

Apprentissage automatique et réseau  
de neurones ( AARN )

## Reconnaissance de la parole par le modèle d'Attention

Travail réaliser par :

- BOUALIT Sabrina 201400008574
- BELKADI Widad Hassina 161631054302

Encadrer par :

- Mr GUESSOUM

USTHB 2019-2020

---

Introduction [ BELKADI ] .....	3
Chapitre1: Modèles de séquences avec le mécanisme d'attention .....	4
<b>1.1. Qu'est-ce que SeqToSeq models ? [ BOUALIT ] .....</b>	<b>4</b>
1.1.1. Présentation du modèle SetToSeq .....	4
1.1.2. Exemple d'utilisation [2].....	4
<b>1.2. Architectures de base de modèle SeqToSeq [BOUALIT] .....</b>	<b>4</b>
1.2.1. Réseau de neurones récurrent .....	4
a. L'architecture d'un RNN est décrite dans la Figure 1 :.....	4
b. Problème rencontré par le RNN [4] .....	4
c. Réseau de mémoire à long terme (LSTMs) .....	4
1.2.2. Encodeur/Décodeur.....	5
1.2.3. L'attention dans les modèles de séquences [BELKADI] .....	5
1.2.3.1. Motivation .....	5
1.2.3.2. Intuition et principe .....	6
1.2.3.3. Architecture « attention de bahdanau ».....	6
1.2.3.4. Autre variante « L'attention de Luong ».....	7
1.2.4. Types attention .....	8
<b>2.1. Speech Recognition [ BELKADI ] .....</b>	<b>8</b>
<b>2.2. Le Modèle .....</b>	<b>8</b>
2.2.1. Data : inputs – outputs [ BELKADI ] .....	8
2.2.1.1. Inputs.....	8
Spectrogramme de mel	On utilisera ce format comme input
Spectrogramme	9
2.2.1.2. Outputs .....	9
2.2.2. Model Listen, Attend and Spell (LAS) [ BOUALIT].....	9
2.2.2.1. Qu'est-ce qu'un modèle Listen, attend and Spell (LAS) .....	9
2.2.2.2. BLSTMs (Bidirectionnel LSTM) .....	11
2.2.2.3. Listen.....	11
2.2.2.4. Attend and Spell .....	12
<b>2.3. L'inférence (Decoding “greedy and beam search “ ) [ BELKADI] .....</b>	<b>13</b>
2.3.1. Greedy Search.....	13
2.3.2. Beam Search .....	13
Chapitre 3. Conception, Expérimentations et implémentation du système de reconnaissance vocale .....	15
3.1. Environnement d'implémentation [ BOUALIT].....	15
3.1.1. Présentation de Keras .....	15
3.1.2. Typologie des modèles Keras .....	15
3.2. Dataset et prétraitement [ BELKADI] .....	16
3.3. Architecture et Apprentissage .....	17

3.3.1. Choix de la fonction d'erreur.....	17
3.3.2. Paramétrage de l'architecture et de l'apprentissage [ BELKADI ET BOUALIT] .....	17
3.3.3. Comparaisons et discussions [ BELKADI ] .....	23
3.3.4. Inférence .....	23
Chapitre 4: Implémentation de l'application [ BOUALIT ] .....	25
<b>4.1. Principe de l'application</b> .....	25
<b>4.2. Réalisation</b> .....	25
Conclusion [ BOUALIT ] .....	27
Références .....	27

## Introduction [ BELKADI ]

Tout ce qu'on exprime, en parole ou même en texte, comprend un énorme volume d'informations pertinentes, tous ce qu'on dit, avec quel ton, dans quel sens... Nous devons extraire les valeurs de ces informations, c'est pourquoi il est primordial de trouver des méthodes pour comprendre, analyser, traduire et raisonner sur le langage humain. Ces techniques sont regroupées dans le NLP « Natural Language Processing ».

Le NLP est la branche de l'intelligence artificielle qui étudie, comprend et donne un sens au langage humain. La plupart des techniques de NLP reposent sur l'apprentissage automatique pour tirer un sens des langues humaines. La conversion de l'audio vers le texte aussi appelée la reconnaissance de la parole ou automatic speech recognition (ASR) prend une place importante dans ce domaine.

La reconnaissance vocale, très largement utilisée ces jours-ci, consiste à analyser la voix humaine captée au moyen d'un microphone pour la transcrire sous la forme d'un texte, ceci permet aux êtres humains d'utiliser leur voix pour parler avec une interface d'une manière qui, dans ses versions les plus sophistiquées, ressemble à une conversation humaine normale.

Traditionnellement, la reconnaissance vocale se réalise en 3 étapes : Un modèle acoustique, ensuite, la modélisation de la prononciation et enfin la modélisation du langage. Chacun conçu séparément avec des objectifs différents. En apprentissage profond, nous pouvons entraîner un modèle unique regroupant les étapes précédentes et ce par le biais des réseaux de neurones récurrents et en utilisant un nouveau concept paru en 2015[10] appelé l'attention qui a fait ses preuves dans les modèles de séquences.

Les systèmes ASR permettent à une personne physiquement handicapée de commander et de contrôler une machine. Même les personnes ordinaires préféreraient une interface vocale à un clavier ou une souris. L'avantage est plus évident dans le cas de petits appareils portables. Comme est le cas de l'application que nous souhaitons développer.

Dans le cadre du projet d'AARN, l'idée de l'application est de faire apprendre aux petits enfants la langue arabe à l'aide de la reconnaissance de la parole arabe, une application mobile composée d'exercices amusants et éducatifs. Les objectifs visés par ce projet sont :

- ✚ S'initier dans le NLP et plus précisément le domaine de la reconnaissance vocale.
- ✚ Manipuler les audio et apprendre à les prétraités.
- ✚ Comprendre les modèles seq2seq et les architectures actuelles pour résoudre ses problèmes.
- ✚ Bien comprendre et maîtriser l'architecture des modèles d'attention.
- ✚ Apprendre à intégrer l'attention pour développer le système ASR en utilisant une architecture robuste appelé « Listen attend and spell » [18].
- ✚ Développer une application mobile éducative pour faire apprendre aux enfants la bonne prononciation les mots, à bien articuler et à bien interpréter les mots et phrases et ceci en passant par une suite d'exercices lucides et facile à utiliser qui encourage les enfants à progresser dans l'orale de la langue arabe.

Pour atteindre ces objectifs, nous passons par les étapes suivantes :

- ✚ Chapitre 1 : intitulé « modèles de séquences avec le mécanisme d'attention » : nous présenterons dans ce chapitre les bases du modèle seq2seq ainsi que les principes et l'architecture de l'attention.
- ✚ Chapitre 2 : portant le titre « l'attention dans la reconnaissance de la parole : Listen attend and spell » : ce chapitre portera sur le système ASR en débutant par ce qui est le ASR, en comprenant ses inputs et outputs, l'architecture LAS et enfin l'inférence.
- ✚ Chapitre 3 : et c'est le chapitre le plus important car il englobe la conception, les expérimentations et l'implémentation du système de reconnaissance vocale arabe, on justifie dans cette partie nos choix pour la conception de l'architecture ainsi que le paramétrage de l'apprentissage et montrons les résultats obtenus.
- ✚ Chapitre 4 : intitulé « Implémentation de l'application », nous présentons l'application avec toutes ses fonctionnalités.

# Chapitre1: Modèles de séquences avec le mécanisme d'attention

## 1.1. Qu'est-ce que SeqToSeq models ? [ BOUALIT ]

### 1.1.1. Présentation du modèle SetToSeq

Le modèle SeqToSeq est un modèle qui a été introduit par Google en 2014 afin d'être employé dans différentes applications dont la traduction, la reconnaissance vocale ainsi que le sous-titrage [Sutskever et al, 2014]. Le modèle a été développé afin de palier un problème récurrent dans les applications précédentes à savoir la nécessité de faire correspondre une entrée de taille  $n$  à une sortie de taille  $m$  qui est différente de  $n$  (ex : traduction d'une phrase vers une langue où la phrase est de taille différente). [1]

### 1.1.2. Exemple d'utilisation [2]

Reconnaissance de paroles	Génération de musique	Classification de sentiments
<b>E/S données séquentielle</b> <b>X : Audio</b> → <b>Y : Texte</b> Séquence de fréquence de voix dans le temps réelle.      Séquence de mots	<b>S : données séquentielle</b> <b>X : genre de</b> → <b>Y : Partition</b> <b>music</b> Séquence de symbole Vide ou      de musique	<b>E : données séquentielle</b> <b>X : Phrase</b> → <b>Y : Évaluation</b> Séquence de      Classe de mots      sentiment

## 1.2. Architectures de base de modèle SeqToSeq [BOUALIT]

### 1.2.1. Réseau de neurones récurrent

Afin de pouvoir manipuler des données séquentielles, le modèle SeqToSeq est basé sur les réseaux de neurones récurrents (RNN) [Sherstinsky, 2020]. Ce sont des réseaux de neurones artificiels qui possèdent des connexions récurrentes, c'est-à-dire qu'il existe au moins un cycle dans la structure de ces réseaux qui permet la prise en compte de séquences de données dont la taille est variable. [3]

#### a. L'architecture d'un RNN est décrite dans la Figure 1 :

Le modèle prend en entrée une séquence d'éléments  $(x_1, x_2, \dots, x_t)$ . Dans le cadre du traitement automatique du langage et notamment la traduction automatique, l'exploitation du texte brute n'est pas envisageable. Une approche consiste donc à décomposer le texte en un ensemble de mots puis à encoder chaque mot en un

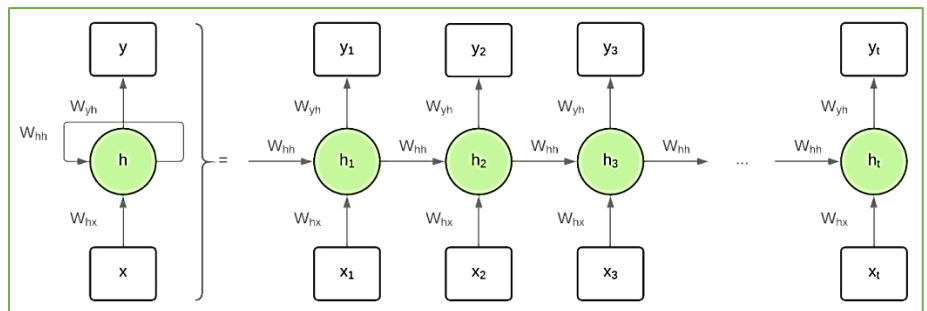


Figure 1 : Architecture d'un réseau de neurones récurrents

vecteur de mot unitaire de dimension  $(D, 1)$  où  $D$  est la taille du vocabulaire qui contient tous les mots, ce vecteur contient des 0 sur toute la colonne sauf à la  $i^{\text{ème}}$  ligne correspondante à l'index du mot dans le vocabulaire.

Lors du traitement de la séquence, l'information extraite à partir de l'élément  $x_i$  correspondante à  $h_i$  est mémorisée afin d'être prise en compte lors du traitement de l'élément suivant  $x_{i+1}$  ce qui permet de prendre en considération les éléments précédents de la séquence lors de la prédiction d'un élément en sortie. Ceci permet notamment d'améliorer la cohérence et la significativité de la séquence prédite par le modèle. Enfin, le modèle produit en sortie une séquence d'éléments  $y(y_1, y_2, \dots, y_t)$ . Afin d'entraîner le modèle et corriger ses poids, on utilise généralement la fonction de perte d'**entropie croisée** afin de calculer l'erreur entre l'élément réel et l'élément prédit puis corriger les poids du modèle en conséquence.

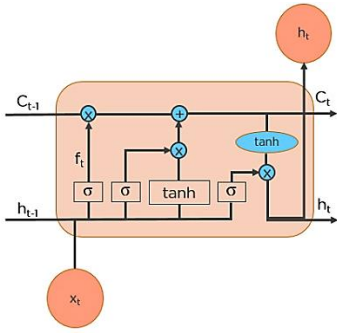
#### b. Problème rencontré par le RNN [4]

problème de la disparition des gradients	Problème de gradient explosif
Les gradients portent des informations utilisées dans le RNN, et lorsqu'il devient trop petit, les mises à jour des paramètres deviennent insignifiantes. Cela rend difficile l'apprentissage pour les séquences de données longues.	Ce problème survient lorsque de grands gradients d'erreur s'accumulent, ce qui entraîne des mises à jour très importantes des poids du modèle de réseau neuronal pendant le processus d'apprentissage.

#### c. Réseau de mémoire à long terme (LSTMs)

Le LSTM pour long short Term memory est une variante du RNN capable d'apprendre des dépendances à long terme. Ils implémentent également la récurrence de la même manière qu'un RNN simple mais au lieu d'une seule couche, il existe quatre couches « d'entrée, de sortie, de candidature et d'oubli » qui interagissent de manière très spécifique appelées portes [11]. Pour cela cette architecture est utilisée afin de détourner les problèmes cités précédemment.

## Fonctionnement des LSTMs [4]



**Première étape :** Décider quelles informations doivent être omises de la cellule dans ce pas de temps particulier. La fonction sigmoïde détermine cela. Il regarde l'état précédent ( $h_{t-1}$ ) avec l'entrée courante  $x(t)$  et calcule la fonction.

**Dans la deuxième couche, il y a deux parties :** La fonction sigmoïde : décide des valeurs à laisser passer (0 ou 1). Et la fonction tanh donne un poids aux valeurs qui sont passées, en décidant de leur niveau d'importance (-1 à 1).

**La troisième étape consiste à décider quelle sera la sortie.** Nous exécutons une couche sigmoïde, qui décide quelles parties de l'état de la cellule parviennent à la sortie. Puis nous mettons l'état de la cellule à travers tanh pour pousser les valeurs entre -1 et 1 et le multiplier par la sortie de la porte sigmoïde.

### 1.2.2. Encodeur/Décodeur

Le modèle SeqToSeq adopte une architecture d'encodeur/décodeur qui consiste en 2 parties [3]:

#### Encodeur

Prend en entrée une séquence  $x$  où  $x_i$  est un élément possédant l'ordre  $i$  dans cette séquence. L'encodeur consiste en un ensemble d'unités récurrentes, chaque unité correspond à un RNN qui traite un élément  $x_i$  de la séquence  $x$  en entrée.

L'information extraite de l'élément  $x_i$  à partir de chaque unité appelée  $h_i$  est propagée vers l'unité suivante. Cette information est calculée selon la formule suivante :

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \text{ où :}$$

- $h_{t-1}$  est l'information extraite de l'unité récurrente précédente
- $x_t$  est un élément d'ordre  $i$  dans la séquence  $x$
- $W^{(hh)}$  est le poids correspondant à  $h_{t-1}$
- $W^{(hx)}$  est le poids correspondant à  $x_t$

L'encodeur produit en sortie un vecteur calculé via la formule précédente qui va encapsuler les informations extraites à partir de la séquence en entrée, ce vecteur sera exploité par la seconde partie de l'architecture, à savoir le décodeur.

