Master Traitement Automatique des Langues - 2020/2021

UE705 Projet et cours de langues

Vers une validation automatique des variantes de prononciation pour la reconnaissance et la synthèse de la parole

Etudiants :

Colleen BEAUMARD
Nicolas PETITJEAN
Tom WYSOCKI

Tuteur :

Denis JOUVET

Réviseur :

Slim OUNI

Juin 2021

Sommaire

1 Introduction

Ce rapport fait suite à celui réalisé au premier semestre sur la validation automatique des variantes de prononciations pour la reconnaissance et la synthèse de la parole. Nous avions dans cette première étude bibliographique présenté plusieurs méthodes existantes pour la conversion graphèmes[1] vers phonèmes[2]. L'objectif de ce semestre est de trouver par l'expérimentation une approche performante pour la prédiction des prononciations de mots d'un lexique donné. Pour cela, nous avons étudié trois approches différentes appliquées sur deux lexiques fournis par notre tuteur. Une première approche repose sur des règles d'alignement entre lettres et phonèmes pour le français. Les deux autres approches reposent sur l'apprentissage machine: l'une utilise une méthode basée sur les réseaux de neurones en séquences-à-séquences et l'autre est une approche statistique exploitant les séquences jointes. Pour utiliser ces modèles, nous nous sommes servis des logiciels OpenNMT pour l'approche par réseaux de neurones, et de Sequitur pour l'approche statistique. Le fait d'utiliser 2 modèles différents nous permet de faire une comparaison sur l'efficacité de chacun.

Le premier chapitre est consacré à la présentation du sujet, ainsi qu'a un rappel des différentes méthodes vues lors du premier semestre. Le second chapitre présente les lexiques de prononciations utilisés ainsi que les outils liés à la conversion graphème vers phonèmes, ou G2P. Le troisième chapitre porte sur le travail effectué lors de ce semestre avec les traitements effectués sur les lexiques de prononciations anglais et français. Le quatrième chapitre est lié à la validation et à la sélection des prononciations. Enfin le dernier chapitre sera la conclusion, suivi de la bibliographie et des annexes.

Dans notre rapport, nous utiliserons les caractères SAMPA français[3] pour transcrire les phonèmes.

Mots-clés : conversion graphèmes vers phonèmes (*Grapheme-to-Phoneme* - G2P), taux d'erreurs sur les mots (*Word Error Rate* - WER), taux d'erreurs sur les phonèmes (*Phoneme Error Rate* - PER), mémoire à court et long terme (*Long Short-Term Memory* - LSTM)

[1] **Un graphème** est une lettre ou un ensemble de lettres exprimant un phonème, par exemple le graphème *eau* exprime le phonème [o].

[2] **Un phonème** est une unité abstraite servant à exprimer un ou plusieurs sons dans une langue.

[3] `https://fr.wikipedia.org/wiki/Symboles_SAMPA_fran%C3%A7ais`

1.1 Présentation du sujet

Le sujet de notre projet tutoré est la validation automatique des variantes de prononciations pour la reconnaissance et la synthèse de la parole. L'objectif initial était l'évaluation de l'alignement à base de règles pour décider si une prononciation prédite (par un modèle neuronal ou statistique) était valide ou non. Les premiers résultats assez décevants ont amené à ré-orienter la suite du travail, et à étudier la possibilité de combiner plusieurs approches pour améliorer les performances de prédictions des prononciations de mots. Voici un rappel sur ce que nous avons étudié et qui nous a servi pour cette partie sur la réalisation de notre projet.

1.2 Méthodes utilisées

1.2.1 Approche à base de règles

Cette approche est constituée de trois parties :

- *Une base de faits*, qui est sa connaissance du monde

  Base de fait : $BF = \{C, H, A, T\}$

- *Une base de règles* qui détermine les interactions possibles avec ce monde

  Règles : $\{C + H \rightarrow ch,\ E + A + U \rightarrow o,\ A + T \rightarrow at\}$

- *Un moteur d'inférence* qui est sa manière de raisonner. Celui-ci peut être par chaînage avant, chaînage arrière, ou la combinaison des deux.

  Avec un chaînage avant, l'algorithme utilise la base de règles pour créer une nouvelle base de faits $BF_2 = \{C, H, A, T, ch, at\}$

On retrouve cette approche dans l'article de H. Elovitz & al.[1], où elle est utilisée pour développer un synthétiseur vocal appelé *Votrax*.

1.2.2 Approche statistique

L'approche statistique est un champ de l'Intelligence Artificielle dans lequel des méthodes statistiques sont utilisées pour faire enrichir la connaissance des machines sur un sujet précis.

Nous avons lu deux articles traitant de cette approche : le premier explique le principe de séquences jointes [2] et le second concerne les champs conditionnels [3]. Les séquences jointes se basent sur l'alignement entre une suite de phonèmes et de graphèmes, comme l'illustre le schéma ci-dessous extrait de l'article de M. Bisani et H. Ney [2].

