Stage de Master Recherche

Rapport bibliographique

Big deep voice : indexation de données massives de parole grâce à des réseaux de neurones profonds

Domaines: Calcul et langage, Informatique neuronale et évolutionnaire

Auteur:
Antoine Perquin

Superviseurs:
Gwénolé Lecorvé
Damien Lolive
Expression - IRISA

ENSSAT
LANNION

Abstract:

Les systèmes de synthèse de parole sont des outils permettant de générer un signal de parole correspondant à un texte. Les solutions actuelles fournissent un signal de qualité, mais la parole générée est peu expressive, notamment en raison de leur jeu de données limité. Pour résoudre ce problème, il est donc nécessaire d'augmenter la quantité de données pour y intégrer de la variabilité. Cela soulève néanmoins deux nouvelles problématiques : « Comment décrire et comparer les données ? » et « Comment indexer et rechercher des données lorsque leurs descripteurs sont de grande taille et que le nombre de données est conséquent ? ». Le but de ce stage est de mettre en place une mesure de similarité entre phonèmes, puis éventuellement un algorithme de recherche de plus proches voisins, qui pourront être intégrés au système de synthèse de parole de l'équipe Expression. Dans cette optique, la problématique de description et comparaison des données est adressée grâce à la propriété de plongement des réseaux de neurones. On propose de répondre à la problématique de recherche de données grâce à une méthode de recherche de plus proches voisins en grande dimension.

Table des matières

Introduction

La synthèse de parole à partir de texte est un domaine qui possède de nombreuses applications. Actuellement, elle permet par exemple de réaliser des interactions humain-machine, notamment sur des serveurs de service clientèle. Elle permet aussi de prêter assistance aux personnes handicapées en permettant la lecture sur ordinateur pour les personnes malvoyantes ou l'expression orale pour les personnes atteintes de troubles de la parole. La synthèse de parole étant un sujet de recherche depuis le XVIIIème siècle, de nombreuses solutions existent déjà et permettent d'obtenir une parole compréhensible. Cependant, l'expressivité des signaux produits n'est toujours pas suffisante pour des applications plus ambitieuses telles que le doublage de films ou la lecture d'*audiobooks*. Ainsi, l'ensemble des recherches actuelles cherchent à améliorer les techniques déjà existantes afin d'obtenir des résultats plus expressifs, c'est-à-dire de la parole capable d'exprimer des émotions telles que la joie ou la colère, mais aussi capable d'imiter un accent ou d'adopter un style d'élocution particulier. Pour cela, de nombreux travaux s'appuient sur les avancées récentes dans le domaine de l'apprentissage automatique.

Les solutions commerciales actuelles en synthèse de la parole à partir de texte s'effectuent majoritairement par concaténation d'unités. Si ces solutions produisent des signaux de bonne qualité, leur expressivité est limitée à celle exprimée dans la parole constituant leur base de données. Ces solutions nécessitent de plus une expertise en linguistique lors de leur élaboration. Pour pallier ces limitations, les recherches se sont portées sur l'utilisation de méthodes statistiques d'apprentissage automatique, principalement les modèles de Markov cachés et les réseaux de neurones. Ces solutions produisent des signaux compréhensibles et relativement expressifs mais qui semblent souvent étouffés. Une tendance actuelle consiste alors à se tourner vers des solutions hybrides comme, par exemple, une sélection d'unités guidée par une méthode statistique. Ce type d'approche est précisément celui qui nous intéresse dans notre travail. Plus précisément, notre objectif est de combiner l'utilisation de réseaux de neurones profonds et d'un système par sélection d'unités pour pouvoir exploiter des corpus de parole de très grande taille, capables de contenir une grande variété d'expressivités. Dans ce document, nous en donnons les problématiques et abordons les différentes solutions proposées dans la littérature.

La force d'une méthode hybride tient dans l'intégration de découvertes en matière de réseaux de neurones. Notamment, la propriété de plongement des réseaux de neurones, ou *embeddings*, terme que l'on utilisera tant il est admis dans la communauté francophone, permet d'obtenir une représentation alternative compacte des données fournies en entrée d'un réseau.

L'introduction d'expressivité et de variabilité en synthèse de la parole implique de posséder de grandes bases de données de parole dont la description se fait à l'aide de vecteurs de grande dimension. Alors, les méthodes de recherches fondées sur des index classiques ne suffisent plus et il convient d'utiliser des techniques adaptées à la recherche de plus proches voisins dans des espaces de grande dimension.

Dans ce document, nous commençons par brosser un historique des évolutions récentes en terme de synthèse vocale, y compris les méthodes hybrides utilisant des réseaux de neurones profonds. Nous étudions ensuite dans les deux sections suivantes les propriétés de ces réseaux, puis les techniques de recherche de plus proches voisins dans un espace de grande dimension.