#### Décodeur

Consiste en un ensemble d'unités récurrentes, chaque unité correspond à un RNN qui produit un élément  $y_t$  à partir d'un élément du vecteur produit par l'encodeur à chaque intervalle de temps  $t$ . Le décodeur produit donc en sortie une séquence  $y$  où  $y_i$  est un élément possédant l'ordre  $i$  dans cette séquence. Comme dans le cas de l'encodeur, l'information extraite de l'élément  $i$  du vecteur produit par l'encodeur à partir de chaque unité appelée  $h_i$  est propagée vers l'unité suivante via la formule suivante :

$$h_t = f(W^{(hh)}h_{t-1}) \text{ où :}$$

- $h_{t-1}$  est l'information extraite de l'unité récurrente précédente
- $W^{(hh)}$  est le poids correspondant à  $h_{t-1}$

La sortie  $y_t$  est calculée à chaque intervalle de temps  $t$  via une fonction d'activation permettant de créer un vecteur de probabilités afin de déterminer l'élément en sortie qui correspond le mieux à la séquence en entrée via la formule suivante :

$$y_t = \text{softmax}(W^S h_t) \text{ où :}$$

- $h_t$  est l'information extraite de l'unité récurrente précédente
- $W^S$  est le poids correspondant à  $h_t$

 Nous représentons ci-dessous l'architecture encodeur/décodeur :

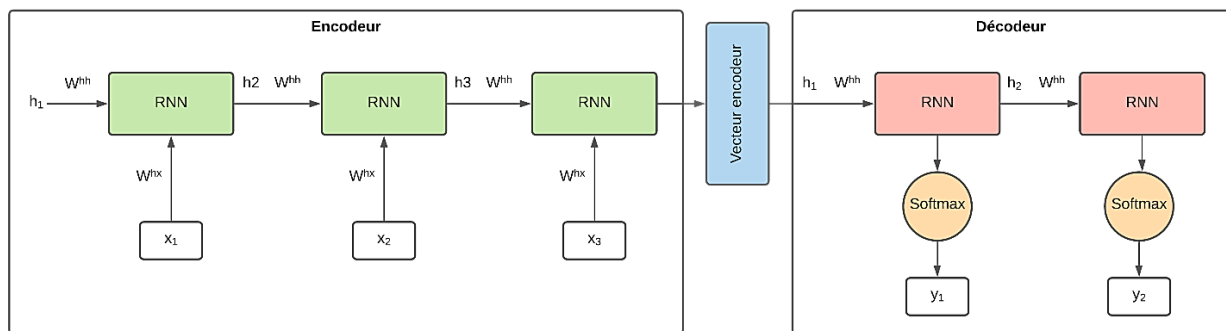


Figure 2 : Architecture du modèle SeqToSeq

### 1.2.3. L'attention dans les modèles de séquences [BELKADI]

#### 1.2.3.1. Motivation

La motivation de l'attention est dû aux limites des encodeur/décodeur dans les modèles à séquences. Comme nous l'avons vu, l'encodeur prend l'entrée et la convertit en un vecteur de taille fixe, puis le décodeur effectue une prédiction et donne une séquence de sortie. Cela fonctionne bien pour une séquence courte mais cela échoue lorsque nous avons une longue séquence car, il devient difficile pour le décodeur de mémoriser la séquence entière dans un vecteur de taille fixe et de compresser toutes les informations contextuelles de la séquence.

L'être humain lors de la traduction par exemple, ne procède pas comme le principe des encodeur/décodeur il prend la phrase source la décompense et fait la traduction morceau par morceau et d'où la motivation des modèles d'attention [5].

### 1.2.3.2. Intuition et principe

Avant d'expliquer le principe donnant le droit aux créateurs de cette magie [9,10] :

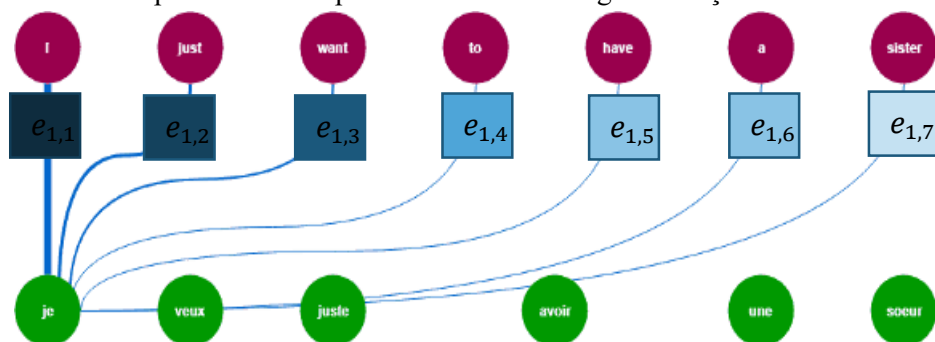
✚ dzmitry bahdanau

✚ Kyunghyun Cho

✚ Yoshua Bengio

Le vecteur unique généré par l'encodeur passé au décodeur porte la charge de coder la phrase entière. L'attention permet au réseau du décodeur de se «concentrer» et «à donner de l'attention» uniquement aux parties importantes des sorties de l'encodeur pour chaque étape des propres sorties du décodeur.

On calcule d'abord un ensemble de poids d'attention. Ceux-ci seront multipliés par les vecteurs de sortie de l'encodeur pour créer une combinaison pondérée. Le résultat doit contenir des informations sur cette partie spécifique de la séquence d'entrée, et ainsi aider le décodeur à choisir les bons mots de sortie. Voici un schéma qui explique nos dires en prenons l'exemple de la traduction anglais-français :



Ce schéma montre les scores d'attention calculé à partir de la sortie de l'encodeur pour le pas de temps « je » du décodeur on peut voir que le poids approprié au pas de temps « I » de l'encodeur vers la sortie « Je » est le plus grand( intensité de la couleur ) , et c'est ce qu'on essaye de faire apprendre au modèle .

Figure 3 mécanisme de l'attention [8]

### 1.2.3.3. Architecture « attention de bahdanau »

Nous avons compris le principe théorique de l'attention mais par exemple comment les poids d'attention sont-ils calculer ? Comment construire la couche d'attention ? Nous allons entamer maintenant les aspects techniques de l'attention.

✚ La couche d'attention suit les étapes suivantes :



✚ Notation : On fixe ces notations pour faciliter l'explication :

$h_1, h_2, h_2 \dots h_{t'} \dots h_n$	États (sorties) de l'encodeur (n= taille de la sortie de l'encodeur)
$y_1, y_2 \dots y_t \dots y_m$	États (sorties) du décodeur (m=taille de la sortie du décodeur)
$S_1, S_2 \dots S_t \dots S_m$	États cachés du décodeur (m=taille de la sortie du décodeur)
$e_{1,1}, \dots, e_{2,1} \dots e_{3,1} \dots e_{t,t'} \dots e_n$	$e_{t,t'}$ représente le score accordé à $h_{t'}$ pour calculer la sortie $y_t$
$\alpha_{1,1}, \dots, \alpha_{2,1} \dots \alpha_{3,1} \dots \alpha_{t,t'} \dots \alpha$	$\alpha_{t,t'}$ représente le poids d'attention donnée à $h_{t'}$ pour calculer la sortie $y_t$
$C_1, C_2 \dots C_t \dots C_m$	Le vecteur de contexte

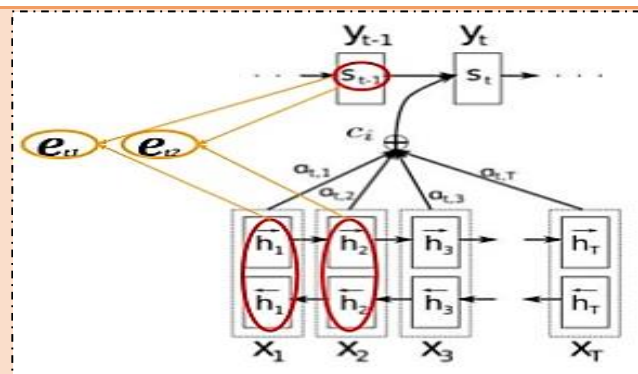


Figure 4 Mécanisme attention [12]

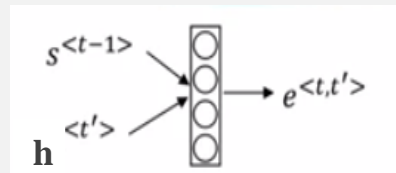
1. Calcul des scores  $e_{t,t'}$

2. Calcul des poids  $\alpha_{t,t'}$



On calcule le score  $e_{t,t'}$  de chaque état  $h_{t'}$  de l'encodeur pour chaque sortie du décodeur  $y_t$ , on reprend l'exemple précédent de traduction, on doit avoir les scores

$e_{1,1}, e_{1,2}, e_{1,3}$  assez élevé par rapport aux autres scores pour prédire le « I ». les scores sont calculés ainsi [9]:



- ✓  $S^{<t-1>}$  : état précédent du décodeur
- ✓  $h^{<t'>}$  : état  $t'$  de l'encodeur.

Et donc les scores sont calculés à partir d'un réseau de neurone.

$$e_{t,t'} = W_{t,t'} \tanh(W_{Enc_i} H_{t'} + W_{Dec_j} S^{t-1})$$

Où  $W$ ,  $W_{Enc}$  et  $W_{Dec}$  sont des matrices de poids.

### 3. Calcul du contexte $C_t$

Après avoir calculé les poids ou montants d'attention  $\alpha_{t,t'}$  on passe au calcul du vecteur de contexte qui sera utilisé par le décodeur pour prédire le mot suivant dans la séquence. Il est calculé ainsi [9]:

$$C_t = \sum_{t'=1}^n \alpha_{t,t'} h_{t'}$$

Si on reprend l'exemple de la traduction on peut voir que si les poids  $\alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}$  sont élevés et les autres sont faibles alors le vecteur de contexte contiendra plus d'informations sur les états  $h_1$  et  $h_2$  et  $h_3$  et relativement moins d'informations sur les autres

Le poids d'attention  $\alpha_{t,t'}$  représente combien doit-on donner de l'attention à l'état de l'encodeur  $t'$  pour produire la sortie  $t$ , ils sont calculés à partir des scores  $e_{t,t'}$  à l'aide de la formule suivante [9]:

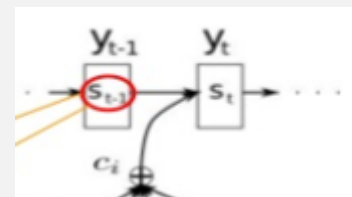
$$\alpha_{t,t'} = \frac{\exp^{e_{t,t'}}}{\sum_{t'=1}^n \exp^{e_{t,t'}}}$$

Cette formule correspond exactement à une couche avec fonction d'activation soft max avec comme condition:

$$\sum_{t'=1}^n \alpha_{t,t'} = 1$$

### 4. Ajout de $C_t$ avec l'état caché du décodeur précédente

Dernière étape consiste à faire combiner le contexte et l'état précédent du RNN pour sortir un nouveau état caché et output par la cellule courante [9]:



Cette procédure de 1 à 4 se répète  $m$  fois (taille de la sortie du décodeur) pour produire à la fin la séquence en output.

#### 1.2.3.4. Autre variante « L'attention de Luong »

Il existe un autre type d'attention celui proposé par Thang Luong [27], la différence entre ces deux types est résumée en deux points [28]:

✚ La façon de calculer les scores  $e_{t,t'}$

✚ La position dans laquelle le mécanisme d'attention est introduit dans le décodeur.

1. Décodeur RNN

2. Calcul des scores  $e_{t,t'}$

3. Calcul des poids  $\alpha_{t,t'}$

4. Calcul du contexte  $C_t$

5. production de la sortie finale

La différence intervient dans les étapes 1, 2 et 5 que nous redéfinissons [28]:

#### 1. Décodeur RNN

Contrairement à l'attention de Bahdanau, l'utilisation du décodeur n'est pas la dernière étape du décodage, mais plutôt la première, l'état caché et la sortie du décodeur précédent sont passés au décodeur RNN pour générer un nouvel état caché pour ce pas de temps.

#### 3ème et 4ème étape : Calcul des poids et du contexte

Ces deux étapes sont similaires à celle de Bahdanau, on fait entrer le soft max pour calculer les poids à partir des scores et on calcule le contexte en multipliant l'état de l'encodeur et les poids.

#### 2. Calcul des scores d'alignement $e_{t,t'}$

On calcule le score  $e_{t,t'}$ , en utilisant le nouvel état masqué du décodeur calculé en étape 1 et les états masqués de l'encodeur, contrairement à Bahdanau, Luong a proposé plusieurs façons de calculer les scores multiplication, générale et concaténation, on présente la première car c'est ce qu'on utilisera [28]:

✚ L'opération dot ou multiplication est la plus simple consiste à multiplier l'état de l'encodeur et l'état caché produit par le décodeur:

$$score_{\text{uneljelement}} = H_{\text{encodier}} \cdot H_{\text{decodier}}$$

#### 5. production de la sortie finale

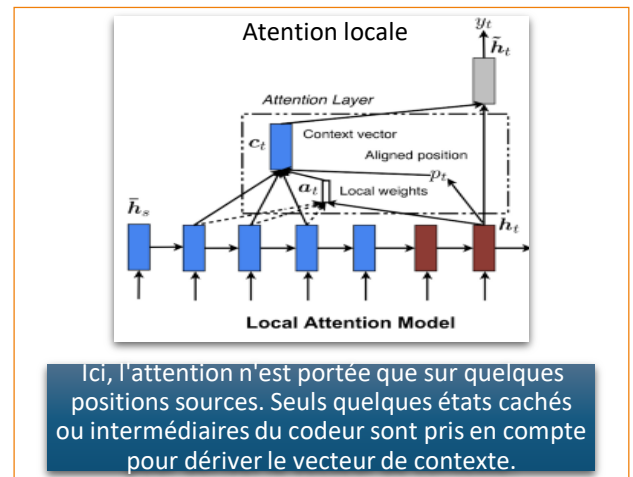
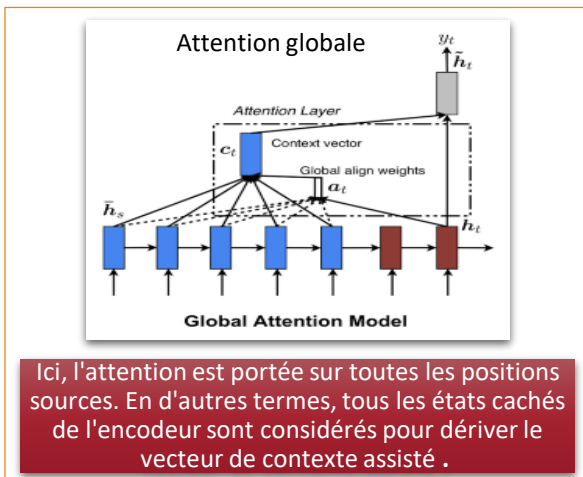
Dernière étape consiste à faire concaténer le contexte et l'état actuel du RNN générer en étape 1 pour sortir un nouvel état caché:

Cette procédure de 1 à 5 se répète  $m$  fois (taille de la sortie du décodeur) pour produire à la fin la séquence en output.