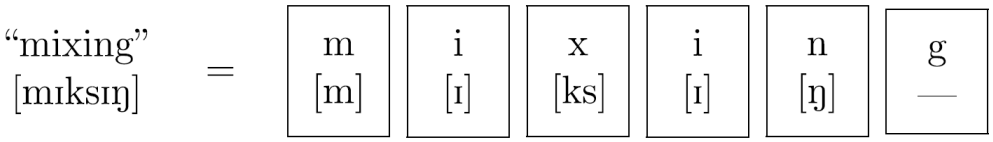

Figure 1: Schéma d'utilisation d'un modèle statistique utilisant les séquences jointes [2].

Dans le schéma ci-dessus nous avons un mot *mixing* et sa traduction phonétique [mIksIN], le but va être de trouver l'alignement parfait entre chaque graphème et chaque phonème. Comme nous pouvons le voir, un graphème peut avoir un phonème vide, comme le *g* qui est muet, ou deux comme le x qui se prononce [ks]. La méthode des séquences jointes est directement liée au logiciel Sequitur, c'est donc cette méthode que nous avons choisi pour l'approche statistique.

1.2.3 Approche par réseaux de neurones

Pour vulgariser, les réseaux de neurones fonctionnent de la même façon que le cerveau humain, c'est-à-dire qu'ils possèdent des couches de neurones connectées entre elles. Notre but est d'avoir une validation automatiques des variantes de prononciations, nous avons donc pour cela un corpus d'apprentissage en plus de cette approche. Plus nous avons de neurones et de couches de neurones, plus notre modèle sera précis. Le réseau de neurones s'entraînera sur notre corpus d'apprentissage, après quoi nous pourrons l'utiliser pour faire des prédictions. L'un des réseaux de neurones les plus utilisé est le *Long Short Terme Memory*, ou LSTM [2]. La seconde méthode que nous avons étudiée est celle des séquences-à-séquences [4]. Une autre méthode est celle des *Transformer based conversion* [5], tandis que la dernière porte sur la *Convolutional Neural Networks conversion.* [5].

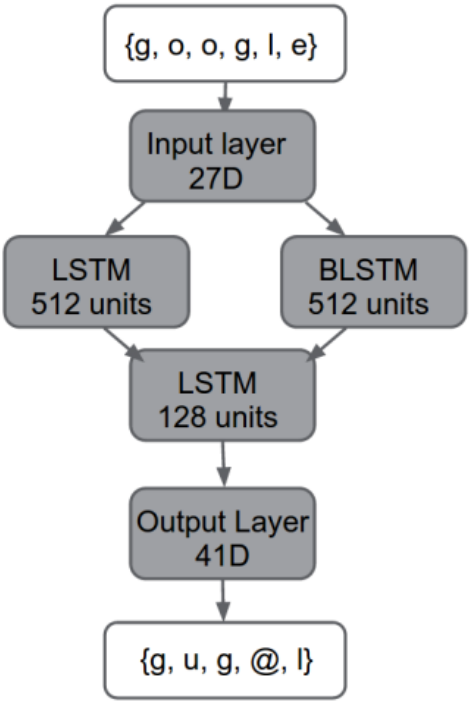

Figure 2: Schéma d'utilisation d'un modèle BLSTM pour la conversion G2P dans une architecture en réseaux de neurones. [6].

Dans le schéma ci-dessus, nous avons plusieurs informations importantes comme l'*Input layer* et l'*Output layer* qui représentent les couches de neurones que l'on va mobiliser pour faire fonctionner notre modèle. Nous avons les types de neurones *LSTM* et *BLSTM*[4] et pour chaque couche exploitant un type ou l'autre, nous aurons 512 neurones. Notre modèle prendre en entrée le mot *google* et il nous donnera sa transcription phonétique [gug@l] en sortie. Nous avons choisi pour l'approche à base de réseaux de neurones les réseaux LSTM car c'est avec celle-ci que fonctionne le logiciel OpenNMT.

[4]La différence entre un réseau de neurones **LSTM** et **BLSTM** est que chaque couche est parcourue dans un seul sens pour le premier tandis que pour le second, chaque couche est parcourue dans les deux sens.

2 Présentation générale

2.1 Lexiques de prononciations

Dans cette section, nous présentons les deux lexiques de prononciations que nous avons utilisés dans le cadre du projet : un lexique de prononciations français et un lexique de prononciations anglais. Ils possèdent une grande quantité de mots avec leur transcription phonétique associée. L'avantage d'utiliser un lexique de prononciations en français et un autre en anglais est que l'on peut faire par la suite des comparaisons entre les résultats de notre modèle pour voir dans quelle langue il excelle le plus.

2.1.1 Lexique de prononciations en français

Le lexique de prononciations français utilisé s'appelle **BDLEX** (Base de Données LEXicales) et comporte un ensemble de 337 550 mots listés par ordre alphabétique. La table 1 présente un extrait de ce lexique.

Table 1: Extrait de mots et de prononciations du lexique français

abajoue	a b a Z u
abajoues	a b a Z u
abat	a b a
abatage	a b a t a Z
abatages	a b a t a Z
abats	a b a
abattis	a b a t i
abattée	a b a t e
abattées	a b a t e
abbevillien	a b @ v i l j e

Chaque ligne contient un couple composé d'un mot avec sa transcription phonétique utilisant les symboles en SAMPA[5] français (*Speech Assessment Methods Phonetic Alphabet*).