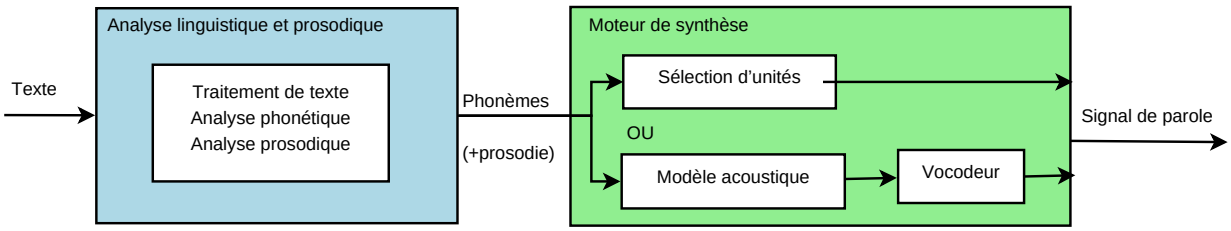

FIGURE 1 – Schéma d'un système de synthèse

1 Synthèse de la parole

De manière générale, un système de synthèse de parole se décompose en deux parties (*cf.* figure 1). La première se charge d'analyser et de traiter le texte en entrée du système, analyse linguistique et prosodique sur le schéma. La deuxième partie réalise concrètement, à partir des informations issues de l'analyse en amont, grâce à un moteur de synthèse, la génération du signal de parole à partir de texte. Cette section a pour but de définir ce qu'est un système de synthèse ainsi que de présenter les différentes solutions actuellement existantes en matière de moteur de synthèse, introduction nécessaire aux sections 2 et 3, plus centrales en terme de problématique. Nous commencerons par présenter le fonctionnement de l'analyse linguistique avant d'examiner différentes solutions de moteurs de synthèse.

1.1 Analyse linguistique et prosodique

Un moteur de synthèse ne peut pas prendre directement du texte en entrée, il nécessite une liste de phonèmes accompagnés éventuellement de consignes telles que la prosodie afin de réaliser la synthèse de parole. Il est donc nécessaire d'effectuer une étape préalable de conversion texte vers phonèmes, c'est le rôle de l'analyse linguistique et prosodique. L'analyse fonctionne en trois étapes ([1], chap. 19) : le traitement du texte, l'analyse phonétique et l'analyse prosodique. Contrairement au moteur de synthèse, l'analyse est dépendante de la langue visée par le système de synthèse.

Lors du traitement du texte, celui-ci est d'abord analysé afin d'en identifier la structure : la ponctuation est considérée et le texte est séparé en phrases ou paragraphes. Il est ensuite normalisé, afin de tenir compte des abréviations et des acronymes. Par exemple, en anglais, *St.* peut être interprété *Street* ou *Saint*. Une étape de balisage du texte peut être effectuée afin de modifier la manière dont une séquence doit être interprétée : une suite de nombre séparés par des points peut être un numéro de téléphone ou une adresse IP. Enfin, la dernière étape du traitement correspond à effectuer une analyse syntaxique et linguistique qui permet de différencier les homonymes comme "parent" qui peut être lu comme un nom ou un verbe conjugué. Une fois le traitement du texte terminé, chaque mot est identifié. On peut alors procéder à l'étape d'analyse phonétique qui permet de transcrire chaque mot en une séquence de phonèmes. Le plus souvent, cette étape est effectuée en parcourant des dictionnaires de prononciation. Enfin, l'analyse prosodique détermine l'intonation de la parole à générer : intensité, hauteur et durée de chaque phonème. Cette analyse peut par exemple aider à distinguer à l'écoute une question d'une affirmation.

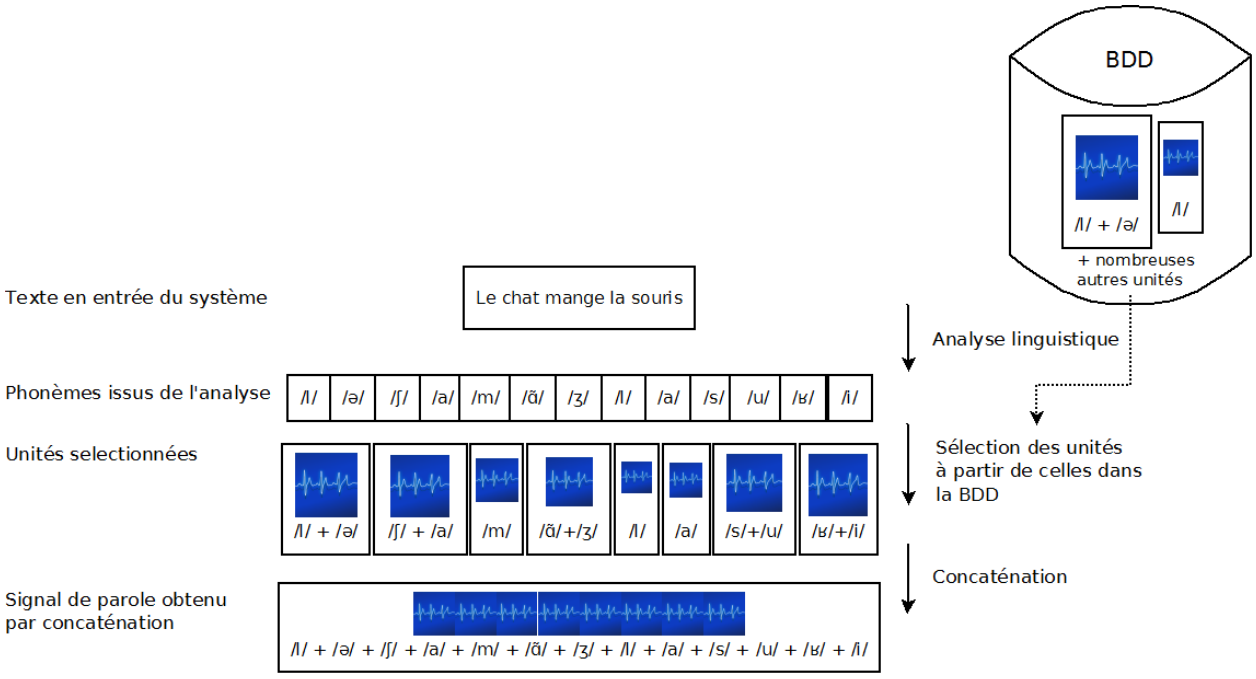

FIGURE 2 – Schéma d'une concaténation d'unités

Une fois prédits les phonèmes et la manière dont ils doivent être prononcés, le moteur de synthèse cherche à produire le signal de parole correspondant. Nous allons approfondir ici cette partie en examinant les solutions dites par sélection d'unités et celles fondés sur des modèles statistiques avant d'examiner les solutions hybrides.

