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

Décembre 2020

Sommaire

1 Introduction

Dans le cadre de notre Master de Traitement Automatique des Langues (TAL), nous avons été amenés à réaliser un projet tutoré sur un sujet de recherche. Ce projet se divise en deux parties: une bibliographie au premier semestre et une partie réalisation au second semestre. Nous avons choisi de travailler sur un projet proposé par Denis JOUVET, chercheur au sein de l'équipe Multispeech au Laboratoire lorrain de recherche en informatique et ses applications (Loria). Le sujet du projet porte sur la validation des variantes de prononciation pour la reconnaissance et la synthèse de la parole. La prononciation d'un mot correspond à une séquence de phonèmes et certains mots peuvent être prononcés de plusieurs manières. Pour reproduire ces symboles en informatique on utilise le langage SAMPA (*Speech Assessment Methods Phonetic Alphabet*) disponible dans plusieurs langues et qui se base sur l'API (alphabet phonétique international). On trouve ici un exemple avec des variantes de prononciation pour les mots "maintenant" et "*the*" :

(French)	maintenant :	m ɛ̃ t n ɑ̃	m ɛ̃ t ə n ɑ̃
(English)	the :	ð ə	ð i

L'étude de la prononciation des mots trouve son utilité dans la reconnaissance de la parole (d'un signal vocal vers une séquence de mots) et la synthèse de la parole (d'un texte vers un signal vocal). L'un des enjeux de cette étude est la validation des variantes de prononciations, car les lexiques ne sont malheureusement pas assez complets. Afin de palier au problème, une méthode qui existe actuellement est de comparer des conversions graphèmes-phonèmes faites avec plusieurs méthodes et d'observer les différences; mais ce procédé n'est pas toujours fiable et présente parfois des erreurs. L'objectif de ce projet est de détecter les prononciations incorrectes dans un dictionnaire de prononciation. Dans cette bibliographie, nous présenterons des méthodes pour la conversion graphèmes-phonèmes : les approches à base de règles, les méthodes statistiques et l'utilisation des réseaux de neurones. Enfin, nous conclurons sur ce qu'apporte l'ensemble de ces procédés et les perspectives sur lesquelles nous nous pencherons au prochain semestre.

Mots-clés : conversion graphèmes vers phonèmes (*Grapheme-to-Phoneme* - G2P), modèle de séquences jointes (*Joint-Sequence Model* - JSM), champs aléatoires conditionnels (*Conditional Random Fields* - CRF), modèle de Markov caché (*Hidden Markov Models* - HMM), mémoire à court et long terme (*Long Short-Term Memory* - LSTM), réseaux de neurones récurrents (*Recurent Neural Network* - RNN) et réseaux convolutionnels (*Convolutionnal Neural Network* - CNN)

2 Les méthodes de conversions graphèmes vers phonèmes

Nous avons pris le parti ici de parler des méthodes utilisées dans les articles que nous avons lu en suivant un ordre progressif de complexité. On retrouve premièrement l'approche à base de règles, suivi des méthodes statistiques et finalement les réseaux de neurones.

2.1 L'approche à base de règles

L'approche à base de règles est plus communément appelée système expert dans le monde de l'intelligence artificielle. Un système expert est composé de trois parties :

- Une base de fait : c'est-à-dire sa connaissance du monde ;
- Une base de règles : c'est-à-dire les interactions possibles qu'il peut avoir avec ce monde ;
- Un moteur d'inférence : c'est-à-dire la façon dont il va raisonner. Il y a généralement 3 façons de raisonner: par chaînage avant, par chaînage arrière ou en combinant les deux méthodes, chaînage avant et arrière.

Le raisonnement par chaînage avant va partir des règles et des faits de notre base de connaissances pour en déduire de nouveaux faits tandis que le chaînage arrière va partir d'une règle et va effectuer le chemin inverse pour se référer à la base de faits.

Soit B_f une base de faits et R un ensemble de règles. L'algorithme de chaînage avant va utiliser les connaissances dans la base de faits pour en déduire de nouveaux faits. Par exemple :

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

Ici notre base de faits est composée de 4 lettres et de 2 règles. Le chaînage avant, va nous permettre de déduire un phonème à partir de plusieurs lettres. A l'état initial $Bf = \{C, H, A, T\}$ et après avoir utilisé l'algorithme de chaînage avant, nous avons une nouvelle base de faits : $BF_1 = \{C, H, A, T, ch\}$. Ici, l'algorithme ne trouvera rien d'autre car il ne peut pas appliquer d'autres règles.

