Study of unit selection text-to-speech synthesis algorithms

David Guennec

ANNÉE 2016

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

David Guennec

préparée à l'unité de recherche IRISA UMR6074
Institut de Recherche en Informatique et Systèmes Aléatoires
École Nationale Supérieure des Sciences Appliquées et de Technologie

---

Étude des algorithmes de sélection d'unités pour la synthèse de la parole à partir du texte

Thèse soutenue à Lannion le 22 septembre 2016

devant le jury composé de :

Nick Campbell
Professeur, Trinity College Dublin / rapporteur

Ingmar Steiner
Senior researcher, Université de Saarland / rapporteur

Yves Laprie
Directeur de recherche, CNRS, LORIA / examinateur

Élisabeth Delais-Roussarie
Directrice de recherche, CNRS, LLF / examinatrice

Philip N. Garner
Senior researcher, Idiap Research Institute / examinateur

Damien Lolive
Maître de conférences, Université de Rennes 1 / Directeur de thèse

"No one regards what is before his feet; we all gaze at the stars."

Quintus Ennius (239–169 BC)

Acknowledgements

Résumé en français

Le présent résumé est une version condensée en français de l'ensemble des considérations, hypothèses et expérimentations, agrémentées de leurs résultats, présentés en langue anglaise dans cette thèse. Dans ce résumé, un soin particulier a été apporté au respect du même ordre de présentation des idées apportées dans la thèse en anglais et dans le présent résumé, de sorte que chaque chapitre du premier correspond à une section du dernier. Par exemple, la section II du présent résumé décrit le contenu du chapitre 2 dans la partie en langue anglaise.

Introduction

Les travaux de thèse présentés dans ce documents portent sur le sujet qu'est la synthèse de la parole à partir du texte, lequel peut être décrit comme un objet d'étude pluridisciplinaire [Boë 1990; Boeffard 2004]. En effet, l'étude de la parole humaine fait autant appel aux considérations anatomiques, et donc issues de la médecine, pour décrire l'appareil vocal humain qu'à la physique pour comprendre la mécanique des flux d'air (l'acoustique) interagissant au sein de ce dernier. En outre, parce que la parole est le vecteur d'un message doté de sens, l'étude de ce dernier, la linguistique, y prend également une place prépondérante.

La synthèse de la parole fait appel à l'ensemble de ces sciences, auxquelles il faut ajouter l'informatique. Avec l'émergence de l'informatique, les « machines parlantes » sont sorties des salles où elles étaient entreposées et où elles étaient souvent commandées à la main pour se retrouver au coeur de calculateurs aux capacités de traitement et de stockage en constante évolution. En effet, la synthèse de la parole visant la production d'un outil capable de produire un signal de parole sans intervention humaine directe, cet outil doit être doté d'un degré minimal d'automatisation. Comme dans de nombreux domaines, la majorité des systèmes de synthèse de la parole actuels ne tentent pas de reproduire le fonctionnement naturel de la bouche, du larynx et des poumons. Des systèmes relevant de cet ordre existent bel et bien (il s'agit de la synthèse dite articulatoire), mais leur intérêt reste académique : ils visent à toujours mieux modéliser l'appareil vocal humain afin de le comprendre, de la même manière que les modèles de systèmes planétaires ou galactiques en physique visent à valider des théories par l'expérimentation. Lorsque la qualité de la parole synthétique est un objectif, les systèmes actuels visent plutôt, soit à reproduire la parole via des connaissances expertes sur la composition du signal vocal une fois produit, soit à utiliser ces connaissances expertes pour sélectionner les meilleures portions d'un corpus de parole pré-enregistré reproduisant le message à produire puis les joindre.

Les travaux présentés dans cette thèse s'inscrivent dans le cadre de la seconde solution, nommée synthèse par sélection d'unités ou encore Synthèse Par Corpus (SPC). Les contributions présentées peuvent être réparties en deux parties. Tout d'abord (section VI), une analyse approfondie et un diagnostic de l'algorithme de sélection d'unités, lequel recherche dans le treillis des portions de corpus (nommées unités) utilisées, sont présentés. L'importance de l'optimalité

de la solution est discutée et une nouvelle mise en oeuvre de la sélection basée sur un algorithme A^* est proposée. La deuxième partie des contributions, elle-même subdivisée en trois portions, traite d'améliorations apportées à la fonction de coût permettant à l'algorithme de sélection de trier les séquences d'unités en vue de sélectionner la meilleure. Une méthode de calcul de coût cible est testée sur un coût visant les durées phonétiques des unités (section VII). Une méthode visant à améliorer l'intonation de la parole synthétisée est ensuite présentée (section VIII). Enfin, un système de pénalités nuancé par une fonction floue, lequel a pour but d'améliorer les jonctions entre unités est décrit (section IX).

Avant la présentation de ces travaux, un état de l'art est dressé dans cet ordre : introduction sur la parole humaine (section I), historique de la synthèse de la parole et présentation de l'état courant des recherches (section II) et enfin présentation des différents éléments constitutifs d'un système de Synthèse Par Corpus (sections III et IV). Le chapitre résumé par la section V se charge, quant-à-lui, de présenter le protocole expérimental, les voix de synthèse ainsi que les outils pour gérer ces voix utilisés dans la thèse.

Pour les travaux présentés dans cette thèse, nous travaillons dans un cadre ou la contrainte sur le temps réel lors de la synthèse est levée (le focus est sur la qualité) mais l'on s'y intéresse ponctuellement tout au long de la thèse, en particulier dans le chapitre traitant des algorithmes de synthèse (résumé en section VI). En outre, les travaux présentés dans cette thèse portent sur la synthèse du français. La majorité des résultats peuvent toutefois raisonnablement s'appliquer à de nombreuses langues.

I De la production de la parole

La parole humaine est un signal acoustique dont l'analyse permet de la subdiviser en deux composantes : la composante phonétique et la composante prosodique. La composante phonétique est la plus évidente. La parole peut être découpée en unités de sens plus ou moins grandes : phrases, groupes de souffle (séparés par une inspiration du locuteur), mots, syllabes et enfin phonèmes. Aux phonèmes, au nombre d'environ 35 en français (le nombre exact peut varier en fonction de composantes régionales ou historiques), on adjoint la notion de phone. Un phone est une réalisation acoustique d'un phonème. La parole étant influencée par de nombreux facteurs de variabilité, des phones correspondant au même phonème (dans ce cas, on parle d'allophones) peuvent être très différents. Cela peut notamment être dû au phénomène dit de coarticulation, qui veut que la prononciation d'un phonème soit directement influencée par celles de son prédécesseur et son successeur, voire par la prononciation de phonèmes plus distants. Enfin, une dernière notion revêt une importance considérable : il s'agit du diphone. Plusieurs recherches entreprises dans les années 50-60 ont en effet montré que les concaténations effectuées en milieu de phone étaient en règle générale de bien meilleure qualité que celles réalisées en frontière de phone [Peterson et al. 1958; Dixon and Maxey 1968]. De cette constatation est née une nouvelle unité : le diphone. Un diphone est une unité dont les deux frontières sont les centres de deux phones successifs, le centre du diphone étant la frontière entre les deux phones. Par exemple, la séquence de deux phones

[a o] contient un seul diphone commençant à la moitié du [a] et finissant à la moitié du [o].

La deuxième composante de base de la parole est sa prosodie. La prosodie regroupe toutes les informations que la composante phonétique ne prend pas en considération. Di Cristo [Di Cristo 2000] caractérise la prosodie comme « la représentation formelle (aspect phonologique) des éléments de l'expression orale tels que les accents, les tons, l'intonation et la quantité, dont la manifestation concrète, dans la production de la parole, est associée aux variations de la fréquence fondamentale (F_0), de la durée et de l'intensité (paramètres prosodiques physiques), ces variations étant perçues par l'auditeur comme des changements de hauteur (ou de mélodie), de longueur et de sonie (paramètres prosodiques subjectifs). » Dans la définition ci-dessus, un élément particulièrement important est la F_0, laquelle est perçue par l'auditeur comme la hauteur de la voix. Il s'agit en fait de la fréquence la plus basse du signal de parole, un signal de parole étant représentable (les portions voisées – générées par vibration des cordes vocales – sont d'ailleurs périodiques) dans le domaine spectral. La F_0 correspond à la première harmonique du signal. Les harmoniques suivantes, F_1 et F_2 du moins, peuvent également être mises en relation avec des aspects physiologiques de l'appareil de production de la parole (ouverture de la bouche et position de la langue).

II Une histoire de la synthèse de la parole

Le chapitre suivant est consacré à un historique de la synthèse de la parole et à un inventaire des techniques actuelles. Depuis la machine parlante du baron Von Kempelen, à la fin du XIII$^{\text{ème}}$ siècle, laquelle était intégralement constituée de composants mécaniques, en passant par le Voder et ses résonateurs électroniques, les méthodes de construction de ces machines ont considérablement évolué. La première tentative documentée de reproduction de sons de parole humaine est due à Christian Kratzenstein qui, en 1779, utilisant des résonateurs acoustiques pour reproduire les sons de 5 voyelles. Douze ans plus tard, le baron hongrois Wolfgang Ritter von Kempelen, publia un livre détaillant plus de 20 ans de recherches et la construction d'une machine parlante de son invention permettant de produire des lambeaux de parole. Au cours des 200 ans qui ont suivi, trois manières de synthétiser de la parole ont été explorées.

La première, historique, consiste à reproduire le système articulatoire humain. Cette méthode, basée sur la résolution d'équations différentielles hautement complexe, demande beaucoup de calculs et reste aujourd'hui trop lente pour une utilisation industrielle. De plus, la parole produite par ce biais reste de faible qualité en comparaison des autres techniques. Rien n'empêche cependant d'imaginer un retour en force de cette approche dans le future si une évolution favorable à cette technique se produisait.

La seconde, développé principalement à des fins de télécommunication (au moins au début), dans les années 1970, est de construire/apprendre des modèles de réalisation de la parole qui peuvent ensuite être utilisés de manière générative. Les méthodes basées sur cette technique ont l'avantage d'être flexibles et jouissent d'un très faible encombrement (les modèles actuels ne font que

quelques méga-octets). Les méthodes actuelles reposant sur ce principe sont regroupées sous le sigle SPSS, pour *Statistical Parametric Speech Synthesis*, et comprennent principalement la synthèse dite par HMM (*Hidden Markov Model*) et plus récemment par DNN (*Deep Neural Network*) [Black et al. 2007; Yamagishi et al. 2008; Hashimoto et al. 2015]. Cette approche statistique paramétrique a été l'objet de nombreux travaux universitaires ces dernières années. Cette méthode offre un contrôle avancé sur le signal et produit une synthèse très intelligible, mais la voix générée manque de naturel.

La troisième façon de produire des énoncés de parole est par concaténation de parole préexistante. La méthode dite par sélection d'unités, ou Synthèse Par Corpus (SPC) [Sagisaka 1988; Black and Campbell 1995; Hunt and Black 1996; Taylor et al. 1998; Breen and Jackson 1998; Clark et al. 2007] qui fait partie de cette catégorie et fait l'objet de notre travail est celle qui produit actuellement les signaux de parole synthétique de meilleure qualité. Le principal inconvénient de cette méthode est la nécessité de garder en mémoire une quantité considérable de parole naturelle pré-enregistrée : le corpus de parole ou voix de synthèse. Ce corpus peut regrouper plusieurs heures, voire plusieurs dizaines d'heures de parole provenant d'un même locuteur. La Synthèse Par Corpus (SPC), est un raffinement de la synthèse par concaténation où l'on dispose d'un simple dictionnaire de diphones, avec plusieurs variantes pour chaque diphonème. La SPC permet la création de synthèse de haute qualité, dont le naturel et la qualité prosodique restent inégalés par les autres méthodes grâce à l'utilisation de parole naturelle pour réaliser la synthèse. La plupart des systèmes industriels actuels fonctionnent grâce à cette méthode qui, outre la taille conséquente du corpus de parole, à quelques inconvénients, telle la difficulté à contrôler la prosodie et le risque d'artefacts de concaténation pénalisant l'intelligibilité. Cette méthode fait intervenir la notion d'unité, laquelle est une liste de segments (des diphones généralement) contigüs dans un corpus de parole correspondant à une partie de la séquence cible de segments à synthétiser.

Fondamentalement, avant les années 50, l'objectif principal de la plupart des études sur la synthèse de la parole était de produire une preuve de concept et d'étudier la parole via un outil. Ensuite, dans les années 50, grâce à un intérêt croissant pour la synthèse de la parole porté par les opérateurs téléphoniques, l'objectif principal est devenu la construction de systèmes de synthèse produisant une parole parfaitement intelligible. Cette étape a été atteinte au cours des années 80/90. La recherche s'est donc portée sur la qualité de la parole, en particulier pour la parole neutre ; et plus généralement la qualité du message porté par la prosodie de la parole synthétique et plus récemment sur l'expressivité.

III L'étage d'analyse et le corpus

Les travaux présentés dans cette thèse portent, au sein d'un système de SPC, sur le module de sélection d'unité. Ce module utilise deux types d'informations : une séquence d'informations représentant le signal de parole à produire et la voix de synthèse, contenant les unités (portions de corpus) à concaténer. Le chapitre 3 présente, d'une part, l'interface entre le texte fourni par l'utilisateur en entrée du système et le bloc de sélection d'unité et, d'autre part, le processus

de création du corpus de parole. Le texte fourni en entrée par l'utilisateur est annoté par une succession d'outils. Le texte est d'abord nettoyé de tout caractère incohérent ou non géré puis découpé en groupes de souffles et en mots. L'étape suivante, dite de phonétisation, génère la séquence de phonèmes reliés aux différents mots. Enfin, la syllabation produit la séquence de syllabes reliées d'une part aux phonèmes et d'autre part aux mots. Ce sont ces annotations qui, sous le nom de séquence cible (de phonèmes, de syllabes, ...), sont fournies au moteur de sélection d'unité.

Le corpus de parole, quant-à-lui, doit être mono-locuteur et couvrir un certain nombre d'attributs comme l'ensemble des phonèmes disponibles dans une langue, l'ensemble des diphones et ce au moins plusieurs fois (surtout pour les plus utilisés) ou bien encore la plus grande partie des syllabes d'une langue. Le script d'enregistrement du corpus, lu et enregistré ensuite en studio, peut être produit par pure construction de phrases en couvrant les attributs requis, par condensation d'un corpus textuel de taille considérable (trop grand pour être enregistré) ou bien via une approche mixte. Dans tous les cas, le compromis entre la taille et la richesse du corpus est le point clé pour obtenir une qualité vocale satisfaisante avec le système de synthèse.

IV Le bloc de sélection d'unité

Le chapitre suivant présente les spécificités du module de sélection d'unités, lequel est immédiatement suivi d'un module réalisant la concaténation des portions de corpus de la séquence sélectionnée et appliquant généralement un lissage à l'emplacement de la concaténation voire des modifications prosodiques comme un ajustement du rythme de la phrase ou de son *pitch* (hauteur de la phrase et perception qu'a un auditeur de la F_0), par exemple via un algorithme nommé PSOLA.

Afin de discriminer les segments provenant du corpus qui correspondent aux besoins exprimés par l'intermédiaire de la séquence cible, la méthode habituelle [Black and Campbell 1995] est de classer les unités en évaluant le degré de ressemblance avec la séquence cible (coût cible) et le risque de créer un artefact lors de la concaténation des unités (coût de concaténation) via des fonctions de coût.

Cette méthode fait intervenir la notion d'unité, laquelle est une liste de segments (des diphones généralement) contigüs dans un corpus de parole correspondant à une partie de la séquence cible de segments à synthétiser.

L'ensemble des unités disponibles correspondant à la séquence cible sont regroupées dans un graphe où une unité est un noeud et où un arc représente une possible concaténation. Plus précisément, ce graphe se trouve être un treillis. Le problème de sélection d'unités est donc un problème de recherche de meilleur chemin (un chemin étant une séquence d'unités correspondant à la séquence cible) dans un treillis. Il existe plusieurs algorithmes résolvant ce problème dont Viterbi et A^*. L'algorithme de sélection utilise la fonction de coût composée des coûts cible et de concaténation pour trier les séquences et sélectionner la meilleure.

V Données expérimentales et méthodologie de test

Dans le chapitre 5, nous présentons les corpus de parole utilisés dans la thèse, les outils permettant de les gérer et la méthodologie de test employée. Les corpus utilisés dans la thèse sont gérés par le *toolkit* ROOTS, développé dans l'équipe. Un autre format, binaire lui, est utilisé lors de la synthèse pour des motifs de rapidité. Nous utilisons deux voix dans nos expériences : *Audiobook*, qui est une voix masculine avec un F_0 moyen très bas (moyenne sur les portions voisées uniquement) à 87Hz et *IVS*, une voix féminine de F_0 moyen de 163Hz. Des sous-portions de ces voix ont été extraites au préalable pour former deux corpus de test auxquels est ajouté un troisième corpus de test de 27 141 phrases. Ces corpus de test sont utilisés pour générer des stimuli de parole à évaluer à la fois via des mesures objective et des évaluations subjectives, ces dernières se traduisant par des tests d'écoute en aveugle. Compte tenu de la complexité de la tâche d'évaluation de la parole synthétique, aucune méthode objective n'a encore réussi a donner pleinement satisfaction à ce jour. Les évaluations sont donc principalement subjectives. Les tests subjectifs peuvent évaluer la préférence des utilisateurs dans une confrontation de systèmes (tests AB) ou bien établir un score pour chaque système évalué (indépendamment (tests MOS, DMOS, ...) ou les deux avec un type de test plus récent : MUSHRA. Les tests cités sont tous utilisés dans la thèse, en priorité les tests AB. En particulier, le protocole décrit dans [Chevelu et al. 2015] est régulièrement utilisé dans nos tests.

VI Concernant le choix de l'algorithme de sélection

Dans le chapitre 6, nous présentons tout d'abord le système de synthèse de parole de l'IRISA, auquel les travaux de cette thèse ont apporté une importante contribution. Notre premier travail a ensuite porté sur l'étude et l'amélioration de l'algorithme de sélection d'unité. Nous avons implémenté une stratégie d'exploration très utilisée dans le domaine (algorithme de Viterbi) et l'avons comparé à une approche utilisant l'algorithme A^*, jugée moins combinatoire. Plus exactement, 3 approches de type *beam-search* (version de Viterbi sous-optimale mais plus rapide car élaguée) ont été testées avec A^*: un *beam-search* avec un fort élagage (faisceau de 10 unités) mais très rapide, une version intermédiaire avec 100 unités et enfin une version quasi-optimale avec 1000 unités. En particulier, la question était d'identifier si oui ou non, l'optimalité de la solution (*i.e.* la séquence de fragments de corpus à concaténer) était importante et sinon, quelle stratégie de recherche était la meilleure.

La comparaison, comprenant des mesures objectives ainsi que plusieurs évaluations subjectives, a été faite en utilisant les mêmes filtres de pré-sélection, la même fonction de coût et les deux corpus présentés plus haut. Les résultats ont montré que A^* dans sa version admissible se montre plus efficace qu'un *beam-search* avec une grande taille du faisceau (celui de taille 1000). Cependant, il a également été montré que les algorithmes explorant un treillis fortement élagué, même s'ils sont perçus comme moins performants (autant dans les tests perceptifs qu'avec les données objectives), ne présentent pas un écart considérable en terme de qualité de synthèse. Même si A^* réalise un nombre de concaténations

inférieur aux autres algorithmes, les évaluations perceptives montrent que cela ne se traduit pas par un écart de qualité considérable. Les résultats semblent indépendants à la fois du style vocal utilisé et de la voix. Cela nous amène à deux conclusions : d'abord, trouver la solution optimale au problème de sélection d'unité semble de peu d'utilité. En effet, des algorithmes modérément élagués présentent une qualité de synthèse perçue identique. Cela s'explique par la variabilité entre les meilleures séquences qui est très faible. En outre, on montre que A^* est mieux adapté que Viterbi au problème de sélection d'unité. Dans le reste du document, nous utilisons toutefois toujours A^* en version optimale lors de la génération des stimuli de test.

En outre, la sélection des unités est fortement dépendante de la stratégie de présélection (qui empêche les unités jugées trop mauvaises d'arriver dans le treillis de sélection en se basant sur un vecteur binaire de caractéristiques à respecter). La présélection pose en effet une contrainte sévère sur le moteur. Nous avons donc évalué l'impact des filtres sur la qualité de la synthèse. Nous avons montré que les filtres que nous utilisons ne dégradent pas la synthèse tout en économisant un temps de calcul considérable (en réduisant la taille du treillis par élagage).

VII Proposition d'un nouveau coût cible de durée phonétique

Le chapitre suivant présente une nouvelle façon – dans le coût cible – de minimiser les différences spectrales en sélectionnant des séquences d'unités minimisant un coût moyen au lieu d'unités minimisant chacune un coût cible de manière absolue. Ce coût est testé pour une distance sur la durée phonémique mais peut être appliqué à d'autres distances. Le but est de sélectionner la séquence complète d'unités qui minimise une distance de durée phonémique avec des valeurs prédites par un ANN (*Artificial Neural Network*) plutôt que de choisir la séquence contenant des unités qui minimisent individuellement la même distance de durée. Ceci est destiné à éviter des cas tels qu'une excellente synthèse pénalisée par quelques très mauvaises unités uniquement en produisant la séquence la plus homogène possible (ce qui est déjà favorisé par le coût de concaténation, bien qu'insuffisamment). Les expériences ont montré que cette nouvelle mesure donne de bons résultats sur les échantillons de parole qui présentent des problèmes de durées, en particulier pour les voix expressives. En outre, nous montrons que la nouvelle mesure ne semble pas affecter les échantillons synthétisés qui ont de bonnes durées depuis le début.

VIII Proposition d'un nouveau coût cible pour le contrôle du *pitch*

Notre deuxième proposition sur la fonction de coût est une fonction de coût cible visant à améliorer l'intonation en se basant sur des coefficients extraits à travers une version généralisée du modèle de Fujisaki. Ce modèle utilise des fonctions gamma modélisant le contour de F_0 appelées atomes. Les paramètres

de ces fonctions sont utilisés au sein d'un coût cible. L'hypothèse sous-jacente est que ces fonctions gamma (dont certains paramètres semblent correspondre à des fonctions physiologiques) sont positionnés à des endroits où les variations acoustiques induites par les facteurs physiologiques auxquels les atomes (nom donné à ces fonctions) semblent corrélés sont importantes et doivent être respectées. Nous supposons également que le processus de sélection d'unité choisit naturellement des unités qui disposent d'une F_0 plus ou moins homogène au niveau de la phrase. En effet, une distance évaluant la différence de F_0 est présente dans le coût de concaténation. De manière à assurer cette hypothèse, le coût de concaténation de F_0 pourrait suivre la formulation proposée dans le chapitre précédent.

Concernant l'intonation des phrases synthétiques, la plupart des problèmes proviennent de segments qui ont un contour local (à l'horizon phone-syllabe) de F_0 très différent de ce qui est attendu. Comme les atomes locaux se situent précisément au niveau des segments voire au niveau syllabique, et comme les atomes locaux pourraient être liés à des facteurs physiologiques liés à l'intonation, nous avons fait l'hypothèse que l'utilisation de ces données via une contrainte de type coût cible peut améliorer la prosodie synthétisée. Une fonction de coût utilisant le contour de F_0 reconstruit à l'aide des atomes a été construite. Une seconde fonction, utilisant uniquement les paramètres des atomes (les paramètres des fonctions gamma) a également été créé et testé. Les expériences menées ensuite ont montré que ces deux méthodes surclassent une distance standard de F_0.

IX Unités sandwich pour le coût de concaténation

Le rôle du coût de concaténation est de s'assurer que l'assemblage de deux segments de parole ne causera l'apparition d'aucun artefact acoustique. Pour cette tâche, des distances acoustiques (MFCC, F_0) [Stylianou and Syrdal 2001; Tihelka et al. 2014] pour évaluer le niveau de ressemblance spectrale entre deux stimuli vocaux sur et autour du point de jonction. Ces coûts de concaténation sont toutefois loin d'être parfaits et de nombreux artefacts apparaissent à la fois dans les systèmes commerciaux et de recherche, même après un traitement post-concaténation. Plusieurs analyses ont montré que ces artefacts se produisent plus souvent sur certains phonèmes que sur d'autres [Yi 1998; Cadic et al. 2009]. Cette observation est à l'origine d'une méthode de construction de script d'enregistrement dans [Cadic et al. 2009] où la couverture de « sandwichs vocaliques » vise à favoriser les concaténations sur des diphonèmes jugés peu risqués. Ainsi, dans le dernier chapitre de cette thèse, nous proposons d'intégrer ces contraintes directement dans la fonction de coût, sans l'aide d'un corpus construit avec des sandwichs vocaliques.

Nous intégrons ainsi une pénalité en fonction de la classe de phonèmes dans la fonction de coût lors de la sélection d'unité. Deux versions sont proposées : d'abord en utilisant une pénalité fixe puis une fonction floue visant à rendre la pénalisation des unités plus flexible. La version faisant appel à une fonction floue est capable de relâcher la pénalité en fonction du positionnement des sous-coûts de concaténation des deux unités à joindre par rapport à sa distribution observée dans le corpus ayant servi à construire la voix de synthèse. En

somme, pour deux unités jouissant par exemple d'un des meilleurs coûts de concaténation possibles, la pénalité induite par la prise en compte des classes phonétiques des diphones composant les deux unités à joindre sera nécessairement faible voire nulle. Une évaluation objective montre que la pénalité est efficace et amène à un meilleur classement des séquences d'unités candidates au cours de la sélection tandis qu'une évaluation subjective révèle une performance supérieure de l'approche floue.

Conclusion

Les contributions apportées par cette thèse se répartissent donc sur deux axes. D'une part, il a été montré que l'algorithme A^* présentait des avantages conséquents sur Viterbi ou même *beam-search*. En outre, comme A^* peut également être élagué, il est une solution tout-à-fait convenable, préférable à Viterbi, pour la sélection d'unité. Notre travail sur les algorithmes a également montré que la recherche de la solution optimale au problème de sélection n'est pas nécessaire puisque qu'un algorithme explorant un graphe de sélection modérément élagué fait jeu égal dans les tests subjectifs. La variabilité entre les meilleures séquences candidates étant faible, la dégradation de la synthèse n'intervient qu'avec un très fort élagage.

D'autre part, nous avons mené trois travaux sur la fonction de coût ayant amené à une meilleur performance générale (en terme de qualité) : un coût améliorant les durées phonétiques, un autre améliorant l'intonation et enfin un système de pénalités nuancé par une fonction floue permettant des concaténations de meilleure qualité.

Les principales possibilités d'extension de la thèse concernent la distance de durée et la pénalité floue sur les sandwichs. En effet, la méthode de calcul du coût de durée pourrait être étendue à toutes les distances (cible et de concaténation) de la fonction de coût, moyennant une adaptation de la méthode pour les coûts de concaténation. Il serait par exemple envisageable de pondérer les coûts de concaténation par une distance comparant les paramètres d'une unité à la moyenne des valeurs des mêmes paramètres observées précédemment dans la portion déjà sélectionnée de la séquence candidate. D'autre part, le système de pénalité pourrait être testé dans d'autres configurations et sur d'autres langues. En particulier, d'autres fonctions qu'une fonction linéaire pourraient être appliquées et les classes de pénalités pourraient être revues. En outre, le coût d'intonation ayant la particularité dans nos travaux de n'avoir été testé que sur un système « oracle », c'est-à-dire sans prédictions depuis le texte mais directement avec des annotations réelles, il serait souhaitable de tenter la prédiction du contour d'atomes reconstruit (prédire les paramètres directement semblent difficile) pour ensuite intégrer ces prédictions et avoir ainsi un coût d'intonation pleinement fonctionnel.

Abstract

This PhD thesis, entitled "Étude des algorithmes de sélection d'unités pour la synthèse de la parole à partir du texte" (Study of Unit Selection Text-To-Speech Synthesis Algorithms), focuses on the automatic speech synthesis field.

Two main strategies are currently under consideration in this field. The first one relies on a statistical parametric approach where models of speech signals are created. Models are then used in a generative way to produce speech utterances. It is widely known as the Statistical Parametric Speech Synthesis approach (or SPSS). The second strategy, which is an evolution of concatenation-based synthesis, consists in preserving and annotating a large speech corpus (usually several hours or even tens of hours), then extract fragments (called units) and paste them together to reproduce a textual utterance to synthesize (called the target utterance). The mechanism (not trivial) by which these fragments are selected is referred to as unit selection. The general technique is called Corpus-Based Speech Synthesis.

My thesis aim is to explore, diagnose unit selection mechanism and suggest improvements. To meet these objectives, a corpus-based speech synthesis was needed. For reasons of independence, flexibility and to ensure a transversal control of the software, it was decided to build a completely new system rather than using and modifying an existing tool. I spent a considerable amont of time during my thesis contributing to the implementation of the synthesis engine within my research team and adding features to it.

I first took interest in evaluating the impact of the search algorithm on unit selection. In particular, I considered whether or not optimality of the solution (*i.e.* corpus units to be concatenated) was important. My conclusion was that the search algorithm sensibly impacts the selection process only when searching for the optimal solution (or near optimal). For most applications, optimality of the solution is not necessary however. Even a very pruned unit selection process can be used with rather few sensible flaws.