1.2 Concaténation d'unités

Comme présenté sur la figure 2, les moteurs de synthèse par concaténation d'unités (partie haute du moteur de synthèse sur la figure 1) consistent à mettre bout à bout des morceaux de parole afin de former la phrase souhaitée. Plus précisément, on suppose disposer d'une base de données contenant des unités de parole prédécoupées. Ces unités peuvent être de différentes tailles : phonèmes, diphones [1], syllabes, mots ou groupes de mots.

Historiquement, les premiers systèmes par concaténation d'unités étaient ceux par diphones. Pour ces systèmes, les seules unités à disposition dans la base de données sont des diphones enregistrés séparément, parcourant l'ensemble des diphones d'une langue mais chacun en un unique exemplaire. S'en tenir à la concaténation pure de diphones donne cependant des résultats non naturels. On peut alors procéder à des transformations du signal, par exemple *via* l'algorithme TD-PSOLA afin de modifier les paramètres prosodiques [2]. Bien que les systèmes de concaténation par diphones présentent l'avantage d'être légers (l'ensemble des unités correspond à seulement quelques minutes d'enregistrement), la parole synthétisée manque toujours de naturel et d'expressivité, ce qui a conduit à la construction de moteurs plus élaborés dits par sélection d'unités.

1. Unité commençant au milieu d'un phonème et s'arrêtant au milieu du phonème suivant.

L'idée de la synthèse par sélection d'unités [3] est de ne pas limiter la base de données à une seule occurrence de chaque diphones. La base correspond alors à un corpus composé de nombreuses phrases dont le flux de parole a été segmenté en unités de tailles différentes. Par opposition avec les moteurs par concaténation de diphones, il y a plusieurs occurrences de chacun d'entre eux dans la base et chaque occurrence correspond à un contexte de prononciation différent, par exemple avec une prosodie et un placement différent dans la phrase. De plus, la base peut contenir des unités de tailles différentes comme des moitiés de phones ou des mots entiers. Il ne convient plus alors simplement de sélectionner le diphone demandé, mais de choisir l'unité ou l'ensemble d'unités qui donne le meilleur résultat. En effet, la synthèse du mot "chat" peut s'effectuer directement à partir de la sélection du mot lui-même s'il est présent dans la base ou à partir des phonèmes /ʃ/ et /a/ si seules ces unités sont présentes. Cependant, si la base est lacunaire, il faudra parfois se rabattre sur le phonème /ɑ/ plutôt que /a/. La sélection d'unités revient donc à résoudre un problème d'optimisation exprimé comme la somme de deux coûts et la séquence minimisant cette somme est la séquence d'unités retenues dont la concaténation donne le résultat en sortie du système de synthèse. D'une part, un coût cible permet de choisir les unités les plus proches de celles décrites par l'analyse linguistique. Ce coût est en général exprimé comme la somme pondérée des différences entre les attributs des unités sélectionnées et celles demandées, les attributs ayant été sélectionnés par des linguistes. D'autre part, un coût de jointure attribue un score lié à la concaténation d'unités qui permet de caractériser si les unités se concatènent bien entre elles sans donner d'artefacts. Ce coût est en général exprimé comme la somme pondérée des différences entre les attributs acoustiques des unités sélectionnées et celles demandées.

Au final, les systèmes de synthèse fondés sur la sélection d'unités permettent d'obtenir une parole synthétisée de qualité et expressive relativement à leur jeu de données, ce qui explique leur utilisation dans la plupart des solutions commerciales actuelles. Cependant, le coût cible est difficile à exprimer, nécessitant l'aide de linguistes pour prioriser l'importance des descripteurs et les performances de ces systèmes diminuent lorsque qu'aucune unité proche de celle cherchée ne se trouve dans la base. Une solution potentielle à ces problèmes se trouve dans l'utilisation de modèles statistiques paramétriques.

1.3 Synthèse statistique paramétrique

Comme présenté sur la figure 1, la synthèse statistique paramétrique consiste à utiliser un modèle acoustique pour prédire, à partir des phonèmes produits par l'analyse linguistique, des coefficients acoustiques qui serviront d'entrée à un vocodeur afin de générer un signal de parole. Pour apprendre un tel modèle, il faut disposer d'un large corpus de parole et de sa transcription phonétique. La transformation du corpus sous forme de coefficients acoustiques permet alors l'apprentissage d'un modèle liant phonèmes et coefficients acoustiques. Ce modèle peut prendre la forme d'un modèle de Markov caché ou d'un réseau de neurone.

Si l'on considère la parole comme une suite d'observations acoustiques correspondant aux phonèmes réalisés, le modèle acoustique utilisé peut être vu comme un modèle de Markov caché (*Hidden Markov Model*, HMM). On parle alors de système HTS [4]. En réalité, ce HMM est construit dynamiquement, à la volée, à chaque utilisation du système de synthèse. Par ailleurs, ce HMM est utilisé de manière non usuelle : on l'utilise pour prédire la séquence d'observations acoustiques correspondant aux phonèmes (et non l'inverse comme d'ordinaire). Pour une phrase à synthétiser, chaque

phonème est modélisé par trois états, chacun étant automatiquement sélectionné à partir d'un ensemble d'états préalablement entraînés. Une fois le modèle construit, les coefficients sont générés trame par trame en cherchant à maximiser la probabilité de la séquence d'états.