### 1.2.4. Types attention

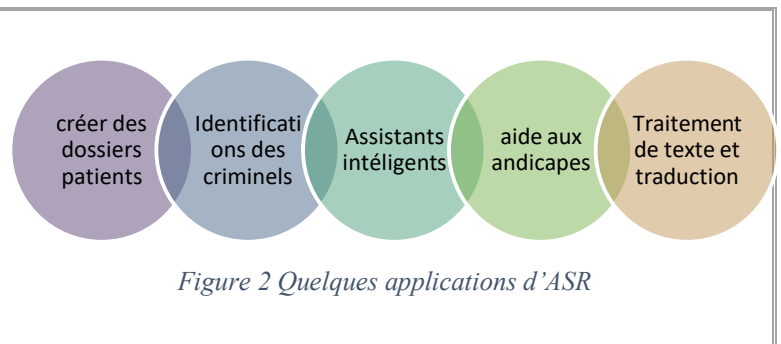
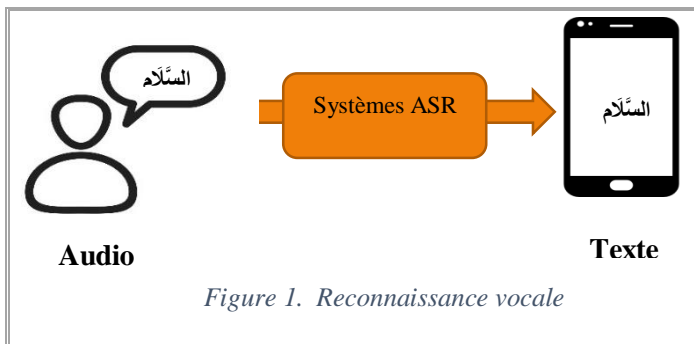
Il existe deux types d'attention [8] :



## Chapitre 2: l'attention dans la reconnaissance de la parole : Listen attend and spell

### 2.1. Speech Recognition [ BELKADI ]

Le traitement du langage naturel, généralement abrégé en NLP, est une branche de l'intelligence artificielle qui traite l'interaction entre les ordinateurs et les humains en utilisant le langage naturel. L'une des applications de cette branche La reconnaissance automatique de la parole, en anglais, automatique speech recognition (ASR), est le problème de transformer un audio (une parole) en texte. C'est une des tâches importantes dans le domaine de l'intelligence artificielle et elle ne manque pas d'applications, Nous utilisons la reconnaissance vocale au quotidien, on montre quelques application dans la figure 2.



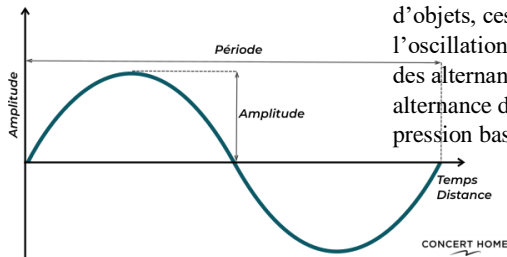
### 2.2. Le Modèle

#### 2.2.1. Data : inputs – outputs [ BELKADI ]

##### 2.2.1.1. Inputs

Nous présentons maintenant le format des entrées du modèle de prédiction, il est claire que nous avons à notre disposition un son (audio) mais comment représenter cette audio ?

## 1. Qu'est-ce que le son ?

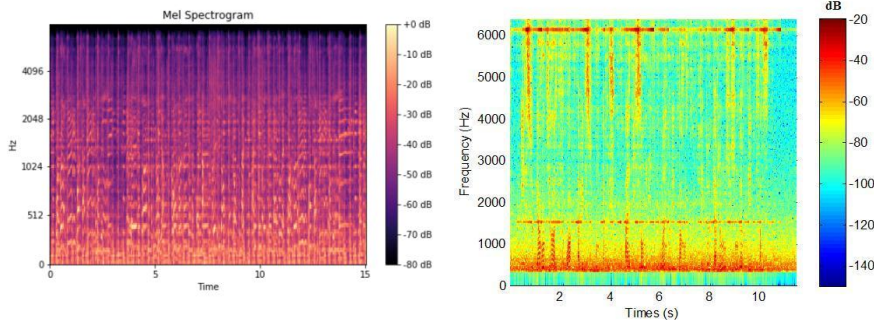


Le son est le produit d'une vibration d'objets, ces vibrations déterminent l'oscillation des molécules d'air qui crée des alternance de pression d'air, une alternance de pression forte et une autre de pression basse produit une vague (wave en anglais).

### Spectrogramme de mel

On utilisera ce format comme input

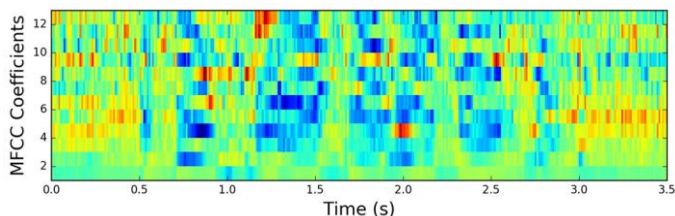
### Spectrogramme



On utilisera aussi ce format comme input

**Mel-Frequency Cepstral Coefficients** sont des coefficients cepstraux calculés par une transformée en cosinus discrète appliquée au spectre de puissance d'un signal. Calculé ainsi :

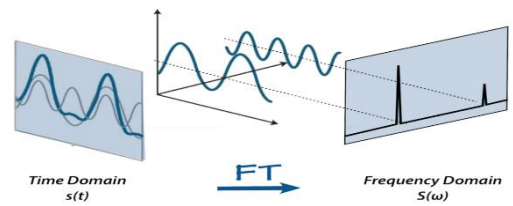
1. Calcul de la transformée de Fourier de la trame à analyser
2. Pondération du spectre d'amplitude (ou de puissance selon les cas) par un banc de filtres triangulaires espacés selon l'échelle de Mel
3. Calcul de la transformée en cosinus discrète du log-mel-spectre.



## 5. MFCCs [16]

## 2. Transformé de Fourier rapide (FFT)

Est une opération qui permet de représenter en fréquence des signaux qui ne sont pas périodiques [12] :



Amplitude en fonction du temps

Magnitude en fonction de la fréquence

## 3. Transformé de fourrier à court terme(STFT)

Ou encore transformée de Fourier à fenêtre glissante, fait plusieurs FFT dans différents intervalles appelées fenêtres. Le carré de son module donne le spectrogramme (fréquence + temps) [13].

## 4. Spectrogramme et spectrogramme de mel

- Un **spectrogramme** est une représentation visuelle du **spectre** de **fréquences** d'un signal tel qu'il varie avec le temps, associant à chaque fréquence une intensité ou une puissance (représenter par des couleurs) en dB (décibel) peut être générer avec la STFT [14] (voir la figure).
- L'être humain est capable de faire la différence entre 500Hz et 1000 Hz mais pas entre des hautes fréquences comme 10000 et 10500Hz malgré que la différence est la même[12].
- L'**échelle de mel** est le résultat d'une transformation non linéaire de l'échelle de fréquence. Son but est de rendre les sons à égale distance les uns des autres sur l'échelle de Mel, «sonnent» également pour les humains car ils sont égaux en distance les uns des autres[15].
- Un **spectrogramme de mel** est un spectrogramme dans lequel les fréquences sont converties en échelle de mel[12].

### 2.2.1.2.Outputs

L'output attendu du modèle est le texte correspondant au vocal en entrée, qui sera représenter comme une suite de caractères, car le modèle est de niveau caractères, on doit donc modifier le format des textes, en prend un exemple explicatif :

Je m'appelle Widad

<start>	J	e	m	'	a	...	d	a	d	<End>
Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>	Y <sub>4</sub>	..	.	.....	...	...	Y <sub>n-1</sub>	Y <sub>n</sub>

## 2.2.2. Model Listen, Attend and Spell (LAS) [ BOUALIT]

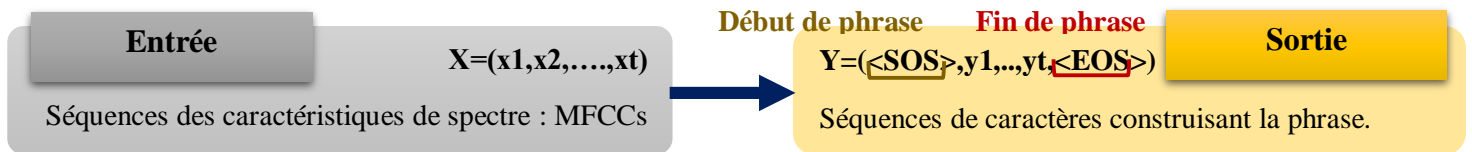
### 2.2.2.1. Qu'est-ce qu'un modèle Listen, attend and Spell (LAS)

Nous présentons Listen, Attend and Spell (LAS), un réseau de neurones qui apprend à transcrire les énoncés vocaux aux personnages. Le modèle apprend tous les composants d'un système de reconnaissance vocale conjointement. Notre système a deux composants principaux [18]:

**Auditeur ( Listener ) :** représente un Encodeur en RNN, son opération principale est : Écouter

**Orthographe ( Speller ) :** représenter par un Décodeur en RNN basé sur l'attention

Dans cette section, nous décrivons formellement le modèle LAS qui permet l'émission d'un ensemble de caractères sans faire d'hypothèses d'indépendance entre les caractères. Il s'agit de la principale amélioration du LAS par rapport aux précédents modèle [18]:



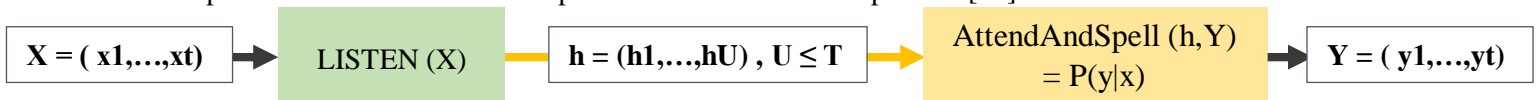
La modélisation de chaque sortie de caractère  $y_i$  est représenté comme une distribution conditionnelle sur les caractères précédents  $y < i$  et le signal d'entrée  $x$  [18]

$$P(y | x) = \prod_{y < i} P(y_i | x),$$

### Remarque

Sur un sous-ensemble de la tâche de recherche vocale Google, LAS réalise un taux d'erreur sur les mots de 14,1% sans dictionnaire ni modèle de langage, et 10,3% avec un modèle de langage recalculé. [17]

La fonction Listen transforme le signal d'origine  $x$  en une représentation de haut niveau  $h = (h_1, \dots, h_U)$  avec  $U \leq T$ , tandis que La fonction AttendAndSpell consomme  $h$  et produit une distribution de probabilité sur les séquences de caractères précédents. Nous fournissons plus de détails sur ces composants [18]:



🌈 Visualisation de l'architecture LAS

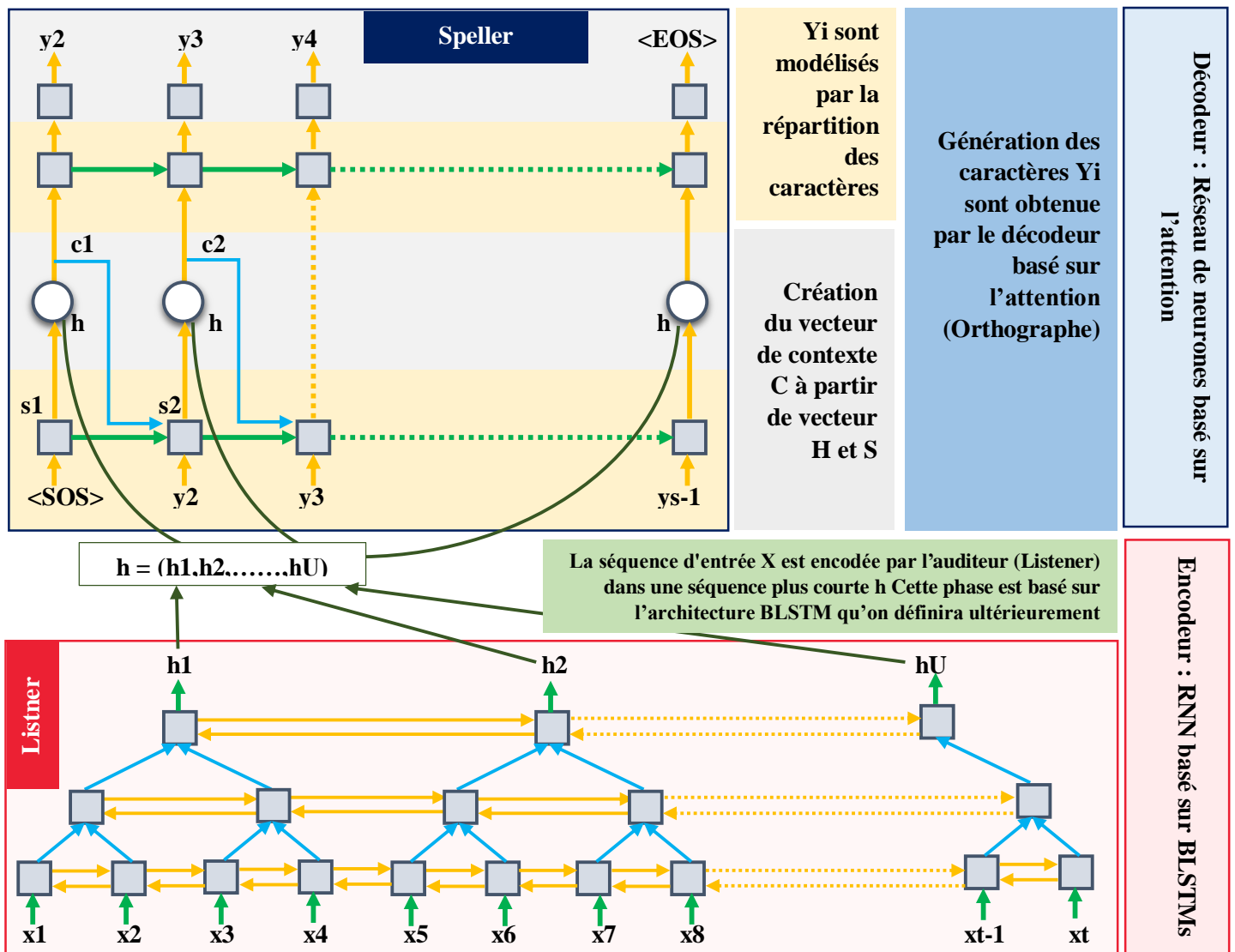


Figure 5: Représentation du model LAS

#### 2.2.2.2. BLSTMs (Bidirectionnel LSTM)

Un RNN bidirectionnel (BRNN) est un modèle proposé pour supprimer diverses restrictions RNN conventionnels. Ce modèle divise les états de neurones RNN réguliers en avant et en arrière.