Dans l'article [1] datant des années 70, on trouve justement un exemple d'utilisation d'une approche à base de règles pour la conception d'un outil pour effectuer une traduction d'anglais vers l'API, dans le cadre d'une création d'un synthétiseur vocal (nommé *Votrax*). Ce système

se basait sur des dictionnaires de prononciations qui faisaient correspondre 1 phonème pour 1 lettre ou plus. Les règles utilisées dans le système pour la traduction fonctionnaient comme ce qui suit : en considérant une séquence de morphèmes ABC, pour la traduction du morphème B en phonème, le système prenait en compte le contexte A et le contexte C pour donner la prononciation D en sortie. Il y a donc une prise en compte des deux contextes: avant et après.

Sur base d'un dictionnaire de 329 correspondances lettre-son et en testant leur modèle sur le *Brown Corpus*[1] ce processus donna des résultats plutôt encourageant avec une bonne prédiction des mots de 90% et de 97% au niveau des phonèmes. Les erreurs étaient principalement dues aux variantes de prononciations des voyelles accentuées (en anglais). L'équipe de chercheurs évoque aussi la préférence d'un mauvais positionnement d'accent plutôt que d'un schwa (voyelle neutre) qui implique une perte d'informations. Il met en lumière la nécessité d'améliorer les système de traduction graphèmes-phonèmes et la fiabilité des dictionnaires de prononciations.

[1]ensemble de textes anglais et américain, c'est le premier grand corpus structuré et stocké en ligne et contenant 500 échantillons d'anglais avec environ un millions de mots, compilé à partir d'ouvrages et publié en 1961

2.2 Méthodes statistiques

L'approche statistique est un champ de l'IA basé sur des méthodes mathématiques et statistiques pour donner la capacité à un ordinateur d'apprendre. Dans cette approche le corpus a une grande importance car la méthode statistique est composée de deux phases:

- Première phase : elle consiste à entraîner notre modèle sur une partie conséquente du corpus;
- Seconde phase : elle consiste à tester la validité du modèle sur la partie restante et en fonction des résultats, ajuster ou non la précision du modèle.

Le choix des proportions du corpus pour l'entraînement et la validation sont très importants. Un modèle entraîné sur un partie du corpus trop petite lui fera perdre en précision; et inversement: un corpus de validation trop faible risque de donner de mauvaise interprétation de la précision du modèle. Il faut donc veiller à garder une taille suffisante pour que l'estimation des performances sur ce corpus soit pertinente. La configuration optimale que l'on utilise le plus est un découpage avec 80% du corpus pour l'entraînement et 20% pour la validation du modèle.

Dans le but de calculer l'exactitude de chaque méthode, on calcule le *Phoneme Error Rate* (PER), c'est-à-dire le taux d'erreurs de mots, et le *Word Error Rate* (WER), le taux d'erreurs de phonèmes.

Dans les méthodes statistiques, nous trouvons plusieurs approches : *Joint-sequence conversion* et *Conditional Random Fields conversion.*

2.2.1 Joint-Sequence Conversion

L'article [2] présente une méthode statistique basée sur les séquences jointes ou *joint-sequences* en anglais. Cette méthode s'appuie sur le principe de l'alignement entre une suite de lettres et une suite de phonèmes. On va chercher à générer pour un mot la meilleure transcription phonétique de celui-ci. Ce qui veut dire que pour une forme orthographique $g \in G^*$ nous cherchons la meilleure prononciation $\varphi \in \phi^*$ où φ est un élément de l'ensemble ϕ^* qui lui représente l'ensemble de toutes les séquences de phonèmes et G^* est l'ensemble de toute les formes orthographiques possibles. Le modèle de séquences jointes se base sur le principe de co-segmentation c'est-à-dire que la relation entre l'entrée et la sortie peut être générée par une séquence jointe qui va couvrir les symboles d'entrées et de sorties. On appelle ces symboles des graphones; par exemple voici un graphone et sa transcription : *ing* et [ɪŋ].

L'apprentissage du modèle s'effectue avec l'algorithme d'espérance-maximisation qui est un algorithme itératif qui va chercher à trouver les résultats les plus optimaux pour un modèle probabiliste.