Lorsque le modèle acoustique d'un système de synthèse est un réseau de neurone profond (*Deep Neural Network*, DNN), l'entrée du DNN correspond au phonème à synthétiser et à d'autres éventuelles informations générées par l'analyse linguistique. Sa sortie correspond aux paramètres attendus par le vocodeur. La durée de chaque phonème peut être calculée en rajoutant une sortie au DNN de base, ce qui revient à entraîner un DNN multi-tâches dont la tâche principale est de prédire les paramètres acoustiques et la tâche secondaire de prédire la durée du phonème. Les modèles acoustiques fondés sur des DNN s'avèrent souvent plus efficaces que les solutions utilisant des HMM [5]. Certaines extensions de ce principe proposent même de se passer de vocodeur pour prédire directement la frame d'onde du signal [6].

Les systèmes statistiques paramétriques permettent en général de synthétiser une voix relativement expressive, même sur des bases de données plus petites qu'en sélection d'unités. Malheureusement, l'utilisation d'un vocodeur produit parfois un son étouffé. Les systèmes hybrides proposent des éléments de solutions intéressants pour concilier les intérêts de chaque approche.

1.4 Systèmes hybrides

Les solutions dites hybrides sont des moteurs de synthèse n'étant ni purement par concaténation ni purement statistique paramétrique. Par exemple, une méthode par concaténation où une solution de construction des unités manquantes dans la base de données est mise en place [7] est considérée hybride. La solution hybride proposée dans [8] consiste à faire une sélection d'unités guidée par un réseau de neurones. Plus précisément, un DNN est entraîné en tant que modèle acoustique et est utilisé pour définir le coût cible d'un système par sélection d'unités en comparant les attributs prédits par le DNN à ceux des unités présentent dans la base de données. Les résultats des tests d'écoute montrent que cette approche est meilleure qu'une méthode statistique pure. La même expérience effectuée en remplaçant le DNN par un HMM ne permet pas de montrer une amélioration significative de l'approche par DNN par rapport à celle par HMM. On peut cependant penser que ces résultats sont améliorables en utilisant d'autres types de DNN ou en tirant mieux parti de ses propriétés d'*embeddings*, comme nous chercherons à le montrer dans la section suivante.

Ainsi, dans le domaine de la synthèse de parole, deux types de méthodes se distinguent, celles par sélection d'unités et celles statistiques paramétriques. Les solutions par sélection d'unités sont celles qui permettent actuellement d'offrir la meilleure qualité. Cependant, cette méthode offre une expressivité limitée à celle contenue dans son jeu de données. Augmenter l'expressivité d'une solution par concaténation nécessite donc d'augmenter la quantité et la variabilité des données, ce qui pose deux problèmes majeurs. Il faut tout d'abord être capable de correctement décrire et comparer les unités de parole, ce qui est rendu possible par les *embeddings* présentés dans la section 2. Il faut de plus, dans le cadre d'une application interactive telle que la synthèse de parole, être algorithmiquement efficace : des algorithmes de recherche de plus proches voisins sont décrits dans la section 3.

2 Réseaux de neurones

Les réseaux de neurones sont une technique d'apprentissage automatique qui permet entre autre de faire de l'apprentissage supervisé, mais aussi de produire des représentations continues, souvent compactes, de données fournies en entrée. Après un rappel sur les réseaux de neurones, ce sont ces représentations, appelées *embeddings*, qui feront l'objet d'une étude particulière. Nous concluons enfin cette section par l'étude d'une technique qui permet d'améliorer les performances d'un DNN, l'apprentissage multi-tâches.

2.1 Généralités

Les réseaux de neurones reposent sur le modèle du perceptron. Un perceptron est un neurone possédant plusieurs entrées x_i auxquelles sont associées un coefficient w_i, une sortie s et une fonction d'activation f à laquelle est associée un coefficient w_0. La sortie est prédite comme la somme des entrées pondérée par leur coefficient, à laquelle la fonction d'activation est appliquée en prenant en compte le décalage introduit par son propre coefficient : $s = f(\sum_i w_i x_i - w_0)$.

Un apprentissage comparant les valeurs prédites par le réseau et les valeurs attendues permet de faire converger les coefficients w_i vers des valeurs permettant une prédiction plus ou moins précise de la sortie en fonction du jeu de données utilisé. On peut alors introduire la notion de couche : plusieurs perceptrons partageant les mêmes entrées et la même fonction d'activation mais possédant des coefficients et sorties distincts. On peut alors considérer une couche de perceptrons sous une représentation vectorielle dont chaque composante correspond à une sortie : $s_i = f(\boldsymbol{w_i}^T \boldsymbol{x} - w_{i0})$ où $\boldsymbol{w_i}$ est le vecteur représentant les coefficients permettant de prédire la sortie i et $\boldsymbol{x}$ est le vecteur représentant les entrées de la couche.

Il est ensuite possible de construire des réseaux de neurones possédant plusieurs couches, on parle alors de DNN. Dans ces types de réseaux, les sorties d'une couche cachée correspondent aux entrées de la couche suivante. Une couche peut être entièrement connectée à la précédente, on parle alors de couche dense, ou partiellement connectée, on parle de couche convolutionnelle. Dans un cas comme dans l'autre le principe reste le même : les sorties correspondent à une combinaison linéaire des entrées. En général, la dimension des couches cachées[2] d'un DNN est plus petite que celle de ses entrées, chaque couche possède ainsi une représentation différente de l'entrée. Ce sont ces représentations alternatives que l'on appelle *embeddings*.