Nous avons également eu besoin d'une version à l'envers de ce lexique de prononciations car nous avons comparé les prédictions à l'endroit et celles à l'envers dans le but de savoir si la prise en compte du contexte avant ou arrière est importante pour les modèles utilisés.

[5]Voici le lien vers la liste de symboles SAMPA français : `https://fr.wikipedia.org/wiki/Symboles_SAMPA_fran%C3%A7ais`

Pour cela, nous avons inversé l'ordre des lettres de chaque mots et de chaque phonèmes, comme le montre la table 2 (qui est la version à l'envers de la table 1).

Table 2: Extrait de mots et de prononciations du lexique de prononciations français à l'envers

euojaba	u Z a b a
seuojaba	u Z a b a
taba	a b a
egataba	Z a t a b a
segataba	Z a t a b a
staba	a b a
sittaba	i t a b a
eéttaba	e t a b a
seéttaba	e t a b a
neillivebba	e j l i v @ b a

2.1.2 Lexique de prononciations en anglais

Le lexique de prononciations en anglais qui a été utilisé est le **CMUdict**[6] (*Carnegie Mellon University Prounoncing Dictionnary*). Il contient un ensemble de 134 304 mots et sa structure est identique à celle du lexique de prononciations français :

Table 3: Exemples de mots et de phonèmes du lexique de prononciations anglais

AAA	T R IH2 P AH0 L EY1
AAAI	T R IH2 P AH0 L EY2 AY1
AABERG	AA1 B ER0 G
AACHEN	AA1 K AH0 N
AACHENER	AA1 K AH0 N ER0
AAH	AA1
AAKER	AA1 K ER0
AALIYAH	AA2 L IY1 AA2
AALSETH	AA1 L S EH0 TH
AAMODT	AA1 M AH0 T

Les caractères utilisés pour la transcription phonétique sont issus de l'ensemble des symboles ARPAbet[7] (*Advanced Research Projects Agency alphabet*). Les chiffres (0, 1 et 2)

[6]Voici le lien d'accès au lexique : `https://github.com/Alexir/CMUdict/blob/master/cmudict-0.7b`
[7]Voici le lien d'accès vers la liste des symboles : `https://en.wikipedia.org/wiki/ARPABET`

après les symboles des voyelles correspondent à l'accentuation de syllabes en anglais (plus communément appelé le stress[8]).

2.2 Outils de conversion graphèmes vers phonèmes

Dans cette sous section, nous détaillons les trois approches que nous avons utilisées sur chacun des lexiques de prononciations. Nous avons appliqué une méthode, celle par réseaux de neurones sur le lexique de prononciations anglais, et l'ensemble des méthodes sur deux variantes du lexiques de prononciations français : une version avec les graphèmes et phonèmes à l'endroit (lexique de prononciations initial) et une version avec les graphèmes et phonèmes à l'envers (l'ordre de chaque séquence de lettres et de chaque séquence de phonèmes est inversée). Ainsi, nous avons deux types d'apprentissages (par réseaux de neurones et par séquences jointes) avec lesquelles nous comparons les résultats dans le but d'évaluer leur performance.

Nous avons découpé les données en trois parties:

- Une partie *train* composée de 70% du lexique de prononciations pour l'apprentissage
- Une partie *dev* composée de 15% du lexique de prononciations pour le développement
- Une partie *test* composée de 15% du lexique de prononciations pour évaluer les performances des modèles

Cette découpe a été utilisée pour tous les apprentissages et évaluations de performances. Pour obtenir ces trois fichiers, nous avons implémenté une fonction en python et utilisé la fonction *shuffle* pour mélanger l'ordre des lignes afin d'avoir dans chacun une mixité équitable, les mots dans les lexiques de prononciations originaux étant rangés par ordre alphabétique.

2.2.1 Règles d'alignement

Les règles d'alignement sont un ensemble de règles phonétiques mettant en correspondances des lettres ou suites de lettres avec des phonèmes. Un ensemble initial de règles nous a été fourni par notre tuteur au début de ce semestre.

[8]Le stress est une caractéristique importante de la langue anglaise car c'est l'accent tonique du mot. Cela signifie que sur un mot de deux syllabes ou plus, nous aurons toujours une syllabe plus accentuée que les autres.

Il s'agit de la première méthode que nous avons utilie afin pour nous familiariser avec les outils de l'apprentissage supervi. La table 4 donne un aperçu des règles utilies pour l'alignement.

Table 4: Exemples de règles d'alignement des graphèmes avec les phonèmes

Graphèmes	→	**Phonèmes**
é	→	e
er	→	e
es	→	e
ez	→	e
e	→	e
ed	→	e
ë	→	e

2.2.2 OpenNMT pour la mise en oeuvre de modèles séquences-à-séquences

OpenNMT est un logiciel open source pour les modèles par réseaux de neurones (ou séquences-à-séquences). C'est un projet né en 2016 et depuis beaucoup utilisé. Il permet d'effectuer des apprentissages machines grâce à 2 couches LSTM avec 500 unités cachées sur l'encodeur et le décodeur. Il y a deux implémentations disponibles : celle utilisant *Tensor flow* et celle utilisant Python (Pytorch); c'est cette dernière que nous avons utilisée. PyTorch est une bibliothèque Python permettant l'apprentissage machine et qui s'appuie sur Torch (développé par Facebook). PyTorch permet d'effectuer les calculs tensoriels[9] nécessaires, notamment pour l'apprentissage profond.