2.2.2 Conditional Random Fields conversion

L'article [3] nous a servi d'introduction à la méthode susnommée. L'objectif de ce dernier est d'établir que le *Conditional Random Fields* (CRF) est performant pour la conversion G2P. Le CRF est un modèle graphique où la probabilité conditionnelle est calculée grâce à des vecteurs, il permet donc la prédiction à long terme. Afin d'évaluer cette méthode, une expérience a été menée dans l'article sur deux corpus de tailles différentes (français et anglais). Cette expérience a nécessité l'utilisation du *Part Of Speech tag* pour le CRF dans le but d'être plus précis.

Il y a deux parties distinctes :

- La première consiste à aligner les graphèmes et les phonèmes au moyen du modèle de Markov caché ou *Hidden Markov Model*, ou HMM. L'algorithme pourra avoir recours à des phonèmes vides au besoin.
- La seconde est l'entraînement du *Grapheme-to-Phoneme*. L'apprentissage définit une distribution par probabilités conditionnelles sur une séquence d'étiquettes, en fonction d'une séquence d'observation. Deux traits sont disponibles : le trait unigram (qui ne prend en compte que le phonème actuel) et le trait bigram (qui prend en compte le phonème précédent en plus du phonème actuel).

Voici les résultats du corpus anglais :

- L'alignement des graphèmes et des phonèmes a été fait à la fois manuellement, et grâce au HMM. Le WER est à 35% pour l'alignement manuel, contre 35,7% pour le HMM tandis que le PER est à 8,2% pour l'alignement manuel contre 8,4% pour le HMM.

Voici ceux du corpus français :

- Le PER et le WER sont égaux pour le CRF et le JMM. Cependant, pour les prononciations multiples, le CRF est meilleur que le JMM.

Pour conclure, le *Conditional Random Fields* est une méthode ayant prouvé son efficacité dans la conversion G2P, puisque plus performante que les méthodes précédemment utilisées.

2.3 Réseaux de neurones

Le nom de réseau de neurones vient de la biologie, car il s'inspire directement de ce domaine. Les réseaux de neurones sont souvent optimisés par des méthodes d'apprentissage de type probabiliste, tel que les réseaux bayésiens, ou en utilisant des algorithmes de descente de gradient.

Vous trouverez ci-dessous un exemple de réseau neuronal :

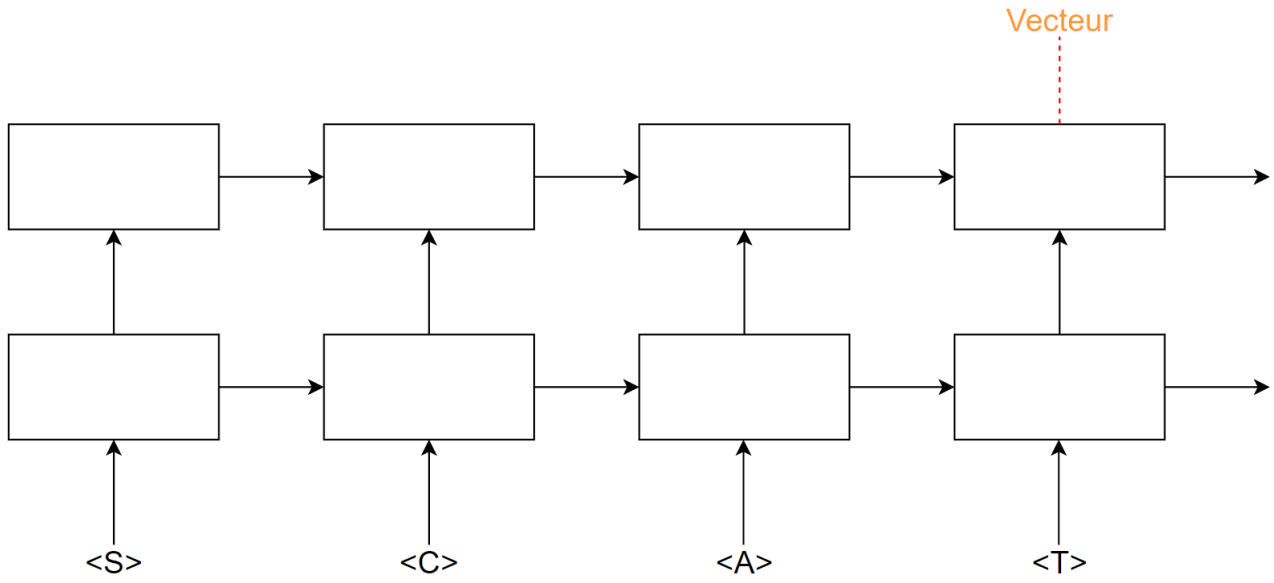

Figure 1: Exemple général d'un réseau de neurones

La figure ci-dessus nous montre un réseau de neurones en 2 couches. Nous avons en entrée une lettre donc $<S>$ pour dire que l'on commence le mot ensuite on a le mot lettre par lettre et à la fin ce mot est véctorisé, c'est-à-dire que pour qu'il soit compréhensible de l'ordinateur on va transformer notre chaîne de caractère en un vecteur.