2.2 Embeddings

Calculer un *embedding* correspond à entraîner un DNN pour une tâche particulière dans le but d'utiliser non pas la sortie prédite par le réseau mais les valeurs en sortie de l'une des couches cachées. Les *embeddings* sont souvent utilisés afin d'encoder un contexte large à l'aide d'un vecteur de plus petite dimension. Par exemple, dans [9] les auteurs introduisent le modèle *skip-gram*. Représenté sur la figure 3, ce modèle est un réseau de neurones entraîné à prédire, à partir de la représentation vectorielle d'un mot en entrée, la représentation des mots l'entourant dans la phrase d'origine : ceux le précédant et ceux le suivant. Une fois entraîné, un tel modèle permet de contenir la représentation

2. Couche d'un réseau de neurone située entre l'entrée et la sortie.

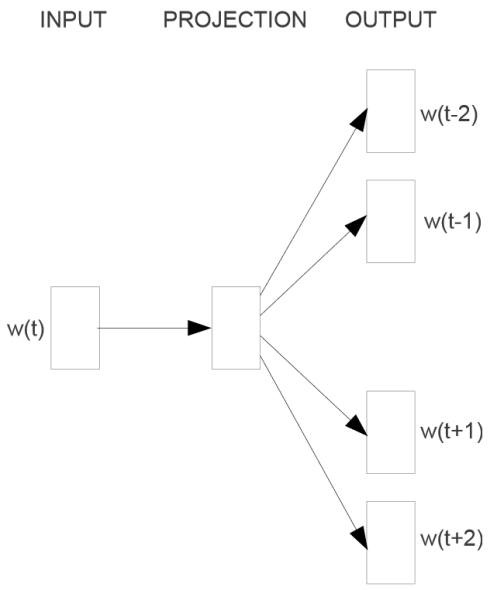

FIGURE 3 – Schéma d'un skip-gram
(Source : [9])

condensée des mots en entourant un autre. De manière inverse [9], on peut tenter de prédire un mot à partir de ceux environnants c'est ce qui s'appelle un CBOW (*Continuous Bag of Words*).

Sous cette forme, on peut considérer que les *embeddings* sont une forme de projection d'un espace vectoriel de grande taille vers un espace vectoriel de taille inférieure. Dans [10], les auteurs mettent en avant un autre aspect intéressant des *embeddings* : ils permettent de conserver les similarités entre objets dans cet espace projeté et les projections de deux objets partagent les mêmes liens que les deux objets originaux. Par exemple, si l'on examine les représentations vectorielles issues de l'*embedding* de noms de différents pays par le *skip-gram*, on observe que ces représentations sont réparties selon un axe. La même expérience effectuée sur le nom de différentes capitales répartit ces dernières sur un même axe parallèle à celui des pays. On remarque de plus que le vecteur reliant l'*embedding* d'un pays à celui de sa capitale est quasi-égal à celui reliant l'*embedding* d'un autre pays à celui de sa propre capitale. Dans le même article, les auteurs proposent de définir des opérations sur les *embeddings* de mots en tant que vecteurs afin de répondre à des requêtes analogiques du type « Quel est le mot qui se rapproche de 'France' de la même manière que 'Madrid' est proche d' 'Espagne' ? » et montrent que les *embeddings* de mots issus de *skip-gram* permettent de répondre 'Paris'.

Les *embeddings* ne sont pas uniquement utilisés pour la représentation de mots mais aussi en traitement d'image ou pour améliorer la reconnaissance de phonèmes. Dans [11], la reconnaissance d'action appliquée à la vidéo nécessite l'apprentissage d'attributs spatio-temporels caractérisant la vidéo avant d'utiliser un classifieur. L'apprentissage d'*embeddings* permet ici d'apprendre des attributs sans nécessiter d'expertise quant au choix de ces attributs. Dans [12], l'auteur tente d'améliorer la reconnaissance de phonèmes grâce à l'utilisation d'*embeddings*. Ceux-ci sont issus d'un réseau en-

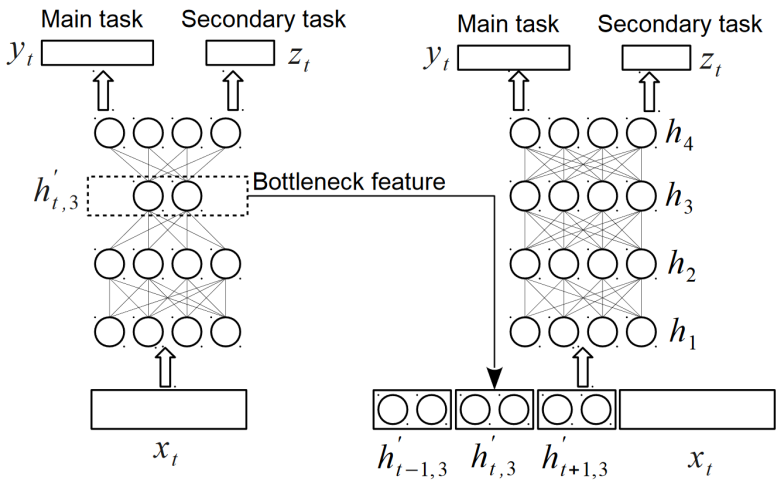

FIGURE 4 – Schéma d'un DNN liant multi-tâches et *embeddings*
(Source : [14])

traîné à prédire la description acoustique d'un phonème à partir de son contexte. Les *embeddings* servent ici à condenser le contexte d'un phonème qui va servir d'entrée à un second DNN. Il est intéressant de noter que comme dans cet article, l'apprentissage d'*embeddings* peut être utilisé conjointement avec les autres techniques usuelles en construction de DNN, et notamment l'apprentissage multi-tâches.