OpenNMT est un logiciel qui s'exécute en ligne de commandes. Nous avons utilisé Ubuntu pour le faire tourner. Le logiciel nous demande une commande avec un fichier de configuration, un fichier d'apprentissage et un fichier modèle. Pour une première prise en main d'OpenNMT, nous avons commencé avec l'exemple fourni lors de son installation qui vise à traduire un texte du français vers l'allemand.
La figure 3 illustre la manière dont sont découpés les mots et comment fonctionne le modèle dans le processus d'apprentissage avec le logiciel.

[9]Les calculs tensoriels sont de manière très générale utilisés pour exprimer les relations géométriques liées aux calculs différentiels.

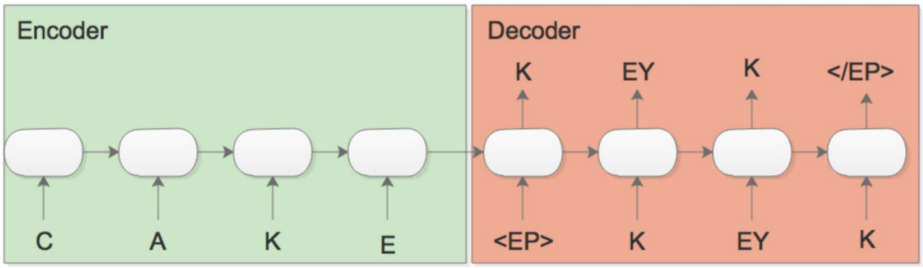

Figure 3: Fonctionnement du modèle avec l'exemple du mot *cake* en entrée dans l'encodeur et la transcription phonétique prédite par le modèle dans le décodeur [5]

Les données fournies pour l'apprentissage du modèle sont celles contenues dans le fichier *train*. Le fichier de configuration utilisé est un fichier YAML (*Yet Another Markup Language*) qui précise les données à utiliser pour l'apprentissage, l'endroit où sauvegarder le modèle, et où sauvegarder les prédictions. Un exemple de fichier YAML est disponible en annexe (voir section 6).

Une fois le fichier YAML prêt, il reste à lancer le logiciel pour l'apprentissage de notre modèle qui traitera d'abord les fichiers *train* et *dev* (cette étape prend environ 6h pour 50 000 itérations). Une fois l'apprentissage terminé, notre modèle est prêt à être évalué avec le fichier *test*, ce qui nous donne en sortie un fichier de prédictions pour les mots présents dans le fichier *test*.

2.2.3 Sequitur pour la mise en oeuvre de séquences jointes

Sequitur est un convertisseur graphèmes vers phonèmes sous licence GPL[10], qui peut être entraîné pour déterminer automatiquement la prononciation de nouveaux mots. Sequitur utilise une approche statistique de séquences jointes, l'article [2] lui est directement lié. Les résultats donnés par le modèle de Sequitur nous ont été directement fourni par notre tuteur; les fichiers utilisés (*train*, *dev* et *test*) sont les mêmes que ceux utilisés avec OpenNMT.

Maintenant que nous avons présenté les deux modèles de conversion, intéressons-nous à la manière d'évaluer leurs performances.

[10]Les licences GPL pour *General Public License* sont des licences fixes, les conditions légales de distribution d'un logiciel libre.

2.3 Mesures d'évaluation

Pour évaluer la qualité des prononciations prédites et savoir si le modèle est efficace, nous utilisons deux scores.

Le premier est le taux d'erreurs au niveau des mots appelé le *Word Error Rate* (WER) en anglais et indique le pourcentage de mots pour lesquels la prononciation prédite n'est pas correcte. L'autre score utilisé est le *Phoneme Error Rate* (PER). Il s'agit de la mesure de Levenshtein[11] entre le résultat de la transcription automatique et la prononciation de référence divisée par le nombre de phonèmes de la prononciation de référence. Le PER est en général plus faible que le WER, puisque la moindre erreur d'un phonème implique une erreur pour la transcription phonétique entière du graphème.

Grâce à ces deux scores, nous allons pouvoir faire des comparaisons entre les différents modèles.

[11] **La distance de Levenshtein** est une distance donnant la différence entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.

3 Travail réalisé

Maintenant que la présentation des lexiques de prononciations et des outils de conversions est faite, nous allons détailler le travail réalisé lors de ce semestre dans le but de tester nos approches pour la conversion des graphèmes vers les phonèmes. Nous nous sommes basés sur les lexiques de prononciations français (3.2) et anglais (3.1), ainsi que les outils de conversion G2P (2.2) présentés précédemment pour ces tests.

Dans le travail de réalisation où nous avons entraîné des modèles, nous avons eu besoin de séparer les lexiques de prononciations en trois fichiers (les détails les concernant sont dans la section 2.2).