Dans cette méthode, nous pouvons retrouver les approches: *Sequence to Sequence*, *Long-Short Term Memory Recurrent Neural Network*, *Transformer based conversion* et *Convolutional Neural Networks conversion*

2.3.1 Long-Short Terme Memory Recurrent Neural Network (LSTM RNN)

L'article [4] nous décrit la méthode des réseaux de neurones récurrents, et plus précisément le *Long Short-Term Memory*, ou LSTM.

Le principe des réseaux de neurones récurrents est la capacité d'utiliser à la fois le contexte gauche de l'input en plus de l'input lui-même. Pour cela, le système enregistre les étapes précédentes dans sa structure interne, puis construit une fenêtre dynamique contextuelle temporelle afin de les garder en mémoire. Cependant, il est possible que les valeurs associées à la récurrence s'effacent ou explosent, selon que les paramètres associés du réseau soient trop petits ou trop grands. Pour les longues séquences, cela peut être compliquées à gérer à cause de ces problèmes.
C'est là l'avantage du LSTM : il s'agit d'un réseau de neurones récurrents (il possède donc les capacités cités précédemment), mais il contient en plus des unités de mémoire dans la couche récurrente cachée. Celles-ci ont des *self-connections*, leur permettant d'enregistrer leur état à un instant t. De plus, à l'intérieur de ces unités, il y a des portes multiplicatives spéciales qui contrôlent le flux temporel d'entrée et de sortie et qui peuvent, au besoin, effacer leur état temporel.

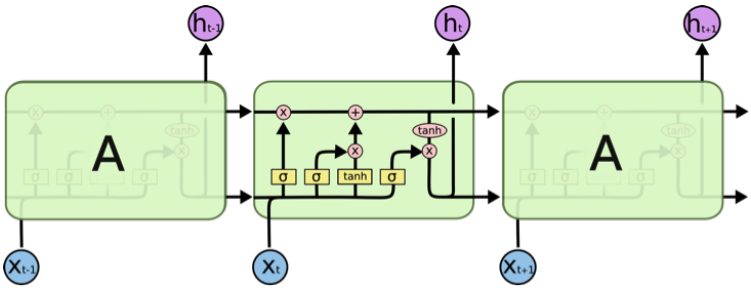

Figure 2: Schéma d'un système LSTM extrait de *Quantmetry* [5]

Afin de prouver que cette approche est utile, les auteurs ont réalisé une expérimentation sur deux types de LSTM : *UnidirectionalLSTM*, ou ULSTM, qui a une sortie retardé, et *DeepBidirectionalLSTM*, ou DBi-LSTM, qui a un étage de classification temporelle de liens en plus.

- ULSTM

Ce type de LSTM a une sortie retardée, ce qui permet de voir n contexte avant de commencer à associer les phonèmes aux graphèmes. Il contient 1024 unités de mémoires LSTM sur une couche avec une couche de sortie, avec l'activation softmax et la fonction *cross-entropy loss*. Pour l'expérience, il a été initialisé avec un poids aléatoire, entraîné avec un taux d'apprentissage de 0.002, et a été arrêté en fonction de la performance du modèle sur un ensemble de données en développement.

Différents délais de sortie ont été testés :

- Pas de délai : il s'agit de l'approche la plus simple, les phonèmes sont associés un par un aux graphèmes (avec l'utilisation d'un marqueur vide au besoin)
- Délai fixe : la séquence d'entrée est bloquée avec un délai fixe (par exemple, 2).
- Délai complet : la séquence d'entrée est entièrement vu et traitée avant d'associer les phonèmes. Pour cela, les auteurs ont utilisés un marqueur de fin.
- Bi-LSTM

Le Bi-LSTM voit le contexte entier d'office en plus de pouvoir lire l'entrée de droite à gauche et de gauche à droite. Il existe aussi le DBi-LSTM, qui est un Bi-LSTM avec plusieurs couches cachées, contrairement à ce dernier qui n'en contient qu'une. Le premier est plus performant que le second.