2.3 Apprentissage multi-tâches

Tout réseau de neurone est entraîné dans un but particulier. L'idée de l'apprentissage multi-tâches est d'ajouter une tâche au réseau de neurone lors de l'apprentissage. On se rend alors compte que la précision des prédictions du DNN augmente généralement. En effet, ajouter une autre tâche permet au DNN de généraliser son analyse des données d'entrée en apprenant des caractéristiques liées à la seconde tâche. Il est intéressant de remarquer que différentes tâches secondaires n'apportent pas le même gain de performances.

Par exemple, dans [13] différentes tâches secondaires sont proposées afin d'améliorer la reconnaissance de phonème. Un DNN est entraîné comme modèle acoustique en tant que tâche principale et les différentes tâches secondaires testées sont les suivantes : prédire les états acoustiques (comparés avec les résultats d'un k-means) ou prédire le contexte de la trame en cours de traitement (trame précédente et suivante). Cette étude montre que la tâche secondaire doit être choisie de manière à être liée à la tâche principale tout en apportant des informations sur la structure du problème, ici la dépendance de contexte.

Dans le domaine de la synthèse de la parole, [14] montre que l'apprentissage multi-tâches permet d'apprendre un modèle acoustique plus performant qu'un modèle fondé sur un HMM ou un DNN simple. Deux réseaux multi-tâches sont entraînés (*cf.* figure 4). Leur tâche principale est de prédire les coefficients acoustiques des descripteurs de phonèmes en entrée et la tâche secondaire varie selon les expériences. Cette étude observe aussi les améliorations apportées par l'utilisation ou non dans le deuxième réseau d'un contexte sous forme d'*embeddings*, ces derniers étant issus du premier

réseau. Des mesures objectives sur les coefficients acoustiques prédits montrent que l'utilisation d'apprentissage multi-tâches améliorent le taux d'erreur et que cette amélioration est renforcée par l'utilisation de contexte sous forme d'*embeddings*. En revanche, les auteurs n'ont pas réussi à montrer d'amélioration significative lors des tests d'écoute.

Au final, grâce à un DNN entraîné pour calculer des *embeddings*, on peut obtenir une projection d'un ensemble d'unités de parole dans un même espace vectoriel. Si l'utilisation d'apprentissage multi-tâches permet d'augmenter la performance du réseau, on peut espérer que les *embeddings* issus de ce réseau soient aussi de meilleure qualité. Dans l'optique d'une synthèse par sélection d'unités, il se pose alors la question de la méthode pour sélectionner une unité voulue dans ce nouvel espace projeté. Pour cela, les techniques d'indexation sont une piste de solution, et même plus particulièrement les techniques de recherche de plus proches voisins en grande dimension. C'est précisément l'objet de notre prochaine section.

3 Recherche des plus proches voisins dans un espace de grande dimension

La recherche de plus proches voisins consiste à rechercher une donnée particulière dans une base. Si celle-ci existe, on doit être capable de la trouver, sinon on doit être capable de trouver la ou les donnée(s) s'en rapprochant le plus. Ce point est intéressant vis-à-vis de la sélection d'unités où cette recherche permettrait de trouver les unités semblables à celles demandées par l'analyse linguistique. Le fait de considérer plusieurs voisins permet alors de choisir parmi toutes les combinaisons possibles à l'échelle de la phrase celle où les unités se concatènent le mieux.

Si une recherche exhaustive de plus proches voisins fonctionne, celle-ci est loin d'être optimale et le recours à des techniques d'indexation s'avère plus efficace. Pour obtenir une voix expressive en synthèse de la parole, la quantité de données stockées et la grande dimension de leurs descripteurs poussent à explorer de nouvelles solutions de recherche, notamment dans le domaine de la recherche d'informations. La plupart des solutions existantes peuvent être réparties en deux catégories : les solutions exactes et les solutions approximatives qui n'assurent pas que les points retournés fassent partis des plus proches voisins du point cherché. Une autre distinction possible consiste à opposer les arbres de partitions et les techniques à base de hachage.

3.1 Arbres de partitions : kd-tree

Les kd-tree permettent de partitionner spatialement les points d'un espace vectoriel en s'inspirant d'un arbre de recherche binaire [15]. Dans le cas d'un arbre de recherche binaire, on sépare un ensemble de valeurs de dimension 1 en construisant un nœud dont la valeur est la médiane de l'ensemble, son fils gauche correspond à l'ensemble des valeurs inférieures à la médiane et le fils droit correspond à l'ensemble des valeurs supérieures à la médiane. La construction est effectuée de manière récursive sur les fils gauche et droit avec l'ensemble de valeurs associées restantes. On arrête la récursivité quand le nombre de valeurs associées à un nœud est suffisamment petit. Si une valeur est seule, on la stocke telle quelle, sinon on stocke l'ensemble des valeurs sous la forme d'une liste chaînée. L'ensemble des valeurs sont donc stockées dans les feuilles, aucune ne se trouve dans un nœud interne.