Nous commençons la présentation du travail réalisé pendant ce semestre avec le lexique de prononciations anglais (3.1) et poursuivons avec le lexique de prononciations français (3.2).

Pour le lexique de prononciations en anglais nous avons utilisé uniquement l'approche séquences-à-séquences avec OpenNMT, par soucis de temps, contrairement au lexique français où les trois approches ont été utilisées.

3.1 Prononciations en anglais

3.1.1 Pré-traitement

Avant d'utiliser le lexique de prononciations en anglais (CMUdict) pour l'apprentissage du modèle au moyen d'OpenNMT, il nous a fallu faire un pré-traitement afin de le nettoyer de tout élément pouvant corrompre l'apprentissage et également de le préparer à ce dernier. Nous avons tout d'abord créé un fichier en enlevant les commentaires présents au début de ce lexique, il ne restait plus que les mots et les prononciations. Enfin, à partir de ce dernier fichier, nous en avons créé un autre où les graphèmes étaient séparés par un espace. Les phonèmes l'étant déjà, nous n'avons effectué aucune modification pour ces derniers (voir la figure 4).

Figure 4: Extrait de mots et de prononciations du lexique de prononciations en anglais

A A A	T R IH2 P AH0 L EY1
A A A I	T R IH2 P AH0 L EY2 AY1
A A B E R G	AA1 B ER0 G
A A C H E N	AA1 K AH0 N
A A C H E N E R	AA1 K AH0 N ER0
A A H	AA1
A A K E R	AA1 K ER0
A A L I Y A H	AA2 L IY1 AA2
A A L S E T H	AA1 L S EH0 TH
A A M O D T	AA1 M AH0 T

Après avoir fini de modifier le lexique de prononciations en anglais, nous l'avons séparé en trois parties pour pouvoir démarrer l'apprentissage avec OpenNMT (voir la section 2.2).

Nous avons ensuite divisé chacun des trois fichiers en trois fichiers de nouveau :

- Un fichier *source* composé uniquement des graphèmes
- Un fichier *target* composé uniquement des phonèmes avec le stress (représenté par les numéros)
- Un fichier *target bis* composé uniquement des phonèmes sans le stress (représenté par les numéros)

Au final, nous avons obtenu plusieurs fichiers : les fichiers *train*, *dev* et *test* originaux, ainsi que trois autres fichiers (*source*, *target*, *target bis*) pour chacun d'entre eux.

3.1.2 Approche par réseaux de neurones avec OpenNMT

Pour l'apprentissage du modèle séquences-à-séquences avec OpenNMT, nous avons dû créer un fichier de configuration au format YAML. Ce dernier permet au modèle de savoir où trouver les fichiers dont il aura besoin pour l'apprentissage du modèle. Il s'effectue en deux phases :

1. Première phase : *sample*

   Cette phase consiste à construire le vocabulaire nécessaire pour entraîner le modèle en précisant le nombre de lignes prélevées sur notre lexique de prononciations (nous avons choisi 10 000 lignes).

2. Seconde phase : *train*

   Cette phase consiste à entraîner le modèle grâce aux fichiers nécessaires, qui sont répertoriés dans le fichier YAML. OpenNMT ira les chercher en se basant sur le chemin écrit dans ce fichier.

La dernière étape consiste à réaliser des prédictions sur le fichier *test* avec le modèle. Les résultats obtenus seront alors comparés aux résultats attendus pour mesurer sa performance.

L'apprentissage a été réalisé avec 50 000 étapes[12], une sauvegarde du modèle ayant été réalisée toutes les 5 000 étapes.

3.1.3 Résultats

Table 5: Résultats avec ou sans prise en compte du stress dans le lexique de prononciations en anglais pour le modèle séquences-à-séquences

Modèle	**PER**	**WER**
Avec prise en compte du stress	10.93%	41.58%
Sans prise en compte du stress	8.71%	36.86%

Les résultats observés dans la table 5 nous montrent que la considération du stress dans la transcription phonétique d'un mot donne un taux d'erreurs plus important, le modèle à donc quelques faiblesses sur ce point.

Nous les avons comparé à ceux présentés dans l'article [2]. Dans ce dernier, les auteurs ont utilisé une approche de séquences jointes (voir la section 1.2.2) en utilisant le même lexiques de prononciations en anglais. Cependant, la répartition entre les fichiers *train* et *test* sont différentes : leur fichier *train* est composé de 79,55% du lexique de prononciations en anglais (contre 70% pour notre modèle) et leur fichier *test* est composé de 8,93% de ce même lexique (contre 15% pour notre modèle). Ces différences sont donc à prendre en considération lors de la comparaison entre les deux résultats.

[12] **Une étape** est la lecture du fichier d'apprentissage.

Table 6: Résultats de notre modèle séquences-à-séquences en comparaison du modèle de séquences jointes avec la prise en compte du stress pour le lexique de prononciation en anglais

Modèle	**PER**	**WER**
Modèle séquences-à-séquences	10.93%	41.58%
Modèle de séquences jointes	7.10%	28.50%

Leur modèle ayant pris en compte le stress, nous avons seulement comparé nos résultats le prenant aussi en compte.