The second part of my work focused on the cost function that allows the search algorithm to rank corpus units according to their suitability to solve the problem. This function is composed of a concatenation and a target cost. The first one measures the ability of a unit to be pasted after another one without causing artefacts. The second one judges the level of dissimilarity between a unit and what is desired. A fuzzy penalty function using the "Vocalic Sandwich" criterion was designed and tested. Its goal is to try avoiding concatenations on corpus units where more atefacts are usually found. This method has the particularity to be flexible and does not always penalize units. Indeed, it also takes into account the value concatenation cost. New target cost strategies have been implemented and tested. A first cost, which integrates long-term constraints in the phonemic duration target cost, was tested. Its aim is to produce a sentence with units of roughly equivalent quality. The second cost is based on the atom-based intonation decomposition technique, a generalized version of Fujisaki's Command-Response model. It aims at enhancing intonation in sentences produced by the Text-To-Speech synthesis system. Both two target costs and the "Vocalic Sandwich" penalty were shown to enhance synthesis quality, especially the last one, which should really be the subject of more research.

Contents

Introduction

Speech synthesis democratization accelerated dramatically in recent decades – and even more in recent years – with the appearance and the popularization of new needs. If foreseeing the future of human-computer interfaces[1] is much of a guessing game, an observation of recent changes in technology shows a clear trend: the "traditional"[2] mouse and keyboard setup tends to be replaced by touch and voice-enabled interfaces whenever and wherever possible. Why is that so? Simply because touching, hand-manipulating and speaking are the essence of interaction of a human being with his environment. But why is this revolution happening now, while it did not occur a few decades ago? After all, touchscreens, speech synthesis and to a lesser extent Automatic Speech Recognition (ASR) are not younger than the computer mouse. Once again, the answer is simple. The technology was not ready until the beginning of the XXIst century.

So, now concerning speech synthesis, what changed between now and then? Speech synthesis began, with baron Wolfgang Ritter von Kempelen during the XVIIIth century, as an experiment focused on testing the feasibility of producing speech artificially. It was also a simple way to better understand speech by reproducing it. The emergence of electronics allowed experiments like Homer Dudley's Voder but it is only with the arrival of the computer age that speech synthesis could be fully automated, this time generating speech stimuli from a textual input. The first Text-To-Speech synthesis systems (TTS) emerged in conjunction with the expansion of telecommunications. And it is precisely telecom operators that first saw a potential in TTS. It enabled them, along with automatic switching systems, to replace human telephone operators with machines. Hence, TTS enabled-telephone servers began to appear in the 90s. With the arrival of unit selection-based speech synthesis, which we will discuss in an instant, TTS ultimately reached a quality level sufficient for commercialization. It is only in the last years though that TTS and ASR increasingly invaded the consumer market, thanks to the conjunction of four elements. First, the dramatic increase in computation capabilities, even in small embedded and mobile systems. Second, the increase of storage capability, especially in mobile

[1] The term human-machine is perhaps even better as identifying new technological devices as computer-driven tends to be more and more difficult for the public.

[2] Can a technology that reached the mass market in the 80s, like the mouse did, be called traditional 30 years later?

devices (allowing TTS directly on a smartphone for instance). Third (and perhaps the most important), the democratization of internet access on all devices, allowing to perform the synthesis on powerful servers before sending the synthesized speech stimuli back to the users. Fourth, the refinement of unit selection TTS and the emergence of SPSS (*cf.* next paragraph) boosted synthetic speech quality. For all these reasons, TTS is taking, along with ASR, an increasingly important place in the human-machine communication paradigm.

Let us now focus on the current state of the art of research in the TTS field. In recent years, research in text-to-speech synthesis essentially focused on two major techniques. The statistical parametric approach (SPSS), which mainly includes HMM and DNN-based systems, is the most recent and has been the focus of many academic work in recent years. This method offers advanced control on the signal and produces very intelligible speech but generated voice lacks naturalness. The historical one, unit selection [Sagisaka 1988; Hunt and Black 1996], is a refinement of concatenative synthesis, which principle is very simple: record speech, split it into small units and paste these units in order to match a textual utterance. Sound created with this method features high naturalness and its prosodic quality is unmatched by other methods, as it basically concatenates speech actually produced by a human being. While most industrial TTS systems rely on unit selection, this method has its drawbacks, for instance the difficulty to force prosody and the possibility to get concatenation artefacts penalizing intelligibility. In the formulation of the unit selection problem, a unit is a list of contiguous segments (in the speech corpus) fitting a portion of the target sequence. In order to discriminate the segments coming from the corpus that fit the requirements expressed via the target sequence, the usual method is to rank the units by evaluating the context matching degree (target cost) and the risk of creating an artefact if concatenating the unit (concatenation cost) via balanced cost functions. Due to the complexity of the problem – unit selection has to process millions of candidate units to synthesize an average utterance – pruning is often needed to give a result in an acceptable time or even in real time for most industrial applications. In this thesis, we take abstraction of this real time constraint as we work on uncompromised synthesis quality. We will nevertheless focus punctually on this constraint, especially in chapter 6, when dealing with unit selection algorithms.

The work presented in this thesis entitled "Study of Unit Selection Text-To-Speech Synthesis Algorithms", we focus on unit selection solely. Our work aims at exploring, diagnosing unit selection mechanism and adding algorithmic improvements both in the unit selection process and in the selection cost. For this, we built a new Text-To-Speech synthesis system within IRISA/Expression team. We decided to build this system instead of choosing an existing one for reasons of independence, flexibility and to ensure a transversal control of the software, especially as an extensive part of this thesis was spent working on the inside of the unit selection engine. A non-negligible amount of time was spent, during this thesis, contributing to the implementation and maintenance of the synthesis engine

and adding features.

This thesis is articulated around tree main parts. In the first part, we describe the state of the art in speech synthesis. We start with a presentation of the basic concepts on which TTS relies in an introductory chapter (chapter 1). The second chapter (chapter 2) show the place of our work within the "TTS world" by making a chronology on speech synthesis. In particular, we highlight the main problems researchers addressed throughout the history of speech synthesis, finishing with the main problems in current research. The two remaining chapters, in the first part deal with the full TTS chain, namely the TTS frontend (chapter 3) and backend (chapter 4) and their sub-components. We present each of them extensively. In the last chapter, we define formally the unit selection problem and we discuss its characteristics in detail. In this thesis, we pay a particular attention to present the problem in its variable-size version, and not as often the reduced diphone-based version.

In the second part of this thesis, we focus on the whole unit selection block and we analyze its key algorithmic components. The first chapter of that part (chapter 5) presents the databases and database management tools we use throughout the thesis. In that chapter, we also give precisions concerning the test methodology used in this document. In the second chapter (chapter 6), we describe the TTS system we built and we present our work on the unit selection algorithm. In most cases, the Viterbi algorithm is used to perform the unit selection task, but it is not the only possible one. Since unit selection can be formulated as a path finding problem, other graph exploration algorithms can also be applied, which is what we do in this chapter. We also present important results on the preselection filters and the cost function.

Finally, the third part focuses on the cost function. A total of three propositions are presented, two within the target cost and one in the concatenation cost (following Black & Campbell's formulation [Black and Campbell 1995]). In chapter 7, a duration cost is presented. Its aim is to select the unit sequence that best minimizes, as a whole, duration distance rather than choosing the sequence containing units that individually minimize a duration distance. This is intended to avoid cases like excellent synthesis penalized by few very bad units. This cost is tested on a phonemic duration cost but can perfectly be applied to any other cost. Chapter 8 presents a target cost aiming at improving the intonation of synthesized sentences. For this task, it uses the parameters of Atom-based intonation decomposition technique, a recent generalization of Fujisaki's Command-Response model. Finally, we present a work on a penalty system using the concept of "Vocalic Sandwiches" first presented by D. Cadic *et. al.* [Cadic et al. 2009] in chapter 9. In particular, a fuzzy membership function depending on the distribution of concatenation costs was designed. It softens the penalty with regard to the relative cost of a unit in the speech corpus cost distribution.

Part I

State of the Art

Chapter 1

On Speech Production

"Speech or human language is the ability to communicate one's feelings or thoughts to his fellows by different voice intonations."

Von Kempelen's definition of speech in 1791,
in "Mechanismus Der Menschlichen Sprache",
Wolfgang Ritter von Kempelen (1734–1804)

Speech *is a multidisciplinary subject of study and can be seen as the crossing between three domains: Medicine and anatomy on one side to characterize the human speech apparatus; Physics for the study of the acoustic dimension and finally linguistics to analyze the message actually conveyed by this way of expression. To these fields we must add Computer Science [Boë 1990; Boeffard 2004]. The main goal in this short introduction chapter is to give some basic vocabulary and definitions. Though we use several other sources, like the Springer Handbook of Speech Processing [Benesty et al. 2008], the main source of inspiration of this chapter is the "La parole et son traitement automatique" book (French) [Calliope 1989]. We first define the nature of speech signals, focusing on its decomposition into meaningful units. We discuss the notion of acoustic variability that makes speech synthesis and recognition particularly difficult. We also focus on the human vocal apparatus and its main components. Then, we define the notion of prosody. Finally, we present briefly speech in the spectral domain.*

1 What Is Speech ?

The distinction between speech signals and other sounds is made thanks to specific characteristics, directly related to the acoustic mechanisms they originate from in the speech apparatus (larynx, voice chords, tongue, hard palate, aperture, *etc.*). There are two ways

to produce speech sounds. First, by vocal folds' vibration, also called voicing and second, by a direct air flow from the lungs into the vocal tract, without the use of vocal folds. A voiced sound is a sound that was produced by a vibration of vocal folds, though it may later be modified by articulators. Vocal sounds may be retained for a time by a closure of the vocal tract at lips level.

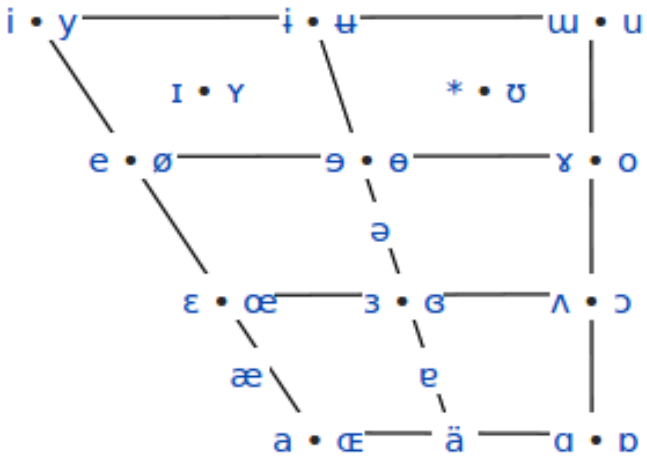

Figure 1.1: The vocalic trapezium as shown on Wikipedia encyclopedia. The horizontal axis corresponds to tongue position (front on the left, central in the middle row and back on the right) while the vertical one is for mouth opening (closed in the upper part and open in the bottom).

Speech sounds in a language – mostly French in our case, are called phones. Phones are sorted in two big categories: vowels and consonants. Vowels are characterized by their intrinsic properties: nasality, opening level of the vocal tract, tongue position (front or back on both horizontal and vertical dimensions) and lips articulation (roundness). Tongue height, backness and roundness may be used for defining the vocalic IPA vowel trapezium, shown on figure 1.1 [1]. On this trapezium, the horizontal axis corresponds to the tongue position, from front position on the left to back on the right, while vertical one is for mouth opening, also called aperture or vowel closeness. Open vowels are at the bottom while closed ones are on top.

Consonants are classified depending on their voicing (vowels are always voiced), *i.e.* the use or not of vocal folds, the manner of articulation (obstruent, sonorant, lateral or trill) and the place of articulation (labial, bidental, coronal, dorsal, laryngeal or peripheral) [Calliope 1989; Benesty et al. 2008].

These different properties inherent to phones allow to gather them into a set of classes that serve to formalize elementary sounds of a language: phonemes. Several phones in a language can be related to a same phoneme. The distinction between phone and phoneme

[1] These characteristics can be analyzed and qualified in terms of frequency through the notion of formant.

lies on a difference of domain: acoustic versus phonologic. A phone is the acoustic realization of a phoneme, the latter being a phonological unit. The classes formed by phonemes can be observed on the general IPA trapezium above for English language. Depending on the language or even the regional accent, the position of these classes may change and some classes may not be present while some others may appear (as all languages don't use the same sets of phonemes). The phoneme usually constitutes the first and smallest entity of a language. As several distinct phones are related – in a language – to the same phoneme, elements of the set of phones related to a same phoneme are called allophones[2].

By chaining phonemes one after the other we get the notion of syllable. A syllable is made out of 3 elements: an onset, a nucleus and a coda. The nucleus is the core of the vowel and is necessarily a vowel. The onset and the coda are the sets of consonants respectively preceding and following the vowel core of the syllable. They may eventually be empty (one or the other or even both). Syllables are usually defined as the smallest meaningful units in a language, as the latter basically happens to be a concatenated chain of them. They have particular importance in syllabic languages, like French, Romanian or Vietnamese. In those languages, on the contrary of stress languages (like English) the rhythm of the sentence is defined by the speed syllables are pronounced, each one of them being pronounced with the same length.

2 Acoustic Variability

A huge number of parameters causes spectral characteristics of speech production to vary – sometimes considerably. Gender, age, ethnic and cultural origin, education level, emotional state, sickness, *etc.* are as many parameters that cause speech stimuli corresponding to a same sentence to be different. Some allophones pronounced by two different speakers, or even by the same speaker in two different contexts, may vary up to a point where a listener might not even recognize them as the realization of a same phoneme. That effect is particularly intense on vowels. As an example why, we will discuss the three biggest sources of differences.

First Male and female speech apparatuses are not entirely identical. Female vocal tracts generally are 15% shorter than male ones. Furthermore, male larynx is located deeper (lower) than female larynx, which induces a different articulation of the tongue for the two genders: masculine tongue articulation is more open (less cramped articulatory channel) than for females.

Second, the shape of the nasal tract changes from an individual to another. Nasal

[2]It has to be kept in mind that the allophone set is language dependent. Phones realizing the same phoneme in one language are not necessarily allophones in another.

sounds, which require a communication between nasal cavity and oral tract, are therefore different among different people.

But then, beyond dissimilarity between male and female articulatory apparatus or even the influence of culture on pronunciation, the key phenomenon to explain such variability is co-articulation. Co-articulation is the influence one sound exerts on its successor(s). In order to produce the first sound, the speaker has to put his articulatory apparatus in the required configuration and then rearrange quickly the apparatus to meet the configuration necessary to pronounce the following sound. Thus, the regular flow in pronunciation and the continuity of the sound dictate a difference in the characteristics of the second sound compared to its archetype. In practice, the speaker has to find a good compromise between the required position his articulators must have to pronounce the right sound and the time he can spend pronouncing this sound. Hence, the sound the speaker actually produces is a compromise of these two prerequisites. As an example, let us focus on French syllable [du], transcribing word *« doux »* (soft). To pronounce it, the speaker must first say phoneme [d] for which the tongue has to be in the front position in the mouth, and then shift the tongue to the rear as quickly as possible for the second sound [u]. As in practice the speaker only has a small amount of time to get his tongue back, the latter generally is not in the usual position for pronouncing the [u]. The faster the speech flow, the most intense the effect. As co-articulation is a matter of compromise, the first phoneme ([d] in the example above) is also impacted by the second one ([u]) though it is to a smaller extent. Co-articulation ranges most of the time over two phones, but it may also occur on a wider range (three or even four phones).

Because co-articulation is so important, unit selection – which will be introduced in chapter 2 and presented in detail in chapter 4 – uses a base unit called diphone instead of phonemes. A diphone is the speech signal segment between the half of a first phone and the half of its successor. Though it was originally invented by Küpfmüller and Warns in 1956, the unit was first introduced in [Peterson et al. 1958], where it was called a "dyad" and then diphone in [Dixon and Maxey 1968]. Figure 1.2 gives an example of phone/diphone segmentation on syllable "aba". On the upper part, we have phone frontiers, with the label of the phoneme each phone is a realization of. The lower part shows diphones, which frontiers are placed on the middle of each phone.

This has the advantage of having the concatenations (in a unit selection synthesizer) on phone centers, which are the most stable parts of the signal and the least impacted by coarticulation.

Beyond stepping up the difficulty of the challenge Automatic Speech Recognition (ASR) systems face, this increased acoustic variability can prove difficult to reproduce in synthetic speech. It is nonetheless essential, to reproduce it in order to keep naturalness of the voice.

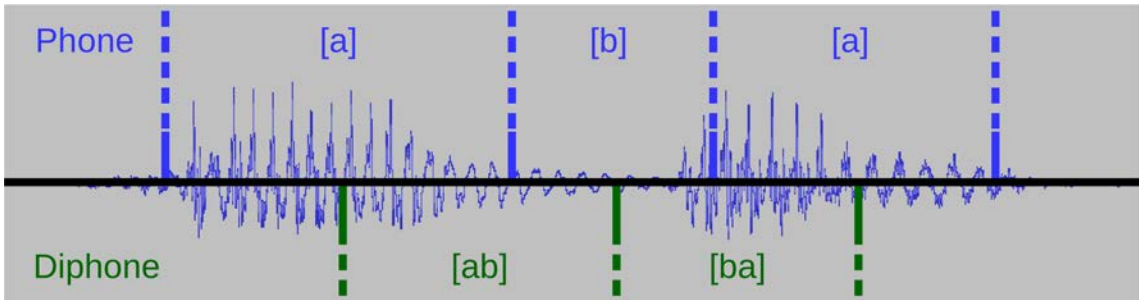

Figure 1.2: An example of phone/diphone annotation. Here, syllable "Aba" is decomposed in phones in the upper part and in diphones, with a frontier in the middle of each phone, in the lower part.

3 Anatomy of the Vocal Apparatus

Before serving for speech production, the human vocal apparatus is a key part of both the respiratory system and the alimentary system. Many of its components are common to most mammals, especially among primates. What makes the distinction of the human system is a set of unique features: a flat tract, small lips, small teeth, short oral cavity, rounded tongue and independent control over phonation and articulation. The incremental usage of the vocal apparatus as a communication tool over human evolution caused dense and direct neural connections from the language related areas to the articulatory system through the cortex [Benesty et al. 2008].

The speech production apparatus, as described in figure 1.3, may be divided into two distinct parts. First, the phonation system composed of both the lungs and the larynx are responsible for the production of the air flow and vocal vibration. Then, the articulatory system adds resonance and modulates the signal. Cavities, situated both in the vocal tract and nasal system, cause the air to resonate.

The lungs, by setting the sub-glottal air pressure create the voice source sound. The air pressure is the pressure caused by the air blocked and accumulated in the entry of the larynx before the vocal folds open and let the air flow. The larynx, pictured on the right part of figure 1.4, contains the glottis and the vocal folds. It creates a vibration of the air by vibration of the vocal folds when the air flows through it. This happens only for voiced sounds though; unvoiced sounds travel through the larynx without being stopped at the level of the glottis. In the beginning of a breath group, the sub-glottal pressure is maximal and evolves by about 15% through time before dropping drastically in the end of the breath group.

The opening of the glottis – housing vocal folds – is called abduction while glottal closure is named adduction. They are permitted by the arytenoid muscle, which moves vocal

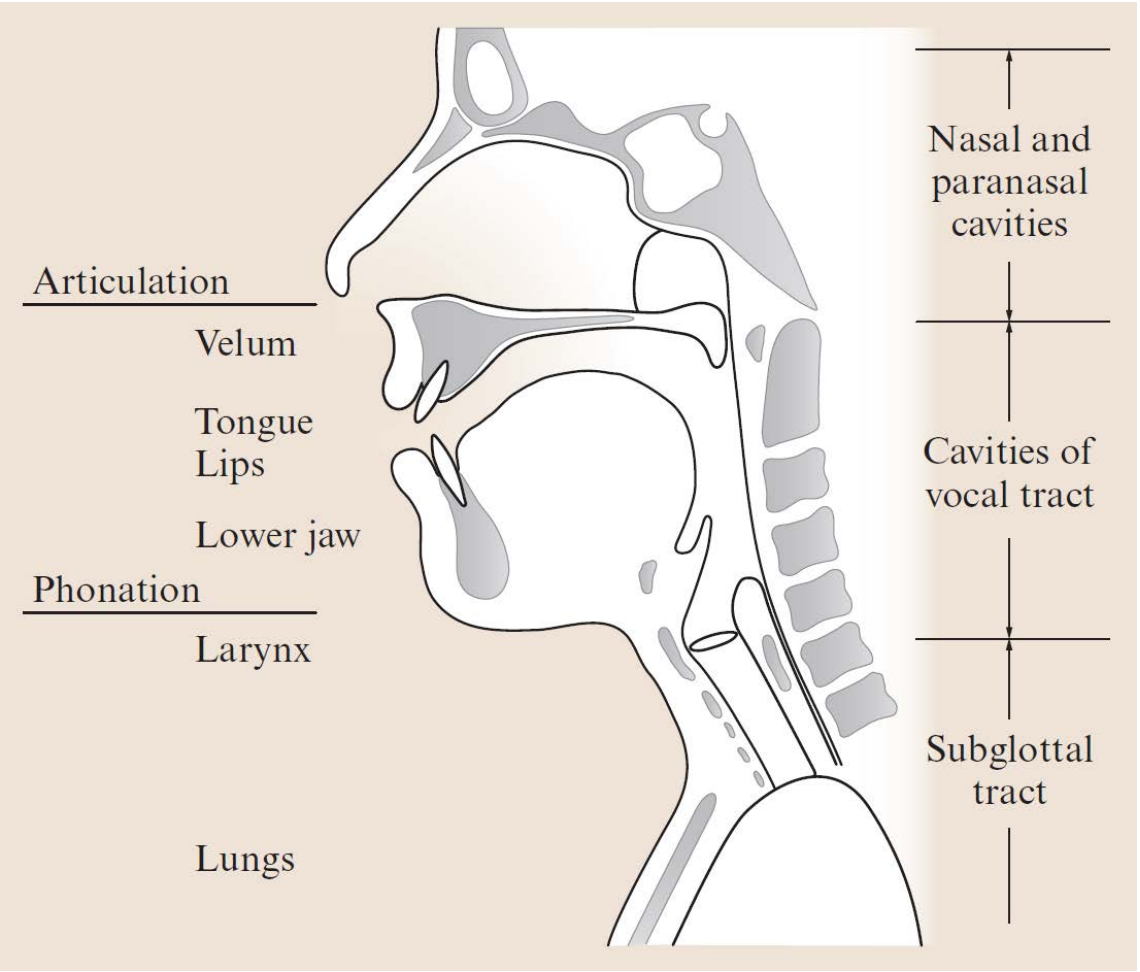

Figure 1.3: A view of the human vocal apparatus, dissociating parts responsible of phonation and the main articulators. The sounds produced by these mechanisms propagate and resonate into three cavities/tracts. Figure extracted from the Springer Handbook of Speech Processing [Benesty et al. 2008].

folds. Both are represented in the lower view of figure 1.4. This cycle of adduction and abduction events has a periodic nature and its frequency is called F_0, or fundamental frequency. It is perceived as the pitch, *i.e.* the sound level in speech. Together with rhythm, pitch forms the melody of a sentence, exactly as they form the melody of a musical piece. In other words, F_0 is the lowest frequency of a periodic waveform (*i.e.* its first harmonic). Higher frequencies in which the signal can be decomposed are associated to F_1, F_2, ... (see section 5).

Left part of the figure positions the articulators that cause a modulation of the air as it travels through the vocal tract. Most articulators are situated in the vocal tract and the mouth, the most important being the tongue, lips, palate and jaws. A part of the air, depending on the position of the velum (that acts like a trap-door between oral and nasal

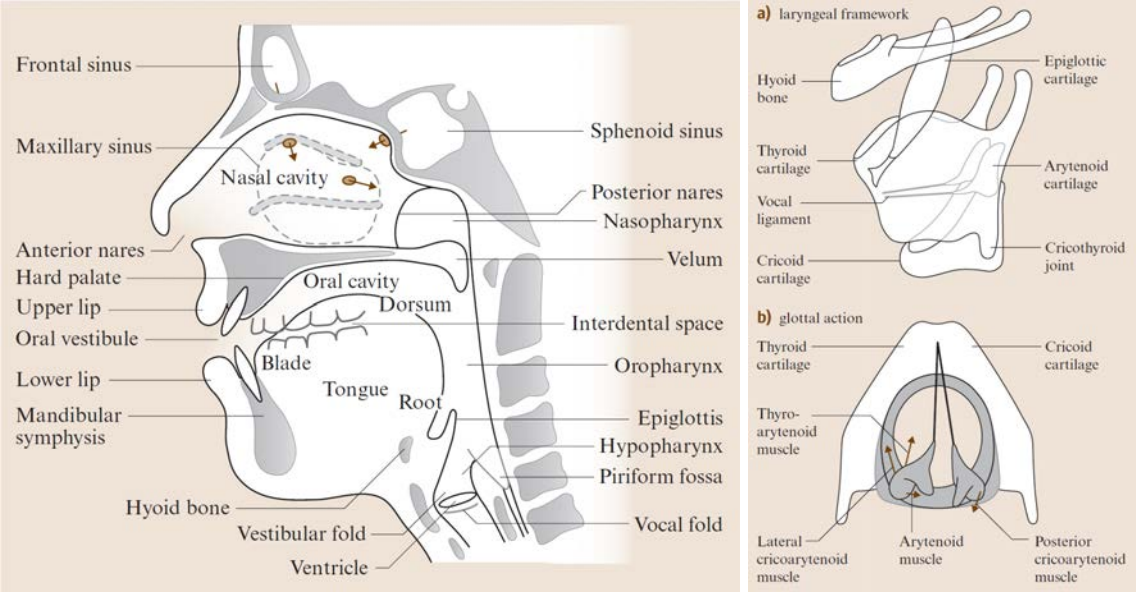

Figure 1.4: *Left:* A global view of the articulators and cavities involved in speech production inside the vocal apparatus. *Right:* Detailed views of the laryngeal framework and the glottal adduction (left part of the glottis sketch) and abduction (right part). Figures extracted from the Springer Handbook of Speech Processing [Benesty et al. 2008].

cavities), also enters the nasal tract. For the production of nasal sounds, the velum let's all the air enter the nasal cavity.

4 Prosody

Speech signals are not exclusively constituted of phones, syllables and words. When we speak, elements like voice height, intensity of speech and speech rate vary in permanence. This complex process, which mainly intervenes at the supra-segmental level, with the notable exception of phonemic duration, is called prosody. It adds to the signal information that meaningful units alone cannot convey. It provides the information of modality[3], shows the relative importance of the different elements in the sentence via intonation and accents and tells the emotional state of the speaker.

Di Cristo [Di Cristo 2000] characterizes prosody as the elements of oral expression like accents, tones, intonation and quantity; which are linked to the temporal evolution of the three prosodic parameters that are fundamental frequency (F_0), duration and intensity. The listener interprets these as melody, length and tonal variations.

[3]The syntactic structure, *i.e.* indication on neutral, interrogative or exclamatory nature, of the sentence.

4.1 Parameters

The three prosodic parameters (F_0, intensity and duration) are all produced relatively independently by the speaker. Nonetheless, relations exist first between F_0 and intensity as both of them are factors of the sub-glottal pressure (F_0 is also a factor of the tension in the vocal folds). Duration also being dependent of the sub-glottal pressure, prosody is usually represented solely by the F_0 curve [Calliope 1989]. While possible ranges of F_0 are about 80-400Hz for males and 120-800Hz for females, they usually vary between 80 and 250Hz for males and 150 to 400 for females [Benesty et al. 2008]. Children F_0 usually varies in the 200-600Hz range, but can go much higher.

4.2 Constituents

Intonation is the variation of pitch aiming at describing attitudes and emotions of the speaker, modality of the utterance. It is one of the main constituents of prosody. Pierre Delattre [Delattre 1966] established a classification of the 10 main types of intonation in French: finality, major or minor continuation, implication, order, question, interrogation, aside, echo and exclamation. Modality can be declarative, exclamatory or interrogative.

The accent component of prosody is what emphasizes elements of the sentence, usually a word or a syllable. Most often, the main way accent is expressed is through a greater amplitude (sound level)[4] and a longer duration (resulting from an accentuated pronunciation). Accent is usually used to define three categories of languages [Lolive 2008]:

- Languages where emphasis is not placed at a fixed position, like English.
- Languages where emphasis placement is not free but is constrained by the number of syllables. French is part of that category.
- Tonal languages, where tone is used as a semantic information.

In French, final accent – on the last syllable is particularly important.

A last constituent of prosody is flow rate, which can, when varying, put emphasis on some part of the spoken message, underline hesitation or insistence or translate some emotion. It is very influenced by pauses. In French, the flow is about 4 to 7 syllables per second.

Prosody is also characterized by the phenomenon of microprosody. As speech is subject to production constraints (*cf.* section 2), prosody is also subject to variations with the nominal values. These variations are called microprosody or micromelody.

[4] More specifically, for a periodic signal, amplitude is the maximum value of the signal. Here, it can be assimilated to the highest sound level for a given cycle. As speech sound waves change through time, so does frequency and amplitude.

5 Spectral Analysis

In the temporal domain, speech is defined – as for other sounds – by the ambient sound level; sampled at a rate usually varying between 8 and 48kHz, depending on the finality. Speech temporal domain characteristics can therefore be studied, but a lot more information is observed in the frequency domain. Such study is mainly based on the analysis of spectrograms, which are time/frequency representations of speech. Time is the x axis while frequency is the y axis. The diagram shows the quantity of energy carried by frequency bands in function of time through a color or gray scale.