Les kd-tree, quant à eux, étendent ce principe à des ensembles de dimension k. Il faut alors pour chaque nœud choisir selon quelle dimension partitionner l'espace. En général, on choisit la dimension selon laquelle la variance est maximale. De manière similaire aux arbres de recherches binaires, on divise l'espace en séparant les points dont la composante est inférieure à la médiane de ceux dont la composante est supérieure à la médiane. On recommence récursivement sur les deux sous-ensembles jusqu'à avoir des ensembles suffisamment petits. La recherche de plus proches voisins dans un kd-tree revient alors à comparer la i-ème composante du point recherché avec la valeur du nœud où i est la composante discriminante choisie lors de la construction. Si la composante est inférieure, on examine récursivement le fils gauche, sinon on examine récursivement le fils droit. Les résultats de la recherche correspondent aux points trouvés au niveau des feuilles.

Malheureusement, les kd-tree ont une complexité linéaire en fonction de la dimension et du nombre de données pour la recherche. Ils ne sont donc pas adaptés aux espaces de grande dimension. Il convient donc de les adapter, quitte à n'obtenir qu'une solution approximative. Nous verrons une adaptation dans la section 3.3, mais commençons par examiner les techniques fondées sur le hachage.

3.2 Locality Sensitive Hashing (LSH)

La méthode LSH se base sur le principe de projections depuis l'espace de recherche vers un espace de dimension réduite en remarquant que deux points proches dans l'espace original ont une forte probabilité de l'être aussi dans l'espace projeté et, de manière similaire, deux points éloignés dans l'espace d'origine ont une forte probabilité d'être éloignés dans l'espace de projection.

Plus précisément, selon [16], on souhaite avoir une famille de fonctions de hachages H telles que pour $h \in H$, on puisse trouver un rayon R, un coefficient c et deux probabilités constantes P_1 et P_2 telles que pour tous points p et q de l'espace de recherche on ait les deux propriétés suivantes :

$$\|p - q\| \leq R \Rightarrow Pr(h(q) = h(p)) \geq P_1 \tag{1}$$

$$\|p - q\| \geq cR \Rightarrow Pr(h(q) = h(p)) \leq P_2 \tag{2}$$

L'algorithme LSH effectue le hachage sur la concaténation de L fonctions de hachages de H : $g_j(q) = (h_{1,j}(q), ..., h_{k,j}(q)), j \in 1, .., L$ où k est un paramètre et q un point de l'espace vectoriel. Alors, on construit L tables de hachages à partir de ces fonctions et on y répartit l'ensemble des points du jeu de données. La recherche de plus proches voisins d'un point q avec LSH se fait en examinant les cases des L tables ayant la même valeur de hachage que q. On examine la distance entre les points retournés par cette méthode et le point cherché. Si la distance est satisfaisante, on considère que le point trouvé fait partie des plus proches voisins.

L'algorithme LSH présente l'avantage de s'exécuter rapidement et de prendre peu de place en mémoire (complexité locale $O(nL)$ où n représente le nombre de données dans la base). Cependant, LSH effectue une recherche approximative et de nombreux paramètres sont à régler et influent énormément sur les performances de l'algorithme. De plus, il faut être capable de définir une distance entre les points de l'espace vectoriel.

3.3 Autres méthodes de recherche approximative

La méthode fondée sur les kd-tree vue précédemment permettait d'effectuer une recherche exacte de plus proches voisins. Cependant, quand la dimension de l'espace de recherche et le nombre de données augmente, le temps d'exécution d'un tel algorithme devient vite incompatible avec une application interactive telle que la synthèse de la parole. Dans [17], les auteurs remarquent que se limiter à une recherche approximative des plus proches voisins permet de retourner 90% de plus proches voisins corrects en obtenant des résultats au moins deux fois plus rapidement. Les auteurs proposent deux méthodes approximatives fondées sur la partition de l'espace de recherche : la construction de kd-tree aléatoires et la construction d'arbres k-means hiérarchiques.

Le premier algorithme consiste à construire plusieurs kd-tree de manière aléatoire. Lors du choix de la composante qui permet de séparer l'espace, plutôt que de choisir la dimension ayant la plus grande variance, la composante est choisie aléatoirement parmi les 5 dimensions présentant la plus grande variance. On obtient alors une forêt de kd-tree. Les différents arbres de la forêt sont explorés parallèlement avec une file de priorité partagée. La file est triée par distance croissante entre la frontière représentée par le nœud en cours d'analyse et le point cherché. Ainsi, la recherche examinera en priorité les feuilles les plus proches du point recherché. Enfin, dès qu'un point est examiné par la recherche dans un arbre, il est marqué comme tel afin de ne pas être à nouveau examiné dans les autres arbres. L'algorithme se termine quand un certain nombre de feuilles ont été parcourues et renvoie les voisins trouvés jusqu'à présent.

Le second algorithme consiste à appliquer un k-means sur l'ensemble du jeu de données, puis continuer récursivement à appliquer un k-means sur chaque cluster obtenu dans l'étape précédente jusqu'à obtenir un certain nombre de points par cluster. On obtient alors une partition de l'espace sous la forme d'un arbre dont les fils de chaque nœud sont les clusters obtenus par application du k-means. Lors de la recherche de plus proches voisins, on parcourt l'arbre en prenant à chaque étape la branche dont le centre de cluster correspondant est le plus proche du point cherché.

Au final, les techniques de recherche de plus proches voisins vues ici imposent de faire un compromis. Si les kd-tree peuvent proposer une solution efficace, cette méthode est sujette au fléau de la dimension. Certaines adaptations permettent aux kd-tree d'être plus efficaces dans un espace de haute dimension. Dans la catégorie des recherches approximatives, la solution LSH semble aussi être efficace mais nécessite de régler de nombreux paramètres. Il est intéressant de tester ces différentes solutions sur son propre jeu de données, car la performance de ces algorithmes semblent en être dépendante.