Le modèle de séquences jointes semble plus efficace pour la conversion graphèmes vers phonèmes (7,10% contre 10,93% concernant le PER, et 28,50% contre 41,58% concernant le WER), néanmoins la répartition des fichiers *train* et *test* a pu avoir une influence sur ces résultats.

Afin de déterminer quelle approche est la plus performante pour ce lexique, il faudrait utiliser les mêmes répartitions pour les fichiers afin de pouvoir comparer les résultats efficacement.

3.2 Prononciations en français

3.2.1 Approche par règles d'alignement

A partir du fichier de règles fourni par notre tuteur (voir la section 2.2.1), nous avons appliqué ces dernières sur les fichier *train* et *dev* afin de voir si des erreurs d'alignement étaient produites. Comme c'était le cas, nous avons ajouté des règles pour corriger ces erreurs, et nous avons appliqué de nouveau le fichier de règles (enrichies par les nouvelles règles) sur le même fichier que précédemment. Ce cycle a été répété jusqu'à ce qu'il n'y ait plus aucune erreur.

Figure 5: Extrait de règles ajoutées à l'ensemble initial des règles d'alignement

um → o∼	columbarium → k O/ l o∼ b a R j O m	
z → ts	breitschwanz → b R a j t S v a t s	
ue → ju	fuel → f j u l	

Lorsque nous avons corrigé toutes les erreurs, nous avons appliqué le fichier de règles sur le fichier *test*, afin de voir si elles étaient performantes. Aucune erreur n'a été produite sur ce fichier, nos règles étaient donc suffisantes pour tous les mots du fichier *test*.

3.2.2 Approche par réseaux de neurones avec OpenNMT

Version à l'endroit

- Les étapes pour le lexique de prononciations français à l'endroit sont identiques à celles du lexique de prononciations en anglais (voir la section 3.1.2 pour plus de détails) :
  - La première phase consiste à construire le vocabulaire pour l'apprentissage du modèle. Celui-ci est prélevé sur un nombre fini de lignes (ici, 10 000 lignes)
  - La seconde phase consiste à entraîner le modèle grâce aux fichiers nécessaires (indiqués dans le fichier de configuration au format YAML)
- Lorsque ces deux phases sont finies, l'évaluation du modèle se fait sur le fichier *test* en comparant les résultats obtenus avec ceux de référence.

Version à l'envers

- Les étapes pour le lexique de prononciations en français à l'envers sont exactement les mêmes que celles du lexique de prononciations français à l'endroit.

3.2.3 Approche statistique avec Sequitur

L'apprentissage avec Sequitur a été réalisé par notre tuteur, qui nous a ensuite envoyé les résultats pour le lexique de prononciations français à l'endroit et à l'envers. Nous avons analysé les résultats, qui sont disponibles dans la table 7.

Nous avons résumé les résultats sur le fichier *test* pour chacune des deux méthodes avec la version à l'endroit et la version à l'envers.

Table 7: Performances de chaque modèle pour le lexique de prononciations en français

Modèle	**PER**	**WER**
OpenNMT endroit	0.59%	3.05%
OpenNMT envers	0.76%	4.24%
Sequitur endroit	0.53%	3.15%
Sequitur envers	0.50%	2.92%

On observe que le PER est sensiblement similaire pour les quatre modèles, mais ceux obtenus avec Sequitur sont légèrement inférieurs à ceux d'OpenNMT, notamment pour le modèle à l'envers (étant le seul avec un WER inférieur à 3% parmi les quatre modèles).

3.3 Comparaison de l'approche par réseaux de neurones sur les deux lexiques de prononciations avec OpenNMT

Afin de comparer l'approche séquences-à-séquences sur les deux lexiques de prononciations (français et anglais), nous allons considérer uniquement la version à l'endroit du lexique de prononciations français.

Table 8: Résultats de l'approche séquences-à-séquences sur les deux lexiques de prononciations

Modèle OpenNMT	**PER**	**WER**
Lexique de prononciations anglais (avec la prise en compte du stress)	10.93%	41.58%
Lexique de prononciations anglais (sans la prise en compte du stress)	8.71%	36.86%
Lexique de prononciations français à l'endroit	0.59%	3.05%

Comme nous pouvons le voir sur la table 8, le modèle OpenNMT est plus performant sur le lexique de prononciations français à l'endroit, même en considérant l'autre lexique sans la prise en compte du stress. Cela pourrait s'expliquer par la grande différence du nombre de mots présents dans chacun d'entre eux (337 550 pour le français contre 134 304 pour l'anglais). Pour pouvoir les comparer plus efficacement, il faudrait alors utiliser deux lexiques de prononciations ayant approximativement la même taille.

4 Validation et sélection des prononciations

4.1 Application de l'approche par règles

Les résultats de cette approche (voir section 3.2.1) ont montrés que les règles étaient parfaitement appliquées, puisqu'aucune erreur ne subsistait dans leur application sur le fichier *test*. Cependant, cette approche ne permet pas de détecter les mauvaises prononciations. Comme vu dans la table 4, un ou plusieurs graphèmes peuvent correspondre au même phonème, ce qui peut compliquer la tâche pour l'application des règles. Par exemple, *e* peut-être prononcé [e] ou [@] en fonction de sa position, ou de ses contextes avant et/ou arrière.