En d'autres termes, il existe deux réseaux récurrents différents, en avant et en arrière. Ces deux réseaux se connectent à la même couche de sortie pour générer des informations de sortie. Avec cette structure, les situations passées et futures d'entrées séquentielles dans un laps de temps sont évaluées sans délai [19].

La version LSTM de la structure BRNN est appelée Bidirectionnel LSTM (BLSTM). Cette version peut améliorer les performances du modèle LSTM dans les processus de classification. Contrairement à la norme Structure LSTM, deux réseaux LSTM différents sont formés pour les entrées séquentielles dans le BLSTM architecture. [19]

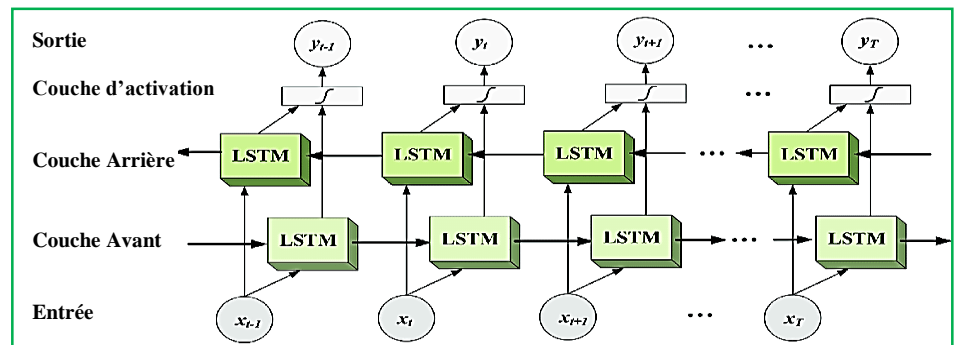


Figure 6: Structure BLSTM de base avec un fonctionnement séquentiel des entrées

#### 2.2.2.3. Listen

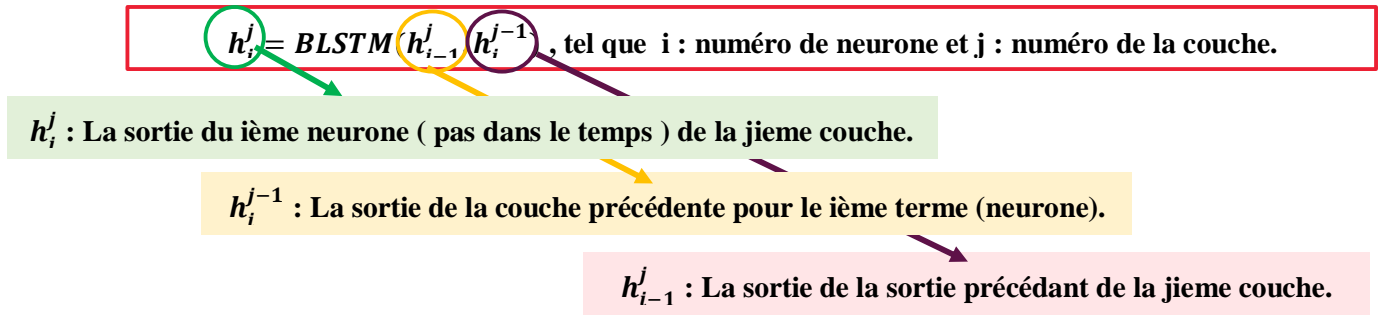
L'opération d'écoute utilise un RNN de mémoire bidirectionnelle à long court terme (BLSTM) Cette modification est nécessaire pour réduire la longueur  $U$  de  $h$  par rapport à  $T$  la longueur de l'entrée  $x$ , car les signaux vocaux d'entrée peuvent avoir des centaines à des milliers de trames. [18]

Une application directe de BLSTM pour l'opération Listen converge lentement et produit des résultats inférieurs à ceux rapportés ici, même après une grande période d'apprentissage. C'est probablement parce que l'opération AttendAndSpell a du mal à extraire les informations pertinentes d'un grand nombre d'entrée. [18]

Nous contournons ce problème en utilisant deux phases :

- Réseau neuronal convolutif (CNN) afin de nous permettre d'utiliser les X pour extraire les caractéristiques et les présenter sous un format adéquat avec le format traité par les BLSTM.
- Utilisation de multicouche BLSTM similaire (3 couches) tel que dans chaque couche BLSTM empilée successivement, nous réduisons la résolution temporelle d'un facteur de puissance 2.

Dans une architecture BTLM profonde typique, la sortie au  $i$ -ème pas de temps, de la  $j$ -ème couche est calculée comme suite [18]:

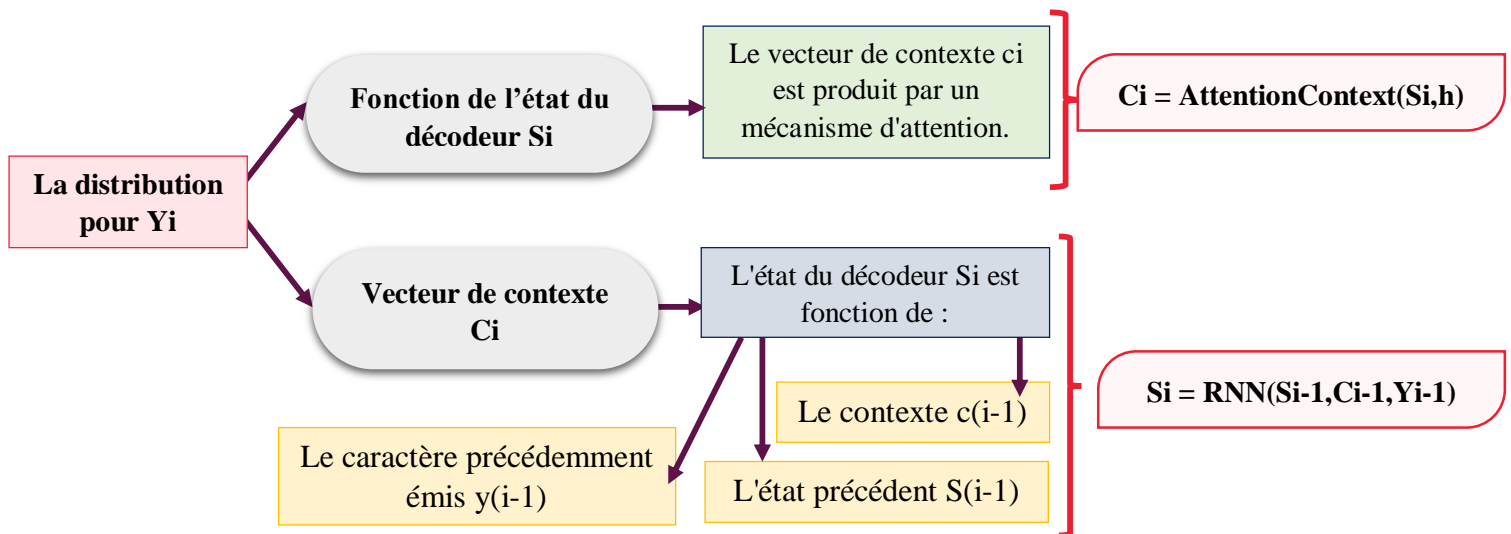


La réduction de la résolution temporelle obtenue grâce à la couche BLSTMs, cela permet au modèle d'attention (détaillé dans section suivante) d'extraire les informations pertinentes à partir d'un plus petit nombre de pas. En plus de réduire la résolution, l'architecture profonde permet au modèle d'apprendre des représentations d'entités non linéaires des données. [18]

#### 2.2.2.4. Attend and Spell

Nous décrivons maintenant la fonction AttendAndSpell. La fonction est calculée à l'aide d'un transducteur LSTM basé sur l'attention [18]. À chaque étape de sortie, le transducteur produit une distribution de probabilité sur le caractère suivant conditionné sur tous les caractères vus précédemment.

La distribution pour  $y_i$  est une fonction de l'état du décodeur «  $s_i$  » et du contexte «  $c_i$  ». L'état du décodeur  $s_i$  est fonction de l'état précédent «  $s_{i-1}$  » du caractère précédemment émis  $y_{i-1}$  et du contexte  $c_{i-1}$ . Le vecteur de contexte  $c_i$  est produit par un mécanisme d'attention. Plus précisément :



Et enfin nous définissons la formule de distribution de  $Y_i$  :

$$P(y_i | Y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

Où CharacterDistribution est un RNN à 2 couches LSTM avec des sorties softmax sur les caractères précédents. Tel que :

## Pour chaque pas de temps $i$ ( neurones) du décodeur

### **C = AttentionContext()**

- C est un vecteur de contexte qui encapsule les différentes informations concernant le signal d'entrée pour générer le caractère suivant.
- $S = (h_1, \dots, h_U)$  représente le vecteur de pas de temps  $u$  de  $h$  pour générer un vecteur d'attention  $\alpha_i$
- $\alpha_i$  est utilisé pour le combiner linéairement avec les vecteurs  $h_u$  pour créer  $c_i$

### 2.3.L'inférence (Decoding “greedy and beam search “ ) [ BELKADI]

Il est temps maintenant d'utiliser le modèle appris pour produire de nouvelles prédictions sur de nouveaux audio. Le but est de produire la meilleure séquence de caractères correspondante à l'audio donné en input. L'output du modèle LAS est un vecteur de distributions probabilistes sur tous les caractères possibles du vocabulaire

de la langue pour chaque caractère de la séquence. Nous illustrons dans ce tableau un exemple d'output du modèle LAS. Soit un vocabulaire de 28 caractères de la langue arabe, et la longueur d'une séquence est de 5 caractères. La somme de chaque ligne est égale à 1.

	P(ل)	P(ب)	P(ت)	P(ج)		P(ي)
Car1	<b>0.5</b>	0.2	0.02	0.001	...	0.31
Car2	0.3	0.06	0.03	<b>0.2</b>	...	0.001
Car3	0.01	<b>0.54</b>	0.06		..	0.1
Car4	<b>0.22</b>	0.06	0.009		..	0.07
Car5	0.0006	<b>0.16</b>	0.12		...	0.008

Pour transformer cet output en un texte arabe compréhensible il nous faut un algorithme de décodage. Vu qu'on est en inférence, l'input de l'encoder est facile à deviner c'est l'audio, tant qu'à l'input du décodeur est inconnu. On initialise le décodeur avec un jeton de début de phrase puis à chaque fois on prédit le caractère suivant. On commence par présenter un algorithme de recherche gourmand ensuite on passe à une recherche plus compliquée mais plus efficace appelé recherche de faisceau.

#### 2.3.1. Greedy Search

Aussi nommée décodeur de meilleur chemin est la plus simple des méthodes de décodage le principe est de prendre à chaque instant le caractère ayant la plus grande probabilité puis concaténer tous les caractères prédits. Selon l'exemple précédent nous aurons comme output : الباب. Nous présentons l'algorithme :

Algorithme Greedy_search	
<b>Input :</b> audio , decoder_inpt : Tableau[1 .. maxCarac]=[startToken,..] , modèle	
<b>Output :</b> prédiction = Tableau [1 .. maxCara] de caractères	
<b>Debut</b>	
	Prédictions = modele.predict(audio , decoder_inpt ) // prédire
	Sequences=[argmax(cara) for cara in predictions] //liste d'index pour la plus grande probabilité chaque ligne
	Retourner séquences
<b>Fin</b>	

Le choix d'un seul meilleur candidat peut convenir pour le pas de temps actuel, mais lorsque nous construisons la phrase complète, cela peut être un choix sous-optimal [5]. Et donc cette approche a l'avantage d'être simple et très rapide mais on reproche à son résultat d'être loin de la prédiction optimale.

#### 2.3.2. Beam Search

Contrairement à choisir pour chaque pas de temps le meilleur caractère, l'algorithme de recherche de faisceau sélectionne à chaque fois plusieurs alternatives comme entrées à chaque pas de temps sur la base d'une probabilité conditionnelle. Le nombre d'alternative est appelé Beam Width (largeur du faisceau) « B ». A chaque pas de temps, la recherche extrait les B meilleurs candidats (Avec les probabilités les plus élevées) comme choix à la prochaine entrée du décodeur pour le prochain pas. Voici une définition plus formelle :

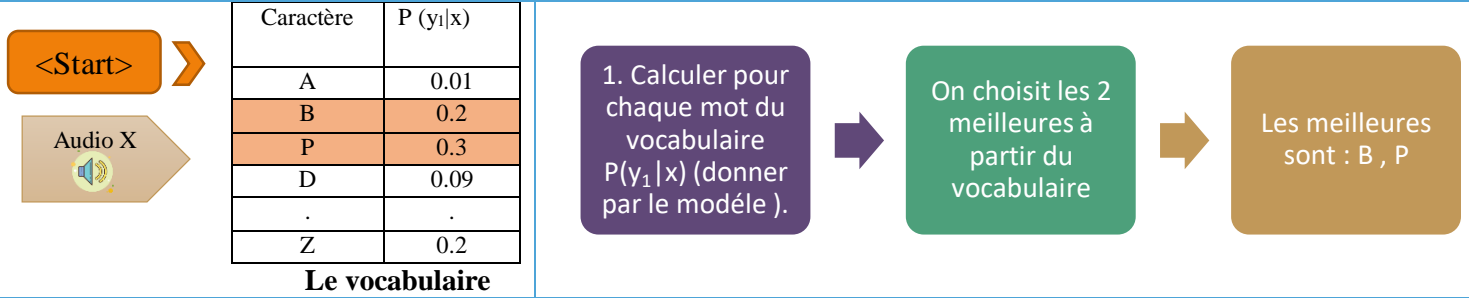
« L'algorithme de recherche de faisceaux local garde la trace de  $k$  états plutôt que d'un seul. Il commence par  $k$  états générés aléatoirement. A chaque étape, tous les successeurs de tous les  $k$  états sont générés. Si l'un d'entre eux est un objectif, l'algorithme s'arrête. Sinon, il sélectionne les  $k$  meilleurs successeurs dans la liste complète et se répète » [6].