Phonemes can be identified on spectrograms based on their spectral, characteristics, the most important being the formants. They are the harmonics that correspond to spectral maxima, *i.e.* frequency ranges possessing peaks of energy. They are called F_1, F_2, F_3, F_4, *etc.*, F_1 being associated to the second lowest frequency of the signal after F_0 and the others being associated to increasingly higher frequencies. Some of them, the main formants F_1 and F_2 for instance, can also be put in relation with physiologic events [Calliope 1989]. F_1 is correlated with mouth opening, and ranges between 320 and 1000 Hz for the different vowels in French. A relation exists between F_2 and the position of the tongue in the mouth. F_2 ranges between 800 and 3200 Hz for French vowels. Formants named F_1 and F_2 are very useful for identifying vowels. F_3 is correlated with the configuration of the lips for vowels. Other formants F_4, F_5, *etc.* are of a more limited use.

6 Conclusion

In this chapter, we briefly presented the basic concepts on which TTS, and more precisely unit selection, relies. We first focused on speech signal, actually defining what is to be reproduced by the synthesizer. In order to introduce one of the main issues in concatenative synthesis – and therefore in unit selection, we presented the notion of speech unit, in particular the diphone. Then, we defined the notion of prosody, detailing its main components and origin, hence showing its importance in a reconstructed speech stimuli. As an introduction to the next chapter, we described the human vocal apparatus and its main components. Finally, we briefly presented the fundamental notion of formant.

Chapter 2

A History of Text-To-Speech Synthesis

"We shall never cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time."

Thomas Stearns Eliot (1888–1965)

Many see in the usage of pipes leading to idols mouths – meant to feign the divinity's response to worshipers' requests – in antique Mediterranean societies the first expression of human desire, if not to recreate speech, at least to make an object talk. But it is only during the Age of Enlightenment that first serious work was made, this time with more selfless goals in mind. From baron von Kempelen's talking machine to Statistical Parametric Speech Synthesis or unit selection text-to-speech synthesis, a number of original ideas have been explored to make machines produce human speech. On evolutions to revolutions, speech synthesizers first tried to mimic the vocal apparatus, then recorded or modeled phonemes and tried to paste them together. They used rules to guess formant trajectories or tried to create models of the human vocal system. Finally, they analyzed speech data to learn how to speak. Or they searched that data to find and paste the right pieces. The scope of Text-To-Speech technologies for synthesis is vast and complex, let's dive into the matter.

The history described in the present chapter is mainly based on three reference articles on the subject: [Flanagan 1972], [Klatt 1987] and [Schroeder 1993], at least for work earlier than the 80s. The description of von Kempelen's machine comes mostly from his book [Kempelen 1791].

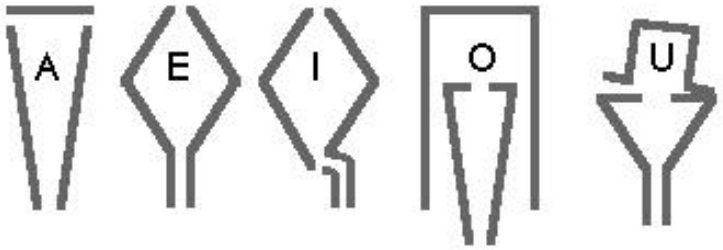

Figure 2.1: Kratzenstein's resonators for synthesizing five vowel sounds. Air was sent inside resonators by blowing a reed attached at its extremity [Flanagan 1972].

1 Inception – Reproducing What Works

The first real efforts to reproduce artificially human speech, during the renaissance period, come bundled into the larger scope of speech study. Hence, the first documented work aiming at reproducing speech artificially comes at the end of the Age of Enlightenment, with two simultaneous attempts by physiologist Christian Gottlieb Kratzenstein in Russia and Baron Wolfgang von Kempelen in Austria.

1.1 Kratzenstein's Resonators

In 1779, the annual prize of the Imperial Academy of St. Petersburg subject was, according to Flanagan [Flanagan 1972]: "(1) What is the nature and character of the sounds of the vowels *a, e, i, o, u*, [that make them] so different from one another? (2) Can an instrument be constructed like the *vox humana* pipes of an organ, which shall accurately express the sounds of the vowels?". Christian Kratzenstein, the winner of the contest, constructed 5 resonators, which dimension was similar to the human apparatus, each one meant to reproduce one of the 5 required vowels. They are pictured on figure 2.1. Each resonator reproduced the believed configuration of the mouth and larynx. They were activated by blowing into a vibrating reed, on top of which sat the resonator.

The overall accuracy was qualified "tolerable" and though the resonators answered the second part of the question. The first one, on the processes at the origin of each vowel sound, wasn't answered.

1.2 Wolfgang von Kempelen's Talking Machine

At the same time in Austria, Hungarian baron Wolfgang Ritter von Kempelen had been undertaking much more thorough research since 1769. In 1791, he published a book, "Mechanismus Der Menschlichen Sprache Nebst Beschreibung Seiner Sprechenden Maschine" (The mechanism of human speech followed by the description of a talking machine) [Kempelen 1791][1], where he gathers the results of his more than 20 years work.

[1]Though the original book was in German, a French translation was also released at the same time, which title was "Le mécanisme de la parole".

In his book, he describes the machine he built to synthesize speech, shown on figure 2.2. The construction of the machine aimed at reproducing the principal components of the human apparatus. The bellows, shown on the central frame reproduced the lungs; the pipe inside the wooden box they blew into reproduced the larynx and comprises a reed reproducing the glottis and vocal cords [2]. The elements attached at the extremity of the pipe – detailed on the top and bottom frames – reproduced both nasal and oral cavities. The conical element at the extremity of the machine was carved out of elastic gum – a matter von Kempelen chose for its excellent elastic properties – and could be shaped by the left hand of the operator. The right hand went on top of the wooden box, fingers taking place on the two spindles and two holes (respectively named s, sch, m and n on the bottom frame) allowing the elbow to take place on the top of the bellows. The spindles are used for the production of unvoiced fricatives. The two holes represent nostrils that can be plugged on the same principle as a flute.

The machine was said to produce honorable accuracy in the reproduction of human speech, but was very difficult to master. Despite its degree of elaboration, the machine couldn't reproduce all plosives, and von Kempelen used phoneme [p] as a replacer. The conical end in elastic gum of the machine was described by its creator as very imperfect as it missed lips, teeth, tongue and reproduces the palate properties quite poorly.

1.3 A Period of Stagnancy

Most attempts to make a talking machine throughout the XIX^{th} century and the beginning of the XX^{th} ended up being improvements of von Kempelen's machine, with various degrees of success. Among the best advances are sir Charles Wheatstone's work, who created an improved version of von Kempelen machine. Then Alexander Graham Bell and his father Alexander Melville refined Wheatstone's machine by adding rubber lips and a wooden tongue. The century wasn't empty for speech synthesis, as Wheatstone, Robert Willis and Herman von Helmholtz laid down the basics of modern understanding of the acoustic processes involved in the production of speech.

In particular, the rise of telecommunication technologies drew a new type of researchers to consider speech synthesis. Before the 30s, speech synthesis was a tool used by physiologists to understand the mechanisms of voice production and test their theories. With the arrival of telecommunication technologies and electronics, people started to think of speech synthesis as part of a vocoder, or voice coder that would ease transmissions bandwidth by translating voice into a set of commands that could be transmitted and reproduced (re-synthesized).

[2] That part of the machine is directly inspired by the acoustics of flutes.

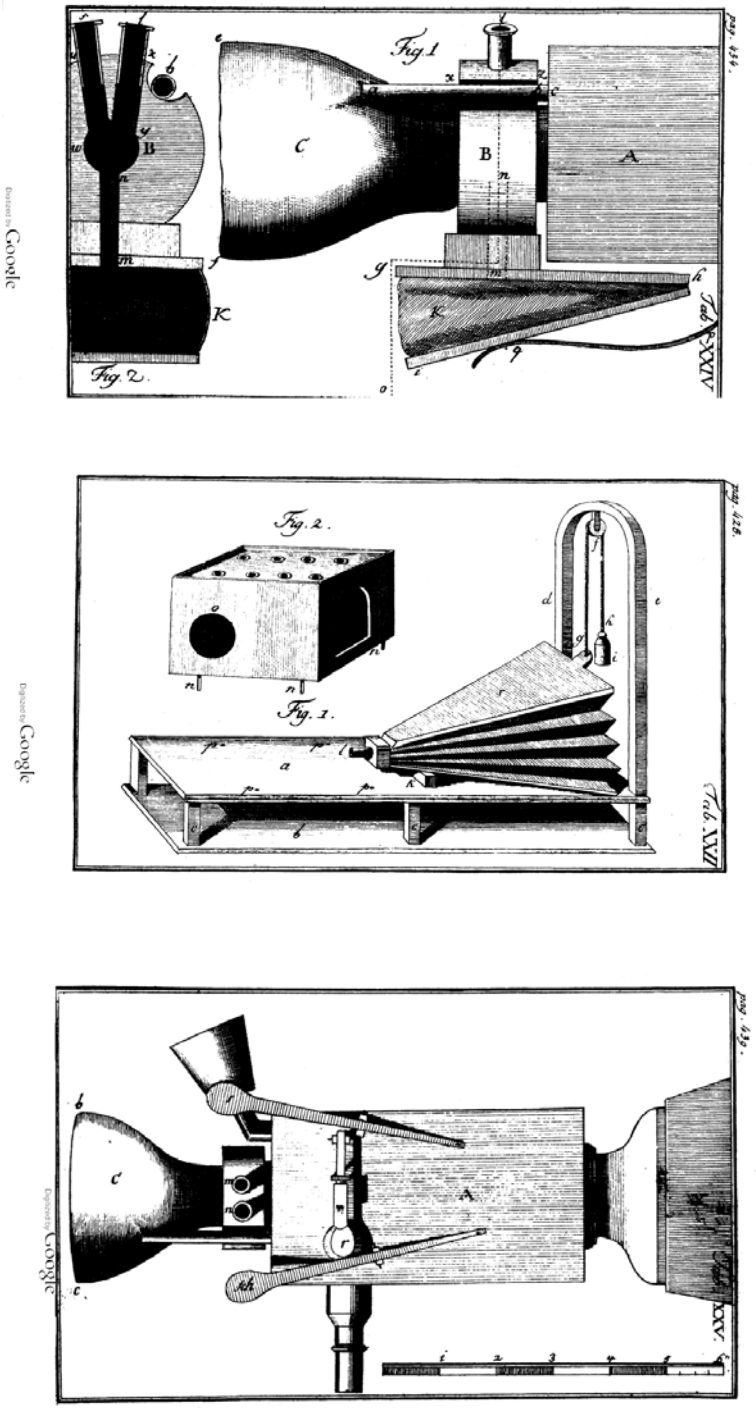

Figure 2.2: Baron Wolfgang von Kempelen's speaking machine. *Top:* Detailed view of the extremity of the machine, reproducing nasal and oral cavities. Part A mainly reproduces the larynx, part B the nasal cavity and part C the mouth. It is elastic and can be modeled by the user's hand to mimic lips articulation. *Middle:* The bellows mimic lungs while the wooden box contains a reproduction of the larynx that include a pipe and a reed. *Bottom:* General view of the machine after the bellows. The two spindles can be operated and the two holes plugged by the user's hand. Figures come from von Kempelen's book published in 1791, digitalized by Google Inc. [Kempelen 1791].

1.4 The VODER

The revolution in the field comes in 1939 with the VODER (or Voice Operation Demonstrator) and the emergence of electronic technologies. During New York Universal Exposition, Homer Dudley from Bell laboratories presented a machine, entirely composed of electronic chips, that was able to produce speech from a bunch of successive narrow band filters (*i.e.* connected in parallel) acting as resonators. It is described on figure 2.3. The device followed the source-tract separation principle, either a voiced or unvoiced sound source could be selected and then modified by the resonance box containing the filters. A wrist bar selected the sound source: a random noise source for unvoiced sounds and a relaxation oscillator for voiced ones. An additional foot pedal was used to adjust the pitch of the oscillator. The filters, each having a resonance similar to individual speech sounds, were controlled through a keyboard of ten keys – one for each filter – that could activate or deactivate them. Filters outputs pass through potentiometer gain controls and are added. Another key controlled the amplifier and three others caused a transient excitation of the filters selected by the ten first keys to simulate stop-constant sounds [Flanagan 1972].

The quality was sufficient to be understood, at least for short utterances. However, operators had to train for at least a year before being able to demonstrate the machine during live sessions.

2 Articulatory Synthesis

All the mechanical methods we described earlier had one point in common: they tried to reproduce the functioning of the human vocal apparatus, and they only got better because we understood better and better the way it works over time. Von Kempelen's, Wheatstone's, Bell's machines, the VODER, *etc.*; all followed that principle. With the arrival of computer era, differential calculus and fluid mechanics allowed a much more formal modernization of the vocal tract, but without notable results on the final quality of produced speech [Mermelstein 1973]. The aim of the technique, nowadays, is to digitally reproduce the human apparatus and all its components and to perform synthesis by simulating its behavior when stimulated by an air flow while taking into account human physiological constraints [Rubin and Baer 1981].

These systems have two major downsides. On the one hand, because of the huge number of parameters that have to be taken into account to reproduce a valid speech apparatus, designers are forced, even today, to carry out a lot of approximations that end up to a severe degradation of the system's efficiency. Hence, final speech ends up being quite poor. On the other hand, systems based on a modelling of the human apparatus are highly computational and the synthesis is still difficult to handle in real time. Those downsides make articulatory synthesis almost impossible to use in real time [Story 2009]. A recent

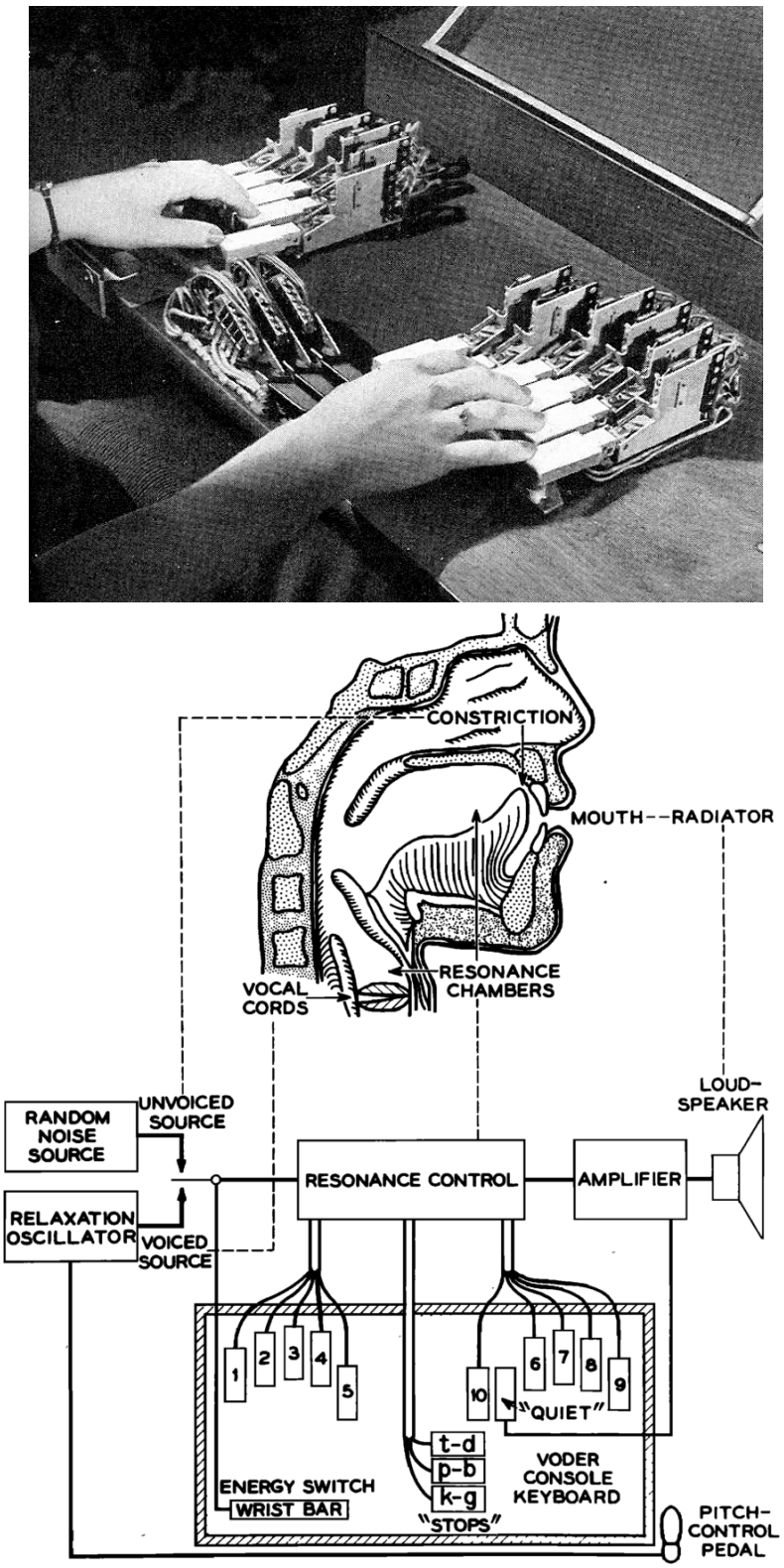

Figure 2.3: *Top:* The keyboard controlling the VODER. An additional foot command controlled pitch. A web page dedicated to the VODER shows pictures and videos of the VODER at work: http://120years.net/the-voder-vocoderhomer-dudleyusa1940/. The top image is extracted from that website. *Bottom:* The VODER consisted of a bank of electronic filters excited by an impulse train or noise. Image extracted from Flanagan's article on the history of TTS [Flanagan 1972].

investigation in that field, generating intelligible voices, is given by Brad Story [Story 2011].

Today, articulatory speech synthesis continues to live and progress, but not for pure TTS purpose. As a matter of fact, other recent speech synthesis methods outmatch it nowadays. However, articulatory synthesis has other arrows in its quiver. It is a fabulous tool to formulate, implement and test new acoustic models of the human speech apparatus; allowing to verify the positions of articulators, like lips, jaw and glottis with an experimental model. It is now the main focus of articulatory systems, though this use of articulatory synthesis is not recent [Maeda 1979].

With the explosion of computational capacity in recent years though, and with the constant refinement of our knowledge on the speech apparatus, it is possible that articulatory speech synthesis becomes, again, competitive.

3 Rule-based and Formant Synthesis

Synthesis techniques that reproduce the way humans talk are not the only ones, many others were developed along decades. The principle of rule-based speech synthesis is to be able to model the different parameters defining the acoustic signal (historically the formants) by using a set of production rules based on linguistic and phonetic analysis of the utterance. Via these rules, the question is principally to be able to represent frequencies, amplitudes and bandwidth of these different formants depending on the constraints of the text, in particular co-articulation.

Generally, the rule-based approach describes evolution rules for a generation model of speech. The models of the speech signal usually rely on a source-filter representation of speech, which assumes that the speech signal is convoluted, at the larynx position, to a filter (most often linear) that characterizes resonance modes of the vocal tract. On the contrary of articulatory approaches, these systems rely on much simpler models and not on complex constraints on the air flow in a dynamic environment, allowing a drastic simplification of the computations which leads to efficient systems that could, back in the 70s-80s perform in real time. The major disadvantage was the lack of naturalness in the tone which emphases the artificial origin of the voice.

Rule-based and formant synthesis usually go together, the rule-based part generating a representation of speech that's possible to synthesize with a formant synthesizer. The formant synthesizer itself is based on the principle that the main perceptual information is carried by formants. F_1, F_2 and F_3 are generally enough to get an acceptable synthesis. A formant synthesizer uses formant-related information – central frequency, bandwidth and amplitude – to reproduce voiced segments while noise bands may be used for unvoiced speech. Other parameters could be taken into account like nasality. Twelve or more fea-

tures are sufficient to output intelligible speech. As no formant analyzer existed, formant synthesizers used the hand-tuned rules described earlier to perform synthesis.

In 1922, J.-Cl. Stewart made the first formant-based synthesizer. According to Jean-Sylvain Lienard, cited in the Calliope book [Calliope 1989], it was composed of a periodical source and of two electric resonators allowing reproduction of vowels, diphthongs and a few words like "mama, Anna"[3]. The VODER, which contained an analyzer extracting acoustic components from speech, can also be considered an ancestor of formant synthesis.

In 1950, Cooper's "Pattern Playback" synthesizer was able to synthesize speech from a spectrogram [Cooper et al. 1951]. It can be considered as the first full automatic speech synthesis system as no intervention from the operator is needed while reading the spectrogram. The two most famous formant-based systems, Walter Laurence's PAT (Parametric Artificial Talker) and Gunnar Fant's OVE (Orator Verbis Electrics), emerged soon after, in 1953. On the contrary of Pattern Playback, both used distinct electronic resonators for generating each formant. This last point is the key to subsequent formant synthesizers, where a set of resonators are placed either in parallel or in cascade to generate formants [Holmes 1983].

More recently, a famous rule-based system, the Klattalk system [Klatt 1982; Klatt 1987], featured about five hundred rules starting from the letters of the utterance plus an exception dictionary of more than 1500 words which translations to speech were hand-made. They generated 20 parameters featuring F_0, duration and stress, among others segmental level phonological features. Then, the result was provided to a formant synthesizer that generated the speech signal.

These rules could be numerous, complex and needed expert knowledge of the field. For example, a work on duration [Bartkova and Sorin 1987] provides complex trees of rules, modulated by elongation and shortening coefficients, solely for estimating duration.

Historically, rules were hand-made and deterministic and were processed by expert systems. For a long time, the couple rules/formant synthesis was the best in the TTS field but the emergence of unit selection in the 90s overcame these systems. The arrival of HMM-based systems (see section 6.1) and to a further extent SPSS (Statistical Parametric Speech Synthesis) in general in early 2000s brought back these ideas when proposing statistical rules. Sets of HMMs are first learnt on a speech corpus in order to obtain the spectral dynamics of temporal events. HMMs are then used in generation mode to get synthetic spectral observations. A vocoder (voice coder), relying on some *ad-hoc* modelization of speech into features, is used to convert these artificial signals into actual speech.

[3]Translated from French: "constitué d'une source périodique et de deux résonateurs électriques permettant de reproduire des voyelles, des diphtongues et quelques mots tels que « mama, Anna »".

4 Linear Prediction Synthesis

Linear prediction was a competing approach to formant synthesis. It was a speech-to-speech analysis/synthesis method, and was mainly studied for telecommunications [Atal 1971]. It is based on the principle that a speech frame is a linear combination of preceding frames. To compute a new frame, an algorithm computes linear combination coefficients of preceding frames [Makhoul 1975]. The coefficients are computed by minimizing the average quadratic error between real and predicted signals in the temporal domain. Coefficients are updated every 5-20ms [Calliope 1989]. This approach, though producing speech of good quality, carried a buzzy noise due to oversimplification of the vocal source and reproduced some phonemes quite poorly, like nasals.

5 Concatenative Synthesis

The idea that's conceptually the simplest way to generate speech is to peek actual speech samples into a dictionary of prerecorded phonetic units and to join them one after the other to reproduce a given phonetic sequence. The choice of the acoustic unit to record, store in the dictionary and concatenate is one of the main problems these systems had to face. The first ones, in the 50s, used phones [Harris 1953]. The discontinuities in the prosody of synthesized sentences, the inexistent management of co-articulation effects and their poor restitution [Pols et al. 1987] caused the resulting speech to be particularly unintelligible. The search for a better way to perform concatenations ended up providing a new acoustic unit in 1956, invented by Küpfmüller and Warns: the diphone. It was introduced as a "dyad" in [Peterson et al. 1958] and then took the name of diphone in [Dixon and Maxey 1968].

The first concatenative speech synthesis system relying on diphones was developed by Estes et al. [Estes et al. 1964]. Diphone-based concatenative synthesis had a far better quality than phones but wasn't perfect either. So other units were tested, each one longer than the previous: half-syllables [Fujimura 1976], syllables and disyllables. Other attempts were made on sub-phonemic units, for instance half-phones [Conkie 1999]. It is important to take note the impact of these units on the size of the dictionary and thus on the footprint of the system. The consequences on the computation charge to browse the dictionary and get data also are important. For instance, let's say the phonetic alphabet in language that has to be reproduced features 35 phonemes, which is about what you get in French, there are 35 phonemes, theoretically 35^2 diphones (a bit less in reality as some combinations cannot occur), 35^3 triphones (about 43 000), hundreds of thousands of syllables and millions of words. The complexity of the problem and the system footprint are even bigger in the case of systems featuring multi-represented units, *i.e.* the dictionary contains more than one instance of each acoustic unit, in different contexts.

After selection, an algorithm is used to perform concatenations, the goal being to offer smooth concatenation areas. Hence, it often does more than just pasting speech samples one after the other. The usual approach is to mix part of the two speech signals over a few frames before and after the concatenation spot; but other techniques, like the generation of a small signal part to make the link, are possible. Algorithms may then be used to modify parameters of the signal prosody like pitch or duration modification, usually Pitch Synchronous OverLap-Add method, or PSOLA [Moulines and Charpentier 1990]. This aims at getting a signal as close as possible to the prosody that has to be reconstructed and that are determined by linguistic processing of the textual input.

The vision of the unit dictionary evolved a lot through time. The idea of adding several representatives of the same unit (multi-represented units), motivated by the huge differences between units depending on their apparition context, caused an explosion of the size of speech recordings; not in a dictionary anymore this time, but as a long sequence of annotated speech. Annotations, distributed on several levels, range from the allophone temporal start and end marks to the same for syllables and finally for words and named entities at the highest annotation level. It was also permitted, for systems with a large footprint, by the increase of computational and storage power. From this new point of view, the speech database ceases to be a dictionary with sound samples mapped to a base unit. It becomes a continuous speech corpus, and this corpus is annotated on several levels with several unit types (allophone, syllable part, syllable, word, lemma, ...). When a unit is needed, an algorithm will select in the corpus the most appropriate one to the desired context during synthesis. This unit can be anything: a diphone, a sequence of diphones, a syllable, a word, a set of words, even the whole utterance if it appears in the corpus. It can even be any set of diphone, with no relation to any linguistic criterion, no matter the size. This implies making a ranking of units, from the least to the most adapted, hence the need to define a concept of cost to minimize. This principle is called unit selection. It was first presented in [Sagisaka 1988] [4] and is depicted on figure 2.4. Sometimes, it is simply called corpus-based speech synthesis, though some other techniques are also corpus-based (SPSS learns models from a corpus for instance).

It's in the 1990s that those systems using "units of variable size" (referred thereafter simply as units) finally emerged. This is particularly the case of the CHATR system [Black and Taylor 1994; Hunt and Black 1996]. In such systems, one sees coexistence between units of different sizes (diphones, triphones, syllables, *etc.*). Because it generates speech from units actually produced by a human speaker, unit selection systems vehicle the identity of the original speaker much more accurately than systems based on speaker models. This in particular gives a much more natural generated voice. In some cases, however, the absence of an acceptable unit in the database requires the use of another unit (for replacement).

[4] Yoshinori Sagisaka's work presents the concept of unit selection but does not present a working implementation at the time

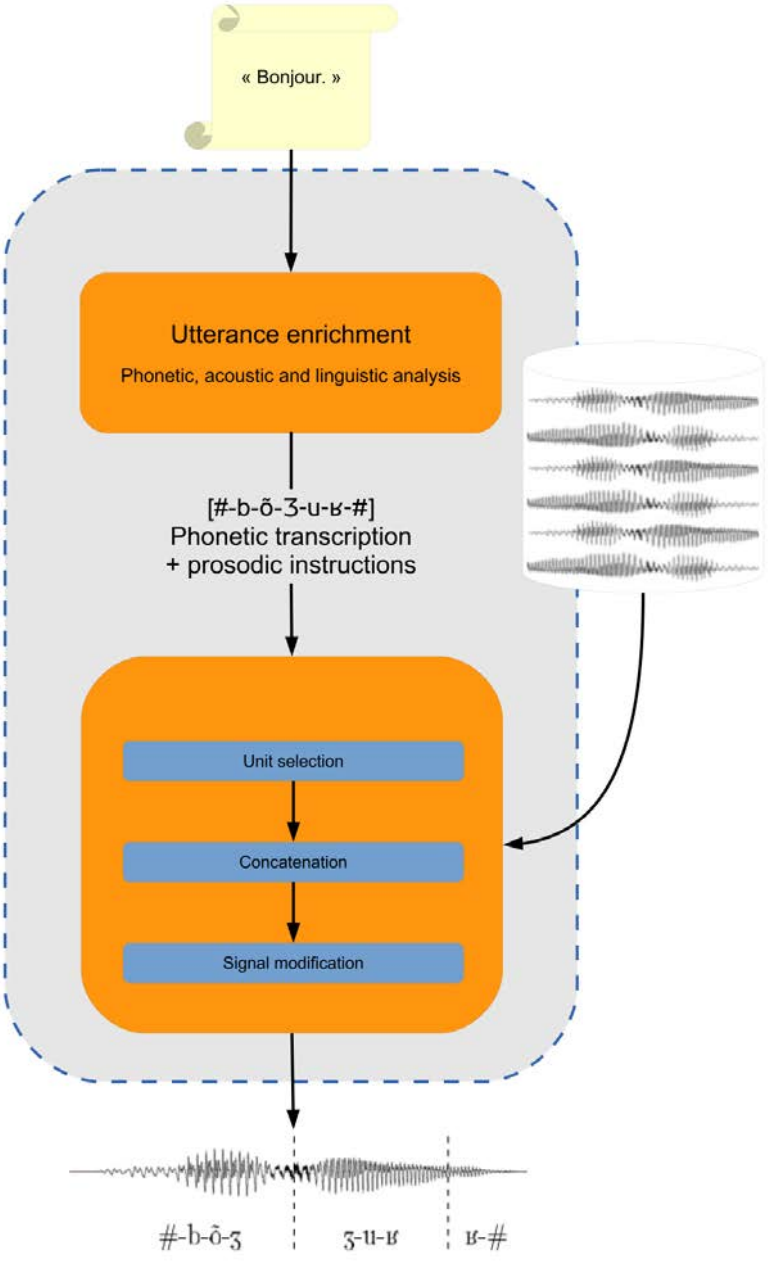

Figure 2.4: The general framework of a unit selection-based TTS system. Figure inspired by [Cadic 2011].