En effet, nous avons séparé les résultats des prédictions par le modèle séquences-à-séquences (OpenNMT, voir section 2.2) en fonction de si elles étaient correctes ou incorrectes, en le comparant à la prédiction de référence. Nos règles ont ensuite été appliquées sur les deux fichiers qui en ont résulté (l'un contenait uniquement les prédictions correctes et l'autre uniquement les prédictions incorrectes) :

- Pour le fichier de prononciations correctes, aucune erreur d'alignement n'a été trouvée ce qui est normal, puisque les règles alignaient les prononciations du fichier *test*
- Pour le fichier de prononciations incorrectes, il n'y a également eu aucune erreur d'alignement, ce qui n'est pas normal

Figure 6: Extrait d'alignements des graphèmes vers phonèmes sur des prononciations incorrectes du lexique de prononciations en français

Mot à l'endroit	abstinent → a b s t i n
Mot à l'envers	sreiuqca → e j k a

Nos règles sont minimales, c'est-à-dire qu'elles ne s'intéressent qu'à un ou plusieurs graphèmes n'ayant qu'un phonème. En conséquence de quoi, elles ne sont pas assez strictes pour détecter les erreurs. Une piste de recherches serait de faire des règles plus complexes, en incluant les contextes avant et/ou arrière afin d'être le plus contraignant possible. Néanmoins, cela signifie qu'il faudrait écrire beaucoup plus de règles afin de couvrir tous les contextes possibles.

4.2 Comparaison des prédictions à l'endroit et à l'envers d'OpenNMT et de Sequitur

Nous avons comparé les prédictions des quatre modèles (OpenNMT à l'endroit, OpenNMT à l'envers, Sequitur à l'endroit et Sequitur à l'envers) en nous basant sur la prononciation de référence. Il y a cinq configurations possibles :

- **4** : les quatre prédictions sont identiques
- **3-1** : trois prédictions sont identiques et une est différente
- **2-2** : deux prédictions sont identiques et les deux restantes sont identiques entre elles
- **2-1-1** : deux prédictions sont identiques et les deux restantes sont différentes entre elles
- **1-1-1-1** : toutes les prédictions sont différentes

Table 9: Résultats de la comparaison des prédictions des différents modèles

Nombre de prédictions identiques parmi les modèles	**Nombre de prédictions (sur un total de 51 612)**	**Proportion des configurations (en %)**
4	48 165	93.32%
3-1	2098	4.06%
2-2	917	1.78%
2-1-1	424	0.82%
1-1-1-1	8	0.02%

Nous pouvons voir que tous les modèles ont réalisé presque totalement les mêmes prédictions comparées à la prononciation de référence. En effet, 93.32% des prédictions de la configuration 4 sont identiques aux prononciations de référence, tout comme 4.06% des prédictions de la configuration 3-1 (pour celle-ci, la comparaison a été effectuée avec la prédiction majoritaire). Il y a donc 97.38% des prédictions dont au moins 3 sont identiques qui correspondent à la prononciation de référence.

4.3 Combinaison des modèles

Nous avons combiné les prédictions de ces quatre modèles afin de choisir la meilleure prédiction pour chaque mot. Le but était d'obtenir un fichier de prédictions encore plus précis qu'avec seulement le modèle le plus performant parmi les quatre. Ainsi, si on se retrouve dans la configuration '2-2' ou '1-1-1-1', nous avons choisis la prédiction donnée par Sequitur envers car il s'agit du modèle le plus performant des quatre. Si la configuration est du type '3-1' ou '2-1-1' alors nous prenons la prédiction majoritaire. Il n'y a aucune ambiguïté, pour la configuration '4' où les prédictions sont toutes les mêmes.

Cette manière de procéder nous a permis d'obtenir un PER de 0.47% et un WER de 2.75%, qui sont les meilleurs scores, sans pour autant être réellement plus performant.

Nous terminons par une analyse de ces erreurs qui perdurent dans la conversion G2P, détaillée dans la section suivante.

4.4 Analyse des erreurs récurrentes

L'analyse des erreurs récurrentes est faite sur les prédictions réalisées par la combinaison des modèles (voir section 4.3).

Nous avons exécuté un script en python que nous avons créé pour retrouver les différentes erreurs, et les classer en trois catégories : *omission*, qui signifie qu'un phonème dans la prédiction est manquant par rapport à la prédiction de référence ; *substitution*, qui signifie qu'un phonème a été remplacé par un autre dans la prédiction ; *insertion*, qui signifie qu'un phonème a été ajouté dans la prédiction par rapport à la prédiction de référence.

Table 10: Présentation des différents types d'erreurs pour la prédiction des prononciations

Type d'erreur	**Prononciation de référence :** abaZu
omission	abZu
substitution	ataZu
insertion	abaZue

Sur 51 612 prédictions, nous en avons eu 1 420 erronées (en comparaison aux prédictions de référence), soit 2,75%.