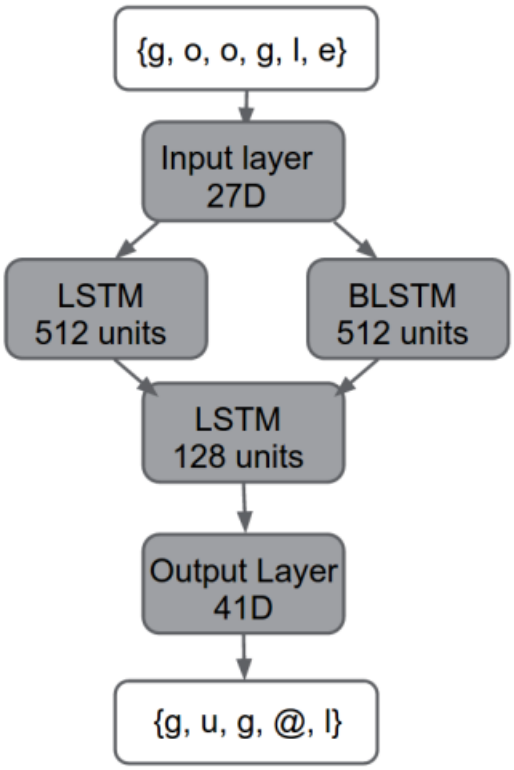

Figure 3: Schéma d'utilisation d'un modèle DBi-LSTM pour la conversion G2P dans une architecture en réseau de neurones [4]

Sur cette figure, la première case est une entrée, suivi d'une couche composée de deux éléments : une couche ULSTM et une couche Bi-LSTM. Celle-ci aboutit sur une couche ULSTM pour ensuite finir sur la couche de sortie.

Le DBi-LSTM maximise les probabilités d'avoir l'étiquetage correct grâce à sa couche softmax en sortie avec 41 unités. Il utilise en parallèle une couche de Classification Temporelle Connectioniste.

Les résultats de l'expérimentation montrent que le délai complet est le plus performant, et que le Bi-LSTM l'est plus comparé au LSTM et aux modèles n-gram (25,8% contre 30,1% et 26,5%). De plus, ceux-ci sont plus légers (3 et 11 MB) et plus rapides (12 et 64 ms/mot) que les modèles n-gram. En combinant ces deux méthodes (Bi-LSTM et les modèles n-gram), on atteint la meilleure performance (21,3%).

Pour conclure, le LSTM est une méthode ayant prouvé son efficacité, mais peut être amélioré en étant combiné avec d'autres méthodes déjà existantes.

2.3.2 Sequence-to-sequence

L'article [6] expose une façon de transformer les graphèmes en phonèmes via une approche statistique. On va utiliser un réseau de neurones récurrents appelé *Long Short Term Memory* (LSTM) ou réseau de neurones récurrents à mémoire court-terme et long terme en français car c'est une structure qui permet de palier au problème de perte de gradient[2]. Ils utilisent cette structure car elle ne nécessite pas d'alignements [3] entre le phonème et le graphème pour fonctionner. La structure qu'ils ont choisi ici est en 2 couches avec un encodeur et un décodeur. L'encodeur va lire une séquence en entrée et va ensuite la transformer en un vecteur, c'est-à-dire que prenons par exemple le mot *chat* on va parcourir le mot lettre par lettre et le vectoriser. Le décodeur va ensuite être utilisé pour prédire le phonème en fonction du graphèmes précèdent. La structure LSTM à besoin d'un alignement pour fonctionner ils ont donc testé 2 alignements pour voir lequel des deux fournit de meilleurs résultats pour la tâche de conversion graphème vers phonèmes. Le premier alignement est dit **unidirectionnel**, dans cet alignement la prédiction du phonème courant dépend de la prédiction du phonème et de la lettre que l'on avait avant. Le second alignement est dit **bidirectionnel**, dans ce processus, nous avons un réseau de neurones qui va lire de gauche à droite tandis que l'on va en avoir un autre qui lui va faire l'inverse et lire de droite a gauche. Ce qui veut dire que dans un modèle bidirectionnel la prédiction sur le phonème va dépendre

[2]Le gradient est un vecteur il a donc une direction et une magnitude.