This unit was most likely produced in a very different context that the unit that is needed. This may reduce the quality of the prosody in the synthesized voice and causes glitches to occur at concatenation points with that unit. This can go up to seriously influence the sentence intelligibility in the worst cases. It does occur however only in few cases[5], usually with a small database, and much less if the base is large. It is, however, a weak point of the method, especially if compared to HTS and its siblings (see section 6.1) for which the intelligibility is nearly guaranteed regardless of the circumstances. Another weakness is the rigidity of the method. In recent years, techniques adapting the speaking style of a corpus from one speaker to another (or learning mean voices and then adapting them [Fan et al. 2015]) became popular. In concatenative synthesis, this kind of speaker adaptation is impossible. Moreover, prosody proves difficult to model and control. The reason for this relies in the temporal width of the signal segments the unit selection algorithm takes into account, which is centered on two units (possibly a few more if wide contextual features are used). The ranking provided by the algorithm and its cost function is therefore based on considerations taken on short time periods which totally contrasts with the nature of prosody; for which most components have an effect on the long term: several syllables at least, more generally at breath group level and even at the sentence level. Another reason is that prosody is highly dependent of the speaker, while unit selection cost function components are not. In particular, strict recording conditions, aiming at getting the best concatenation experience, strip corpora of any expressiveness. Nowadays, to give the system more flexibility, large corpora are recorded; sometimes featuring different emotions along with neutral speech.

As our work focuses on unit selection, this technique will be discussed in detail in the next chapter.

6 Statistical Parametric Speech Synthesis

6.1 HMM-based Speech Synthesis

Until recently, with the generalization of model-based synthesis in what's now called SPSS (see next section), only HMMs were used for statistical parametric modelling of speech. This method, named HMM-based synthesis, still is – now with other SPSS techniques – the main competitor of unit selection. This method relies on HMM-based (for Hidden Markov Model-based) speech synthesis. Initiated by K. Tokuda in 1995 [Tokuda et al. 1995] with the HTS (HMM-based speech synthesis) system [Zen et al. 2007], it is still a very active research field, though HMMs are more and more abandoned for DNNs. While being a more recent approach than unit selection, HTS and its pairs can be seen as distant descendants of the couple synthesis by rules/formant synthesis, as the probabilistic models placed in decision trees used in the method are from this point of view the equivalent to the rules

[5] Generally when using uncommon words containing rare diphones.

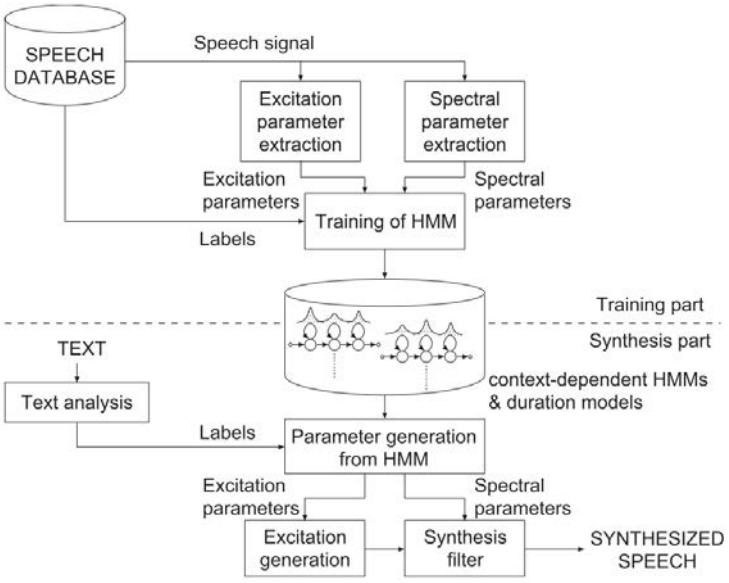

Figure 2.5: A block diagram representing the general learning and synthesis processes in HTS, the main HMM-based TTS system. Figure extracted form [Black et al. 2007].

of synthesis. In addition, HMMs based synthesis uses much more parameters than single formant trajectories.

Decision Trees and H(S)MMs

HMM-based TTS systems depend on two elements: HMMs and decision trees [Yoshimura et al. 1999]. The HMMs (HSMMs in reality, for Hidden Semi-Markov Models) are used to model spectrum and pitch information. In the HTK framework, which HTS relies on, one HMM corresponds to one phoneme. In order to make sentence reconstruction possible, each HMM have to be concatenated. To enable this, HTK adds two non-emitting states at start and end of the HMM; on which concatenation is done [Le Maguer 2013]. Every state of the Markov models is associated to a decision tree describing the different prosodic and linguistic contexts (constructed through predefined set of features, also called questions) affecting this state. Each node of the tree corresponds to one contextual property of the question set. Each leaf contains a statistic distribution that's refined during learning. During synthesis, the leaf that corresponds to the right context is selected. The emission probability of its associated HMM state then becomes tied to the statistic distribution contained in the selected leaf. Another decision tree, independent of the HMMs, is used to define the duration of the various states [Yoshimura et al. 1999].

General Process of the Synthesis

The general process of both learning and synthesis tasks is summarized on figure 2.5. First of all, learning the model (HMMs and decision trees) is made with a speech corpus. Though

a very small corpus is sufficient to learn a voice (one of the method main strengths), it is preferable to have a few hours of speech in the learning corpus; as more data in the corpus yields to better modelling and therefore better speech quality. In addition, SPSS is quite sensitive to the quality of annotations (more than unit selection) and voice quality is strongly impacted by the quality of automatic annotation tools. Speech does not necessary come from one and only one speaker contrary to unit selection: both mono-speaker and multi-speaker modeling is possible [Yamagishi et al. 2009]. One can even learn an average voice of many different speakers, then adaptable to a particular voice with only a few data (100 utterances of the target voice is usually enough) [6]. Synthesis process begins the same way as unit selection: target textual utterance is parsed and converted into a sequence of descriptors on several levels (phonological and linguistic plus some prosodic features). From this sequence of descriptors, a HMM matching the sentence to produce is reconstructed by concatenating HMMs retrieved by exploration of the decision tree. Then, this new model is used to synthesize the trajectories the speech signal will take on each parameter taken into account in the system (parameters of the vocoder). As an example, the HTS system features include: MFCC coefficients, aperiodicity coefficients [Zen et al. 2007], prosodic parameters, F_0 (fundamental frequency), *etc.* Finally, parameter trajectories are provided to a vocoder, usually STRAIGHT [Kawahara et al. 2008] or SPTK[Fukada et al. 1992], to generate the sound signal.

HMM-based synthesis has the advantage of providing a very intelligible voice in almost all contexts. It is also very adaptable as speech is represented by a model. Production of creaky or muffled voice for instance, with few data only, is possible through adaptation of a model learnt with another voice corpus. However, its overall quality remains quite lower than unit selection. In particular, synthetic voices feature a buzzy background noise due to an oversimplification of the model used to describe speech in the vocoder. The voice also appears muffled usually. This is due to the over-smoothing of speech frames during training of the statistical model [King 2010]. These systems are therefore, for the moment, less close acoustically to the identity of the speaker's voice used for learning.

6.2 The (Re-)mergence of DNNs

More recently, in the last three years, parametric synthesis has been the epicenter of a new revolution: the great return of deep Artificial Neural Networks (ANN) to the stage. The use of neural networks in both ASR and TTS fields is not new, and a lot of work has been done in the 80s-90s to model speech or some of its components with them. We will discuss it further when introducing our work on duration modeling for unit selection, but let's just say the use of ANNs for modeling duration has been investigated in the 90s. Work was done to explore the feasibility of using Deep Neural Networks (DNN) for ASR in the 80s, with

[6] The possibility to build average voices raised an interesting question: "Is the average of all male/female voices the best male/female voice ?".

limited results. At that time, HMMs proved to be better and work focused on them. Today, DNNs achieve better performance than HMMs in many domains. So what did change between then and now? Almost nothing, basically, if not the number of features that drastically increased to reach several hundreds and even more than a thousand sometimes. We speak of DNNs today but they already existed in the 90s, it wasn't only about simple ANNs. What really changed is the size of learning/validation/test sets. Learning a DNN on a few tens of minutes of speech seems meaningless today, though studies twenty-five years ago rarely used more. But the evolution of both storage and computational power (especially recent improvements on GPUs) now allows learning of deep neural architectures over several hours or even tens of hours of manually annotated speech.

Currently, systems use DNNs, RNNs (Recurrent Neural Networks - a variant of DNNs) or more specifically LSTM-RNNs (Long Short-Term Memory Recurrent Neural Networks) both in learning and generative ways. LSTM-RNNs seem to show the best results so far [Zen and Sak 2015]. DNNs may be used either for mono or multi-speaker learning, as in [Fan et al. 2015] where a 3-layers (hidden) DNN is learnt, the DNN having as much output layers (79 features per output layer) as the learning set has speakers. In particular, this work shows the adaptation of such a multi-speaker model to the acoustic space of a speaker who has an unusual way of speaking [7] actually keeps the identity of the speaker as intended but also tends to correct the speaker's pronunciation problems.

Nonetheless, it would be quite short-sighted to think DNNs are the solution to all problems in synthesis (regarding the current trend to put neural networks everywhere, the question seems valid). In a study on the potential benefit of integrating DNNs in HTS, [Hashimoto et al. 2015] shows that DNNs handle the learning task better than the usual decision trees and HMMs, but show lesser performance than SPTK's Maximum Likelihood Parameter Generation algorithm (MLPG) for trajectories generation. In that study, the couple DNN and MLPG shows the best performance while the usage of DNNs for trajectory generation (along with HMMs for learning) shows worse performance than the usual HMMs and MLPG.

7 Conclusion and Graphical Summary

To conclude this section, we present a chronological summary of speech synthesis in figure 2.6. Basically, over the last 200+ years since Kratzenstein and his acoustic resonators, speech synthesis techniques explored three main ways to produce speech. The first one, historical, is to reproduce the human articulatory system. The second, mainly developed for telecommunication purposes (at least in the beginning), is to build/learn models that can afterwards be used in a generative way. Methods based on that point of view have the

[7] In mandarin in the experiment.

advantage of being very flexible and have a very small footprint. The third way to produce speech utterances is by concatenating pre-existing speech. unit selection, which is part of that category and is the subject of our work produces the best sounding synthetic speech nowadays, but has a very large memory footprint, isn't adaptable to new voices very quickly and is less flexible than leading SPSS techniques. Finally, the wish to make SPSS/unit selection hybrids, where the unit selection is guided by trajectories/parameters produced by a statistical parametric model is always tempting. Basically, before the 50s, one of the main objectives of most studies on speech synthesis was to produce proof of concepts. Once this step was passed, around the 50s, and with the increasing interest in speech synthesis, the main issue became the construction of very intelligible speech synthesis systems. This step was reached during the 80s/90s, and research then focused on speech quality, especially for neutral speech; and more generally quality of the message carried by synthetic speech prosody, especially expressiveness.

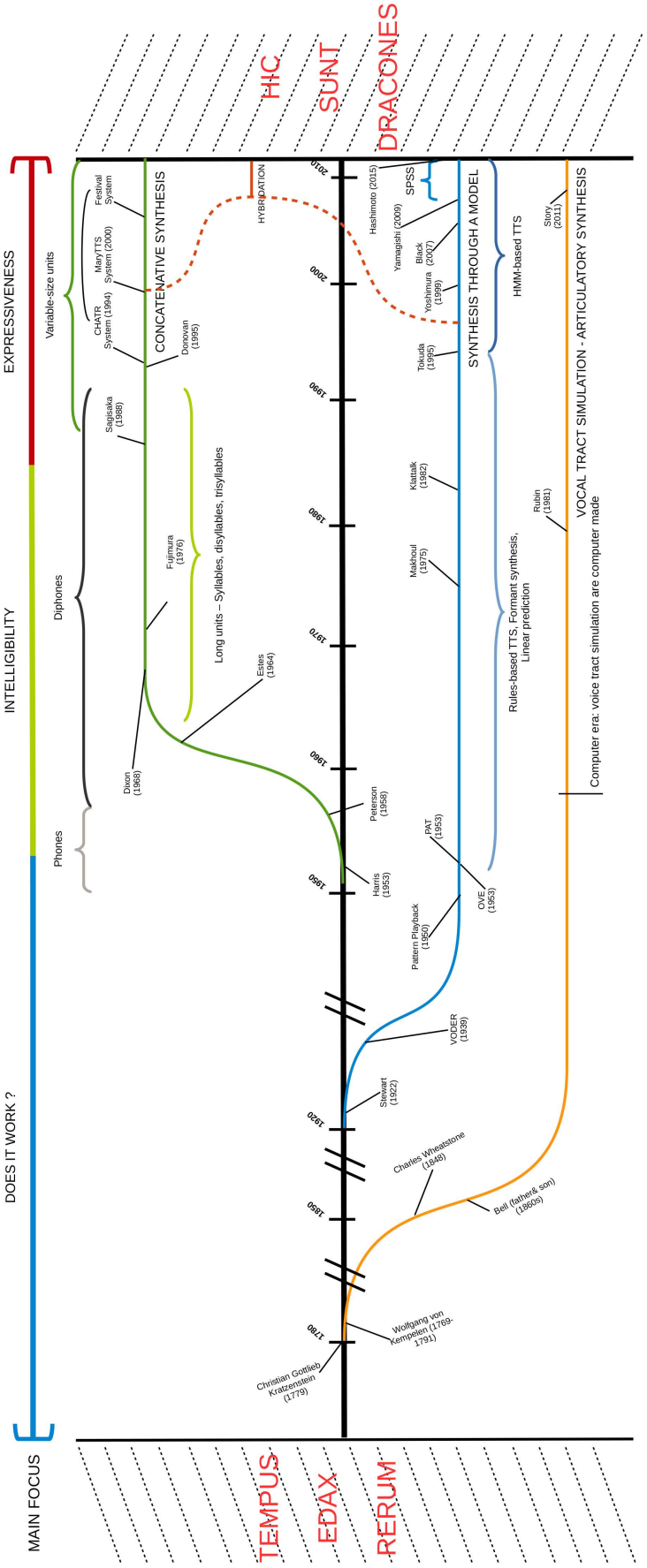

Figure 2.6: Timeline of the speech synthesis field and main research focus evolution.

Chapter 3

The TTS Frontend and Corpus

"From a drop of water a logician could infer the possibility of an Atlantic or a Niagara without having seen or heard of one or the other."

Arthur Conan Doyle (1859–1930)

From *now on, our main focus shall be on unit selection speech synthesis. In the two next chapters, we will describe in detail the complete organization of a speech synthesis system based on that principle, describing each block and the main technics explored with unit selection. We will organize this chapter in two parts. In the first one, as a reminder, we will come back again on the general organization of unit selection systems. In particular, we will describe the TTS frontend, which shows how the input text is transformed into a data sequence fit for the synthesizer's own input. As little work was carried on the frontend during my PhD, we will not discuss it further. The second one describes the corpus building process, along with the issues it brings. This part is particularly important because unit selection is heavily dependent on the corpus, especially the method used to make it. In addition, there is a link between the corpus creation method (covering features especially) and the unit selection strategy (cost function particularly).*

1 Conceptual Overview

As we saw in the last chapter, any TTS system may be divided into a frontend and a backend part. This is shown on figure 3.1.

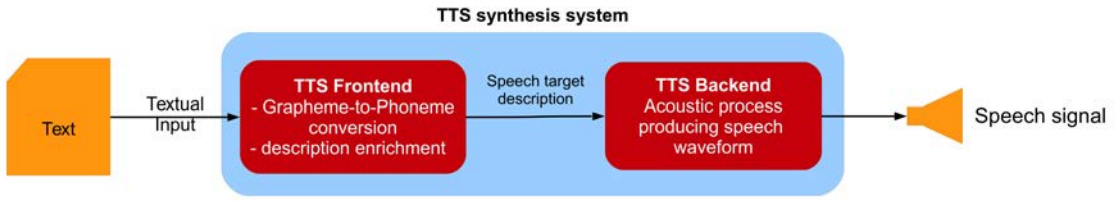

Figure 3.1: General block diagram of a TTS system.

1.1 Frontend Block

The frontend is responsible for translating the user's input into a message that's actually understandable and rich enough in terms of information to be used by the TTS system for speech generation. This means first processing the text in order to correct or delete words and punctuation mistakes. This allows to get valid sentences with valid punctuation symbols and pronounceable words. Lemmas can then be derived from words. This step is called tokenization as the output is a vector of normalized entities also called tokens.

Once the input utterance has been tokenized, it is processed by the phonetizer, in order to get a vector of phonemes representing the oral version of the utterance. This is usually called the grapheme-to-phoneme conversion task. Finally, the sequence of syllables that matches with generated phonemes is extracted using a syllabication algorithm.

In some cases, some additional processes may be done, like adding target prosodic information predicted from the text or provided by the user (context for generating expressiveness for example).

It is interesting to note that, though whe focus on Text-to-Speech synthesis, the input of synthesis processes is not necessarily raw text. For instance, Steve J. Young introduced in 1979 the notion of Concept-to-Speech synthesis [Young 1979]. In that work, a concept is a sentence portion for which we know the syntax. Sentence "Georges irons his blanket.", for instance, can be transformed into the single concept (.IRON, Georges, blanket) where verb "IRON" is a function taking two parameters "Georges" and "blanket". Infering these concepts from text is possible, though not straightforward.

1.2 Backend

The TTS backend, often referred to as "the synthesizer", aims at transforming this description of the target utterance into an acoustic voice signal. Examples of such a thing are the formant, linear prediction, diphone-based, statistical parametric speech synthesizers we saw in the last chapter; and of course unit selection.

Let us now consider the case of the unit selection backend from now on.

2 The Corpus

Before addressing the unit selection problem, let us focus on the speech corpus. After getting the utterance processed by the frontend, the corpus is the only external component the unit selection block will use.

2.1 Presentation

The corpus can be represented as a long strip of speech, annotated on several levels with linguistic, phonetic, and acoustic information. The choice of what to include or not in it is not obvious at all. Neither are the size of the corpus, its encoding quality, the inflexion the speaker should use, the literary genre to employ, *etc.* In order to build a quality non-specialized voice, several hours of neutral speech are a minimum for unit selection. But the content of the corpus itself, in terms of phonological units, has to be varied and rich enough in terms of current phonological units in the target language to guarantee that the Unit selection algorithm will find units that are sufficiently close to the target.

The corpus size depends directly on two elements: the aim of the voice and the technical constraints of the platform where it should be deployed.

The aim of the voice characterizes the way the voice will be used. The main question is: will it be used for some domain-specific synthesis or does it have to be generalist, or is it in the between? For instance, a GPS voice comes with many prerecorded sentences/words from the driving vocabulary like "road", "roundabout" or "turn". But as it may also have to pronounce street or city names, it has also to provide more generalist units, like a basic diphone covering. The other question is: what is the target audience? A voice reading books to children will probably not be fit for the same task with adults.

For the technical constraints, it results from four main axes:

- Does speech have to be generated in real time?
- What are the computational power, storage and RAM access speed of the device that will have to perform the synthesis task?
- What is the storage capability available on the device?

Once the size and desired content have been selected, the corpus is recorded by a speaker (usually a professional actor), who reads a *recording script*. The recording script is the list of all sentences contained in the corpus, accompanied by reading constraints and instructions. For instance, a same sentence can appear twice in the script, first asking the reader to emphasize a particular word and then without that constraint.

2.2 Corpus Creation Methods

To create that recording script, two methods are widely exploited in the literature. The first one is corpus condensation, the second one being sentence construction. Corpus condensation is by far the most explored in the literature and consists in selecting one by one the sentences that improve most some covering strategy (*cf.* section 2.3) within an initial bigger corpus (made out of books, letters, political speech, theater plays, *etc.*). The other one is sentence construction. Its aim is to generate sentences of the new corpus that maximize the covering objectives. The problem in that case is to generate sentences that may be pronounced in a real context, for the speaker would otherwise have difficulties pronouncing a sentence that's deprived of any sense.

2.3 Corpus Condensation

The corpus condensation method assumes that a corpus of considerable size is available, we will call it the initial corpus. This corpus is supposed to present a very wide range of linguistic, phonetic and prosodic attributes, but is too huge to be recorded. The goal of corpus condensation is to solve that situation by reducing the size of the corpus as much as possible. This is usually done by choosing the n utterances, usually sentences, that contain the most interesting attributes. The corpus condensation hence forms a tradeoff between size and abundance of covered attributes.

Problem Representation

In the corpus covering problem, the starting point is a huge initial corpus in which one has to pick up the most interesting utterances, which optimize a cost function. To find the most interesting utterances, a set of attributes must be defined. For instance, a corpus covering objective can be to have in the final corpus a covering of some unit distribution (phonemes, diphonemes, syllables, vocalic sandwiches – *cf.* chapter 9 –, *etc.*). Other objectives than unit distributions may be used as well. For instance, expressiveness type of an utterance: depending on the aimed content of the corpus, it may appear useful to have a certain amount of utterances pronounced with different emotional states. So in order to realize corpus covering, a criterion has to be selected. The cost function to optimize is usually the following:

$$utterance\ cost = \frac{number\ of\ new\ attributes\ covered}{utterance\ length} \tag{3.1}$$

Given an initial corpus including a total of n utterances and an alphabet of k units to

cover, the problem can be represented by the following matrix:

$$A = \begin{array}{c} \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{array} \begin{array}{c} \begin{array}{ccccc} u_1 & u_2 & u_3 & \dots & u_n \end{array} \\ \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \dots & \alpha_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{k1} & \alpha_{k2} & \alpha_{k3} & \dots & \alpha_{kn} \end{bmatrix} \end{array} \tag{3.2}$$

where each utterance u_1, u_2, ..., u_n is represented by a column of values indicating the number of times it matches covers attribute $a_i, i \in [1, k]$. In other words, for $i \in [1, k]$ and $j \in [1, n]$, α_{ij} represents the number of times attribute a_i is covered in utterance u_j. [1] Using that matrix, the question is: which utterances should be chosen to cover attributes $a_1, ..., a_k$ with a minimal number of utterances?

Let B be the column vector of constraints a set of utterance must meet to be considered as a solution to the covering problem:

$$B = \begin{pmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_k \end{pmatrix} \tag{3.3}$$

where b_i is the minimal number of occurrence of attribute a_i in the corpus. Let also $X \in \{0, 1\}^n$ be a column vector of binary values:

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix} \tag{3.4}$$

where $x_j = 1$ if utterance u_j is part of the condensed corpus and 0 otherwise. Hence, the corpus covering problem can be defined as the search for the vector X that best minimizes the sum of all selected utterance costs:

$$X^* = \underset{A \cdot X \geq B}{\arg\min} \left(\sum_{j=1}^{n} c_j \cdot x_j \right) \tag{3.5}$$

where c_j is the cost of utterance u_j (using for instance the cost function defined in equation

[1] Due to the usually huge quantity of units to cover, a lot of zeros appear in each column. From a global point of view, only a few percent of the α_{ij} differ from 0. Matrices are hence represented as sparse matrices.

3.1).

This problem is actually a specialized case of the Set Covering Problem, or SCP (which is known to be NP-complete). For this reason, corpus condensation has to be performed on a corpus and with a vector of attributes of reasonable size (*e.g.* covering phones, diphones and the most common syllables in a language with an initial corpus of 5 million sentences and having at least 100 occurrences of each diphone).

Covering Objectives

In order to build the corpus, two strategies may be used:

- The size of the final corpus is fixed to a certain corpus size that must maximize the covering of the attributes (with $a_i = 1, \forall i$) while minimizing utterance cost (a cost has to be set in order to avoid extremely long sentences).

- The final corpus must cover the constraints up to a certain acceptable percentage. A minimum corpus size is also set.

For measuring the level of covering, a handful of criteria may be used, in particular:

- Having x_i realizations of constraint c_i within the final corpus. For example, a constraint could be to include at least x_i times the diphone [ub]. Diphone synthesis dictionaries in the 90s were based on that principle.

- Using a natural distribution to define x_i. It can be one existing in the initial corpus, like A. Krul *et al.* did [Krul et al. 2006]. More generally, the goal can be to keep the quality of the initial corpus covering within a smaller one.

Several criteria may be used for the same covering. For unit distributions, Zipf and Zipf-Mandelbrot laws, describing the appearance frequency of a base unit (phone, diphone, *etc.*) in a language, have to be taken into account: One has to privilege the most frequently used units in the target language[2], most frequent units being present in a wider number of contexts while the least frequent appear in less contexts. They may even be absent depending on the constraints on the database size.

In any case, the choice of the constraints to cover is not trivial and may have a significant impact on the final corpus size. A diphone covering is of course the minimum, but is not enough to obtain smooth concatenations.

[2]This can only be a sub-set of the language. For instance, in order to produce speech for a telephone dialog voice, it is preferable to use the units that are frequent in telephone conversations or in text messages rather than those frequent in poems or literary work.

Covering Algorithms

In order to get the final corpus from the initial one, given a set of constraints, two algorithmic approaches are relevant: approaches that rely on heuristics with no guaranty of optimality (greedy one mainly) and approaches with heuristics giving a guarantee. These algorithms have three optimization axes: the quality of the final corpus, the size of that corpus and the time needed to find a solution. The algorithm has to find the right tradeoff between those axes. Greedy methods are the most popular, as they are easy to implement and yield good results. The most popular versions of the algorithm are the following:

Agglomerative: The agglomerative is the most popular way to build a corpus. It consists in iterating over the initial large corpus, picking up the utterance that best improve the score function, until these objectives are met. The final corpus is composed of the selected utterances. Jean-Luc Gauvain *et al.* [Gauvain et al. 1990] presented such an approach.

Spitting: The reverse, as it is presented in [Francois and Boeffard 2002], is also possible. Instead of adding utterances to the final corpus, the spitting algorithm starts from the initial big corpus and removes the least helpful utterances iteration after iteration, until no utterance can be removed without breaking the covering criteria.

Pair exchange: The pair exchange algorithm differs from the agglomerative and spitting algorithm in the sense that it tries to optimize an existing corpus by exchanging its content with the initial corpus. This initial corpus can either have been constructed randomly, as it is done in [Kawai et al. 2000] for instance; or built with another corpus building algorithm [Francois and Boeffard 2002]. It works as follows: an initial set of utterances is extracted from the main corpus to build a first version of the target corpus. The algorithm then iterates over the main corpus and, at each iteration, picks up an utterance that isn't in the target corpus and selects another, this time in the target corpus. The two utterances are switched and the new corpus covering is computed. If it is better than before the switch, the exchange is validated; otherwise, it is discarded. Finally, the algorithm terminates when the covering goal is reached.

As said in the last point, combinations of these algorithms are possible. The comparison Hélène François and Olivier Boeffard carried out in 2002 showed that the best greedy approach is an agglomerative algorithm followed by a spitting one [Francois and Boeffard 2002]. They also emphasized the slowness of the pair exchange algorithm. The comparison was made using a corpus of 3 000 French sentences over several comparison criteria (sentence usefulness for instance).

Other Approaches

While greedy approaches are easy to implement and give good results, other ones may be used that can yield to better covering and provide some guarantee over the quality of the

covering. Chevelu *et al.* [Chevelu et al. 2007; Barbot et al. 2012] present an algorithm that builds a corpus based on the principle of Lagrangian relaxation. The algorithm is called LamSCP for Lagrangian based Algorithm for Multi-represented SCP and gives some indication of the quality of the covering that is obtained. Lagrangian relaxation has the advantage of producing an optimal solution to the Set Covering Problem for problems of reasonable size, which is unfortunately not the case for greedy implementations. In the case of greedy techniques, the order of complexity is too high, with typically thousands of sentences and millions of units. LamSCP presents the advantage to provide a value for the cost of the optimal covering.

The algorithm is composed of three phases, first computing an approximation of the optimal Lagrangian multipliers vector and a first under-optimal solution (obtained by computing a first solution to the problem with a greedy algorithm). This under-optimal solution is assimilated to an upper bound to the cost function evaluating the quality of the covering. The second step consists in an exploration of the neighborhood of the first solution and greedy functions process the neighbor Lagrangian vectors to obtain the new covering, the best one becoming the new upper bound. The third phase aims at reducing the size of the problem with the help of heuristics. After these 3 phases, the algorithm comes back to phase 1 in a loop that lasts until either the residual sub-problem is empty or if the Lagrangian function becomes more expensive that the best solution yet found (which acts as an upper bound).

The real quality of the final solution given by the algorithm can therefore be assessed and in their tests, the authors get coverings only 0.8% bigger than the optimal phones and diphones covering. Furthermore, it gets a 10% better covering than a standard greedy algorithm (agglomerative followed by spitting algorithm).