Prenons un exemple pour expliquer comment marche cette recherche dans la reconnaissance vocale, on se base sur le cours [5], On déroule la recherche en faisceau :

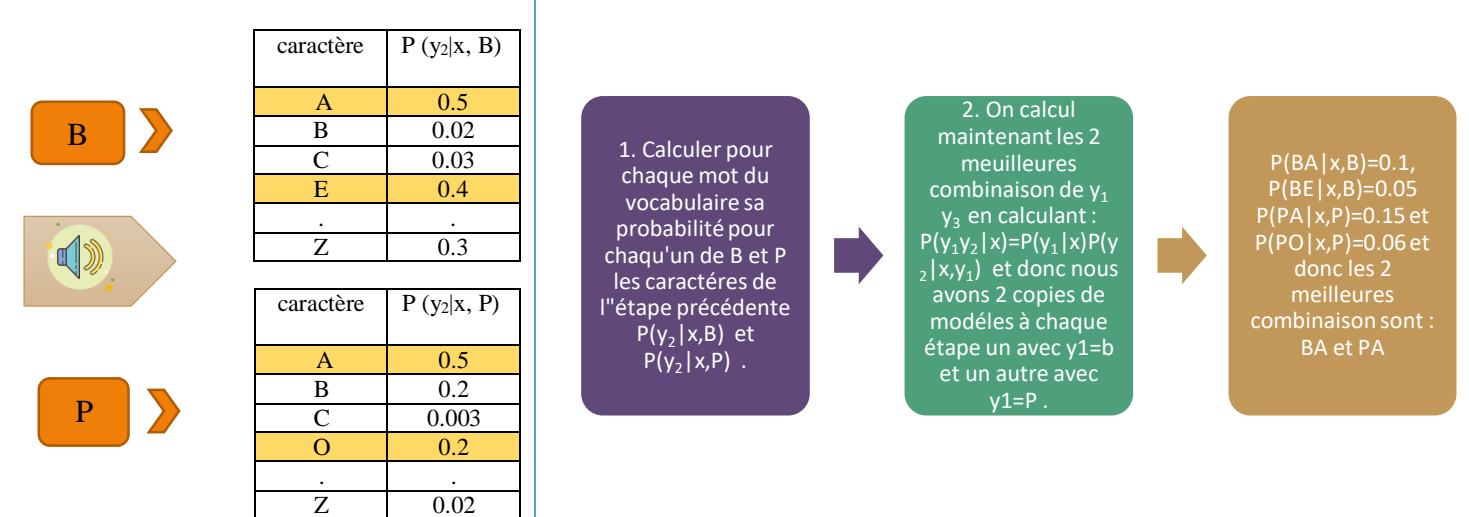


## Déroulement Beam search , avec B=2

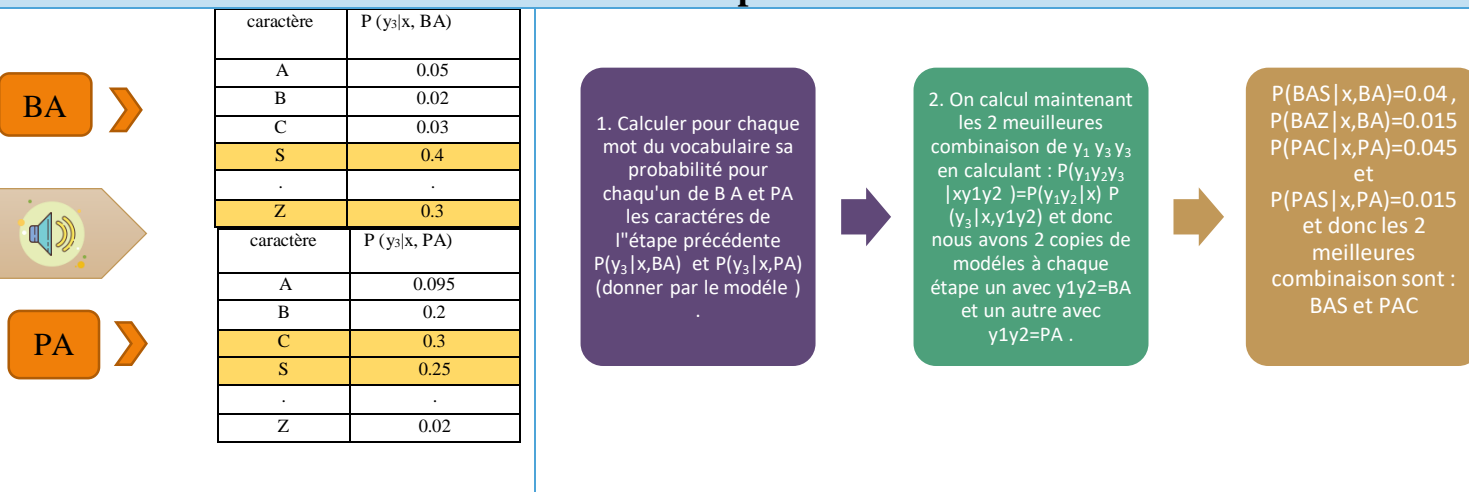
### Etape 1



### Etape 2



### Etape 3



Et ceci se répète jusqu'à rencontre du token de fin de phrase ou atteindre le nombre max de caractères de la phrase .

On peut remarquer que si B=1 la recherche devient une recherche gourmande .

- Il est donc question de calculer la formule suivante, soit  $T_y$  la longueur de la plus grande phrase : 
$$\text{Argmax} \prod_{t=1}^{T_y} P(y_t|x, y_1, y_2 \dots y_{t-1}) \dots \dots (1)$$
- Mais il y'a problème lors du calcul de cette formule car si la longueur de la phrase est grande le produit de nombres  $< 1$  devient un réel très petit et tendra vers 0. La solution est de faire rentrer le log cette méthode s'appelle log-vraisemblance et est calculé ainsi [5] : 
$$\text{Argmax} \sum_{t=1}^{T_y} \log P(y_t|x, y_1, y_2 \dots y_{t-1}) \dots \dots (2)$$
- Mais avec cette formule le modèle va favoriser les petites phrases de grandes malgré le fait que les grandes sont les bonnes sorties, la solution est de faire une normalisation de la longueur, Enfin la formule deviendra ainsi, avec  $0 < \alpha < 1$  : 
$$\text{Argmax} \frac{1}{T_y^\alpha} \sum_{t=1}^{T_y} \log P(y_t|x, y_1, y_2 \dots y_{t-1}) \dots (3)$$

Nous présentons maintenant l'algorithme du beamSearch [7] :

#### Algorithme Beam\_search

Input : audio , modèle , B ,  $\alpha$  , sequence\_max\_len



Output : prédiction = Tableau [1 .. maxCara] de caractères

Debut

```
K_beams = [(0,[reel(token_start_sentence)]*(sequence_max_len) //tableau de structure (proba,phrase pr prédite
Pour index_cara ->1 à sequence_max_len :
    All_k_beams=[]
    Pour proba, sent_predict dans k_beam:
        Predictions=model.predict(input_audio,sent_predict) //maintenant on prends les top k candidats
        Possible_k=MeilleureCandidats(prediction[index_cara].index_sort(),B) //ajouter les k possibles candidats à
        k_beams
        Pour prochain_cara dans possible_k :
            Next_prediction=sent_predict[0..index_cara+1]+[prochain_cara]+[PAD_token]*(sequence_max_len-1)
            All_k_beams.insert((Log_vraisemblance(prediction,sent_predict,  $\alpha$ ), Next_prediction)
    K_beam=Meilleures_combinaisons (all_k_beam,B)
Retourner k_beam
```

Fin

## Chapitre 3. Conception, Expérimentations et implémentation du système de reconnaissance vocale

Nous nous sommes référencés pour construire le système de reconnaissance vocale sur le modèle LAS (voir chapitre 2.2.2). Dans ce chapitre nous allons expliquer les étapes suivies et expérimentations pour la conception du modèle, en justifiant pour chaque étape nos actions et choix :



### 3.1. Environnement d'implémentation [BOUALIT]

#### 3.1.1. Présentation de Keras



Keras<sup>1</sup> est une librairie en apprentissage profond codée via langage de programmation Python. Elle représente une abstraction de haut niveau de la plateforme d'apprentissage automatique Tensorflow qui permet de réaliser des expérimentations rapides.

#### 3.1.2. Typologie des modèles Keras

La librairie Keras est basée sur des structures de données de base que sont les **couches** qui constituent les **modèles** sachant que le type de modèle varie selon la manière dont ces couches sont empilées [Hanifi, 2019].

Le modèle le plus simple qui existe se trouve être le modèle séquentiel dans lequel les couches sont empilées d'une manière linéaire où chaque couche possède exactement un **tenseur d'entrée** et un **tenseur de sortie** [21]. Même si le modèle séquentiel est le plus simple à comprendre et à mettre en place, il est néanmoins **limité** en termes de possibilités puisqu'il ne permet pas de créer un modèle avec des **couches partagées** possédant **plusieurs entrées** et **plusieurs sorties** [21]. Dans le cadre de notre travail, étant donné les limitations précédentes nous avons opté pour le modèle dit **fonctionnel** qui présente plus de flexibilité par rapport au modèle séquentiel permettant ainsi de créer des modèles ayant une topologie non linéaire sous forme de graphe acyclique dirigé avec des couches partagées possédant une ou plusieurs entrées et une ou plusieurs sorties.

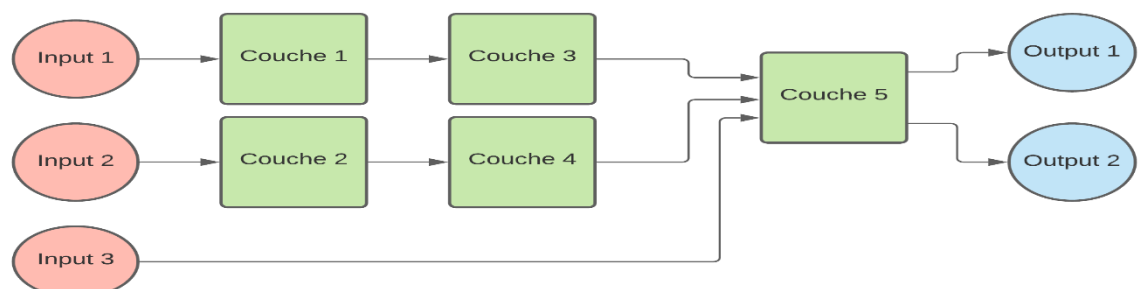


Figure 7 :  
Modèle  
fonctionnel

<sup>1</sup> Keras est disponible sur : <https://keras.io/>

### 3.2. Dataset et prétraitement [ BELKADI]

Le dataset utilisé est « [Arabic Speech Corpus](#) », Le corpus a été enregistré en arabe du sud du Levant (accent damascien) dans un studio professionnel [20], ce dataset contient :

- 1813 audio .wav et fichiers .lab contenant les labels l'orthographe est au [format Buckwalter](#) ce qui est plus convivial lorsqu'il existe un logiciel qui ne lit pas l'écriture arabe. Il peut être facilement reconverti en arabe.
- 100 audio .wav et .lab contenant les labels en texte pour le test.

✓ Aperçu sur le dataset :

ARA NORM 0002.wav	Durée : 00:00:15 Taille : 1,41 Mo
ARA NORM 0003.wav	Durée : 00:00:06 Taille : 423 Ko
ARA NORM 0004.wav	Durée : 00:00:13 Taille : 1,28 Mo
ARA NORM 0005.wav	Durée : 00:00:03 Taille : 344 Ko
ARA NORM 0006.wav	Durée : 00:00:10 Taille : 960 Ko
ARA NORM 0007.wav	Durée : 00:00:18 Taille : 1,66 Mo
ARA NORM 0008.wav	Durée : 00:00:18 Taille : 1,66 Mo
ARA NORM 0009.wav	Durée : 00:00:16 Taille : 1,52 Mo
ARA NORM 0010.wav	Durée : 00:00:04 Taille : 447 Ko
ARA NORM 0011.wav	Durée : 00:00:04 Taille : 409 Ko
ARA NORM 0012.wav	Durée : 00:00:03 Taille : 350 Ko
ARA NORM 0013.wav	Durée : 00:00:07 Taille : 721 Ko

```

"ARA NORM 0002.wav" "waraAaHa AltAqoriyru AlAa*iy >aEadAhu maEohadu >abohaA'i haDabapi AltAibiti fiy Alo>akaAdiymiyAapi
"ARA NORM 0003.wav" "minAa qado yu&adAiy <ilaY taraAjuEi masaAhaAti Alo>anohaAri AljAaliydyiAapi waAnoti&Ari AltAaSaHUri
"ARA NORM 0004.wav" "waAkara AltAqoriyru >ana taraAjuEa masaAhaPi AloJaliydi yumokinu >ayuxilA bimudAdAaLaTi <imodaAda
"ARA NORM 0005.wav" "bayonahaA nahoraA yalun wayaAnogotsiy - fiy AlS-iyno"
"ARA NORM 0006.wav" "wafiy Al$>awoTi AlAaAniy AsotaEaAda baAriysu saAno jiyoramAnu musotawaAhu waHaqAqa AltAaEaDula Eano
"ARA NORM 0007.wav" "yatamaAalu Alo<ibodaAEu AlofanAiyu waAloHaDaAriyu AlAa*iy yako&ifu Eanohu AloMaEoriDu fiy tuHaFk ka
"ARA NORM 0008.wav" "<iDaAfap <ilaY EaroDi >anowaAEK EadiydapK mina Alo>aAa'i Alomuzaxorafi waAlSAnaAdiyqi waAlo>awaAniy
"ARA NORM 0009.wav" "yaxotalifu AlohaAtifu AlzAkiyu Eani AlohaAtifi AltAqoliydyiA fiy >anAhu yugadAimu EadadAF mino wa
"ARA NORM 0010.wav" "wakaAminaAti AltAaSowiYri Alomudomajapi *aAti AltAiqAapi AloEaAliyapi"
"ARA NORM 0011.wav" "waAlomuEaAlijaAti AlAaunaAjiyapi waAlrubaEiAiyapi AlnAwal"
"ARA NORM 0012.wav" "waHaAlapi Alo<isotiqaTaAbi AlS-iyaAkiyiA AloHaAd-i"
"ARA NORM 0013.wav" "liyatamakana AloqaAriju mino mutaAbaEapi taTawuri AloqisAapi Aloxabariyapi bi&akolk salisK wafaEwaA