[3]L'alignement est la correspondance entre un graphème et plusieurs phonèmes, ce qui veut dire qu'un graphème peut avoir entre 0, 1 ou plusieurs phonèmes associés.

de toute les lettres de la séquence. Les résultats sont meilleurs avec le modèle bidirectionnel qu'unidirectionnel car il permet d'éviter une ambiguïté sur le phonème qui va correspondre à un graphème donné.

2.3.3 Transformer based conversion

Cette méthode nous a été présentée dans l'article [7], qui décrit la conversion à base transformationnelle, composée de deux éléments :

- Un encodeur-décodeur
- Un mécanisme de l'attention

L'encodeur-décodeur consiste à avoir un encodeur qui convertit l'entrée en vecteur, et d'un décodeur qui génère la sortie grâce à la représentations de ce vecteur, apprise par le décodeur.

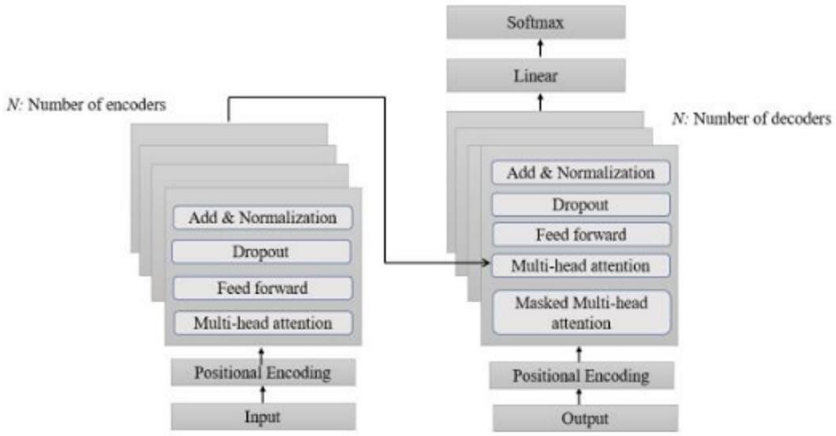

Figure 4: Schéma du fonctionnement pour un modèle en base transformationnelle [7]

Le mécanisme de l'attention a pour principe d'aligner et de traduire chaque mot avec un vecteur. Cela sert à cartographier la requête, à avoir un set composé de paires de clés-valeurs associées à une entrée. Le *Multi-head* construit un produit scalaire en calculant la requête, les clés et les valeurs. On peut avoir, grâce à lui, plusieurs couches parallèles. Celles-ci permettent au système d'apprendre les différentes représentations, de calculer leurs produits scalaires respectifs, et de concaténer les résultats avant de projeter cette concaténation sur l'étage *Feedforward*, pour la prédiction.

L'expérience réalisée dans l'article a été faite sur deux dictionnaires de prononciations : le NetTalk et le CMUDict. Les tokens <START> et <END> ont été utilisés pour le début et la fin des séquences de graphèmes et de phonèmes, et <PAD> pour compléter les parties vides des séquences plus courtes que la séquence la plus longue (fixée à 24). La meilleure performance a été celle du *Transformer 4x4* (quatre couches d'encodeur et quatre couches de décodeur). Pour le NetTalk, le PER est à 6,87% et le WER à 29,2%, pour une taille de 1,95M de paramètres pour le modèle. Pour le CMU, le PER est à 5,23% et le WER à 22,1%, pour une taille de 2,4M de paramètres pour le modèle.

Cette méthode est donc efficace pour la conversion des graphèmes en phonèmes.

2.3.4 Convolutional Neural Networks conversion (CNN)

Dans cette dernière partie nous retrouvons une étude récente sur la conversion G2P se basant sur les réseaux convolutionnels dit CNNs (pour *Convolutional Neural Networks*). L'article [8] propose plusieurs systèmes d'encodeurs-décodeurs basés sur diverses méthodes de conversion et reprenant en partie celles décrites dans les sections précédentes en mettant en évidence la pertinence des réseaux convolutionnels dans un tel système. Tout d'abord l'ensemble des modèles détaillés fonctionnent avec un système d'encodeur-décodeur comme ce qui suit :