2.4 Sentence Construction

An entirely different approach is described in Didier Cadic's work [Cadic et al. 2010; Cadic 2011]: sentence construction. In all the strategies presented earlier, sentences where considered as atomic, *i.e.* they couldn't be cut. The problem consisted in knowing, for the bigger corpus, whether or not a sentence had to be included into the final corpus. The idea of the sentence construction problem, as its name indicates, is to build sentences appearing in the final corpus. Sentences are built so that they increase covering as much as possible. Having the possibility to create sentences allows the algorithm to increase corpus covering more easily than corpus covering methods.

The difficulty with this method is that all sentences cannot be constructed. A first constraint is that the sentence has to be pronounceable, for obvious reasons. But the correctness of the sentence grammatical structure is important as the speaker would find it

difficult to read a sentence that is grammatically wrong. In that case, the laboriousness of the task would make him/her bored or exhausted quickly. But this is not enough, because sentence prosody also has to be accurate, which means the speaker should read sentences that have a minimum of sense. If the meaning of the sentence is aberrant, the speaker might be perturbed in his reading.

As a fully automatic algorithm that builds sentences is very difficult to design, an intermediate – semi-automatic – method is often chosen. In his thesis [Cadic 2011], Didier Cadic describes an approach based on sentence building that uses a distributed architecture with Weighted Finite State Transducer (WFST) that builds and proposes sentences to a number of human operators. The latter can either validate or reject each sentence and all validated sentences are added to the final corpus. This method yields very good results, with a consequent densification of the corpora, but it is very costly both in time, money and human intervention. It also caused a degradation of the sentences consistency.

2.5 Corpus Building Method Choice

To conclude on the corpus construction problem, one can say the choice of the corpus building method is directly linked to:

- Nature of the corpus;
- Storage capability;
- Budget;
- Human resources;
- Time delay to build the corpus.

The corpus characteristics will be determined by the nature of the task it is built for. If the corpus is very specialized (*e.g.*: a voice for a city's bus system), the corpus will simply consist of important keywords and sentences, possibly completed by a simple diphone covering obtained with a greedy algorithm. If the corpus is more generalist, bigger corpora will be necessary, and covered features will have to be chosen appropriately. If resources are limited, greedy algorithms will be the obvious choice. However, with more time and technical and physical resources, a better corpus can be obtained with an algorithm based on Lagrangian relaxation. Finally, the corpus building technique yields the best covering, but this comes to the price of a high technical complexity and high human, time and financial cost.

Finally, corpus building is directly impacted by the quality of annotations, particularly in the case of automatic annotations. In 2007, Lambert *et al.* published a paper where two corpora where built from the 2007 Blizzard Challenge corpus [Lambert et al. 2007]. One

was phonetically balanced while the other consisted of random sentences. The authors stated that automatic annotations where much better in the randomly selected corpus. This resulted in better unit choices and better prosody modelling for the Unit-Selection TTS system and eventually to better rating in both objective and subjective tests. The authors stated that this better performance of the random corpus was certainly caused by quality of alignments in the original corpus.

3 Conclusion

This chapter was devoted to the presentation of a classical TTS front-end and main corpus building methods in the scope of unit selection-based synthesis. We have first shown how the input text is transformed into a multi-leveled representation of the requested utterance, through the use of successive annotation tools. Then, we focused on the problem posed by the speech corpus: we saw that the size and richness of the corpus was the key point for unit selection in order to get a satisfactory speech quality. However, we saw that algorithmic and contextual constraints limit the actual size of the corpus, which led us to present the problem of corpus reduction that aims to get the best possible tradeoff between corpus size and corpus richness.

Chapter 4

The Unit Selection Backend Block

"All life is a concatenation of ephemeralities."

Alfred Edward Kahn (1917–2010)

In the last chapter, the TTS fronted and the construction of the corpus were presented. This chapter will now focus on the second part of the TTS process: the TTS backend. This part takes the information inferred by the frontend, the description of the utterance to produce. Its task is to make the most of the corpus to get a sequence of concatenated speech segments as close as possible to the description given by the frontend. This part will be the one to be given the greatest attention as it is the heart of the thesis. It will consist of a detailed description of the backend of the speech synthesis system, often called the synthesizer.

In this part, we will now focus on the unit selection backend block. Having the annotated corpus and a textual utterance enriched with linguistic and phonetic information (the sentence to produce) obtained with the TTS frontend, the last part of the process is to determine which portions of the corpus should be used to reproduce the utterance, extract them and carry out concatenations (and possibly prosodic modifications).

The term "unit selection" is often used in literature to refer to the whole backend block (in the title of this section for example), which means in that case "Backend block based on the unit selection technique", including the concatenation and smoothing steps. In fact, the unit selection process is only the first part of the block, *i.e.* the search for the right units to concatenate. But as the steps that follow in the backend block depend on the selection process, and as the unit selection process is the differentiating part with other concatenative systems, it is possible to refer to the whole block as unit selection.

From now on though, in order to avoid any confusion, the term unit selection will only refer to the actual selection step alone.

Moreover, synthesis in the unit selection block is done breath group by breath group, mainly in order to decrease the complexity of the problem as concatenation over two silences (as there is between breath groups) is straightforward.

1 Topology of the Problem

The unit selection block has two inputs: the utterance generated by the TTS frontend, which represents the target to synthesize, and the speech corpus. This is represented in figure 4.1. An important point is that the corpus is not part of the selection block: it must be seen as completely interchangeable without any modification on the unit selection engine. In output of the block is the sequence of corpus units that has been selected for concatenation.

1.1 The Base Unit

The problem of finding the best mapping between units from the corpus and the input utterance requires the definition of a base unit. In chapter 2, we discussed this problem, showing that a large range of units were investigated for usage in speech synthesis. The problem is a footprint/performance tradeoff. Long units like syllables or words ensure good TTS performance but creating a corpus with a full covering of words or even syllables, not even speaking of multi-representation, is very difficult to achieve in non-domain-specific

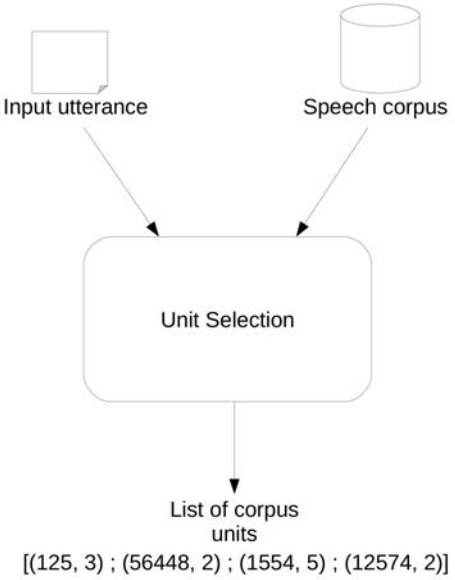

Figure 4.1: Black box view of the unit selection process. The process looks for the best way to reconstruct acoustically the input utterance by joining units from the corpus. The process is completely generic (*i.e.* independent of the corpus). The output of that process is an ordered list of corpus units; here represented by couples where the first number is the position of the first phone in the unit and the second is the number of subsequent phones that belong to that unit. The units in this list are those to be concatenated.

synthesis. Indeed, a significant amount of time and substantial resources would be needed to record it and its footprint would be considerable, which would also make it difficult to use in practice. Short units (demi-phones, phones, diphones, 2-phones, 3-phones, ...) require a much smaller corpus but imply more concatenations as less contexts are represented in the database. In addition, some base units make more sense in some languages than other. Syllable, for example, are more likely to help increase synthesis quality for syllabic languages than tonal languages. For many languages though, the diphone remains the reference unit. It allows the creation of small corpora (less than half an hour) that can still be enriched by adding more covering constraints during the creation of the corpus. As we will see in the following, a diphone corpus can perfectly be used to carry out synthesis by concatenating on bigger units than diphones. In the following, we shall consider the diphone as the base unit.

1.2 The Notion of Sequence

In our representation of the problem, the corpus is viewed as a long stream of speech, where sub-parts are accessible by using the absolute phone annotation. Therefore, units are defined by a couple (index, size) where the index is the absolute position of the first phone of the unit in the corpus and the size is the number of phones in the unit. A unit ranges from position *index* to *index+size-1*. The part that will actually be concatenated will begin by the second part of the first phone of the unit and end with the first part of the last phone. Hence the impossibility to have sequences smaller than 2 phones[1]. An illustration of this concept is given by figure 4.2, which shows the joining of two units [ʁe] and [ey].

As the input utterance is basically a sequence of diphones enriched with additional information – mainly over the characteristics of the wanted diphones, it is broadly referred to as the target sequence (of diphones). In the same way, a sequence that matches the target sequence is called candidate sequence, itself composed of candidate units matching subparts of the target sequence. As synthesis is done breath group by breath group, the target sequence (and therefore candidate sequences) always begins and ends with a silence, which we will refer to with a # symbol in the following.

1.3 Speech Units

The unit selection problem, first defined in Sagisaka's work in 1988 [Sagisaka 1988], differs to basic concatenative synthesis in two points:

1. Multi-representation of units: a same unit is represented several times in the corpus.

2. Variable length units: the size of the unit is not fixed, it can be any number of base unit, possibly the whole target sequence.

[1] The corpus annotations are made by phone while concatenations are made on diphones.

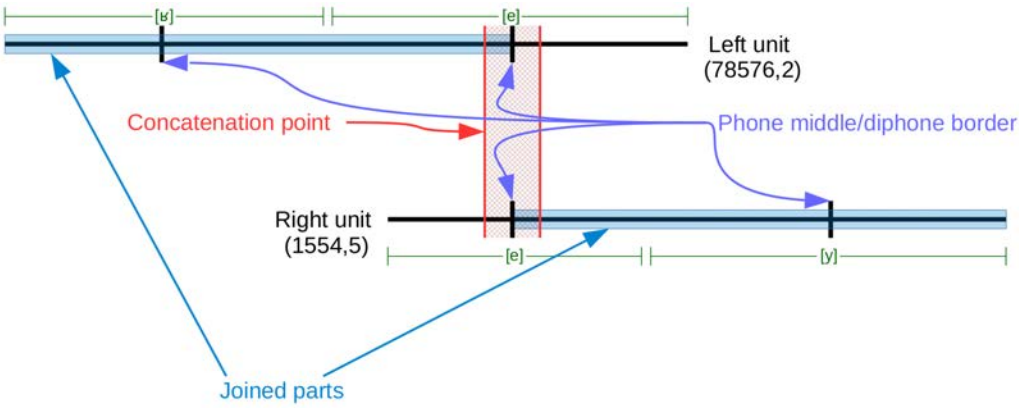

Figure 4.2: Joining unit [ʁe] with unit [ey] requires concatenating on the last phone of left unit [ʁe] and the first phone of right unit [ey], *i.e.* on phone [e]. Concatenations are made on diphone borders, hence in the middle of the [e]. Once centers of [e] phones of both units are aligned, the left part of the concatenated phone is taken from the left unit, the other from the right unit. A PSOLA merge is carried out (red area) to soften the junction (see section 4). Final unit is represented by the blue stripes. The left unit is composed of two phones of index 78576 and 78577 and the right unit of 5, from 1554 to 1558 for the right one in the corpus.

While the first point is also encountered in advanced concatenative TTS systems (that are not unit selection based)[2], the second is a particularity of unit selection. The difference on the first point between unit selection and basic concatenative systems holds in the representation of the corpus. For unit selection systems, the corpus is a whole "speech strip" where speech is extracted only during concatenation while other systems use a dictionary structure to classify speech sounds.

For one target sequence, there is a number of possible candidate sequences, a huge number in fact when the corpus is hours long. The following equation, from H. François's thesis (proof and formula are on pages 79-80), gives an idea of the number of candidate sequences in the case of a corpus with featuring the same number M of representatives of each phone (which is often not the case in unit selection, the number of each phone being usually different) [Francois 2002]:

$$N_{paths} = \sum_{c=0}^{N} M^{c+1} \binom{c}{N-1} \tag{4.1}$$

where N is the number of *phones* in the target sequence (meaning there are up to N-1 concatenation points, each on diphone boundaries). c is the number of concatenations made in the sequence, which varies from 0 to N-1 concatenations. Finally, $\binom{c}{N-1}$ is the number of combinations of c in $N-1$. To illustrate the difficulty of the problem, let us take

[2]Concatenative systems may or may not handle multi-represented units but all unit selection systems do.

the example of sentence "Yes.". If we consider a constant M of 10 phones, there is 6 410 candidate sequences. With a more likely M=1 000 (for a 10 hours corpus for instance), the number of candidate sequences jumps to over 6 billion. And the problem is the same for a bigger target sequence, "Bonjour tout le monde." (French for "Hi everyone."), there are 4.3×10^{23} candidate sequences only for M=10, *etc.*

As we said earlier, a unit is not necessarily a diphone. Longer units, which size is variable and thus undefined before execution, composed of contiguous diphones (in the corpus) are also to be considered (and usually favored). As an example, for 10 hours of speech, the number of units (no matter the size) is superior to 5 million while there is approximatively around 400 000 diphones. Such a quantity of data means all possible paths cannot be parsed in real time, as of 2015. Two possibilities are then feasible to make the problem solvable in real time. First, the corpus can be explored by a greedy algorithm that only considers what it expects to be the best combinations. This leads to under-optimal solutions. It is typically that approach that is used in most beam-search Viterbi-based algorithms used for unit selection. Secondly, algorithms that do not need to parse all units to find the (optimal) solution of the problem might be used instead. This leads to much lower processing time. Unfortunately, with a large speech corpus of several hours, this is not enough to get an optimal solution in real time. To achieve that, non-trivial admissible heuristics would have to be found. In particular, the framework for such an heuristic is offered by A^* algorithm. With A^*, if the heuristic gives an estimation systematically inferior or equal to the real cost, it is insured that the algorithm is admissible. A proof of this property is provided in N. Nilsson's book, "Principles of Artificial Intelligence" [Nilsson 1982]. Nonetheless, such an heuristic is very hard to find for the unit selection problem. But finally, this real time TTS problem relies on the following two questions:

1. Does getting an under-optimal solution to the unit selection problem consistently degrade synthesis?

2. How much does pruning affect perceived quality of speech synthesis? Or, how much pruning can be allowed in the selection algorithm so that degradation is not perceived?

Those concerns will be addressed in chapter 6 where my work on the unit selection algorithm is detailed.

1.4 The Candidate Units Graph

Now, in order to solve the unit selection problem, the algorithm must have some way to structure corpus data and explore it efficiently. As the search algorithm has to find a way to concatenate a finite number of units that have to be selected into a finite (though considerable) number of units with an ordering relation between them and a cost for passing from one unit to the other, the problem finally comes back to a shortest-path finding problem in a directed weighted graph.

Formally, units can be organized in a graph $G = (V, A, C)$ where the nodes set V represents the set of all corpus units that can be used to match the target sequence. A is the set of arcs transcribing possible concatenations between units. As the graph is directed, there is an ordering between the nodes: an arc from a node A to a node B means B can be joined after A. C is the set of costs c_{ij} linked to each arc $(i, j) \in E$ of the graph. It quantifies the risk of creating audible artefacts when concatenating two units.

Several nodes in the graph can represent the same corpus parts. For example, a unit corresponding to the phonetic sequence [# - s - u] is composed of diphones [# - s] and [s - u]. While there is a node representing that unit, there are two other distinct nodes representing the two smaller corpus units composed by the diphones. A same unit can also be used more than one time in the graph as it might be candidate for matching the target sequence several times. For instance, our unit [s - u] from the last example can be used twice to synthesize the target utterance [# - s - u - s - u].

Arcs between the units only transcribe concatenations that mark a progress in matching the target sequence[3]. This means two things. First, that the selection graph is a sub-graph of a global non-oriented graph that models all possible concatenations in the corpus. This graph is fully connected. Secondly, this means the graph is always built during the selection stage, as it is related to the target sequence. It cannot be constructed prior selection as the target sequence is unknown then.

Furthermore, the graph matches following properties:

Finite The corpus, though huge, contains a finite number of nodes.

Acyclic As each unit introduced into the graph is linked to a particular portion of the target sequence, once it is selected, it cannot loop. If the same unit is selected twice, even one after the other, it will be represented by two different units in the graph, each one linked to a particular part of the target sequence.

Directed An edge linking two units transcribes a progression in the construction of a sequence matching the target sequence. Hence, there is a time relation between the two units: one is necessarily preceding the other. In addition, going from the second unit to the first is impossible because (1) it would not mark any progression in the target sequence (2) it would not match the target sequence and (3) it would break the acyclicity constraint. The graph is therefore directed and edges are called arcs.

Weighted An arc from one node to another transcribes the fact of joining two units. Joining two units comes to a cost. Therefore, arcs bear a weight formed by the cost of the operation of adding the new node to the existing candidate sequence.

Furthermore, in order to simplify the work for the algorithm, two non-synthesizable nodes may be added. An "init" node is added to the graph with arcs bearing an empty

[3]Concatenations that would cause a candidate sequence to diverge from the target sequence are forbidden.

cost to every possible unit (*i.e.* node) that match first diphones of the target sequence. Similarly, an "end" node is added as a successor to every node that match the end of the target sequence. As a result, the graph matches all the properties of a lattice. Indeed, with these nodes, every couple of nodes of the graph possess at least one predecessor and one successor in common. The fact that the graph is a lattice enables the usage of dynamic programming for the selection algorithm, which we will detail in the next section. The modeling of the problem via a lattice is widely used in the literature (in [Donovan 1996], [Yi 1998], [Klein and Manning 2003] or [Vepa 2004] for instance). Another representation has been used in the literature, mainly in the 90s: modeling of the problem via a tree. In 1999, Taylor and Black proposed an algorithm called PSM (Phonological Structure Matching) which was based on a hierarchical selection of units, which were placed in a phonological classification tree [Taylor and Black 1999]. The tree represents the target utterance, with nodes depicting first words and then syllables, stress information and finally phone sets from the corpus on the leaves. The phones in the leaf sets are those that match the information represented by parent nodes in the graph. The algorithm then uses a scoring function to get the best phone of each leaf and concatenates them. Of course, this technique answers to the problem of unit selection only if the scoring function takes into account the relation between each set in order to ensure that phones in two sets are considered as part of a same bigger unit if they are contiguous in the corpus. Nodes in the sets could also be rearranged into a small graph and a new (smaller and easier to solve) unit selection problem emerges (this is in particular the way PSM works). In that case, the construction of the tree and its leaf sets can be viewed as a method of preselection, or pruning.

As in this thesis my work is aiming to be as generalist as possible, we will keep considering the most generalist point of view: a graph.

In order to illustrate previous paragraphs, let's consider we want to synthesize the utterance *"Sous une autre forme."* (Under another form.). Figure 4.3 shows the beginning of the related graph. The start point of any search algorithm, in the graph, is the node "init". From that node, there is a link to every node that represent a corpus unit matching the beginning of the target sequence. This node is introduced in order to avoid arbitrary choice of the first node. We make a similar choice by introducing a unique end (target) node. Several units are only a diphone [# - s], while some also match the third phoneme of the sentence, giving a longer unit: [# - s - u]. There might have been even longer units, to the limit of the target sequence size. The size of a unit is determined by the number of related diphones in the corpus that match the target sequence. In particular, if an utterance in the corpus happened to correspond to the complete target sequence, a unique node corresponding to that utterance would be added to the graph and it would be linked on one side to the init node and on the other side to the end node. Nodes representing a diphone [# - s] only have arcs that link them to units that match the next part of the target sequence, beginning with diphone [s - u]. The longest possible size for the nodes

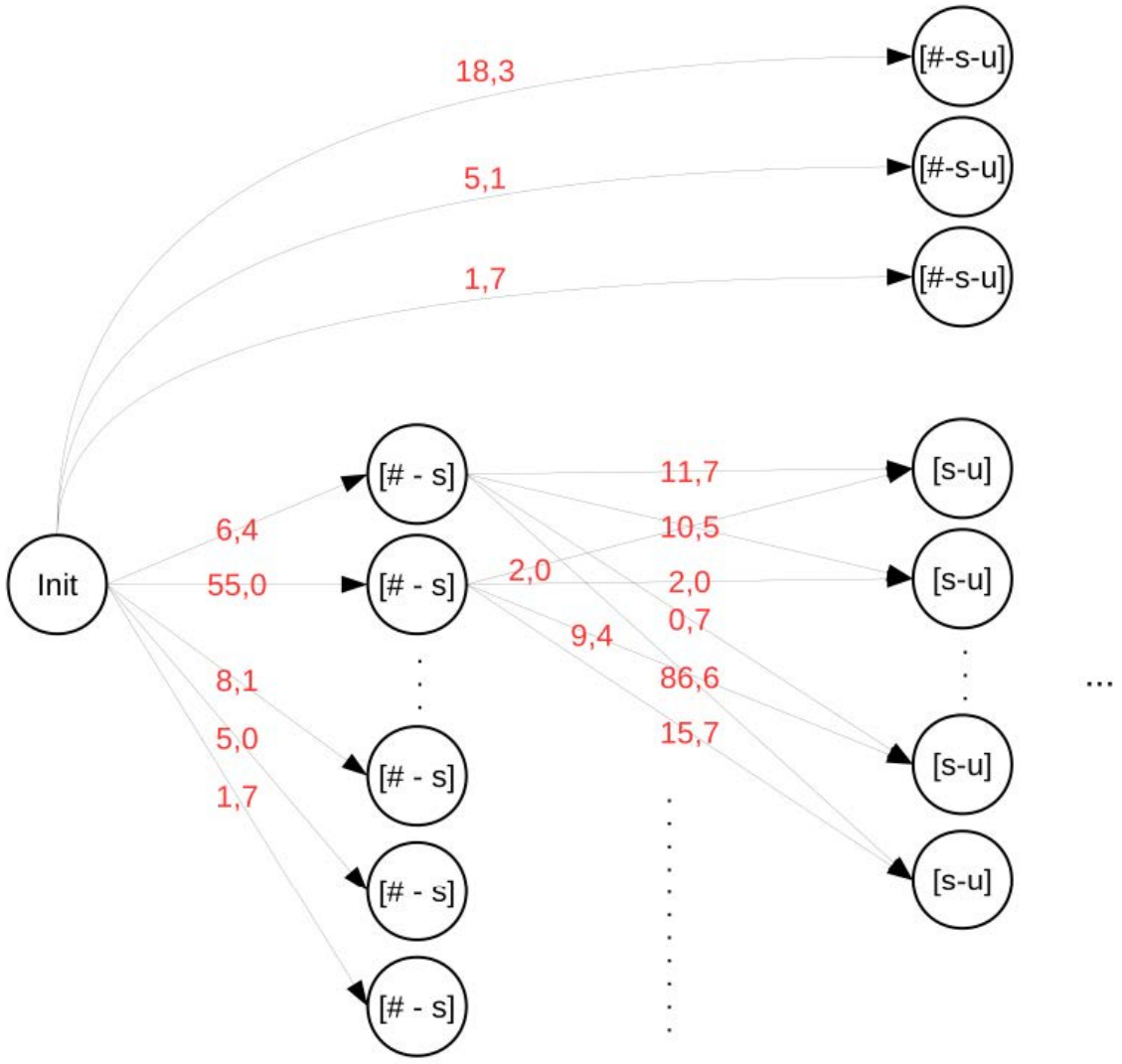

Figure 4.3: Example of the unit selection graph modeling where nodes are corpus units. Each arc is possible way to bind two units. The binding's cost is in red.

accessed by [# - s] nodes is $N-1$ where N is the size of the target sequence of diphones $T = (\hat{d}_1, \hat{d}_2, ..., \hat{d}_N)$.

Another modeling, completely equivalent, is worth mentioning, as it has been widely used in the literature (for instance in Hunt and Black's unit selection founder article [Hunt and Black 1996]). It is to consider V as the set of possible states while browsing the target sequence. In that case, the arcs become the corpus units and each arc bears the cost for the selection of that unit, in the context of the emitting and receiving states. As there is very likely much more than one unit that allows passing from one node to another, the graph is multivalued. For example (see figure 4.4 for an illustration of that example), the first state of the graph the selection algorithm will visit in sentence *"Sous une autre forme."* (after the initialization node) is state [# - s] which we reach when a diphone [# - s] has been

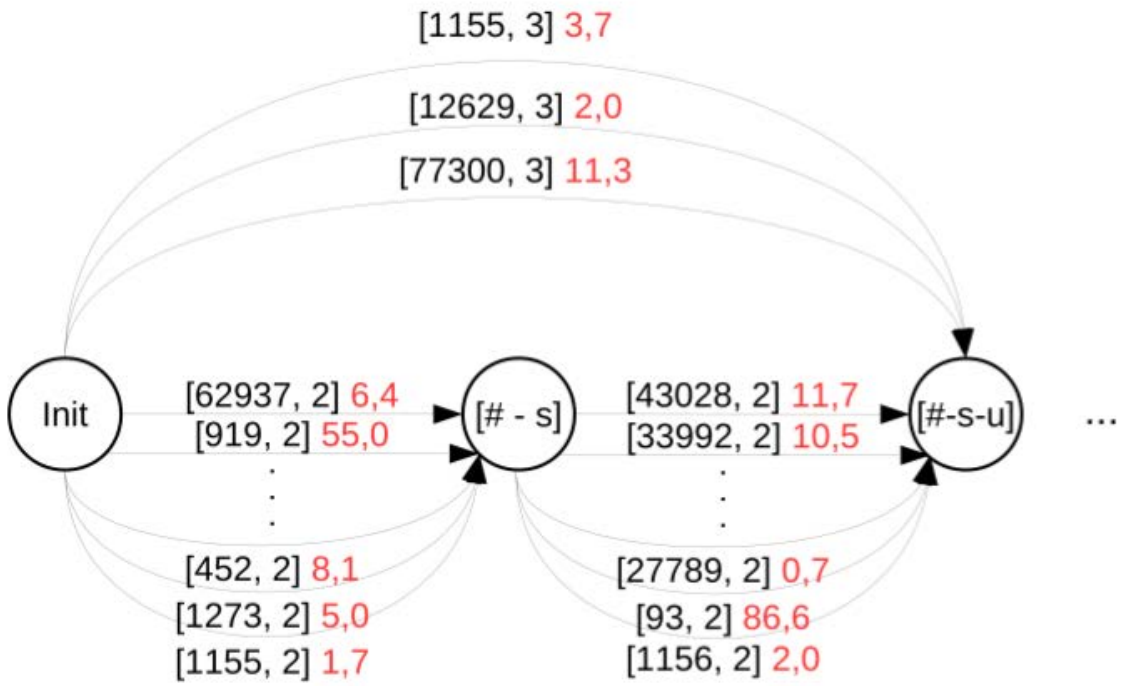

Figure 4.4: Example of the unit selection graph modeling where nodes are states in the target sequence. Each arc is a corpus unit (black part of the label) with a selection cost (in red).

selected. This state is directly linked to state [# - s - u] where two diphones [# - s] and [s - u] have been selected by the selection algorithm. As there is probably a big number of diphones [s - u] in the corpus, every one of them is represented by an arc between states [# - s] and [s - u]. The two phonemes can be on a contiguous segment, in which case they form only one unit. In that case, this unit was selected by taking an arc from the initialization node directly to [# - s - u].

A complete example of that modelization is given by figure 4.5 [Francois 2002]. This modeling is more compact than previous one, and may be easier for presenting the unit selection algorithm but it is also very different from what actually happens in the heart of the algorithms implemented in my work. This is why in the following we will use the first modeling presented where each node is a corpus unit and an arc represents the action of concatenating two units at some cost (potentially null).