```

arabic-speech-corpus\arabic-speech-corpus\wav

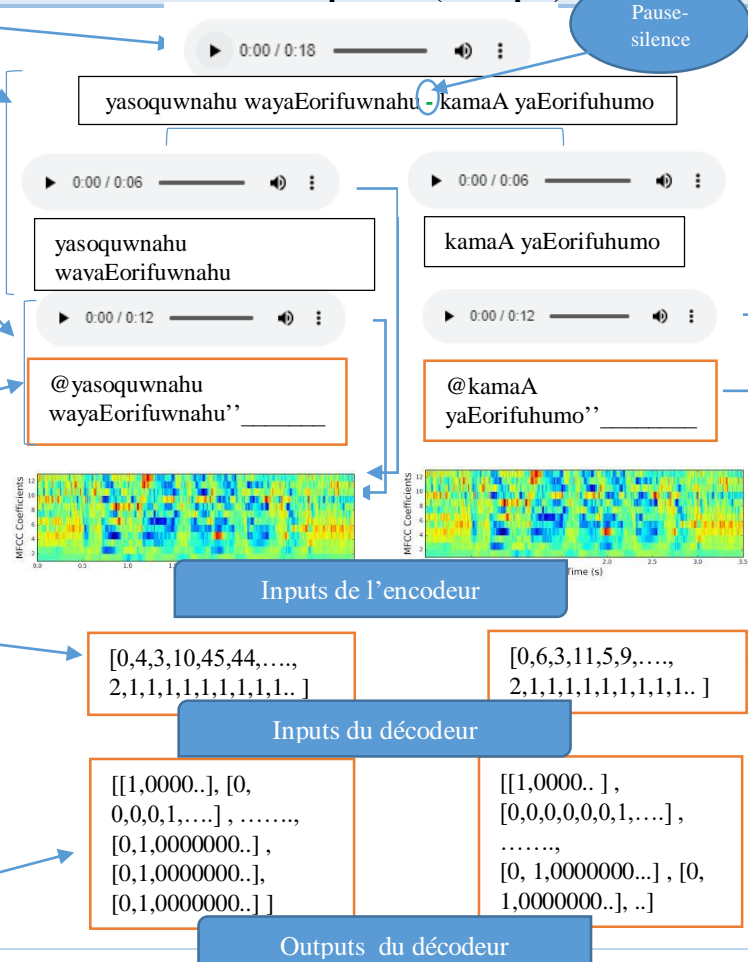
arabic-speech-corpus\arabic-speech-corpus\orthographic-transcript.txt

Ce dataset nécessite un prétraitement, nous présentons les prétraitements faits sur les entrées et les labels :

#### Etapes de prétraitement

1. Lire tous les audios et leurs transcriptions.
2. Découper les audios et les textes par pauses (silence) afin d'agrandir le dataset horizontalement (3068 instances résultantes à partir de 1913) et réduire la durée max des audios vu qu'elle était égale à 32s ce qui correspond à 463 caractères.
3. Calculer la durée max des audios (égale à 12s) et la longueur max des textes (150 caractères)
4. Faire le remplissage (padding) des audios dont leurs période est inférieur à la durée max avec du silence et aussi des textes dont la longueur est inférieur à 150 avec un caractère spéciale « \_ ».
5. Ajouter les jetons de début de phrase « @ » et de fin de phrase « ' » pour chaque phrase.
6. Générer les MFCCs (voir définition chapitre 2) et les spectrogrammes de mel nous feront par la suite une comparaison afin d'en tirer le meilleur format qui nous a donné le meilleur résultat.
7. Les phrases seront découper en caractères chaque caractères sera encoder à un entier unique (selon un dictionnaire qui comporte un index (clé) unique de chaque caractère).
8. Les inputs de l'encodeur seront soit les MFCCs ou des spectrogrammes de mel.
9. Les inputs du décodeur sont les phrases en enlevant le dernier caractère.
10. Les outputs du décodeur sont les phrases décalées d'un caractère qui est le caractère de début encodé en « one hot » donc nous créant ces vecteurs en se basant sur le dictionnaire des caractères possibles.

#### Schéma explicatif (exemple)



### 3.3. Architecture et Apprentissage

#### 3.3.1. Choix de la fonction d'erreur

Pour choisir la fonction de perte, il faut classer le problème à résoudre et aussi voir le format des sorties. En outre, la configuration de la couche de sortie doit également être appropriée pour la fonction de perte choisie.

Speech to texte peut être vu comme un problème de classification à multi classe en considérant les classes : les différents caractères du vocabulaire. En plus, l'encodage de l'output est « one-hot » [26]. **L'entropie croisée** est la fonction de perte par défaut à utiliser pour les problèmes de classification multi-classes pour le format one-hot des sorties. Mathématiquement, il s'agit de la fonction de perte préférée dans le cadre d'inférence du maximum de vraisemblance [26]. Nommé sous l'environnement keras : [categorical\\_crossentropy](#) , en ce qui concerne la couche de sortie sera une couche dense avec comme nombre de neurone la taille de notre vocabulaire (50 caractères ) et une fonction d'activation soft max.

#### 3.3.2. Paramétrage de l'architecture et de l'apprentissage [ BELKADI ET BOUALIT]

L'architecture a été bien expliqué dans le chapitre 2, il est temps maintenant de trouver le meilleure paramétrage, ce schéma montre l'architecture ainsi que les noms des paramètres à fixer, nous expliquons aussi la procédure choisit pour le paramétrage et les différentes comparaisons effectuées :

1. Paramétriser la structure du RDN	
<ul style="list-style-type: none"> <li>✓ MF/Mel : MFCCs ou spectrogramme de mel .</li> <li>✓ N : nombre de couches CNN</li> <li>✓ M : nombre de couches BLSTM</li> <li>✓ m : liste qui montre le nombre de neurones de chaque couche BLSTM .</li> </ul>	<ul style="list-style-type: none"> <li>✓ K : nombre de couches LSTM du décodeur .</li> <li>✓ k : liste qui montre le nombre de neurones de chaque couche LSTM .</li> <li>✓ dim : Dimension de l'enrobage (embedding) dense.</li> <li>✓ C : nombre de couches Dense pour la distribution probabilistes de l'output</li> <li>✓ Params charDistr : le genre de couche, nombre de neurones et la fonction d'activation.</li> </ul>
Listner	Speller
<b>2. Choisir le format d'input : comme déjà dit nous allons comparer les résultats en utilisant en entrées : les MFCCs et les mels spectrogramme.</b>	
<b>3. Paramétriser la couche d'attention : On a vu déjà les 2 catégories d'attention en chapitre 1.2.3. et leurs types, nous allons faire une comparaison entre celles-ci.</b>	
	<ul style="list-style-type: none"> <li>✚ L'attention de bahdanau développé par keras [10] est celle présenté en chapitre 1.2.3, avec ces 2 types globale et locale.</li> <li>✚ L'attention de Luong développé par keras avec ces deux types globale et locale.</li> <li>✚ L'attention de Luong représente notre implémentation en s'inspirant de [29] .</li> </ul>
4. Paramétriser les paramètres d'apprentissage .	
<ul style="list-style-type: none"> <li>✚ <b>Epochs :</b> Nombre d'époques pour entraîner le modèle. Une époque est une itération sur l'ensemble des données inputs et outputs fournies [22].</li> <li>✚ <b>K-fold validation :</b> nous avons utilisé K-fold cross validation pour tirer plusieurs ensembles de validation d'une même base de données et ainsi d'obtenir une estimation plus robuste , nous paramétrons le nombre de blocs .</li> </ul>	<ul style="list-style-type: none"> <li>✚ <b>Optimiseur (fonction d'apprentissage) :</b> on veillera à comparer les résultats de ses optimiseurs : <ul style="list-style-type: none"> <li>▪ <b>SGD :</b> la base des autres optimiseurs (gradient descente avec momentum) [23].</li> <li>▪ <b>Adam :</b> estimation du moment adaptatif, Cet optimiseur est devenu assez répandu et est pratiquement accepté.</li> <li>▪ <b>Nadam :</b> version améliorée de adam</li> <li>▪ <b>Adamax :</b> parfois supérieur à Adam, en particulier dans les modèles avec embeddings (c'est notre cas) [23].</li> <li>▪ <b>Rmsprop :</b> Cet optimiseur est généralement un bon choix pour les réseaux de neurones récurrents [23].</li> <li>▪ <b>Adadelta :</b> recommandé pour des données « speech ».</li> </ul> </li> </ul>
Les métriques [24] et visualisation	
<ul style="list-style-type: none"> <li>✚ <b>CategoricalAccuracy (acc et val_acc) :</b> Calcule la fréquence à laquelle les prédictions correspondent aux étiquettes one-hot.</li> <li>✚ <b>lienArchi :</b> un lien vers le résumé de l'architecture .</li> </ul>	<ul style="list-style-type: none"> <li>✚ <b>CategoricalCrossentropyLoss (loss et val_loss) :</b> Représente l'erreur entropie croisée.</li> <li>✚ <b>LienGra :</b> un lien vers les graphes des métriques.</li> </ul>

	paramètres de l'architecture										paramètres d'apprentissage				Métriques						
Num confie	MF/Mel	N	M	m	K	k	dim	C	TypeAtten tion	Params hardistr	Epochs	K-fold (np-split)	Optimizer	paramOpt	Acc	Error	Val_acc	Val_error	lienArchi	LienGra	
1	MF	2	1	8	1	1 6	3 2	2	Bahdan aus keras global	Dense-32- relu Dense-16- relu	60	4	adam	learning_rate=0 .001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8714	0.4503	0.8714	0.4640	<a href="#">Modele_Spe echReco_15 99655183</a>	Accuracy : <a href="#">Modele_SpeechReco_1599655183</a> Loss : <a href="#">Modele_SpeechReco_1599655183</a>	
2	Mel	2	1	8	1	1 6	3 2	2	Bah_ke ras , global	Dense-32- relu Dense-16- relu	60	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8539	0.5137	0.8597	0.5202	<a href="#">Modele_Spe echReco_15 99561100</a>	Accuracy : <a href="#">Modele_SpeechReco_1599561100</a> Loss : <a href="#">Modele_SpeechReco_1599561100</a>	
3	MF	2	1	16	1	3 2	3 2	2	Bah_ke ras , global	Dense-32- relu Dense-16- relu	60	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8725	0.4071	0.8791	0.4254	<a href="#">Modele_Spe echReco_16 00028067</a>	Accuracy : <a href="#">Modele_SpeechReco_1600028067</a> Loss : <a href="#">Modele_SpeechReco_1600028067</a>	
4	Mel	2	1	16	1	3 2	3 2	2	Bah_ke ras , global	Dense-32- relu Dense-16- relu	60	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8631	0.4827	0.8561	0.5466	<a href="#">Modele_Spe echReco_16 00080901.</a>	Accuracy : <a href="#">Modele_SpeechReco_1600080901</a> Loss : <a href="#">Modele_SpeechReco_1600080901</a>	
5	MF	2	1	32	1	6 4	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	60	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8916	0.3542	0.8959	0.3742	<a href="#">Modele_Spe echReco_16 00098240</a>	Accuracy : <a href="#">Modele_SpeechReco_1600098240</a> Loss : <a href="#">Modele_SpeechReco_1600098240</a>	
6	Mel	2	1	32	1	6 4	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh		4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8824	0.4079	0.8822	0.4157	<a href="#">Modele_Spe echReco_16 00098794</a>	Accuracy : <a href="#">Modele_SpeechReco_1600098794</a> Loss : <a href="#">Modele_SpeechReco_1600098794</a>	
7	MF	2	1	32	1	6 4	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	100	4	nada m	Default	0.8972	0.3514	0.8972	0.3531	<a href="#">Modele_Spe echReco_16 00258118</a>	Accuracy : <a href="#">Modele_SpeechReco_1600258118</a> Loss : <a href="#">Modele_SpeechReco_1600258118</a>	

8	MF	2	1	32	1	6 4	5 0	2	Bah_keras , global	Dense-32- tanh Dense-32- tanh	127	4	adam ax	Default	0.8797	0.4194	0.8844	0.4021	<a href="#">Modele Spe echReco_16 00291335</a>	Accuracy : <a href="#">Modele SpeechReco_1600291335</a> Loss : <a href="#">Modele SpeechReco_1600291335</a>
9	MF	2	1	32	1	6 4	5 0	2	Bah_keras , global	Dense-32- tanh Dense-32- tanh	400	4	rmspr op	Default	0.8916	0.3729	0.8927	0.3681	<a href="#">Modele Spe echReco_16 00355173</a>	Accuracy : <a href="#">Modele SpeechReco_1600355173</a> Loss : <a href="#">Modele SpeechReco_1600355173</a>
10	MF	2	1	32	1	6 4	5 0	2	Bah_keras , global	Dense-32- tanh Dense-32- tanh	400	4	Adad elta	Default	0.7627	0.9384	0.7435	0.9057	<a href="#">Modele Spe echReco_16 00203183</a>	Accuracy : <a href="#">Modele SpeechReco_1600203183</a> Loss : <a href="#">Modele SpeechReco_1600203183</a>
11	MF	2	1	32	1	6 4	5 0	2	Bah_keras , global	Dense-32- tanh Dense-32- tanh	400	4	Adad elta	Lr=1.0, rho=0.95, epsilon=1e-6	0.8526	0.5248	0.8601	0.4991	<a href="#">Modele Spe echReco_16 00266814</a>	Accuracy : <a href="#">Modele SpeechReco_1600248442</a> Loss : <a href="#">Modele SpeechReco_1600248442</a>
12	MF	2	1	32	1	6 4	5 0	2	Bah_keras , global	Dense-32- tanh Dense-32- tanh	100	4	SGD	Default	0.8057	0.7083	0.8145	0.6805	<a href="#">Modele Spe echReco_16 00270138</a>	Accuracy : <a href="#">Modele SpeechReco_1600270138</a> Loss : <a href="#">Modele SpeechReco_1600270138</a>
13	MF	2	1	64	1	1 2 8	5 0	2	Bah_keras , global	Dense-32- tanh Dense-32- tanh	294	4	nada m	Default	0.9003	0.3461	0.8958	0.3610	<a href="#">Modele Spe echReco_16 00297447</a>	Accuracy : <a href="#">Modele SpeechReco_1600297447</a> Loss : <a href="#">Modele SpeechReco_1600297447</a>
14	MF	2	1	64	1	1 2 8	5 0	2	Bah_keras , local	Dense-32- tanh Dense-32- tanh	400	4	nada m	Default	0.8999	0.3442	0.8969	0.3587	<a href="#">Modele Spe echReco_16 00302374</a>	Accuracy : <a href="#">Modele SpeechReco_1600302374</a> Loss : <a href="#">Modele SpeechReco_1600302374</a>
15	MF	2	1	64	1	1 2 8	5 0	2	Luong_keras , local	Dense-32- tanh Dense-32- tanh	100	4	nada m	Default	0.8992	0.3481	0.8955	0.3615	<a href="#">Modele Spe echReco_16 00338247</a>	Accuracy : <a href="#">Modele SpeechReco_1600338247</a> Loss : <a href="#">Modele SpeechReco_1600338247</a>
16( gpu )	MF	2	1	32	1	6 4	5 0	2	Luong_keras global	Dense-32- tanh Dense-32- tanh	400	4	Nada m	Default	0.89	0.3793	0.8884	0.39	<a href="#">Modele Spe echReco_16 00530096</a>	Accuracy : <a href="#">Modele SpeechReco_1600530096</a> Loss: <a href="#">Modele SpeechReco_1600530096</a>