Parmi ces prédictions erronées, nous avons :

- 349 phonèmes *omis*, soit 24,59%
- 864 phonèmes *substitués*, soit 60,84%
- 207 phonèmes *insérés*, soit 14,57%

La substitution d'un phonème est l'erreur la plus courante, avant l'omission et l'insertion. Nous avons analysé les erreurs pour savoir quels étaient les phonèmes les plus touchés par ces erreur. Pour le type *substitution*, nous avons mis le phonème remplacé suivit de ":" et du phonème remplaçant.

Table 11: Présentation des différents types d'erreurs pour la prédiction des prononciations

Type d'erreur	**Exemples de phonèmes**
omission	t, a, R
substitution	9:@, E/:@ , e∼:a∼
insertion	j, i, a

Nos analyses ont montrées que parmi les erreurs liées à la substitution d'un phonème, 53,95% sont liées au degré d'ouverture (par exemple, 9:@ ou E/:@).

5 Conclusion

L'objectif initial du projet était la validation des prononciations avec l'utilisation des règles d'alignement. Pour cela, nous avons étudié différentes approches pour effectuer ces conversions, et utilisé deux lexiques de prononciations différents. Pour chacune des approches, nous avons utilisés le lexique de prononciations anglais et deux versions du lexique de prononciations français (endroit et envers) afin de comparer les performances. Cependant les premiers résultats peu concluants des règles d'alignement nous on amené étudier la combinaison de plusieurs approches de conversion G2P pour améliorer les performances des prononciations des mots. Nous avons alors effectué une comparaison entre les approches restantes et avons combiné les prédictions de chacun des quatre modèles pour sélectionner la plus probante à chaque fois.
Ce choix nous a permis d'obtenir un résultat encore plus précis que l'utilisation d'un seul modèle. Cependant, ce système n'est pas fiable à 100% et nous avons vu que des erreurs de prononciations perdurent malgré tout. Certaines sont de simples différences de degrés d'ouvertures comme nous l'avons vu, et peuvent être tout de même considérées comme acceptables en tant que variantes de prononciations, mais certains phonèmes récurrents sont mal prédits par moments et ne peuvent pas être considérés comme une variante.

Au cours de cette partie réalisation, nous avons appris à nous servir d'un outil de conversion graphèmes vers phonèmes utilisant les réseaux de neurones. Nous avons appris une méthodologie concernant la séparation des données pour l'apprentissage machine. Malgré quelques manques de connaissances fondamentales sur le fonctionnement des modèles à base de réseaux de neurones, le projet nous a permis de découvrir leur fonctionnement en détail, chose que nous avons vue beaucoup plus tard dans nos cours. Nous avons aussi appris comment évaluer les performances de ces modèles, comment les analyser et les améliorer. Enfin, nous avons appris à combiner différents modèles de conversions pour en tirer le meilleur résultat possible.

Ce fut un projet très instructif sur la manière de traiter les données et sur le fonctionnement de la conversion graphèmes-phonèmes, fondamentale pour la reconnaissance et la synthèse de la parole. Au-delà de la connaissance pratique, le projet tutoré nous a donné un aperçu de ce qu'est un travail de recherche, de ce qu'il exige en terme de recherche bibliographique, de réalisation pratique et de rigueur.

6 Annexes

Voici le contenu du fichier YAML pour utiliser OpenNMT sur le lexique de prononciations en anglais. Les lignes commençant par # sont des commentaires.

```
# en_cmudict.yaml

## Where the samples will be written
save_data: run/fr_data

## Where the vocab(s) will be written
#src_vocab: run/en_cmudict.src
#tgt_vocab: run/en_cmudict.tgt

## Prevent overwriting existing files in the folder
overwrite: False

## Corpus opts:
data:
    corpus_1:
        path_src: fr_data_train.src
        path_tgt: fr_data_train.tgt
    valid:
        path_src: fr_data_dev.src
        path_tgt: fr_data_dev.tgt

## Vocabulary files that were just created
#src_vocab: run/en_cmudict.src
#tgt_vocab: run/en_cmudict.tgt

## Train on a single GPU
#world_size: 1
#gpu_ranks: [0]

# Where to save the checkpoints
```

```
save_model: run/model
save_checkpoint_steps: 5000
train_steps: 50000
valid_steps: 500
```

Liste des tables

Liste des figures

References

[1] H. Elovitz, R. Johnson, A. McHugh, and J. Shore, "Letter-to-sound rules for automatic translation of english text to phonetics," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, pp. 446–459, Dec. 1976.

[2] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, pp. 434–451, May 2008.

[3] I. Illina, D. Fohr, and D. Jouvet, "Grapheme-to-Phoneme Conversion Using Conditional Random Fields," *12th Annual Conference of the International Speech Communication Association - Interspeech 2011*, p. 4, 2011.

[4] K. Yao and G. Zweig, "Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion," *arXiv:1506.00196 [cs]*, Aug. 2015. arXiv: 1506.00196.

[5] S. Yolchuyeva, G. Németh, and B. Gyires-Tóth, "Transformer Based Grapheme-to-Phoneme Conversion," in *Interspeech 2019*, pp. 2095–2099, ISCA, Sept. 2019.

[6] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (South Brisbane, Queensland, Australia), pp. 4225–4229, IEEE, Apr. 2015.