2 Selection Algorithm

The goal of the unit selection algorithm is to find the cheapest path in the graph. Given the nature of the graph, especially the fact that it is actually a lattice (or a tree if the end node is removed), dynamic programming has been the most popular way to perform selection, since Hunt and Black's original work on the CHATR system in 1996 [Hunt and

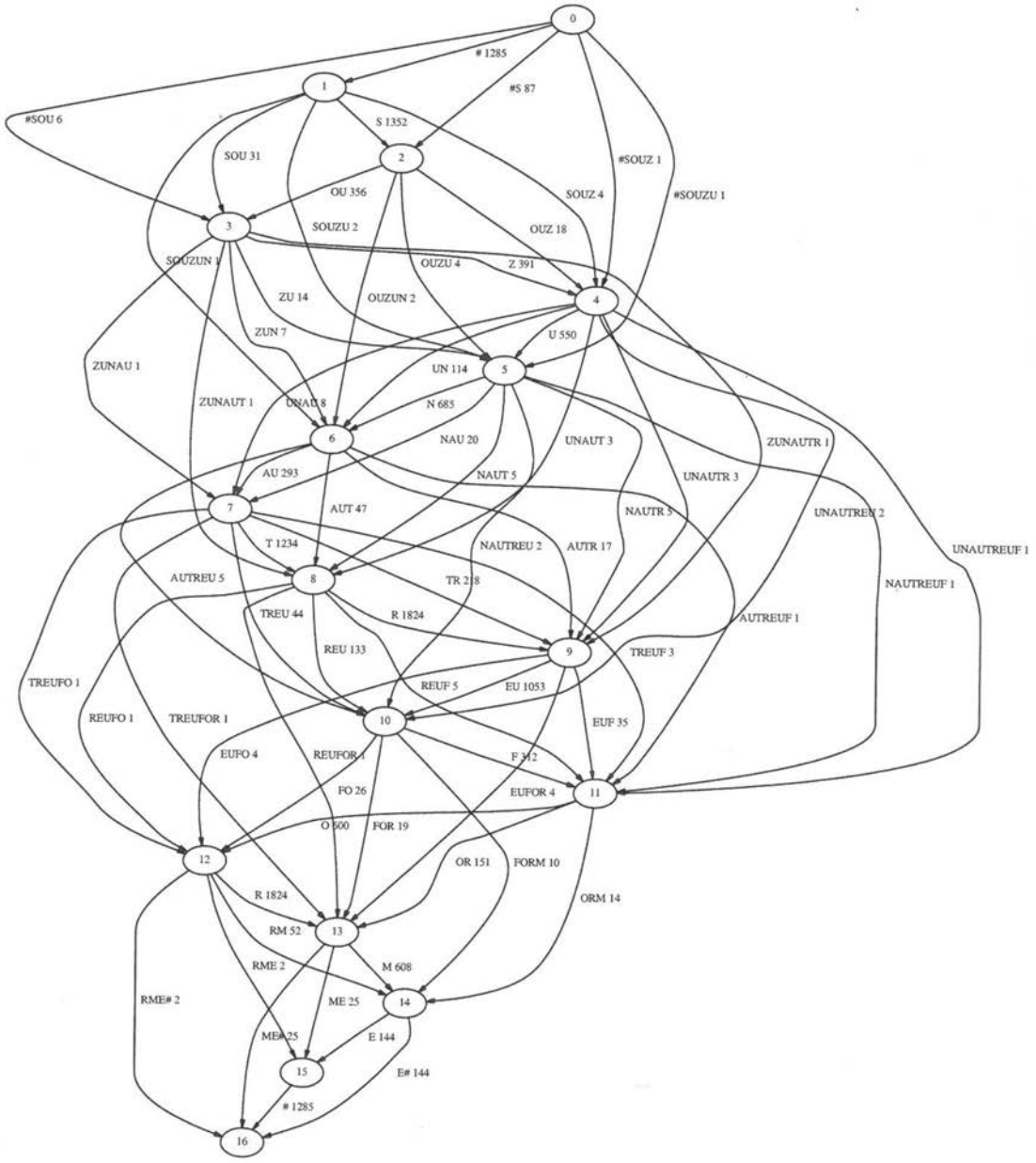

Figure 4.5: The unit selection sequence graph for the French sentence *"Sous une autre forme."*. Each edge has a label and a number. The label represents the unit that's added to the sequence and the number is the quantity of units of that type in the corpus. To reproduce the whole graph, this number would need to be replaced by as many edges, along with the cost of that particular unit in sequence (*i.e.* taking into account the previously selected unit in the sequence). Such a representation is of course impossible given the size of the problem. Nodes represent a common state in the sequence: 0 means nothing was selected, 1 that the first phone related to the target sequence was selected, *etc.* Figure extracted from Hélène François' thesis [Francois 2002].

Black 1996]. In that work, a lattice of phones representing the content of the database[4] is created and pruned. Viterbi algorithm is used to find the best sequence of graph nodes, hence the best unit sequence. The solution is under-optimal as the lattice was pruned. In 1998, M. Beutnagel, A. Conkie and A. Syrdal used the same principle but replaced phones by diphones [Beutnagel et al. 1998]. Then, this method has been reproduced in most publications on unit selection until today, for instance in D. Schwarz's presentation of concatenative sound synthesis (musical synthesis, explorative synthesis, artistic synthesis [Schwarz 2007]) or in a work on the sub-costs of the concatenation cost function by Blouin *et al.* [Blouin et al. 2002].

Over time, several enhancements of the algorithm were proposed, for instance in S. Sakai, T. Kawahara and S. Nakamura's work [Sakai et al. 2008] where stopping criteria aiming at pruning the Viterbi lattice are presented. It was further refined and perceptually evaluated in a work by D. Tihelka, J. Kala and J. Matoušek [Tihelka et al. 2010], where no perceptual degradation caused by the pruning criteria is spotted while the algorithm is up to 58 times faster than the baseline algorithm (a beam-search Viterbi).

2.1 Viterbi Algorithm

Now, to simplify, let us put aside the init and end nodes that just serve to initialize, launch and terminate the search. Let $T = (d_1, d_2, ..., d_N)$ the target sequence of diphonemes of size N, with d_k being the k^{th} diphoneme of the sequence, for any $k \in [\![1; N]\!]$. We note $\Psi_k = \left\{ d_k^1, d_k^2, ..., d_k^{M_k} \right\}$ the set of the M_k candidate diphones in the corpus that match the target diphoneme $\hat{d}_k$. With $i, j \in [\![1; N]\!]$, $i < j$, a corpus unit matching target diphonemes i to j is noted $\mathcal{U}_{i,j}^x$, x meaning the unit is the x^{th} matching the target sequence from d_i to d_j [5]. Hence, we define $\Omega_{i,j} = \left\{ \mathcal{U}_{i,j}^1, \mathcal{U}_{i,j}^2, \ldots, \mathcal{U}_{i,j}^{M_{i,j}} \right\}$ as the set of all corpus units that match the target sequence from diphonemes d_i to d_j included. In the following, we use $\mathcal{U}_{i,j}^{\omega_{i,j}} \in \Omega_{i,j}$ to refer to any unit of $\Omega_{i,j}$. The set of all units in the corpus is the following:

$$\Omega = \bigcup_{\substack{i,j \in [\![1;N]\!] \\ i<j}} \Omega_{i,j} \tag{4.2}$$

Using these notations and with $1 \leq h < i$, the unit selection problem can now be written as an optimization problem, aiming at finding the unit sequence that minimizes a cost function C:

$$\mathcal{U}^* = \underset{\mathcal{U}=\mathcal{U}_{1,h}^{\omega_{1,h}},\ldots,\mathcal{U}_{j,N}^{\omega_{j,N}}}{\arg\min} \left(C(\mathcal{U}_{1,h}^{\omega_{1,h}}, \mathcal{U}_{h,i}^{\omega_{h,i}}, \ldots, \mathcal{U}_{j,N}^{\omega_{j,N}}) \right) \tag{4.3}$$

[4] They are called states though not in the sense of the state in the target sequence, as in the modeling presented in the last section.

[5] Here, ranking of the corpus units with attribute x is purely arbitrary, as no ordering of the candidates $\mathcal{U}_{i,j}^x$ is necessary. x is here used as an identifier for the unit among its siblings.

The cost of each unit is determined by two cost functions C_t and C_c. Hence, the general cost function evaluating each node takes the form $C(\mathcal{U}_{i,j}^{\omega_{i,j}}) = C_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) + C_c(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}})$ where $\mathcal{U}_{h,i}^{\omega_{h,i}}$ is the predecessor of unit $\mathcal{U}_{i,j}^{\omega_{i,j}}$ in the candidate sequence. Equation 4.3 then becomes:

$$\mathcal{U}^* = \underset{\mathcal{U}=\mathcal{U}_{1,h}^{\omega_{1,h}},\ldots,\mathcal{U}_{j,N}^{\omega_{j,N}}}{\arg\min} \left(W_{tc} \sum_{\mathcal{U}} C_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) \; + \; W_{cc} \sum_{\mathcal{U}} C_c(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}})\right) \tag{4.4}$$

C_t is called the target cost and measures the degree of suitability of a corpus unit to represent the corresponding part of the target sequence. C_c is called the concatenation cost. It evaluates the expected quality of the joining point in the signal after concatenating a unit with its predecessor in the candidate sequence (*i.e.* the preceding graph-node in the path being built by the algorithm). We will focus on these costs in section 3.

The Viterbi algorithm was first introduced in a 1967 article by A. J. Viterbi [Viterbi 1967] to find an upper bound to the probability of error in decoding an optimal convolutional code. More recently, Viterbi gave a simpler and more generalist description (though still centered on digital sound processing) of the algorithm in the IEEE Signal Processing Magazine [Viterbi 2006]. The algorithm is based on dynamic programming, with the following recursion formula[6]:

$$C(\mathcal{U}_{i,j}^{\omega_{i,j}}) = \begin{cases} C_t(\mathcal{U}_{1,j}^{\omega_{1,j}}) & \text{if } i = 1 \\ \min\limits_{\Omega_{h,i}} \Big(C(\mathcal{U}_{h,i}^{\omega_{h,i}}) + C_c(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}}) \Big) + C_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) & \text{otherwise.} \end{cases} \tag{4.5}$$

Starting from the last units, the algorithm directly calls the preceding unit on the optimal path, which is computed with the minimum on the recursive call, in practice expressed with a breadth-first search (no recursion in the algorithm since it's dynamic programming).

The algorithm's asymptotic computational complexity is $\mathcal{O}(N * K + N * K^2)$, K being the number of candidate phones in the corpus and N being the number of diphonemes in the target sequence [Schwarz 2007]. More precisely, the target cost function counts for $\mathcal{O}(N \; * \; K)$ computations while the concatenation cost is computed $\mathcal{O}(N \; * \; K^2)$ times.

2.2 Beam-search Algorithm

This complexity can be reduced to $\mathcal{O}(N * K + N * B^2)$ with a beam-search optimization of the Viterbi algorithm, B being the size of the beam in question. Beam-search strategy is, given a unit, to compute the target costs of all possible preceding units in the graph (as it is usually done in the baseline Viterbi algorithm) and only keep the B best ones for further consideration. Hence, at each level of the search, only B solutions for matching the target

[6] Again with $h, i, j \in [\![1; N]\!]$, $h < i < j$.

sequence are considered, as all others have been pruned at the previous level (remember this is a breadth-first strategy, so we are progressing level by level, even if each level has variable-size units).

It is actually this technique that is used in most (if not all) unit selection engines around the world, for a simple reason: the baseline Viterbi algorithm cannot compute the solution to the unit selection problem in real-time. Of course, using a beam-search strategy makes Viterbi algorithm under-optimal, but we will see in chapter 6 that with a reasonable beam size, optimality of the solution to the unit selection problem is not crucial and that reasonably good solutions can be obtained in a very short time with (almost) no perceived degradation of synthesized speech.

2.3 Non-Viterbi Approaches

In most cases, the Viterbi algorithm is used to find the best unit sequence, but it is not the only possible one. As the unit selection problem is about finding the shortest path in a graph, all algorithms able to solve that problem are susceptible to be used.

The most famous of these surely is Dijkstra's algorithm (constantly cited as one of the most used algorithms in the world). Dijkstra's algorithm the shortest path problem in $\mathcal{O}(|V^2|)$, *i.e.* $\mathcal{O}(n^2)$ time according to S. Saha Ray [Saha Ray 2013], n being the number of nodes traversed by the algorithm [7].

A more general algorithm, the Floyd-Warshall All-Pairs-Shortest-Path algorithm, is also fit for the task. Unlike Viterbi or Dijkstra, which have a single source (or init) node, it can compute the shortest path between any two nodes or vertices in the network [Saha Ray 2013]. It relies on dynamic programming and has a time complexity in $\mathcal{O}(n^3)$, thus making Dijkstra better for the unit selection task.

Another algorithm is often used (even more than Dijkstra actually) for shortest path problems: A^*. Contrary to the Viterbi algorithm, A^* algorithm develops a graph. At each time instant, it explores the best node of the graph using a cost function that depends on both the path from the source node and the estimated cost to the target. Originally introduced in 1968 by P. E. Hart, N. J. Nilsson and B. Raphael [Hart et al. 1968], the algorithm basically operates by searching for a path in a directed graph, whose nodes only have a finite number of successors, between a start node and a target node.

At each step, A^* takes the most promising node according to a cost function $f(\mathcal{U}_{i,j}^{\omega_{i,j}}) = C(\mathcal{U}_{i,j}^{\omega_{i,j}}) + h(\mathcal{U}_{i,j}^{\omega_{i,j}})$ and expends its successors (computing their cost by the way) until the target node t is reached. $h(\mathcal{U}_{i,j}^{\omega_{i,j}})$ is a heuristic that enables to speed up the algorithm by privileging the nodes that seem to be on an optimal path over those which have a better cost but may lead to greater costs in the future [Nilsson 1982].

Considering a unique target node, one of the main advantages of A^* is that the algorithm delivers an optimal solution if the heuristic is admissible, *i.e.* if $h(\mathcal{U}_{i,j}^{\omega_{i,j}}) \leq h^*(\mathcal{U}_{i,j}^{\omega_{i,j}})$, where $h^*(\mathcal{U}_{i,j}^{\omega_{i,j}})$ is the real minimum value of the distance to the target node. In particular, note

[7] $|V|$ is the number of nodes in the graph.

that the algorithm is optimal in the trivial case $h(\mathcal{U}_{i,j}^{\omega_{i,j}}) = 0$, *i.e.* if there is no heuristic, and turns out to be equivalent to Dijkstra's algorithm.

Other algorithms, like the D algorithm presented in 1959 by E. Moore or Busacker and Saaty's dynamic programming implementation may also be used.

Very different techniques may also be used for unit selection, the most exotic being perhaps Rohit Kumar's work [Kumar 2004], where a genetic algorithm is used, relying on genetic operators very similar to the target and concatenation costs. This algorithm achieves to find an acceptable (*i.e.* under-optimal though the genetic algorithm converges to the optimal solution) solution faster than using an optimization based algorithm.

Evaluating the interest of these alternatives to Viterbi algorithm has been the first task I have undertaken in my work [Guennec and Lolive 2014a; Guennec and Lolive 2014b]. In chapter 6, we present the results of that work.

2.4 Concerning Variable-size Units

A particular point is that the term unit selection is used indifferently in the literature to describe two processes:

- Parsing a graph where nodes represent diphones in the database (most publications);
- Parsing a graph modeling variable-size units.

While the second point follows exactly the definition of unit selection, the first one seems to match more concatenative synthesis (without variable-size units) than unit selection. Actually, provided that all arcs between nodes that model contiguous diphones in the corpus get a null cost, both come back to exactly the same thing. Hence, using only diphone units provide an immense advantage: the graph is much smaller when including only diphones and no longer units and it is easier to work on as all units have the same size. That means also that the size of the problem is much smaller, leading to both algorithmic space and time economy. In this thesis, we use the general point of view. Variable-size units are used in order to make the discourse as comprehensive as possible, but in practical applications referring to the diphone graph only is sufficient, if and only if arcs between nodes that model contiguous diphones in the corpus get a null cost.

3 Selection Cost

As previously said, Hunt and Black first came with this target cost/concatenation cost formulation [Hunt and Black 1996]. This division into two procedures is particularly adapted to the problem and has been wildly reproduced until today. It presents nonetheless one important disadvantage: global constraints are not taken into account as the costs are local to the nodes of the graph. We will come back on that matter in section 3.5.

Before going further, an important remark has to be kept in mind: the goal of the selection cost construction task is to build a function that penalizes units the way a human

would, with quality of experience in mind. What this means is that a unit sequence, once concatenated, can present discontinuities that human ears cannot discern (*i.e.* buzzing at 50 kHz). Such things should not be taken into account in the cost function. Doing so may cause a unit with perceptible issues to be selected instead of a unit bringing multiple but imperceptible problems.

So the main issue of the selection cost is the following: what criteria should be assessed and what weighting should be operated between them? Of course, the answer to that question is language dependent.

It is also important to note that the constraints used into the selection cost (especially the target cost) need to be as close as possible to the constraints that were used to build the corpus. The ideal case is to have exactly the same costs in the function that scores the utterances during corpus construction and in the unit selection cost function. Indeed, the criteria used to build the corpus have an impact on its composition (phonetic, linguistic, *etc.*). Using the same (or close enough) criteria for selection allows to take advantage of this exotic corpus content. Of course, the selection cost function should ideally be as independent of the corpus as possible. Corpora should be commutable without modifying the cost function. Actually, this is not the case in reality: the cost function efficiency is directly affected by the corpus content. Thus, using the same constraints for corpus construction and unit selection is a welcomed optimization.

3.1 Target Cost

The target cost aims at sorting each candidate unit set $\Omega_{i,j}$ according to its level of correspondence to the description of the corresponding portion of the target sequence. This knowledge can be based on:

- Linguistic and phonetic target sequence descriptors acquired with automatic annotation (only a few predicted)
- Predicted values like F_0, phonemic duration, energy, *etc.*
- Rules (especially prosodic)

Starting from these descriptors and predictions, distance functions are built to compare this data to each corpus unit annotations, which are stored in the corpus (*i.e.* not predicted). This set of distance or difference functions is called the set of target sub-cost functions. Together, they form the unit selection target cost:

$$C_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) = \sum_{k=1}^{K} w_k C_t^k(\mathcal{U}_{i,j}^{\omega_{i,j}}) \tag{4.6}$$

K being here the number of target sub-cost functions, and $C_t^k(\mathcal{U}_{i,j}^{\omega_{i,j}})$ being the function associated to k^{th} sub-cost.

The actual nature of the sub-costs is rarely stated in the literature, for example A. Black and N. Campbell where the target cost is described as containing 20 to 30 sub-costs without giving more detail [Black and Campbell 1995]. Nonetheless, the features usually chosen are linguistic/phonological attributes (it may also be implemented via a preselection filter which would make selection much faster, *cf.* paragraph 3.4), pitch, energy or phonemic duration. For example, Alías *et al.* use predictions of normalized pitch, energy and duration for their target cost [Alías et al. 2011].

The main problem of this formulation concerns the reach of most features used in the cost (this will be further discussed in section 3.5): as the formulation of the problem forces the use of local features, usually centered around the elementary unit (*e.g.* diphone), possibly with contextual information ; no or few attention is given to long term constraints (integration of the unit in regard to the expected prosody of the breath group, consistency with the rest of the candidate sequence for example). This is one of the points we will discuss later on in chapter 7.

3.2 Concatenation Cost

The concatenation cost goal is to prevent concatenation of units susceptible to cause the appearance of a concatenation artefact or any other inconsistency by awarding a cost to any candidate unit considered to be added to the sequence. This cost measures the difference between a candidate unit and the last unit of the candidate sequence under construction. The computed cost is then added to the cost of the candidate unit.

Equation 4.7 presents the integration of concatenation sub-costs in the global join cost function:

$$C_c(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}}) = \sum_{k=1}^{K} w_k C_c^k(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}}) \tag{4.7}$$

When using acoustic or prosodic features in the target cost, the target data is generated by a model, which induces a bias. In the case of the concatenation cost, the advantage is that this data comes from corpus annotations of both left ($\mathcal{U}_{h,i}^{\omega_{h,i}}$) and right ($\mathcal{U}_{i,j}^{\omega_{i,j}}$) units. Distances over these attributes – and most concatenation sub-costs are based on acoustic features – is therefore more reliable.

Empirically, synthesis made with a concatenation cost but no target cost is usually acceptable while synthesis made without any concatenation cost often results in unintelligible sentences. One could hence say that the concatenation cost is the main element of the selection cost.

As the concatenation cost composition is particularly important, literature gives it more focus than it does with the target cost. Thus, many spectral distances are implemented in the concatenation target sub-costs though the most important distance is probably F_0 as say D. Tihelka *et al.* in a paper presenting a refined F_0 cost computing the slopes between the 5 F_0 measures around the concatenation point [Tihelka et al. 2014]. Elsewhere in the

literature, Black and Campbell used normalized pitch in 1995 [Black and Campbell 1995], which was also the case of F. Alías *et al.* more recently [Alías et al. 2011]. The latter also used an Euclidean distance on MFCCs around the joining point. Cepstral distance is also quite popular: FFT-based (Fast Fourier Transform) and LPC-based (Linear Prediction Coefficient) cepstral distances were used in many publications, like M. Macon, J. Wouters and A. Cronk's work [Macon et al. 1998] [Wouters and Macon 1998], A. Black and N. Campbell's [Black and Campbell 1995], A. Hunt and A. Black [Hunt and Black 1996] or Y. Stylianou and A. Syrdal's [Stylianou and Syrdal 2001]. The same also used Line Spectral Frequency (LSF) [Macon et al. 1998; Wouters and Macon 1998; Stylianou and Syrdal 2001]. M. Macon, J. Wouters [Macon et al. 1998] and A. Cronk [Wouters and Macon 1998] also tested symmetrized Itakura distance and Log Area Ratio (LRA). In Y. Stylianou and A. Syrdal's work, LSF (with a Kullback-Leiber distance) computed either by LPC or by PLP (Perceptual Linear Prediction) is evaluated [Stylianou and Syrdal 2001]. They also test log power spectrum (Euclidean distance) computed by FFT, LPC or PLP and power spectrum (Kullback-Leiber distance) also computed by FFT, LPC or PLP. They found Kullback-Leibler distance between FFT-based power spectra and the Euclidean distance between MFCCs to perform the best, while M. Macon and J. Wouters claim a mel-based Itakura distance yields to the best prediction of concatenations discontinuities, but the authors of both studies warn these costs alone are not sufficient to make an adequate candidate unit ranking (allophones in their case).

Many works focus on predicting concatenation issues on resynthesized speech. They do not use the costs they develop to generate synthetic speech: the cost is used on data that was already generated to find the position of concatenation artefacts (if any). These are then compared with human predictions. This comes back to the same problem: predicting which unit is likely to engender an annoying concatenation artefact. Actually, this method is the best way to test a new concatenation cost, as its first quality must be not to detect whether a concatenation artefact exists or not, but whether or not a human can perceive it and be annoyed by it. The issue is that, in the unit selection engine, this cost will be among other measures, and its ability to keep working well with the bias induced by other costs also needs to be investigated.

Concerning synthesis of French, Blouin *et al.* [Blouin et al. 2002] used distances based on energy and energy derivative, F_0 and F_0 derivative, delta energy, duration and predicted duration[8], phonological identity, various phonemic characteristics and positional information (position in syllable, word and breath group) to compute the concatenation cost function.

In order to speedup unit selection, a popular approach is to pre-compute the concatenation cost as the target utterance isn't needed. This approach was first presented in 1999 by M. Beutnagel *et al.* [Beutnagel et al. 1999]. Doing so avoids millions of concatenation cost calculations during synthesis but comes at the expense of memory storage. In effect,

[8]Using duration in the concatenation is quite odd as the concatenation cost should only focus on a restricted part of the signal and not be influenced by target cost specific features (like duration).

corpora usually reaching hours to tens of hours, pre-computing all possible unit concatenations requires considerable storage amount and computation time. Accessing the stored information also becomes complicated at that stage. Also, the slightest change in the formulation of the cost forces to recompute everything. Avoiding the concatenation cost by caching is therefore usable for some precise applications, when the system or the voice are not likely to be altered in the short run and when there is sufficient storage available.

3.3 On Weighting Issues

The toughest problem concerning unit selection cost functions is not which sub-costs should be used but what weighting should be made between each sub-cost.

Let us take back equation 4.3, integrating by the same occasion equations 4.6 and 4.7:

$$\mathcal{U}^* = \underset{\mathcal{U}=\mathcal{U}_{1,h}^{\omega_{1,h}},\ldots,\mathcal{U}_{j,N}^{\omega_{j,N}}}{\arg\min} \left(W_{tc} \sum_{\mathcal{U}} C_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) \; + \; W_{cc} \sum_{\mathcal{U}} C_c(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}})\right) \tag{4.8}$$

$$= \underset{\mathcal{U}=\mathcal{U}_{1,h}^{\omega_{1,h}},\ldots,\mathcal{U}_{j,N}^{\omega_{j,N}}}{\arg\min} \left(W_{tc} \sum_{\mathcal{U}} \sum_{k=1}^{K} w_k C_t^k(\mathcal{U}_{i,j}^{\omega_{i,j}}) \; + \; W_{cc} \sum_{\mathcal{U}} \sum_{k=1}^{K} w_k C_c^k(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}})\right) \tag{4.9}$$

In this equation, two types of sub-cost weights can be distinguished:

- W_{tc} and W_{cc} are the weights given to the whole target cost and the whole concatenation cost respectively. Their aim is to give a balance between the two costs. Either to favor one over the other or to give them the same average magnitude if the costs are not normalized.

- The w_ks are the weights given to each sub-cost inside the target and concatenation costs. Their purpose is the same as W_{tc}/W_{cc}.

These weights are usually fixed in the TTS engine but nothing forbids imagining a system where they would be updated on the go, depending on expert knowledge on linguistic features of the candidate unit for example.

In the literature, the weight tuning problem has been intensively explored. Weight tuning methods can be classified in two categories: objective or subjective. C. Blouin *et al.* [Blouin et al. 2002], comparing several versions of a unit selection cost, also compared objective and subjective tuning of the w_k weights. They proposed to optimize w_ks automatically using the average of the costs found using the content of a speech corpus. In the same paper, they also proposed a system with hand-made weights chosen in function of the phonetic class of the demi-phones composing the unit. The automatically tuned version proved to fare better than its manual counterpart in every tested case.

Main objective methods use mathematical optimization techniques to search a discretized reduction of the weight space. Another choice based on mathematical optimization is the use of multilinear regression between the sub-costs and some objective measure can be computed to refine the weights. As an example of objective tuning, F. Alías along with

X. Llorà proposed in 2003 a genetic algorithm for tuning the w_k weights of both target and concatenation costs simultaneously [Alias and Llorà 2003].

On the subjective side, hand-tuning the weights is by far the most popular option, done mostly using expert knowledge. Another method is to use subjective tests as a post-mapping stage to refine the weights, but this is a costly practice, both in terms of resources and time. Semi-automatic algorithms, based on interaction between the algorithm and a human operator can also be used.

F. Alías *et al.* provide a very good review of all proposed weight tuning techniques in their paper [Alías et al. 2011].

3.4 Concerning Preselection

As the problem of searching for variable-sized units in a corpus is computationally expensive, preselection filters are often implemented to (drastically) speed up the unit selection process, as for example in A. Conkie *et al.* work [Conkie et al. 2008]. They are used to prune very different units (according to the target cost philosophy) added to the graph or the lattice and contains phonetic, linguistic and prosodic related information. This technique was first presented in a 2000 paper by A. Conkie, along with another preselection method consisting in doing massive synthesis and remembering which triphones (the biggest unit) were used so that consecutive synthesis only uses these triphones [Conkie et al. 2000].

To achieve preselection with filters, a key containing discrete information (mostly binary) is created for each speech segment (phoneme or non-speech sound) in the corpus. That enables the algorithm to take or reject the unit quickly by just comparing the values in the key with target values. The key may contain phonetic, linguistic and prosodic information. Hereafter is an example of a set of filters that may be used for the preselection task, for each speech segment constituting the unit:

1. Is the segment a non-speech sound?
2. Is it in the onset of the syllable?
3. Is it in the coda of the syllable?
4. Is it in the last syllable of its breath group?
5. Is the current syllable in word end?
6. Is the current syllable in word beginning?

In this example (used in the IRISA TTS system), if no unit corresponding to the current set of filters is found, the preselection filters are relaxed one by one, starting from the end of the list. This mechanism ensures finding a path in all cases, but the drawback is that we can explore candidates far from the target features we want, thus risking to produce artifacts.

Actually, the purpose of the preselection filters is twofold. First, as we just said, it considerably prunes the graph explored by the unit selection algorithm, making the selection process faster. Second, it serves as a set of binary target cost functions relying on the assumption that if a unit doesn't respect the required set of features, it can't be used for selection. The preselection filters should therefore be seen as part of the cost for a node.

3.5 On Global Constraints

The selection algorithm and costs are particularly fit for short context problems. Fortunately, most problems unit selection faces concern the short context: concatenation of two speech segments on a few hundred signal samples and similarity measure between a unit and its target description. In addition, the formulation of the problem easily allows the use of contextual data (data concerning the predecessor/successor in the target sequence or in the corpus, data from predecessors in the candidate sequence) to improve selection. For example, for a pitch target cost, it may be more efficient to perform a distance over some considered diphone plus its predecessor and its successor than the sole diphone so that the global trend in the corpus part the candidate diphone comes from is also captured.

There is one thing that is not taken into account though: integration of long term constraints, *i.e.* constraints ranging on several syllables, words, constraints on the breath group or even on the sentence or more.

A few work proposed ways to integrate the missing information. For instance, an attempt to address that problem has been made by A. Popescu *et al.* [Popescu et al. 2006] by integrating a new set of sub-costs in equation 4.3, targeting wide-range constraints. As this drastically increased the complexity of the problem, making the optimal solution impossible to compute in reasonable time, a simulated annealing variant constructing an approached solution was also proposed.

In chapter 7, we will discuss another method to adapt the duration target cost so that it does not necessarily try to find the sequence that gives the best approximation of the predicted target duration but tries to obtain a sequence with an homogeneous distance to the target. The goal of this method is to guarantee that the selected sequence is free of compensation effects (*i.e.* none of the selected units presents an outlier duration that would have been compensated).

4 Signal Concatenation

The output of the unit selection stage is as simple as a sequence of unit positions in the corpus. The last component of the TTS chain is therefore the module handling retrieval and concatenation of the selected units. In addition to that task, it can also perform limited prosody modifications (or prosody adaptation) over the complete signal, like changing the pitch or accelerating speech rate.

The concatenation of two signals is greatly improved, as we said earlier, when done on the central part of phones, alias on diphone boundaries. An alternative worth mentioning is performing a search of the best point of concatenation between the two units, as done in A. Conkie and S. Isard's work in 1994 [Conkie and Isard 1994] where the authors propose a method to find the minimal spectral mismatch frames between the units. But even then, putting phonemes end to end is very insufficient to prevent the appearance of artefacts, which can be sorted in two categories:

- Pure concatenation distortions, caused for instance by unsuccessful interpolation or smoothing.
- Prosodic breakage, when joining diphones with very different prosody.