17( gpu )	MF	2	1	32	1	6 4	5 0	2	Luong global	Dense-32- tanh Dense-32- tanh	400	4	Nada m	Default	0.8982	0.3812	0.8888	0.3929	<a href="#">Modele Spe echReco_16 00529643</a>	Accuracy : <a href="#">Modele SpeechReco_160 0529643</a> Loss : <a href="#">Modele SpeechReco_1600529643</a>
18	MF	2	1	32	1	6 4	5 0	2	Luong_ keras global	Dense-32- tanh Dense-32- tanh	127	4	Adam ax	Default	0.8775 -	0.4229	0.8824	0.40599,	<a href="#">Modele Spe echReco_16 00361983</a>	Accuracy : <a href="#">Modele SpeechReco_160 0361983</a> Loss : <a href="#">Modele SpeechReco_1600361983</a>
19	MF	2	1	32	1	6 4	5 0	2	Luong global	Dense-32- tanh Dense-32- tanh	400	4	Adam ax	Default	0.8889	0.3831	0.8889	0.3949	<a href="#">Modele Spe echReco_16 00365698</a>	Accuracy : <a href="#">Modele SpeechReco_1600365698</a> Loss : <a href="#">Modele SpeechReco_1600365698</a>
20	MF	3	1	12 8	1	2 5 6	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	100	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8972	0.3549	0.8919	0.3730	<a href="#">Modele Spe echReco_16 00430113</a>	Accuracy : <a href="#">Modele SpeechReco_160 0430113</a> Loss : <a href="#">Modele SpeechReco_1600430113</a>
21	MF	3	1	25 6	1	5 1 2	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.9026	0.3422	0.8909	0.3789	<a href="#">Modele Spe echReco_16 00442835</a>	Accuracy : <a href="#">Modele SpeechReco_160 0442835</a> Loss : <a href="#">Modele SpeechReco_1600466146</a>
22	MF	3	1	51 2	1	1 0 2 4	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8777	0.4371	0.8759	0.4434	<a href="#">Modele Spe echReco_16 00466146</a>	Accuracy : <a href="#">Modele SpeechReco_160 0466146</a> Loss: <a href="#">Modele SpeechReco_1600466 146</a>
23	MF	3	2	8 8	2	1 6 1 6	5 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8690	0.4485	0.8708	0.4472	<a href="#">Modele Spe echReco_16 00446884</a>	Accuracy : <a href="#">Modele SpeechReco_1600446884</a> Loss : <a href="#">Modele SpeechReco_1600446884</a>
24	MF	3	2	8 8	1	1 6	5 0	2	Luong	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8717	0.4401	0.8774	0.4178	<a href="#">Modele Spe echReco_16 00459560</a>	Accuracy : <a href="#">Modele SpeechReco_1600459560</a> Loss : <a href="#">Modele SpeechReco_1600459560</a>
25	MF	3	2	16 16	1	3 2	1 0 0	2	Bah_ke ras , global	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8781	0.4271	0.8828	0.4106	<a href="#">Modele Spe echReco_16 00448832</a>	Accuracy : <a href="#">Modele SpeechReco_1600448832</a> Loss : <a href="#">Modele SpeechReco_1600448832</a>

26	MF	3	2	32 32	1	6 4	1 0 0	2	Bah_ker as , global	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8910	0.3753	0.8921	0.3750	<a href="#">Modele Spe echReco_16 00461118</a>	Accuracy : <a href="#">Modele SpeechReco_1600461118</a> Loss : <a href="#">Modele SpeechReco_1600461118</a>
27	MF	3	2	64 64	1	1 2 8	1 0 0	2	Luong	Dense-32- tanh Dense-32- tanh	300	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8979	0.3526	0.8945	0.3675	<a href="#">Modele Spe echReco_16 00505280</a>	Accuracy : <a href="#">Modele SpeechReco_1600505280</a> Loss: <a href="#">Modele SpeechReco_1600505280</a>
28	MF	3	2	12 8 12 8	1	2 5 6	1 0 0	2	Bah_ker as , global	Dense-32- tanh Dense-32- tanh	200	4	adam	learning_rate= 0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.8913	0.3891	0.8909	0.3909	<a href="#">Modele Spe echReco_16 00510387</a>	Accuracy : <a href="#">Modele SpeechReco_1600510387</a> Loss: <a href="#">Modele SpeechReco_1600510387</a>
29	MF	2	1	25 6	2	5 1 2, 5 1 2	1 0 0	3	Bah_ker as , locale	Dense-64- tanh Dense-32- tanh Dense-32- tanh	172	4	Nada m	Default	0.9133	0.3175	0.8806	0.4292	<a href="#">Modele Spe echReco_16 00533306</a>	Accuracy : <a href="#">Modele SpeechReco_1600533306</a> Loss : <a href="#">Modele SpeechReco_1600533306</a>
30( gpu )	MF	2	1	25 6	2	5 1 2, 5 1 2	1 0 0	3	Bah_ker as , locale	Dense-64- tanh Dense-32- tanh Dense-32- tanh	100 0	3	adam	Lr = 0.005 beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.9037	0.3379	0.8926	0.3791	<a href="#">Modele Spe echReco_16 00538173</a>	Accuracy <a href="#">Modele SpeechReco_1600538173</a> Loss: <a href="#">Modele SpeechReco_1600538173</a>
31( gpu )	MF	3	1	12 8	2	2 5 6	1 0 0	3	Bah_ker as , locale	Dense- 128-tanh Dense-64- tanh Dense-32- tanh	100 0	4	Nada m	Default	0.8960	0.3660	0.8890	0.3941	<a href="#">Modele Spe echReco_16 00541474</a>	Accuracy : <a href="#">Modele SpeechReco_1600541474</a> Loss: <a href="#">Modele SpeechReco_1600541474</a>
32	MF	3	1	12 8, 12 8	2	2 5 6, 2 5 6	1 0 0	3	Bah_ker as , locale	Dense- 128-tanh Dense-64- tanh Dense-32- tanh	100 0	4	Nada m	Default	0.9069	0.3307	0.8860	0.4023	<a href="#">Modele Spe echReco_16 00544867</a>	Accuracy: <a href="#">Modele SpeechReco_1600544867</a> Loss: <a href="#">Modele SpeechReco_1600544867</a>
33	MF	2	1	25 6	2	5 1 2, 5 1 2	5 0	2	Bah_ker as , global	Dense-32- tanh Dense-32- tanh	300	4	nada m	Default	0.90 87	0. 32 10	0.8 908	0.38 64	<a href="#">Modele Spe echReco_16 00530676</a>	Accuracy : <a href="#">Modele SpeechReco_1600530676</a> Loss: <a href="#">Modele SpeechReco_1600530676</a>



34	MF	3	1	25 6	1	5 1 2	5 0	2	Bah_ke ras , global	Dense-64- tanh Dense-32- tanh	300	4	nada m	Default	0.90 96	0. 32 39	0.8 898	0.39 34	<a href="#">Modele Spe echReco_16 00533490</a>	Accuracy: <a href="#">Modele SpeechReco_1600533490</a> Loss: <a href="#">Modele SpeechReco_1600533490</a>
35	MF	3	1	25 6	1	5 1 2	5 0	2	Bah_ke ras , global	Dense-64- relu Dense-32- tanh	300	4	nada m	Default	0.91 00	0. 32 74	0.8 833	0.41 82	<a href="#">Modele Spe echReco_16 00533605</a>	Accuracy : <a href="#">Modele SpeechReco_1600533605</a> Loss : <a href="#">Modele SpeechReco_1600533605</a>
36	MF	2	1	12 8	2	2 5 6, 2 5 6	1 0 0	3	Bah_ke ras , locale	Dense-64- tanh Dense-32- tanh Dense-32- tanh	100 0	3	adam	Lr = 0.005 beta_1=0.9, beta_2=0.999, epsilon=1e-07	0.9047	0. 33 96	0.8 931	0.37 51	<a href="#">Modele Spe echReco_16 00535561</a>	Accuracy: <a href="#">Modele SpeechReco_1600535561</a> Loss: <a href="#">Modele SpeechReco_1600535561</a>
37	MF	3	2	32 ,3 2	2	6 4, 6 4	2 0 0	3	Bah_ke ras , locale	Dense-64- tanh Dense-64- tanh	100 0	4	Nada m	Default	0.9001	0. 34 32	0.8 879	0.38 80	<a href="#">Modele Spe echReco_16 00548343</a>	Accuracy: <a href="#">Modele SpeechReco_1600548343</a> Loss: <a href="#">Modele SpeechReco_1600548343</a>
38	MF	3	1	12 8	1	2 5 6	1 0 0	2	Bah_ke ras , globale	Dense- 128-tanh Dense-64- tanh	20	4	nada m	Default	0.92	0. 22	0.9 021 098 017 692 566	0.33 88	<a href="#">Modele Spe echReco_15 99299782</a>	Accuracy : <a href="#">Modele SpeechReco_1599299782</a> Loss: <a href="#">Modele SpeechReco_1599299782</a>

Les 5 meilleures configurations sont coloriées .

### 3.3.3. Comparaisons et discussions [ BELKADI ]

#### 1. Comparer les résultats obtenus pour chaque format des données MFCCs et Mel spectrogramme :

Architecture	Architecture 1		Architecture 2		Architecture 3	
Num config	Config1(MFCC)	Config 2(Mel)	Config 3(MFCC)	Config4 (Mel)	Config 5(MFCC)	Config6 (Mel)
Erreur	<b>0.4640</b>	0.5202	<b>0.4254</b>	0.5466	<b>0.3742</b>	0.4157
Accuracy	<b>0.8714</b>	0.8597	<b>0.8791</b>	0.8561	<b>0.8959</b>	0.8822

#### Conclusion

Les architectures avec MFCCs donnent des résultats meilleurs, donc nous avons optés pour le format MFCCs avec 13 coefficients pour le reste des architectures.

#### 2. Comparer les résultats obtenus pour les optimiseurs :

Nous avons pris la config num 5 et changer les optimiseurs pour prendre conclusion des 3 meilleurs optimiseurs en prenant les paramètres par défaut (de learning rate et autres selon l'optimiseur) proposé par keras après par la suite nous paramètrons le learning rate des 3 meilleures optimiseurs :

Num config	Config5 (adam)	Config 7(nadam)	Config 8(adamax)	Config9 (rmsprop)	Config 11(Adadelta)	Config12 (SGD)
Erreur	<b>0.3742</b>	<b>0.3531</b>	0.4021	<b>0.3681</b>	0.4991	0.6805
Accuracy	<b>0.8959</b>	<b>0.8972</b>	0.8844	<b>0.8927</b>	0.8601	0.8145

#### Conclusion

Les 3 meilleurs optimiseurs sont : Adam, nadam et Rmsprop.

#### 3. Comparer le type d'attention :

Type d'attention	Attention globale de Bahdanau de keras	Attention locale de Bahdanau de keras	Attention locale de Luong de keras	Attention globale de Luong de keras	Attention de luong implémenter par nous
Num config	Config13	Config14	Config15	Config16	Config17
Erreur	0.3610	0.3587	0.3615	0.39	0.3929
Accuracy	0.8958	0.8969	0.8955	0.8884	0.8888

#### Conclusion

On remarque que la différence n'est pas très grande mais on peut juger que l'attention locale de bahdanau est la meilleure.

#### 4. Meilleure architecture :

La configuration 38 est le meilleur modèle enregistré, avec comme résultat :

Accuracy	Loss	Val_accuracy	Val_loss
<b>0.92</b>	0.22	0.9021	0.3388

✚ On peut voir que le risque de surapprentissage est minim car la différence entre les résultats dans le train et la validation n'est pas très grande.


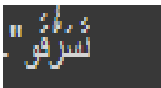
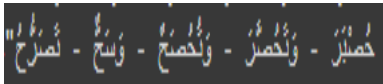
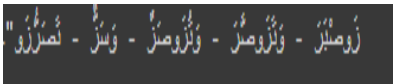
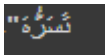
✚ Vous pouvez trouver les modèles enregistrés dans le dossier suivant : [fitted\\_models](#)  
Ou chaque modèle a le nom de format : Nom\_modele\_[Val\_loss,Val\_acc]

### 3.3.4. Inférence


Nous avons utilisé le meilleur modèle obtenu pour tester celui-ci nous avons utilisé la recherche des faisceaux avec la taille de faisceau = 64, nous présentons quelques résultats les prédictions ainsi que la probabilité de vraisemblance calculé nous rappelons la formule :

$$Argmax_{T_y} \frac{1}{T_y} \sum_{t=1}^{T_y} \log P(y_t | x, y_1, y_2 \dots y_{t-1}) \dots (3)$$

## 1. En utilisant les audios du dataset :

Output attendu	Prédiction obtenu avec probabilité de vraisemblance :
	<pre> output beam : -16.032804663322167 (['@', ' ', 'f', 'i', 'y', ' ', 'A', 'l', 'o', 'E', 'a', 'A', 'l', ' ', @ _____ "في العالم الأمريكي" output beam : -15.965092312311754 (['@', ' ', '&gt;', 'a', 'l', 'o', '&gt;', 'a', 'm', 'i', 'r', 'i', 'k', ' ', @ _____ "الأمريكية المأخوذة"</pre>
	<pre> output beam : -5.398794485466624 (['@', ' ', 't', 'a', 's', 'a', 'r', '~', 'u', 'q', 'a', 'w', @ _____ "لسرقو"</pre>
	<pre> output beam : -16.107256015437827 (['@', 'x', 'a', 'y', 'S', 'o', 'b', 'a', 'r', 'a', ' ', ' ', ' ', ' ', ' ', @ _____ "خيمنت - وتخيمنت - وتخيمنت - وتصرتو"</pre>
	<pre> output beam : -16.86764855856231 (['@', 'x', 'a', 'y', 'S', 'o', 'b', 'a', 'r', 'a', ' ', ' ', ' ', ' ', ' ', @ _____ "خيمنت - وتخيمنت - وتخيمنت - وتصرتو"</pre>
	<pre> output beam : -5.878133965878078 (['@', ' ', 't', 'a', 's', 'a', 'r', '~', 'u', 'h', 'a', 'y', @ _____ "تسرقو هي"</pre>

## 2. Exemple en utilisant nos voix :

Output attendu	Prédiction obtenu avec probabilité de vraisemblance :
	<pre> output beam : -58.05919406338944 (['@', 'H', 'a', 'y', 'o', '^', 'u', ' ', 'a', 'n', @ _____ "حي أن العالم العالمية في العالم الأمريكية وال" </pre>

🔊 Il y'a du bruit dans le vocal, ce qui justifie ce résultat.

## Problèmes rencontrés qui peuvent justifier nos résultats :

### Problèmes associés au dataset et au langage

- Dataset insuffisant ( meme pas 10h d'énoncés ) nous avons réussi à l'alargir mais se n'est pas suffisant pour trouver de bon résultats
- On prends l'exemple de LAS : ils ont utilisé un ensemble de données d'environ trois millions d'énoncés de recherche vocale Google (représentant 2000 heures de données). Et environ 10 heures d'énoncés ont été choisis au hasard commeun jeu de validation retenu [18].
- le modèle n'est pas capable de prédire les langues phrases .
- La langue arabe est vaste et sa pronciation est assez difficile .

### Problèmes associés à l'environnement d'execution

- Nous avons essayer d'ajouter un dataset de chiffres arabes mais la RAM n'a pas supporté la taille des données (Avec google colab RAM de 35Mg et avec TPU) , donc impossible de faire l'apprentissage .

