisateurs d'enregister leurs voix et de transcrire la parole en langue Tamazight.

Mots clés : python, apprentissage automatique, apprentissage profond, reconnaissance automatique de la parole, flutter, kabyle, tamazight

Abstract

Speech recognition systems are very useful in everyday life. Our language, Tamazight, does not have such a system, which is why we have undertaken this work.
We trained a deep learning model on pairs of audio recordings and their transcriptions.
The model is then utilized through a backend that generates a transcription from an audio file.
Finally, an Android application made with Flutter allows users to record their voices and transcribe speech into the Tamazight language.

Keywords : python, machine learning, deep learning, automatic speech recognition, flutter, kabyle, tamazight