Conclusion

Actuellement, les solutions commerciales de synthèse vocale suivent l'approche par sélection d'unités, ce qui permet d'obtenir une voix naturelle de qualité. Cependant, cette technique nécessite de grandes quantités de données de parole ainsi qu'une expertise linguistique afin d'obtenir des résultats expressifs. Pour pallier ce problème, des approches statistiques ont été proposées dans la littérature. Celles-ci consistent à apprendre un modèle acoustique pour fournir des paramètres à un vocodeur effectuant la synthèse. Lors de ce stage, nous tenterons de mettre en place un système

de synthèse dont le moteur de synthèse est hybride : un système par sélection d'unités guidée par un DNN. Ici, le DNN devrait avoir un rôle équivalent à celui d'un coût cible défini de manière non supervisée, sur des attributs qui n'ont pas été sélectionnés à la main.

L'idée de mon stage est d'utiliser la propriété d'*embedding* des DNN afin de projeter l'ensemble des unités de la base de données dans un espace où les *embeddings* serviraient de coordonnées. Ces coordonnées permettraient d'appliquer une recherche de plus proches voisins afin d'obtenir les unités les plus semblables à celles demandées par l'analyse linguistique pour effectuer une synthèse par concaténation. Cependant, il faut réussir à construire des *embeddings* ayant de bonnes propriétés. Notamment, il faut que la projection issue des *embeddings* rapproche deux unités de parole semblables et éloigne des unités différentes. Dans le cas contraire, les techniques de recherche de plus proches voisins ne donneront pas de bons résultats. Il faudra donc mettre en place de multiples réseaux pour comparer leurs *embeddings*, avec ou sans multi-tâches, entraîné comme auto-encodeur [3] ou comme modèle acoustique, etc.

Ensuite, toutes les méthodes de recherche de plus proches voisins ne sont pas équivalentes en performances en fonction du jeu de données et de la dimension des descripteurs de données. Il faudra donc déterminer la méthode optimale dans notre cas. Le principe de LSH repose sur des projections et parait donc semblable à celui des *embeddings*. Ce rapprochement laisse penser que leur association devrait être efficace. Cependant, LSH nécessite de régler de nombreux paramètres et de construire plusieurs fonctions de hachage. Si ces conditions s'avèrent trop compliquées à mettre en place dans notre cas, d'autres solutions s'offrent à nous, notamment les kd-tree et plus particulièrement sa variante, les forêts de kd-tree.

Si le travail précédent aura réussi à être effectué, une extension est possible. En effet, la synthèse par sélection d'unités nécessite la définition d'un coût de jointure. La définition d'un tel coût par des DNN dans notre situation pourrait être une poursuite intéressante.

Références

[1] Jacob Benesty, M Mohan Sondhi et Yiteng Huang. *Springer handbook of speech processing*. Springer Science & Business Media, 2007.

[2] Eric Moulines et Francis Charpentier. "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones". In : *Speech communication* 9.5-6 (1990), p. 453–467.

[3] Andrew J Hunt et Alan W Black. "Unit selection in a concatenative speech synthesis system using a large speech database". In : *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings.* T. 1. IEEE. 1996, p. 373–376.

[4] Heiga Zen, Keiichi Tokuda et Alan W Black. "Statistical parametric speech synthesis". In : *Speech Communication* 51.11 (2009), p. 1039–1064.

3. Réseau entraîné à prédire ses entrées.

[5] Yao Qian, Yuchen Fan, Wenping Hu et Frank K Soong. "On the training aspects of deep neural network (DNN) for parametric TTS synthesis". In : *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, p. 3829–3833.

[6] Keiichi Tokuday et Heiga Zen. "Directly modeling speech waveforms by neural networks for statistical parametric speech synthesis". In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, p. 4215–4219.

[7] Stas Tiomkin, David Malah, Slava Shechtman et Zvi Kons. "A hybrid text-to-speech system that combines concatenative and statistical synthesis units". In : *2011 IEEE Conference on Transactions on Audio, Speech, and Language Processing* 19.5 (2011), p. 1278–1288.

[8] Thomas Merritt, Robert AJ Clark, Zhizheng Wu, Junichi Yamagishi et Simon King. "Deep neural network-guided unit selection synthesis". In : *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, p. 5145–5149.

[9] Tomas Mikolov, Kai Chen, Greg Corrado et Jeffrey Dean. "Efficient estimation of word representations in vector space". In : *arXiv preprint arXiv :1301.3781* (2013).

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado et Jeff Dean. "Distributed representations of words and phrases and their compositionality". In : *Advances in neural information processing systems*. 2013, p. 3111–3119.

[11] Quoc V. Le, Will Y. Zou, Serena Y. Yeung et Andrew Y. Ng. "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis". In : *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2011, p. 3361–3368.

[12] Leonardo Badino. "Phonetic Context Embeddings for DNN-HMM Phone Recognition". In : *Proceedings of Interspeech*. 2016.

[13] Michael L Seltzer et Jasha Droppo. "Multi-task learning in deep neural networks for improved phoneme recognition". In : *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, p. 6965–6969.

[14] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts et Simon King. "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis". In : *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, p. 4460–4464.

[15] Jon Louis Bentley et Jerome H Friedman. "Data structures for range searching". In : *ACM Computing Surveys (CSUR)* 11.4 (1979), p. 397–409.

[16] Alexandr Andoni et Piotr Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions". In : *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE. 2006, p. 459–468.

[17] Marius Muja et David G Lowe. "Scalable nearest neighbor algorithms for high dimensional data". In : *2014 IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), p. 2227–2240.