### problèmes associés aux problèmes de speech recognition

- Les voix différent (hommes, femmes, enfants) ceci peut etre un grand problème car notre application est faite pour les enfants .
- les Accents
- Vitesse de parole
- bruit et c'est un grand problème on a vu que le modèle a pris le bruit comme étant des paroles ..

## Chapitre 4: Implémentation de l'application [ BOUALIT ]

### 4.1. Principe de l'application

Une application mobile permettant l'apprentissage de l'arabe pour les enfants tout en appliquant et en s'amusant, cette application est basée principalement sur quatre exercices chacun utilise l'écoute ou la lecture comme aspect de pratique.

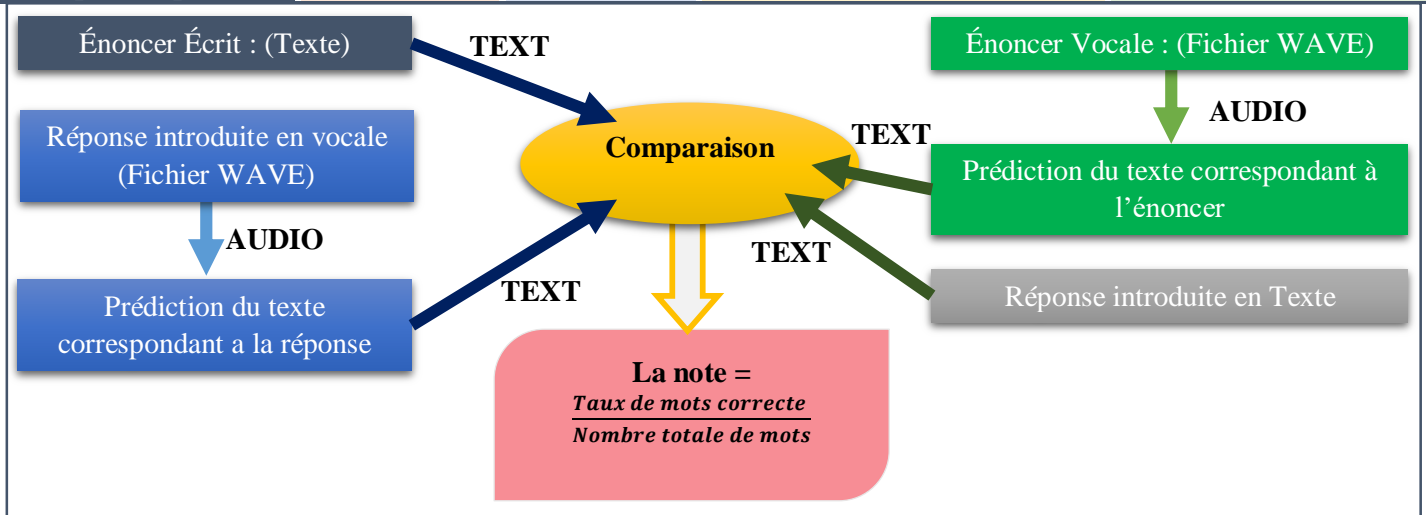
L'application de la reconnaissance vocale en arabe a été utilisée lors de l'évaluation de chaque exercice. Dans ce qui suit nous représentons les quatre exercices de base de notre application ainsi leur méthode d'évaluation :



### Les méthodes d'évaluation

Le principe d'évaluation se base sur la reconnaissance vocale (speech recognition) pour l'arabe, nous avons deux types d'évaluation et réponse : écrite ou vocale, nous détaillons l'évaluation pour chacune.

Numéro de l'exercice	N°1 : Lecture	N°2 : Dictée	N°3 : Que représente l'image	N°4 : Former un mot
Type de question	Écrite	Vocale	Image et écrite	Ensemble de caractères
Type de réponse	Vocale	Écrite	Vocale	Écrite et vocale

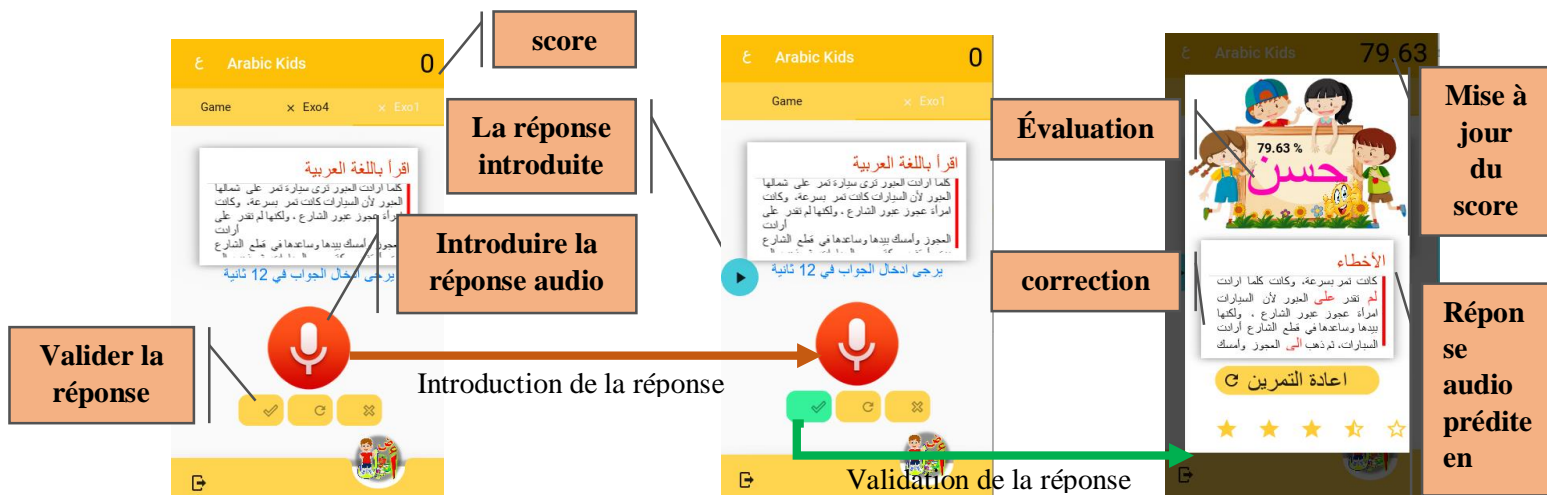


### 4.2. Réalisation

Nous décrivons les fonctionnalités principales de l'application, comme nous avons cité précédemment l'application contient quatre exercices que nous détaillons leur fenêtre comme suite :



### Exemple d'exécution :



## Conclusion [ BOUALIT ]

Ce projet a consisté en l'utilisation d'architectures de réseaux de neurones profonds œuvrant dans le domaine du traitement du langage naturel (NLP) plus précisément dans l'aspect de reconnaissance vocale ( Speech Recognition ) en utilisant les mécanismes d'attention afin de se concentrer sur les informations les plus pertinentes lors de la prédiction pendant l'apprentissage du modèle tout en ignorant les informations superflues qui peuvent biaiser celui-ci.

Ce projet s'est déroulé dans son intégralité à distance, ceci nous a amené à nous organiser en tâche où chacun est impliqué dans les différentes phases du projet : de l'identification des besoins et la compréhension de la problématique jusqu'à la conception et la réalisation du travail.

Notre projet s'articule autour du traitement de la langue arabe à des fins éducatives. Nous avons pu atteindre cet objectif à travers les contributions suivantes :

- Développement d'un système permettant le traitement de séquences vocales et la manipulation de sons en langue arabe dans le cadre d'exercices éducatifs,
- Adaptation de la solution pour le problème de reconnaissance vocale pour la langue arabe en se basant sur le Framework Keras via une utilisation avancée par le modèle fonctionnelle.
- Implémentation d'un réseau de neurones pour le traitement de la langue arabe avec l'intégration de mécanismes d'attention permettant de renforcer d'avantage les prédictions du modèle,
- Implémentation du modèle LAS (Listen, Attend and Spell), un réseau de neurones qui apprend à transcrire les énoncés vocaux en séquences de caractères sans faire d'hypothèses d'indépendance entre ceux-ci,
- Enfin, nous avons réalisé une application permettant la mise en pratique des modèles développés dans le cadre du traitement de la langue arabe.

Ayant conscience que tout projet peut être amélioré, nous estimons que les perspectives suivantes peuvent être mises en œuvre :

- Réaliser un transfert d'apprentissage à partir de réseaux pré-entraînés sur des bases de données en langue arabe,
- Implémenter la solution proposée sur des machines plus performantes que ce que propose l'environnement de développement Google Colab, ce qui permettrait de réaliser des entraînements de plus longue durée sur des bases volumineuses,
- Étendre les modèles proposés à la compréhension d'autres langues pour des exercices éducatifs multilingues,
- Expérimenter l'architecture "Transformer" qui est plus efficace en représentation du langage et en capacités de calcul par rapport à la convolution et aux opérations récursives,
- Améliorer les modèles proposés de manière à prendre en considération la vocalisation (at-tachkil) des séquences vocales,
- Étendre les fonctionnalités du système proposé de manière à intégrer des classes en ligne avec un professeur assistant les enfants dans leur apprentissage et proposer d'autre type d'exercice tel que des discussions automatiques par robot pour les enfants

## Références

- [1]. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to [15]. Dalya Gartzman , Apprendre à connaître le



- Sequence Learning with Neural Networks.
- [2]. Coursera:nlp-sequence-models-why-sequence-models  
<https://www.coursera.org/lecture/nlp-sequence-models/why-sequence-models>
  - [3]. Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306. <https://doi.org/10.1016/j.physd.2019.132306>
  - [4]. Deep-learning-tutorial-Lesson 8 of 10 - Recurrent Neural Network Tutorial : <https://www.simplilearn.com/tutorials/deep-learning-tutorial/>
  - [5]. Andrew ng, Sequence models & Attention mechanism, Beam SEARCH, coursera, [en ligne] <https://www.coursera.org/lecture/nlp-sequence-models/beam-search-4EtHZ> , dernière consultation le 03/09/2020 .
  - [6]. - Pages 125-126, Peter Norvig , Artificial Intelligence: A Modern Approach (3e édition), 2009.
  - [7]. Kartik Chaudhary, Boosting your Sequence Generation Performance with 'Beam-search + Language model' decoding when, why and how of beam-search and LM decoding , medium , [En ligne] , <https://towardsdatascience.com/boosting-your-sequence-generation-performance-with-beam-search-language-model-decoding-74ee64de435a> , dernière consultation le 06/09/2020.
  - [8]. Ajay jangid, Intuitive Understanding of Seq2seq model & Attention Mechanism in Deep Learning , Sep 12, 2019, medium , [en ligne] , <https://medium.com/analytics-vidhya/intuitive-understanding-of-seq2seq-model-attention-mechanism-in-deep-learning-1c1c24aace1e> , dernière consultation le 06/09/2020
  - [9]. Andrew ng, Sequence models & Attention mechanism, Attention model , coursera, [en ligne] <https://www.coursera.org/lecture/nlp-sequence-models/attention-model-lSWVa> , dernière consultation le 03/09/2020 .
  - [10]. Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio , Neural Machine Translation by Jointly Learning to Align and Translate , [Submitted on 1 Sep 2014 (v1), last revised 19 May 2016 (this version, v7)].
  - [11]. Aurélien Géron, Machine Learning avec Scikit-Learn, Traduit de l'anglais par Anne Bohy, Dunod, Paris, 2017
  - [12]. Leland Roberts , Comprendre le spectrogramme Mel , Medium , [en ligne] , <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53> dernière consultation : 07/09/2020
  - [13]. Transformée de Fourier à court terme , Wikipédia , [en ligne] , [https://fr.wikipedia.org/wiki/Transform%C3%A9e\\_de\\_Fourier\\_%C3%A0\\_court\\_terme](https://fr.wikipedia.org/wiki/Transform%C3%A9e_de_Fourier_%C3%A0_court_terme) , consulté le 07/09/2020
  - [14]. Spectrogramme , Wikipédia , [https://fr.wikipedia.org/wiki/Spectrogramme#:~:text=Le %20spectrogramme%20est t%20un%20diagramme,peuvent%20%C3%AAtre%20lin%C3%A9aires%20ou%20logarithmiques.](https://fr.wikipedia.org/wiki/Spectrogramme#:~:text=Le%20spectrogramme%20est%20un%20diagramme,peuvent%20%C3%AAtre%20lin%C3%A9aires%20ou%20logarithmiques.) Consulté le 07/09/2020
  - spectrogramme Mel , medium , towards data science , <https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0>
  - [16]. MFCCs, Wikipédia , <https://fr.wikipedia.org/wiki/Cepstre#:~:text=Les%20coefficients%20r%C3%A9sultants%20de%20cette%20DCT%20sont%20les%20MFCCs.> Consulté le 07/09/2020
  - [17]. Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. Convolutional, Long ShortTerm Memory, Fully Connected Deep Neural Networks. In IEEE International Conference on Acoustics, Speech and Signal Processing, 2015.
  - [18]. Listen, Attend and Spell William Chan Carnegie Mellon University, Navdeep Jaitly, Quoc V. Le, Oriol Vinyals Google Brain {ndjaitly,qvl,vinyals}
  - [19]. M. Schuster, and K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing, 45(1997) 2673–2681.
  - [20]. Nawar Halabi, dataset Arabic Speech Corpus, University of Southampton, <http://en.arabicspeechcorpus.com/>
  - [21]. Hanifi, M. (2019, septembre 22). Sequential API vs Functional API model in Keras. Medium. <https://medium.com/@hanify/sequential-api-vs-functional-api-model-in-keras-266823d7cd5e>
  - [22]. Documentation de keras, Model training APIs, [https://keras.io/api/models/model\\_training\\_apis/#fit-method](https://keras.io/api/models/model_training_apis/#fit-method) dernière consultation 15/09/2020.
  - [23]. Documentation de keras, Optimizers, <https://keras.io/api/optimizers/> , dernière consultation : 15/09/2020.
  - [24]. Documentation de keras , Metrics , <https://keras.io/api/metrics> , dernière consultation : 15/09/2020.
  - [25]. Documentation de keras , Losses , <https://keras.io/api/losses> , dernière consultation : 15/09/2020.
  - [26]. Comment choisir les fonctions de perte lors de l'entraînement des réseaux de neurones d'apprentissage profond, Jason Brownlee , 30 janvier 2019 , [en ligne] , consulté le 15/09/2020.
  - [27]. Minh-Thang Luong, Hieu Pham, Christopher D. Manning , Effective Approaches to Attention-based Neural Machine Translation Submitted on 17 Aug 2015
  - [28]. Gabriel Loya , Attention Mechanism , DEEP LEARNING , 15 SEPTEMBER 2019 , [en ligne] , <https://blog.floydhub.com/attention-mechanism/> , consulté le 16/09/2020
  - [29]. wanasit, Attention-based Sequence-to-Sequence in Keras , 10 September 2017, [en ligne] , <https://wanasit.github.io/attention-based-sequence-to-sequence-in-keras.html> consulté le 15/09/2020