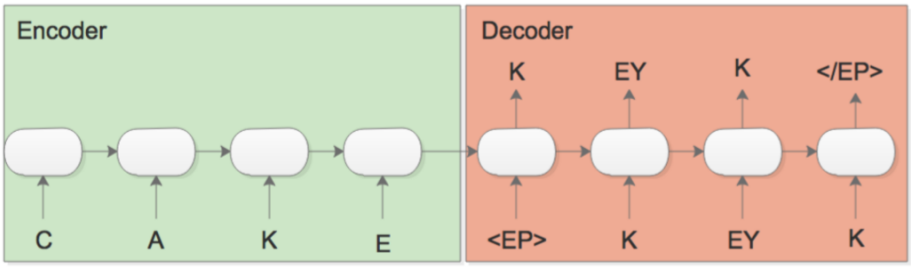

Figure 5: Système avec en entrée le graphème *cake* et en sortie sa forme phonétique [8]

L'encodeur prend une séquence de graphèmes en entrée (*input* en vert) et le décodeur donne en sortie (*output* en rouge) la séquence de phonèmes correspondante. Les chercheurs ont d'abord présenté un premier modèle utilisant un encodeur et un décodeur basés sur des cellules LSTM ; l'architecture est la même que dans la section 2.3.2. Le problème de ce modèle est que toutes les séquences d'entrées sont encodées dans un espace de taille fixe. Pour y remédier ils ont alors proposé un deuxième modèle cette fois-ci Bi-LSTM, qui possède donc deux couches LSTM unidirectionnelles (de gauche à droite et de droite à gauche) et permet

d'avoir en sortie une prédiction de phonèmes qui prend en compte les deux contextes.

Le troisième modèle implique cette fois un CNN en encodeur et un décodeur en Bi-LSTM. Dans l'encodeur on trouve 524 filtres (*filters*),de longueur 23 (*length*) et avec un pas de 1 (*stride*) ; le décodeur en Bi-LSTM contient 1024 cellules. La normalisation par lots (appelé aussi *batch normalization*) présenté sur la figure 5, correspond à le technique qui permet de réajuster le poids de certains nœuds du réseau de neurones après un certain nombre d'exemples (128 ici). C'est l'une des étapes qui permet d'améliorer le réseau, plutôt que de faire un réajustement après chaque itération qui risque de trop dérégler l'équilibre des poids en complément de prendre un temps considérable à traiter.

Le quatrième modèle quant à lui est un encodeur-décodeur exclusivement en CNN en plus de blocs appelés *residual blocks* :

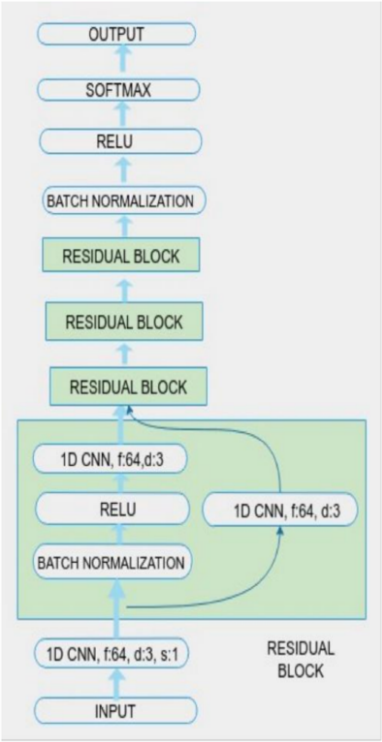

Figure 6: Conversion G2P basée sur des CNN avec blocs et connexions résiduelles (modèle 4) [8]

Les blocs présentés sur la figure 6 fonctionnent ainsi : si les cartes caractéristiques (*features maps*) ont la même taille, les connexions partagent les mêmes hyperparamètres, et chaque

fois que ce nombre est réduit de moitié, le nombre de filtres est doublé. Ceci permet d'avoir une meilleure efficacité tout en apportant un gain de temps de traitement. Il est constaté que le WER était bien plus grand sans connexions résiduelles qu'avec, ce qui prouve leur efficacité.

Ils ont également proposé un 5ème et dernier modèle combinant les modèles 3 et 4 avec un encodeur en CNN avec connexions résiduelles et la normalisation par lots (encodeur du modèle 4) et un décodeur en Bi-LSTM (décodeur du modèle 3):

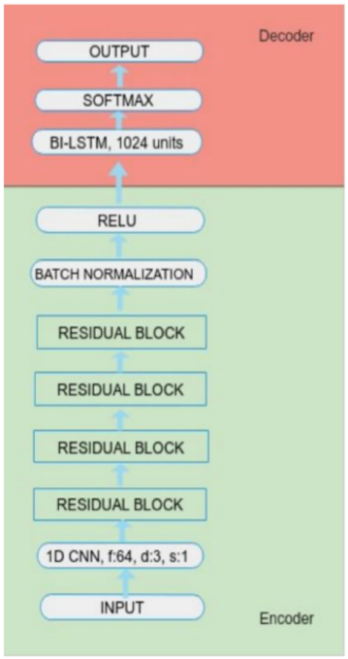

Figure 7: Encodeur en CNN et décodeur en Bi-LSTM (modèle 5) [8]