Thus, many techniques have been tested to eliminate these discontinuities but all actually end up spreading (but also attenuating) the artefacts on long portions of the concatenated speech segments. This contributes to the reasons for the quality drop between natural and concatenated speech: the latter is a set of speech fragments not intended to be put together, joined by a series of unnatural speech portions of non-negligible length where concatenations were performed.

The most common way to minimize distortions is by interpolation [9] of the signals to join on a few pitchmarks on the end of the left unit and on the beginning of the right unit. The corresponding interpolated segment is then used in replacement of the originals. This approach is directly inspired of TD-PSOLA algorithm [Moulines and Charpentier 1990]. The drawback is that this method is very basic and can cause a perceptible breakage of formantic trajectories.

A spectral smoothing solution presented by H. Pfitzinger in 2004 [Pfitzinger 2004] took that problem into account. The two signals derivative logarithmic magnitude spectra are first estimated, then the spectra are aligned using Dynamic Frequency Wrapping (DFW) which allows computing smoothed interpolated frequency responses (with a weighted linear interpolation between the two spectral representations). This spectrum is then converted to auto-regressive filter coefficients realizing a smoothed transition between units.

Another method, leading to better results than TD-PSOLA [Syrdal et al. 1998] but significantly more complex, has been developed by Y. Stylianou in 1996. The Harmonic plus Noise Model (HNM) represents the spectrum in two components: quasi-periodic harmonically related sinewaves and a noise component for representing non-periodic speech sounds (*e.g.* produced by friction). This model is used for performing concatenations by interpolating harmonic parameters of the model [Stylianou et al. 1997; Stylianou 2001], keeping a soft evolution of the spectrum parameters.

In 2001, J. Wouters and M. Macon also proposed a method that builds "fusion units" on the concatenation point. These units are built by reproducing the Line Spectrum

[9] According to [Syrdal et al. 1998; Laprie and Colotte 1998], among others.

Frequencies (LSF) parameters of a natural example of the concatenation point [Wouters and Macon 2001].

On the prosody modification stage, methods like PSOLA and its variants, HNM or STRAIGHT [Kawahara et al. 2008] can perform limited prosodic-order modifications of the signal: pitch adaptation and speech rate adjustment mainly. Modification must remain modest (no more than a $\pm 1.5\times$ modulation of the speech rate for example) in order not to degrade generated speech.

However, the prosody adaptation part is less and less used/developed in recent years, for two reasons: first, the risk of degradation is important and secondly, SPSS methods allow much better (and safer) prosody control for a decent output quality, most work on speech control is therefore centered around SPSS now.

5 Conclusion

In this chapter, we presented the second part of the TTS process: the TTS backend. Having a multi-level representation of the target utterance (from the input text) and the TTS voice, we presented how the unit selection engine builds a graph of units and we presented the graphs properties in detail. In particular, we showed that the unit graph is actually a lattice. We then presented the nature of the unit selection problem, a minimum cost pathfinding problem in a lattice, and the usual way to solve: via a pathfinding algorithm. These algorithms were then introduced, especially the most employed for the task: Viterbi. We then presented a literature review of the content and nature of the selection target and concatenation costs that drive the selection algorithm. Finally, we showed the standard ways unit joining is performed once the sequence of corpus units to concatenate was obtained by the selection algorithm.

Part II

Work on the Unit Selection Process

Chapter 5

Experimental Data and Evaluation Methodology

"Imagine how hard it would be to use a dictionary if its words were not alphabetized!"

Donald Ervin Knuth (1938–)
The Art of Computer Programming
Volume 3, chapter 5

We will now focus on the data, data storage tools and test protocols used through this document. This chapter is split into two main parts. In the first one, we will begin with a description of the ROOTS *toolkit that is used to store the data and then the conversion to a lighter format, used by the TTS system. Then, we will describe the automatic annotation process developed in the Expression team. We will finish that first part by a description of the two voices used in our experiments: Audiobook and IVS.*
Finally, the evaluation methodology used throughout the thesis will be detailed in the last section, with a presentation of the evaluation technique focusing on differences we presented recently [Chevelu et al. 2015].

1 Speech Synthesis Data Management

Speech data used for this thesis is managed by the ROOTS toolkit [Chevelu et al. 2014], developed within the team. ROOTS allows to store, analyze and manipulate speech data

conveniently. The base problem it aims to solve is the disparity of tools available for speech analysis. These tools rarely use the same input/output format and are rarely fully compatible from the beginning. The solution to that problem is to have a tool able to represent and store data and at the same time give an easily usable interface able to import and export data from and to analysis tools, see figure 5.1 for an illustration of this philosophy with ROOTS. The way data is represented in that system must be consistent, ordered and should be able to transcribe the totality of the information provided by analysis tools so that analysis is made possible. In addition, it must be able to represent as much annotation levels as possible and it ought to be upgradable so that new data can be represented by it. It is also preferable to make it usable from a wide variety of computer programming languages, especially script languages. The ROOTS toolkit was designed to respond to these issues.

Although other solutions exist such as Atlas, Agtk, Emu, Hrg or IrcamCorpus-Tools to cite a few, the wish to have a representation format as complete and upgradable as possible (contrary to Hrg which favors access speed over comprehensiveness) is one of the few factors that triggered the development of ROOTS.

1.1 ROOTS Toolkit

The ROOTS toolkit is based on the Object paradigm. It describes a corpus with the following concepts, from the smallest to the greatest reach:

Items: The basic element in ROOTS is the item. An item describes an element of annotation corresponding to a certain type. For example, it can be a phoneme, which in the ROOTS toolkit translates into an instance of the class Phoneme, a syllable, a lemma, a word, *etc.* An object can be a specialization of another object (Object notion of inheritance): A F0Segment and a SpectralSegment are specializations of a

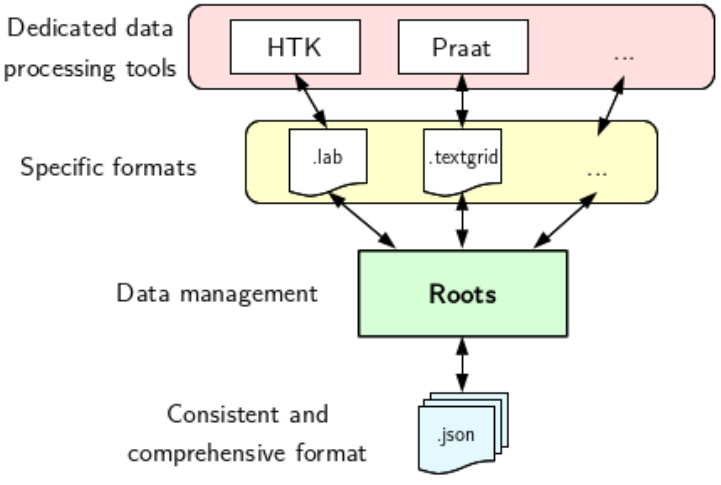

Figure 5.1: Positioning of the ROOTS toolkit in the hierarchy of speech analysis and management tools.

Segment.

Sequences: Sequences describe a temporal ordering of several homogenous items. Depending on the sequence, items may or may not be immediately subsequent in time but a same sequence can only represent one type of item. For instance, a sequence containing instances of the class Phoneme cannot contain an instance of class Word. Furthermore, a given item can be present in one and only one sequence.

Relations: A relation marks a link between the items of two sequences. Relations link two sequences that are part of the same utterance. They are not binary: an item in a sequence can be linked with none, one or several items of the other sequence. A relation between two items is directed. When loading a corpus, ROOTS creates a graph of all existing relations between the sequences. An example is given on figure 5.2. Provided existing relations in the graph form a link between two sequences not directly put in relation, ROOTS can compute the direct relation form existing ones. This is why, when building a ROOTS utterance, only a partial set of relations is sufficient to be able to derive a complete graph of relations between sequences. This is not always the case though as some relations may be meaningless (*i.e.* a relation between the phoneme sequence and the non-speech sounds sequence).

Layers: A self-sufficient group of sequences and relations can be put in a layer of an utterance. This allows to segment the utterance data in function of the annotation level it refers to. For instance, if only phonetic data is needed, the corresponding layer is the only one that needs to be accessed. This component of the ROOTS toolkit is only present to enhance utterance management and is in no way mandatory.

Utterances: All sequences and relations referring to a same corpus part, organized into layers or not, are grouped into an utterance. The Utterance class provides means to manage sequences relations and layers directly and can even retrieve item-level information.

Chunks: Huge corpora can be divided into chunks, each containing a given number of utterances.

Figure 5.3 illustrates the arrangement of these elements in a ROOTS corpus. Relations, though not presented on this figure, are usually represented as sparse matrices. Furthermore, items (as described previously) feature one particular case: some sequences use special “compound items”. A compound item is an item that cannot exist without one or several other items (usually elementary). For instance, a syllable cannot exist if the phonemes related thereto are not present in the corpus. This leads to the notion of embedded relation. Compound elements embed links to the items that are in relation with, even though these items are part of another sequence. ROOTS mechanisms ensures that when an elementary element is deleted, related compound elements are destroyed. This concept is implemented in ROOTS for two cases: syllabic and syntactic trees.

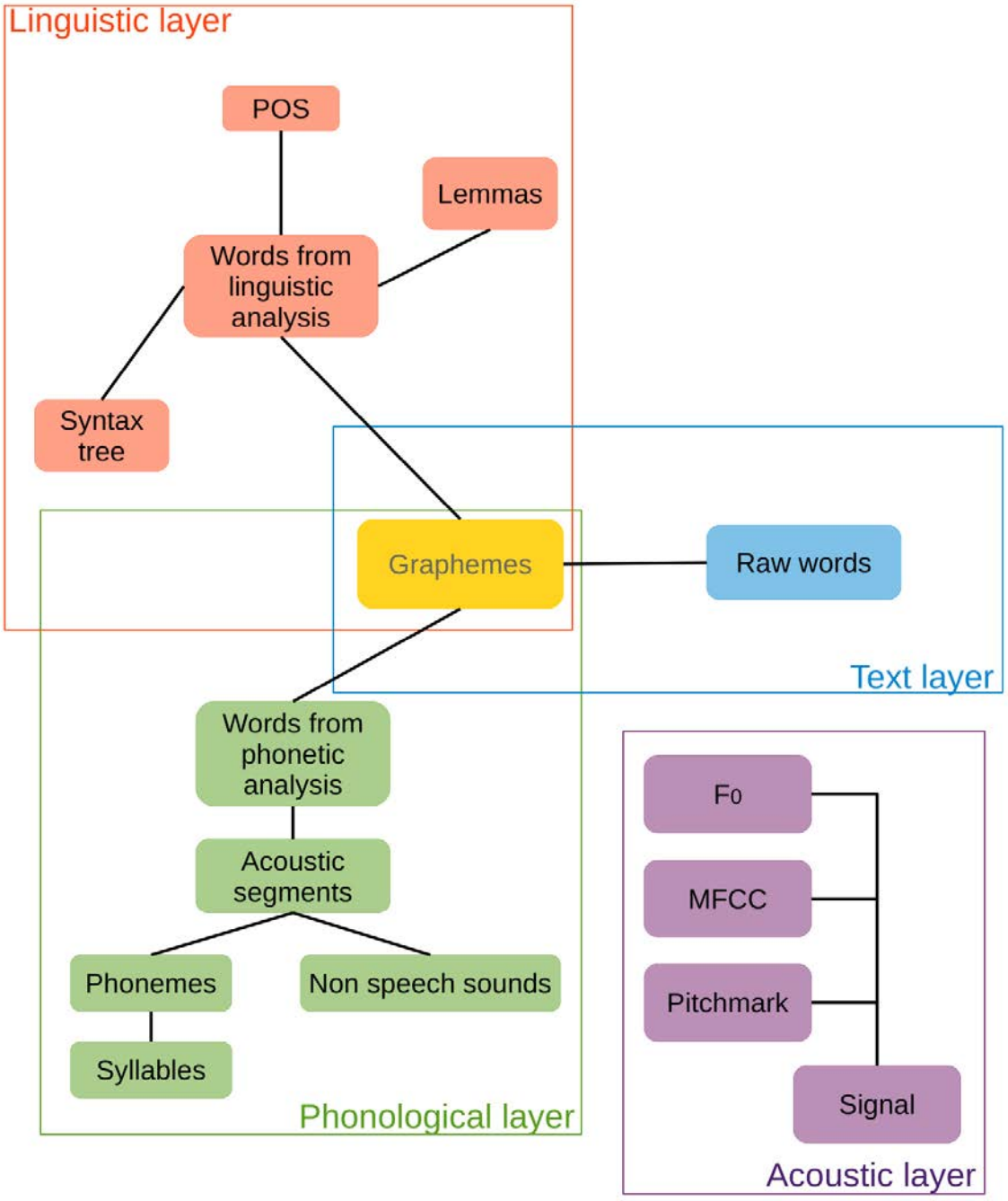

Figure 5.2: Simplified view of sequences and relations present in the data used for the thesis and managed using ROOTS. The sequences are grouped by thematic layer. The grapheme sequence is present in all layers to ensure syncing of the layers (the grapheme sequence is then used as a hub for joining other sequences and relations.)

Layers for ROOTS corpora are grouped according to the following (expandable) base groups:

- Linguistic data;

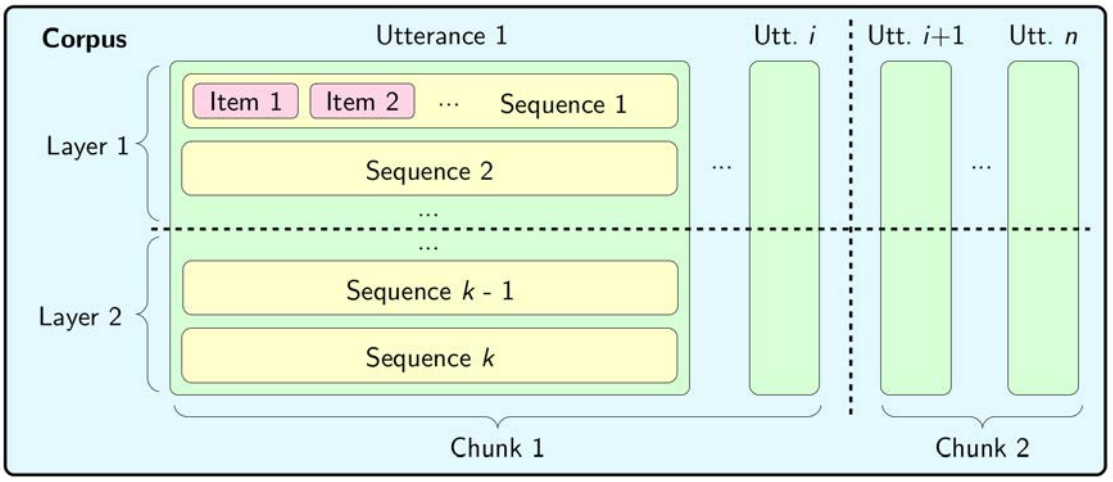

Figure 5.3: ROOTS toolkit data hierarchy.

- Phonological data;
- Textual data;
- Acoustic data.

The physical organization of ROOTS corpora puts utterances within separated JSON files, one for each layer, allowing to load only some layers in spite of the whole utterance. A metafile links them all together.

Corpus data in this thesis includes sequences and relations summarized in figure 5.2. It is divided into four distinct parts, according to the specification of the ROOTS layers presented previously. In order to make the link between each layer so that they can be merged when loading a full utterance, a pivot sequence is needed. This pivot is the sequence of graphemes, which is copied into each layer, with the exception of the acoustic one. The reason is that sequences in the acoustic layer all contain only one ROOTS item, which makes the link between the utterance and an external resource. For instance, the F_0 sequence contains an interface to exploit a file containing all F_0 marks concerning the current utterance. The information is sufficient for a program using ROOTS to make the link between items in other sequences with information provided by the files interfaced in the acoustic layer. The signal sequence allows to manipulate the wav file linked to the utterance.

1.2 Automatic Voice Creation Process

Based on ROOTS, the annotation method summarized in 5.4 has been developed in the team to create a corpus from rushes and import the data into ROOTS [Boeffard et al. 2012]. This process performs automatic annotation of an audio signal given the full version of the script. It was intended for audiobook automatic annotation, like *Audiobook*, one of

our two voices. The process performs annotation on several levels: linguistic, phonologic and acoustic mainly.

The process is divided in two main parts. The first one aligns every sentence in the text with the corresponding speech signal using an ASR system. In order to allow further manual checking and subsequent analysis, the speech signal is cut on pauses at the beginning of the process. In other words, they are cut by breath group. This cutting process is based on pause length and energy level in the speech signal with thresholds depending on the speakers' flow speed and recording level. ASR is carried out on each token and the resulting utterance is aligned to the original text. The matching portions serve as landmarks to align the non-matching parts of the recognized text with the original. Matching text associated with the right speech signal is then removed from the complete text during the alignment process. The ASR word recognition error rate can be used to signal a portion of the alignment that has to be controlled by an operator. Then, breath groups are joined to respect strong punctuation marks and exported to ROOTS. Information on the breath group boundaries is kept.

The second step enriches this simple alignment by performing syntactical, grammatical, phonological analysis and more. During the process, once the sentence alignment is achieved, corresponding ROOTS sequences and relations are created. Starting from that

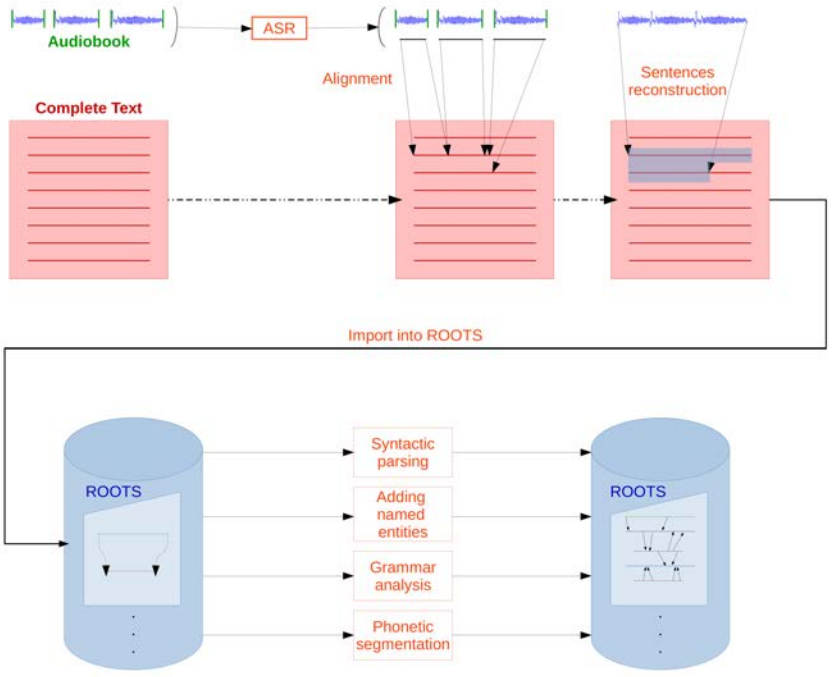

Figure 5.4: General description of the annotation operation. The process begins with two inputs, the complete book text and the audio signal coming from the corresponding text. It is divided in two steps; the first one aligning every sentence in the book with the corresponding speech signal using an ASR system. In the second step, the aligned data is analyzed and annotated using the ROOTS toolkit. This figure is inspired from [Boeffard et al. 2012].

point, all annotation steps directly enrich that simple ROOTS annotation by adding new sequences and relations.

In some cases, annotation data is available along with the original corpus. This is for instance the case for *IVS* corpus – which we will present in section 2.1 – where annotations, manually corrected phone segmentation in particular, were available. In that case, these annotations are added as new sequences and relations with scripts specifically built for the task.

1.3 TTS Corpus Format

The ROOTS toolkit is very efficient when it comes to managing, importing, exporting or analyzing speech data. However, it is not designed to be fast in read/write. Accessing ROOTS data is slow, although there are cache mechanisms to allow faster reading of already used relations/sequences. Storing data in text files makes the access operation slow (even with a format as compact and fast reading as JSON). The fact that all data of at least one sub-file has to be read to allow accessing a single acoustic segment data is particularly problematic. This makes directly interfacing ROOTS with the TTS system impossible.

This is why we designed a compact but very fast representation structure for communicating data to the TTS system. The goal is to be able to load a voice into the TTS system hash tables as fast as possible. For this, annotations are stored in the structure presented on figure 5.5, recorded in a single binary file. In this format, the base unit is the acoustic segment, phone or NSS, which is put in a fixed-size sub-structure. Each item is composed of a header, a key used for unit selection and sub-items with additional acoustic and prosodic information that might be needed both for selection acoustic costs and when extracting and concatenating the stimuli. The information available in the key is given in appendice A.

The segment sub-item makes the link between annotations and the speech data, stored in a separated signal file[1]. A third file contains pitchmarks associated to each item. Acous-

[1]The signal file is a lossless PCM flux encoded in mono with a depth of 16 bits. Sampling frequency depends on the voice (16, 44,1 or 48kHz).

Figure 5.5: Structure of the TTS corpus. Each segment is prefixed by a header and a key and has a fixed size to make recovery and reading speed faster

tic segments are stored in a table-like structure with a header containing the alphabets needed to interpret some item components and as every structure has a fixed size, the location in memory of all elements can be inferred easily. This, with the binary format by which data is stored, allows to load a full 10 hours voice in less than 15 seconds while loading from ROOTS would take several minutes. Once loaded in the TTS system hash tables, the voice can be used for as much synthesis as needed. Access time to the corpus during unit selection is reduced to hash-tables access time. During the signal generation step, access to the corpus is in real time, while loading data from ROOTS would once more perform much worse.

2 Corpora

In this thesis, we use two different speech corpora as our TTS voices. All synthesized speech samples we use in the experiments are synthesized with the two voices created with sub-sets of these corpora. In the major part of this section, we will discuss the content of the two voices we used in this thesis, *IVS* and *Audiobook*. We will finish by describing our various test and validation corpora. Some of them are extracted from *IVS* and *Audiobook* and also feature spoken samples, another one is entirely textual. To complete this section, appendix B provides detail on the exact content of our voice corpora from the phonetic point of view. All corpora are in French language.

2.1 Voice Corpora

Audiobook

The first of our two voice corpora (*i.e.* TTS corpora) is an expressive corpus built from an audiobook. Thereafter, we call it *Audiobook*. The speaker is a male and the mean F_0 value for voiced segments is particularly low, with an average of 87 Hz on voiced segments. Speech signal is sampled at 44.1 kHz, in mono (1 channel). It is stored with a lossless encoding. The corpus is automatically annotated using the global process described in section 1.2 and is represented using the ROOTS toolkit.

Since it is an audiobook, the content of the corpus and expressivity are completely uncontrolled. In particular, as the voice has not been recorded for TTS purposes, prosody is sometimes exaggerated. Nevertheless, it features homogeneous speech and recording was made under excellent conditions yielding to excellent signal quality.

The *Audiobook* corpus is composed of 3 339 distinct utterances, ranging from a few words to the size of a small paragraph. An extreme case is the utterance composed solely of the onomatopoeia “Ah !”. The average length of an utterance is of 39.92 phones. Each utterance is actually a paragraph in the book that is read. The corpus counts 10 hours and 45 minutes of speech. It contains a total of 11 305 distinct words (133 277 occurrences in total), including 6 909 nouns. 91% of that amount consist of non-proper nouns. Due to the literary nature of the corpus, distinct proper nouns in *Audiobook* are few: 13% of observed

proper noun occurrences are distinct[2]. In comparison, on our second corpus, *IVS*, this number is 62%. Names of a few important characters of the novel are repeated more than a hundred times. The main character, Albertine, has her name cited 689 times. Finally, sentences are mostly affirmative (149 exclamatory utterances and 223 interrogative).

For the needs of several experiments, speech samples from the same speaker annotated the same way (and possibly from the same literary style) that are not included in the TTS voice are necessary. To satisfy that need, 200 utterances were removed randomly from the corpus to create two 100 utterances corpora: *Audiobook test* and *Audiobook validation*. The remaining part of the corpus (3 139 utterances) constitutes the actual TTS voice. It is called *Audiobook learning*. Besides its use as our TTS voice, *Audiobook learning* is also employed to train our models, like the ANN used in chapter 7. Validation set is used to verify the efficiency of the models after training. The test corpus is used during the training process of our models (when using some) to control training quality at each epoch. But *Audiobook test* is mainly used as a test corpus for cases when an original speech stimulus is necessary for comparison to natural speech. For the tests, sentences including the most frequent proper nouns are avoided as much as possible for obvious reasons.

Thereafter, when using the term *Audiobook*, we will be referring to the full *Audiobook* corpus. *Audiobook learning*, *Audiobook test* and *Audiobook validation* will be employed when speaking of the 3 sub-corpora. The same will be done with our other corpus, *IVS*, which is also cut in 3 sub-corpora in the same way.

Main statistics of corpus *Audiobook*				
Corpus	Full corpus	Learning	Test	Validation
Utterances	3 339	3 139	100	100
Acoustic segments	404 279	376 418	14 875	12 986
Phones	379 897	353 691	13 987	12 219
Non-speech sounds	24 382	22 727	888	767
Syllables	165 320	153 917	6 102	5 301
Words	133 277	124 110	4 901	4 266
Length	10h45'12s	10h00'14s	24'02s	20'56s

Figure 5.6: Main statistics of *Audiobook* corpus and its sub-corpora *Audiobook learning*, *Audiobook test* and *Audiobook validation*.

Table B shows the main statistics concerning *Audiobook* corpus and its three sub-corpora. The distributions observed on the full corpus (phones, syllables, words) is uniformly distributed over the tree sub-corpora. Non speech sounds are mainly pauses and inspirations. The distribution of phonemes, showed in figure 5.7, is uniformly distributed among all four corpora and consistent with appearance frequencies of phonemes in spoken French as observed by F. Wioland [Wioland 1985]. Appendix B provides a comparison of Wioland's phoneme appearance frequencies with those observed on *Audiobook* and *IVS*. The diphoneme coverage of the learning corpus is not full (78%) but all the most commonly

[2] 390 distinct words for 3 064 occurrences.

used diphonemes are present. The missing ones are mostly very rare or even impossible diphonemes in French language.

IVS

Our second voice corpus, which we will refer to as the *IVS* voice, is also used for the experiments. It was recorded for TTS purposes within an Interactive Vocal System with a hand-made recording script which aim was to cover all diphonemes present in French and comprises the most used words in telecommunications vocabulary. It features a Female voice sampled at 16kHz (lossless encoding, 1 channel) with a mean F_0 at 163Hz for voiced segments, which is quite low for a female speaker. The corpus expression style is completely neutral and very controlled. In terms of intended goal and speaking style, *IVS* is *Audiobook* opposite. *IVS* was built for synthesis purposes while *Audiobook* wasn't. It is neutral and controlled while *Audiobook* is uncontrolled and quite expressive. The other strong contrast between the two corpora is over the annotation process because *IVS* annotations were manually corrected, which is not the case for *Audiobook*. In these conditions, opposing the results for the two voices in experiments is particularly useful.

The corpus is composed of 7 855 utterances, 245 232 phonemes and 20 961 Non Speech Sounds for 7h48' of speech data. Utterances in *IVS* are much shorter than in *Audiobook* with an average of 10.4 phonemes per utterance. In practice, they correspond to the utterances of the reading script, so almost always a sentence. Among the 81 662 words in *IVS*, 13 511 are distinct, 1 642 of which being proper nouns while 5 880 are non-proper. Few proper nouns are present several times and only six appearing more than 20 times [3]. Proper nouns are present in only 1 794 utterances out of 7 855. Sentences, as for *Audiobook*, are mostly affirmative (115 exclamatory utterances and 454 interrogative).

As for *Audiobook*, Agnes is sub-divided into three sub-corpora: *IVS learning*, *IVS test* and *IVS validation*. The three corpora are used for the same tasks, with *IVS test* and *IVS validation* also featuring 100 sentences, though this means the two corpora are smaller (as utterances are). As for *Audiobook learning*, the corpus *IVS learning* is the TTS voice. Thereafter, we simply refer to *IVS learning* voice as *IVS*.

Table 2.1 gives the main data for *IVS* corpus and its 3 sub-corpora, which shows data is uniformly distributed among the corpora. Figure 5.8 shows *IVS* phonemes distribution, which is also uniformly distributed among the sub-corpora of *IVS*. It is also very close to Wioland's phoneme appearance frequencies, much more than *Audiobook*, as it is more representative of spoken French than a novel.

[3] Words "Aujourd'hui" (today, used as a proper noun like in the newspaper "Aujourd'hui en France"), "Europe", "État" (the state) and "Jean" (John) appear 22, 25, 27 and 29 times respectively. "Paris" and France appear 41 and 75 times respectively.

Figure 5.7: Appearance frequencies of the phonemes in *Audiobook* corpus. The general distribution is close to that of spoken French [Wioland 1985].

Figure 5.8: Appearance frequencies of the phonemes in *IVS* corpora. For *IVS*, the general distribution is very close to that of spoken French [Wioland 1985].

Main statistics of corpus *IVS*				
Corpus	Full corpus	Learning	Test	Validation
Utterances	7 855	7 655	100	100
Acoustic segments	266 193	259 227	3 548	3 418
Phones	245 232	238 820	3 253	3 159
Non-speech sounds	20 961	20 407	295	259
Syllables	106 587	103 794	1 417	1 376
Words	81 662	79 511	1 138	1 013
Length	7h48'06s	7h36'26s	3'37s	2'53s

Figure 5.9: Main statistics of *IVS* corpus and its sub-corpora *IVS learning*, *IVS test* and *IVS validation*.