Pour les apprentissages des modèles et les tests ils ont utilisé les mêmes deux dictionnaires de prononciations : CMUDict et NetTalk. Voici un récapitulatif de l'ensembles de données utilisées pour leur étude :

	Taille pour l'apprentissage (nb de mots)	Taille pour les tests	Taille pour le développement	Nombre de graphèmes	Nombre de phonèmes
CMUDict	106 837	12 000	2 670	27	41
NetTalk	14 851	4 951	ND	26	52

Figure 8: Caractéristique des modèles

Pour chacun des modèles, la taille des couches dans l'encodeur est égale à la séquence d'entrée la plus grande et de même pour la sortie: la taille des couches dans le décodeur correspond à la séquence de phonèmes en sortie la plus grande.

Pour CMUDict on a par exemple un matrice de dimension 27 en entrée et de dimension 41 en sortie :

- la longueur de la plus grande séquence d'entrée (22) $*$ le nombre de graphèmes (27)
- la longueur de la plus grande séquence de sortie (22) $*$ le nombre de phonèmes (41)

Concernant NetTalk on a un matrice de dimension 26 en entrée et de dimension 52 en sortie :

- la longueur de la plus grande séquence d'entrée (19) $*$ le nombre de graphèmes (26)
- la longueur de la plus grande séquence de sortie (19) $*$ le nombre de phonèmes (52)

Les résultats finaux ont montré que le modèle 5 (figure 7) (encodeur CNN avec connexions résiduelles et décodeur en Bi-LSTM) avaient les meilleurs résultats avec un PER de 4.81% pour CMUDict et 5.69% pour NetTalk, et un WER de 25.13% pour CMUDict et 30.10% pour NetTalk. Cependant en terme de rapidité de traitement le modèle 4 est de loin le plus rapide, grâce à l'encodeur-décodeur en CNN, mais c'est celui aussi avec le taux d'erreur le plus important.

Ainsi, l'utilisation d'un modèle plus qu'un autre va dépendre de la priorité : une précision plus faible mais un temps d'exécution plus rapide ou bien un temps d'exécution plus long pour une meilleure précision en utilisant un encodeur en CNN et un décodeur en Bi-LSTM.

3 Conclusion

Nous avons présenté dans cette bibliographie une partie des méthodes existantes dans la conversion de graphèmes vers phonèmes. Les articles que nous avons étudiés exposent des approches à base de règles, de méthodes statistiques ou bien de réseaux de neurones afin d'effectuer ces conversions avec des temps d'exécution et des précisions variant d'une méthode à l'autre. Cependant, comme nous l'avons vu des erreurs de conversion demeurent, et cela peut poser problème dans les applications, dans la reconnaissance des variantes de prononciations, dans différentes langues, pour les noms communs et les noms de famille par exemple.

C'est là l'objectif de la deuxième partie du projet. Lors du prochain semestre nous allons étudier l'application de la procédure d'alignement des phonèmes pour détecter les variantes de prononciation incorrectes dans le lexique de prononciation généré automatiquement. La langue ciblée sera le français dans un premier temps avant de s'étendre vers l'anglais, voir l'allemand. Nous verrons comment utiliser des outils tels que ceux que nous avons présenté dans ce rapport bibliographique, et les évaluer sur des ensembles de données (une partie entraînement, une partie développement et une partie test). Il s'agira d'appliquer et adapter des outils d'alignement entre une séquence de phonèmes et une prononciation, et de vérifier par exemple si une absence d'alignement correspond à une mauvaise prononciation. Nous allons alors ajuster et ajouter des règles pour étudier leur impact et étendre ce processus à un large lexique de prononciations utilisé dans la reconnaissance vocale.

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," p. 4, 2011.

[4] 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.

[5] "Sous le capot d'un Chatbot : le cerveau de Skynet," May 2017. Section: Méthodologie Machine Learning.

[6] 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.

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

[8] S. Yolchuyeva, G. Németh, and B. Gyires-Tóth, "Grapheme-to-Phoneme Conversion with Convolutional Neural Networks," *Applied Sciences*, vol. 9, p. 1143, Jan. 2019. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.