Test Corpora

As described in the preceding sections, we use *Audiobook test* and *IVS test* for subjective and objective evaluations when a natural reference or specific annotations are needed.

When this is not the case, we use two other test corpora, only textual. The first one is named *Combescure* (after the name of Pierre Combescure, who designed it) and features 100 phonetically balanced sentences. It is used to get reliable statistics for French.

The second text corpus consisting in 27 141 French sentences extracted from a wide variety of audiobooks, featuring many different styles. It will be called *Various Styles* thereafter. A part of that corpus was recorded by a male speaker. Speech was recorded in mono with a sampling frequency of 48 kHZ. The speaking style adopted by the speaker was neutral as the voice was recorded for TTS purposes (in order to be used as a TTS voice, which it is not in this thesis). This speaker is different from those that recorded *Audiobook* and *IVS* corpora. Recorded data for that corpus is meant for the same use as *Audiobook test* and *IVS test*. An extract of the 27 141 sentences of corpus *Various Styles* is given in appendix C. This corpus is used when no natural reference or specific annotation is needed.

3 Evaluation Methodology

In this section, we will review the protocol observed for all subjective tests performed in the following chapters.

When it comes to proposing a new feature in a TTS system, comparing features or even present a whole new system, a listening test is almost compulsory. The main goal being to produce speech that will be targeted at human listeners, human beings are the final judges concerning quality of speech synthesizers. Classically, both objective and subjective evaluations can be used. On the one hand, objective evaluations have the big advantage of being cheap and fast but no matter how pertinent they are, they still cannot replace subjective tests. On the other hand, to be interesting, subjective evaluations need a large number of samples to be evaluated and also a large number of listeners both chosen depending on the application domain of the system.

3.1 Objective Evaluation of Speech

On the matter of objective measures, considerable research has been put on the development of a good evaluation system that would evaluate speech with the right criteria, but none of the methods that have been proposed are sufficiently reliable, especially for evaluation of unit selection. Some measures were proposed for evaluating degradations on a voice signal, one of the most famous being PESQ (Perceptual Evaluation of Speech Quality) [ITU-T 2001], originally developed to assess the quality of voice codecs on telephone infrastructures. PESQ was tested as a potential subjective test replacer by M. Cernak and M. Rusko [Cernak and Rusko 2005]. The study showed that PESQ offered high correlation with MOS (Mean Opinion Score) perceptive tests, but the experiment was done under particular circumstances. Data from the same speaker was used for reference and test stimuli in one to one matching and synthesis consisted in groups of ten (called "decades") uncorrelated phonetically rich words chosen for their covering of Slovak language phonemes. The test was done using a diphone-based system with 3 different output stages. The authors insist on the fact that PESQ measures cannot be used directly on full test sentences. Indeed, this would require a guarantee on the quality of the time alignment between the two stimuli evaluated, which is hard to achieve. Instead, they segment synthetic and reference speech into words and compute the PESQ score and average for the whole test corpus. While very promising, this methodology remains restrictive as natural stimuli are very different to the "decades" the authors used. Another work, by F. Hinterleitner *et al.*, tests 3 objective measures (POLQA, DIAL and PESQ) on data from the Blizzard challenge (2008-2010 editions) [Hinterleitner et al. 2011a]. They confirm the time alignment issue but get much lower correlation with MOS values, especially while comparing full sentences.

Assessment of global speech quality is not the only type of objective measure available. Specifically focused on unit selection, some work tries to predict presence and position of concatenation artefacts, as for example J. Přibil *et al.* [Přibil et al. 2015]. In that work, a GMM-based statistical method is used with promising result.

In general, though, objective evaluation of speech stimuli remains very imperfect, mainly due to the difficulty to find measures that reproduce the cognitive processes happening in human minds. So subjective perceptual evaluations are still the main element to prove a new system or a new feature.

3.2 Subjective Evaluation of Speech

Several different types of perceptive evaluations are commonly used. Among all the methods, we can distinguish preference tests like AB and ABX, score tests like MOS, DMOS and more recently MUSHRA. All these methods serve the same purpose, which is ranking systems according to some subjective criteria.

In the literature, most of the propositions are perceptually evaluated. For instance, for the Blizzard challenge, a large scale evaluation campaign is used [King and Karaiskos 2012; Prahallad et al. 2014], but each time the number of utterances under test is restricted.

The same is true in the majority of the evaluations done. To cite a few examples, we can mention Inaki's work with 350 sentences [Sainz et al. 2014], Garcia's with 7 sentences for 5 systems [Garcia et al. 2006] or Hinterleitner with two blocks of 18 stimuli [Hinterleitner et al. 2011b]. Usually, the explanation for these low numbers of stimuli is that perceptual evaluations are really time-consuming. Some recent work have questioned the evaluation methodology, like [Latorre et al. 2014] which investigates the impact of listeners mental reference on perceptual tests results, or have proposed protocol modifications as in [Hinterleitner et al. 2011b; Viswanathan and Viswanathan 2005]. Even some alternatives to classic methodologies have also been used, based on crowdsourcing as described in [Buchholz et al. 2013].

More important than the small number of samples chosen, the fact that they are chosen randomly and not for their significance to the evaluated systems may bias the results of evaluations. In a work made in the Expression team [Chevelu et al. 2015], contrary to what is usually done, we proposed to synthesize a large number of samples (several thousands), using texts from various domains. Considering the high number of samples, we introduced an alignment cost between samples from a pair of systems to rank the samples by similarity. In order to do this, the alignment cost is based on Dynamic Time Wrapping (DTW). Once it is done, a perceptual evaluation using the most different samples was made. This way, no assumption is made concerning the quality of a system among the other, we simply focus the evaluation on what may make a difference between the systems. Such a strategy enables reducing the size of a perceptual evaluation to assess the difference significance between systems evaluated. This methodology was successfully tested both with a statistical system (HTS) and a corpus-based one. The results we obtained for AB preference tests are clearly significant while it is not the case when randomly choosing the samples.

3.3 Methodology Followed in the Experiments

For the experiments presented in this thesis, we use a group of 10 expert listeners. A larger group (20 listeners) would be preferable, but the lack of resource did not enable us to gather such a group (except for chapter 7 where 3 more testers were available). All testers are native French speakers with experience with synthetic speech.

We will mainly use AB tests to evaluate the new features we proposed. When circumstances do not permit the use of AB tests, typically when the goals of our experiments make it unusable, we will make use of MOS tests (or MOS derivatives like DMOS for evaluation of degraded speech). We will also make use of a MUSHRA test. The choice to use mainly AB tests, besides the obvious fact that we mostly want to assess user preference between several systems, was made for several reasons.

First, our experience with MUSHRA tests showed that this kind of test is very difficult to build and very difficult for listeners to perform. It is difficult to build because it normally needs systems serving as anchors to give higher and lower bounds to interpret test results. It is difficult for listeners to perform it because they are often proposed a test with 5 or

more speech stimuli per test step and interpreting each one's quality, finding how each one compares to the others and finally marking them is simply too much of a task. This results in an increase of fatigue for testers and might consequently cause a drop in quality of the results to occur. This is the reason we do not make intensive use of MUSHRAs in this thesis. Nonetheless, MUSHRA tests also present appreciable advantages. The lower and higher bounds, when present, allow some control on the results of the tests: results must be between the two bounds. More importantly, they offer, in only one test, the possibility to rank systems and thus get information on which system is preferred to which one, appreciate the performance gap between two systems. This is typically the conjunction of all major results MOS on the one side and AB tests on the other side can provide. Finally, less users are needed than for other tests to get meaningful results [ITU-R 2015] (even though the norm recommends 20 non-expert listeners). For these two reasons, especially the latter, we perform a MUSHRA test in chapter 8. Indeed, not all our testers were available for this test (only 7 out of 10 listeners).

Secondly, MOS tests , probably the most popular listening tests class, should only be used for an absolute ranking (in terms of absolute performance[4]), and not for direct confrontation in a 1 *versus* 1 opposition. In the literature, it is sometimes used to get conclusions on user preferences, which is biased as basic MOS tests never compare systems directly one against the other. In chapter 6, we will make use of MOS and degradation MOS tests exclusively in order to get an absolute mark and thus an absolute ranking of the systems we will compare. Degradation MOS (DMOS) compares a reference system to others, these other systems being considered as degraded versions of the first one. The mark, from 1 to 5, measures the harm caused by the degradation from "very important degradation" to "inaudible degradation". User preferences will be assessed only through AB tests. In addition, an issue with MOS tests is the – possibly huge – difference of rating scale from a tester to the other. A second issue is the possible presence of outliers. Removing these outliers means to privilege the majority of testers/answers and smooth the final results while including them may be seen as a potential noise (especially if it corresponds to a novice or unknown tester).

Finally, AB tests present the considerable advantage to be very easy to perform for testers, thus minimizing mistakes. They face only two stimuli and have a very limited choice (see next paragraph), so there is no fatigue, difference of rating scale or outlier problem with that kind of test. It also has the advantage of being and looking (which is even more important) fast and easy to perform, so more listeners are likely to make the effort to answer the test.

Unless stated otherwise, our AB tests will always have three possible answers: System A, System B or Indifferent. Including a fourth choice "Unable to answer the question" might sometimes be a good idea for AB listening tests, allowing to remove irrelevant data. It was implemented in some of our tests, but given the very few answers of that type, the

[4]Performance here refers to how well a system does what the *question* posed in the test asks.

answer was removed in subsequent tests. In most cases, the question focuses on overall quality. It is the following: "Of A or B, which sample seems to be of the best quality for you?" (French: « De A ou de B, quel échantillon vous paraît de meilleure qualité ? »). We will state the question when it is different but evaluating general quality is usually the best option as more specific questions can be biased by tester incomprehension of the question or side effects influencing her or his judgment.

In order to have results as general as possible, our evaluation protocol targets various points that may strongly influence listening tests. Indeed, focus will be given on the following aspects:

- Automatic or Manual segmentation/annotation;
- Gender of the corpus speaker;
- Impact of the proposition on randomly selected samples versus samples that are the most affected by the new features (as is [Chevelu et al. 2015]).

In order to assess the two first points, we conduct every AB test on our two voices *Audiobook* and *IVS*. *Audiobook* represents the male voice and the voice with automatic annotations while *IVS* is the female hand-checked voice. The same is done for the third point: a first AB test is made with randomly picked sentences in *Various Styles*, then a second test is done with the pairs of sentences that were attributed the biggest difference score. Most different sentences might be selected through a DTW, as we did in [Chevelu et al. 2015], or with other criteria that will be detailed in the related experiments. Hence, in order to perform one comparison of two systems, we carry out 4 AB tests:

- *IVS* voice, randomly picked sentences;
- *IVS* voice, most different sentences;
- *Audiobook* voice, randomly picked sentences;
- *Audiobook* voice, most different sentences.

Concerning objective measures, we did not define a global methodology, given what was presented earlier. Consequently, objective measures used in the experiments will be directly linked to the nature of the problem and will depend of the goal set in each case.

4 Conclusion

In this chapter, we presented in a first part the speech corpora we will use in the rest of the thesis. The first part of the chapter concerned the description of the tools we use to generate, store and exploit our speech data and corresponding annotations. We began by presenting the ROOTS *toolkit that is used to store the data and then the lighter TTS-corpus format, used by the TTS system. We finally presented the two voices used in our experiments:*

Audiobook and IVS, along with the test corpora we use to generate speech stimuli. The second part was devoted to the description of the test protocols. We first discussed the need of perceptual measures in the literature and then gave the considerations, hypothesis and the protocols we followed in this thesis.

Chapter 6

On the Choice of the Selection Algorithm

"With a whole assortment, we will have more choice."

Extract from *IVS* voice recording script.

In this chapter, we will first give a technical description of the TTS system developed in the team, called hereafter the IRISA TTS system. The TTS backend is the place where we conducted our work. Once the TTS system was available, our work was divided into two successive steps. First, we explored the impact of the search algorithm on unit selection. In particular, one of the questions is to assess the ranking made by selection costs and therefore to know whether optimality of the solution to the unit selection problem was necessary (or at least preferable). Once that step is realized, we focused on the cost function components. We will investigate the first step on unit selection algorithms in the second part of this chapter. The second step will be dealt with in the following chapters.

Speech synthesis systems usually use the Viterbi algorithm or sub-optimal variants – most notably beam-search algorithm – to solve the unit selection path-finding problem. However, this is not the only possible choice. In this work, we study a speech synthesis system relying on the A^* algorithm, which is a general pathfinding strategy developing a graph rather than a lattice. Using state of the art techniques, the algorithm is analyzed and compared to Viterbi before being evaluated through objective and perceptive experiments. Before conducting this comparison though, the first task is to assess the impact of preselection filters and cost functions on the selection to assert that they actually work as expected. In particular, a subjective evaluation of the N-best paths returned is made.

The conclusions drawn from the experiments are twofold. First, the A approach is an excellent alternative to beam-search, achieving better performance in the optimal case and also allowing optimizations to speed it up. Second, the impact on quality of the pre-selection filters set, used to restrict the number of candidates to the most promising ones, is low while they improve significantly the search performance. However, testers feedbacks show that filters improve prosody and naturalness perception of synthesized speech. In definitive, this study is a proof of concept aiming at demonstrating the feasibility and usefulness of using an A* algorithm to drive the unit selection process.*
In this chapter, we first describe the IRISA TTS synthesis system, which is the base we use for our work in section 1. We then introduce a modification to the selection block in which an A algorithm is used for unit selection (section 2). Our evaluation of these strategies, in the following sections (3 – 5), are distributed into three axes. We first study preselection filters and selection cost function to make sure they sort the selection graph as intended. Subsequently, we evaluate our A* algorithm and compare it to the usual beam-search strategies. Finally, we investigate whether an optimal solution to the unit selection problem is necessary.*

1 The IRISA TTS Synthesis System

The system that we use as the basis of all the work presented in the present document is the IRISA Text-To-Speech system. This system was in large part developed during the time of the PhD and most of its features were implemented to respond to research needs. All propositions in this thesis were implemented on successive refinements of the system. However, all results presented in this thesis are based on the last (2016) version of the system. The descriptions of the system in our various publications, despite the fact they do not refer to the same iteration of the system are nevertheless close to the current state of the software.

1.1 General View

Figure 6.1 gives a general view of the IRISA Text-To-Speech system. The first part of the process – the frontend – creates a ROOTS utterance enriched with annotations like phoneme, syllable or word sequences from the input text. It is done with automatic tools. The second part – the backend – is made of the unit selection block and the signal generation block. It communicates with the corpus through a hash table[1] that is precomputed before launching the TTS process and placed in shared memory.

The link between frontend and backend is performed by a conversion stage that converts the ROOTS utterance created along the frontend into the binary format of the TTS corpus.

[1] A set of hash tables actually. Each hash table contains references to the corpus with a fixed unit size. There are as much hash tables as there are possible unit sizes. In practice, units are set to be as long as two to three phonemes but it is possible to go much higher.

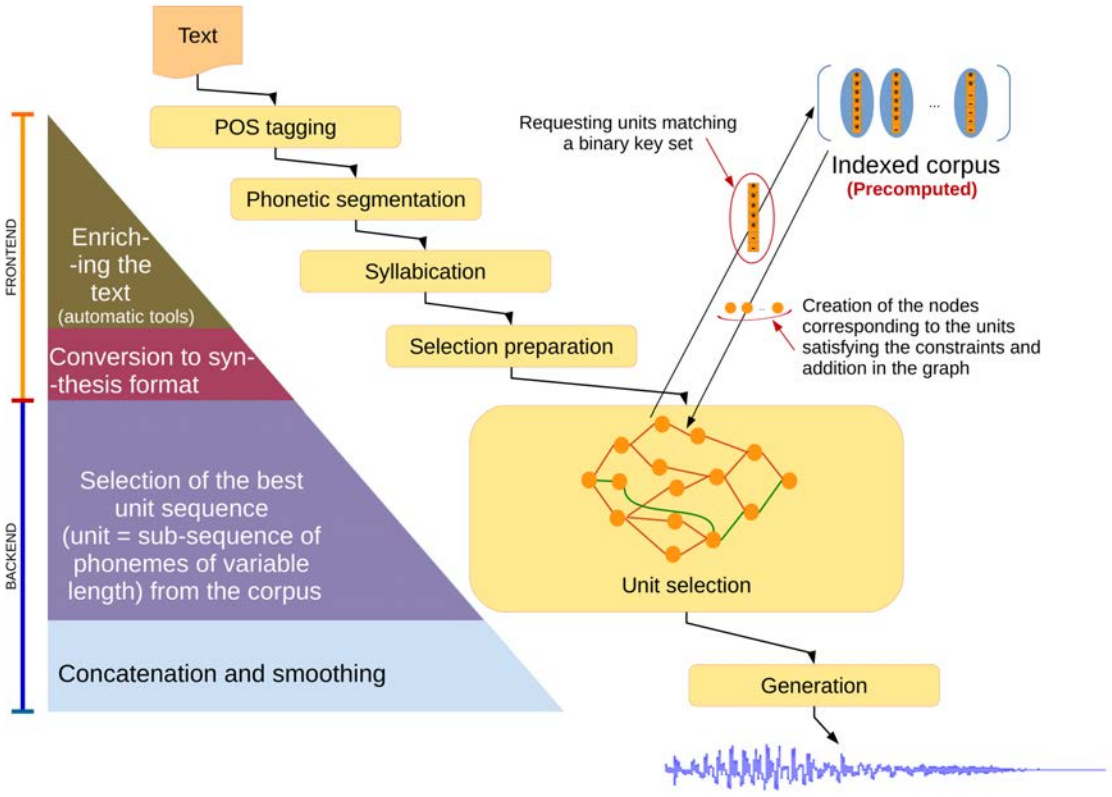

Figure 6.1: Workflow view of the IRISA TTS system.

1.2 Frontend

The first step in the TTS frontend is to convert the textual input into a ROOTS entry. This is done by a set of tools that produces a ROOTS utterance with a word sequence and a grapheme sequence linked by a relation. If the text to synthesize is consequent, the ROOTS entry can be split into several utterances according to strong punctuation marks. Thereafter, to simplify, we will consider the case of only one utterance. In the case of a multi-utterance entry, utterances are processed one after the other at each step. When this is done, this ROOTS utterance is enriched with three successive tools:

POS tagging: This step performs a Part Of Speech analysis and adds a POS sequence in the ROOTS utterance. To do so, two tools are available: The Stanford and Synapse POS taggers. In this work, we use the Stanford POS tagger, on account of its support of multiple languages.

Phonetization: The phonetization step adds a phoneme sequence, NSS sequence, word sequence (from phonetic analysis, as on figure 5.2) and relations between phoneme, grapheme words and NSS sequences. In our engine, three phonetizers are available: Liaphon [Bechet 2001], Espeak and an experimental tool in development in the team. In this thesis, we use Liaphon for the *Audiobook* voice as it was annotated with that

phonetizer and Espeak for *IVS*.

Syllabication: ROOTS syllabication algorithm is used to produce the corresponding sequence. The phoneme sequence is modified to make the link with the syllable sequence.

This is the default steps included in the frontend. Other steps may be added as well, for instance prediction steps to add acoustic or prosodic sequences. In chapter 7 for example, we add a frontend step that predicts and includes durations into the phoneme sequence. This step, when activated, takes place immediately after syllabication.

1.3 Backend

The backend part, described in detail on figure 6.2, starts with feature extraction from the ROOTS file generated by the frontend containing the target sequence with the needed annotations. The advantage of using a ROOTS file for the interface between the frontend and the backend is that almost any ROOTS file, from any corpus, can also be provided to the backend for synthesis. This is particularly useful for a task like synthesis using manually checked annotations or comparison with natural stimuli. Then the unit selection step is done using one of our selection algorithms. This step is parameterized by a cost function and user parameters (for example, requesting the best path or the N-best paths, requesting a sub-optimal version of the selection algorithm, deactivating the preselection filters, etc). Finally, signal generation is performed by mixing each two units with a PSOLA-like algorithm on a horizon of 2 pitch periods, using Hann windows.

Basic Concatenation Cost

The baseline concatenation cost is composed of MFCC ($\Delta\Delta$ coefficients), amplitude and F_0 Euclidean distances; three sub-costs, well rated in the state of the art (see section 3.2 for a review of interesting subcosts). Basic rules addressing duration were first included and then dropped, first because they did not show a real improvement and also because generated speech seems generally well enough. Nevertheless, the inclusion of a target cost or intonation models are interesting matters, which will be discussed in the following chapters. Equation 4.7 can thus be specified:

$$C_c(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}}) = C_{mfcc}(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}}) + C_{amp}(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}}) + C_{F_0}(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}}), \qquad (6.1)$$

where $C_{mfcc}(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}})$, $C_{amp}(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}})$ and $C_{F_0}(\mathcal{U}_{h,i}^{\omega_{h,i}},\mathcal{U}_{i,j}^{\omega_{i,j}})$ are the three sub-costs for MFCC, amplitude and F_0.

The corpus data for each cost is *z-score* normalized during the conversion of the ROOTS corpus to the TTS format. Therefore, all sub-costs are given equal importance to each sub-cost on a phoneme basis. Though all cost weights (W_{tc}, W_{cc} and w_ks in equation 4.9) are fixed at 1, the engine keeps the ability to introduce a different weighting. We use it in chapter 7 for manipulating the target cost/concatenation cost magnitudes.

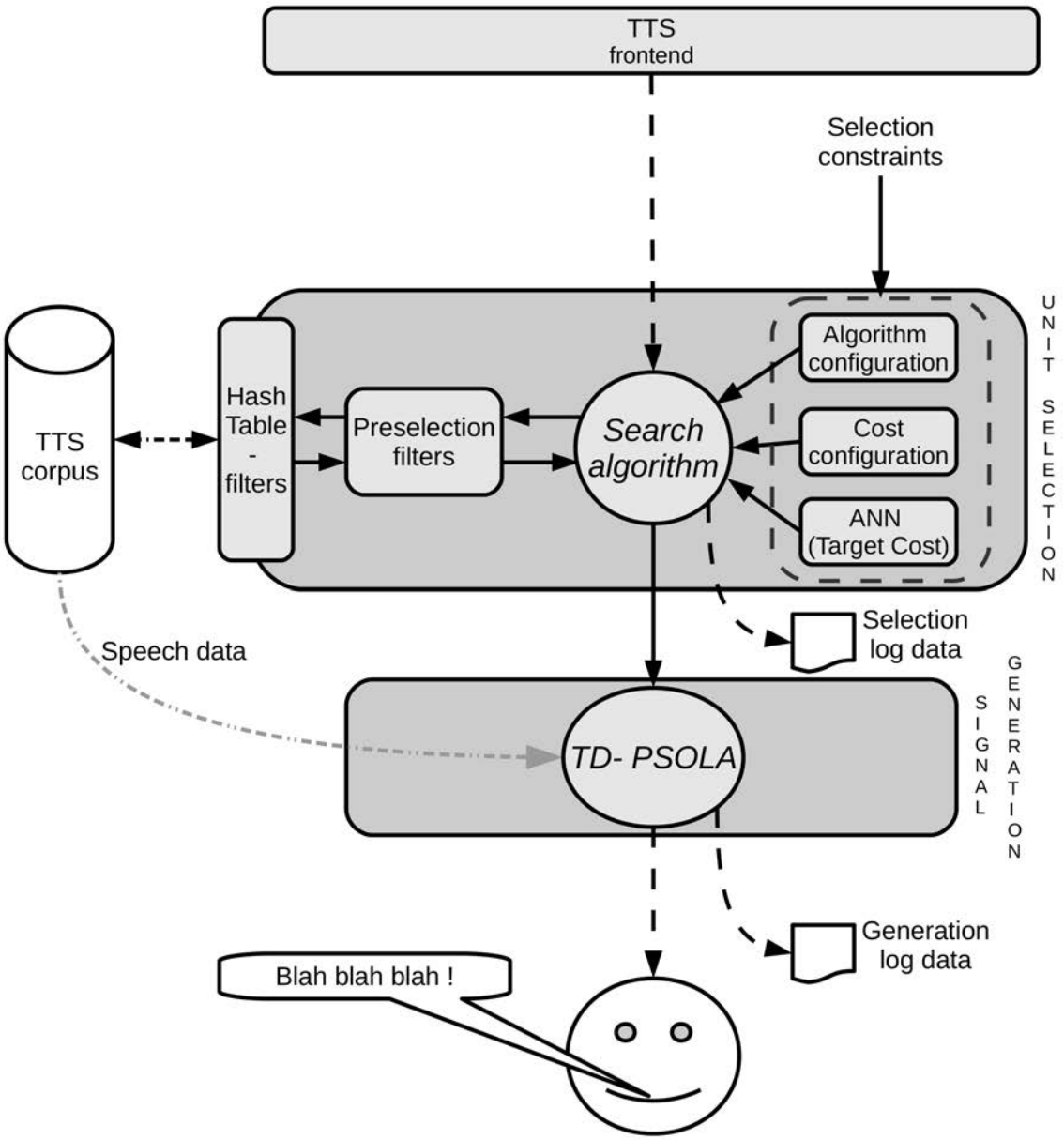

Figure 6.2: Technical description of the IRISA TTS system unit selection and signal generation blocks.

Preselection and Basic Target Cost

As the problem of searching for variable-sized units in a corpus is computationally expensive, hash tables and pre-selection filters are implemented to speed up the unit selection process [Beutnagel et al. 1998].

In the basic configuration of our system, we do not use any target cost and we set C_t to 0. Instead, we filter the candidate units from the corpus, by including in the selection graph only those matching a set of linguistic and phonetic features, which we call preselection filters [Donovan 2001].

Formally, let $D_{i,j}$ the sequence of target diphonemes from diphoneme d_i to d_j. we consider that we have a tuple of J filters modeled by indicator functions $f_j(\mathcal{U}_{i,j}^{\omega_{i,j}}, D_{i,j})$ ($j \in [0\,;J]$) equal to 1 if each diphone in $\mathcal{U}_{i,j}^{\omega_{i,j}}$ respects the condition posed by filter j on the corresponding target diphone of $D_{i,j}$ and 0 otherwise. We consider the set of units satisfying the I first filters for the target sub-sequence $D_{i,j}$:

$$O(I_{i,j}, D_{i,j}) = \left\{ \mathcal{U}_{i,j}^{\omega_{i,j}} / \prod_{i=1}^{I_{i,j} \leq J} f_i(\mathcal{U}_{i,j}^{\omega_{i,j}}, D_{i,j}) = 1 \right\}. \tag{6.2}$$

The preselection step aims at searching, for each target diphone $D_{i,j}$, the set $O(I_{i,j}, D_{i,j})$ of candidate nodes for which $I_{i,j}$ is maximal:

$$I_{i,j} = \arg\min card(O(I_{i,j}, D_{i,j})) \geq MIN_u. \tag{6.3}$$

Concretely, filters are implemented as keys containing discrete information (mostly binary) for each speech segment (phoneme or non-speech sound) in the corpus. This is the "item key" element described in section 1.3. When looking for a unit, the search algorithm asks for a particular key to a hash table, which takes or rejects the corpus items quickly by just comparing the values in their keys with the one provided by the algorithm. Matching elements are sent back to the algorithm which inserts them in the search graph. Binary masks are used to get access only to the desired information during runtime.

All the elements presented in appendix A are not used as filters. The reason is that too many filters tend to degrade the quality of synthesis, which is logical: as filtering is done before adding a node to the search graph, it can select or reject units based on its own criteria, without considering concatenation cost criteria. As the cost is a subtle balance between target and concatenation costs, giving too much importance to one (the target cost here), deprives the other of its working base and therefore renders it useless. Two filters are mandatory and are always included:

1. Unit label (*always active*).
2. Is the segment a non-speech sound (*always active*)?

These filters cannot be relaxed. As for other preselection filters, the default set we use in our experiments is the following, for each speech segment constituting the unit:

Table 6.1: List of the preselection filters for the French Language. The filters are sorted by scope width. Filters in bold are part of the filter set used in the thesis, others are discarded in this work but were taken in consideration and can be used in the engine.

Preselection filters for the French Language
Syllable level
Is the current syllable F_0 on a rising pattern?
Is the current syllable F_0 on a descending pattern?
Is the current segment in the syllable onset?
Is the current segment in the syllable coda?
Is the current segment the first phone of its syllable?
Is the current segment the last phone of its syllable?
Does current syllable have a coda?
Does current syllable have an onset?
Is the current segment in the onset of the syllable?
Word level
Is the current syllable in word beginning?
Is the current syllable in word end?
Sentence level
Is current segment in the last syllable of its breath group?
Is current segment in the last syllable of its sentence?

1. Is it in the last syllable of its sentence?

2. Is it in the last syllable of its breath group?

3. Is the current syllable in word end?

4. Is the current syllable F_0 on a rising pattern?

5. Is the current syllable F_0 on a descending pattern?

The priority order of the filters is the one given above. The pre-selection filters are relaxed one by one, starting from the end of the list. In the case of a non-speech sound, the only feature that matters is the first one, the others being all set to false. In our experiments, small variations to this set may be used, with very little or no impact at all. It is a subset of a wider filters set, which we tested extensively during the thesis (table 6.1).

The remaining information in the key may be used on particular occasions or for other languages than French. For example, the default filter set for English also included filters for the stress information.

Our default sets were constructed mainly using empirical knowledge. If all units are rejected, the filter set is relaxed until a sufficient number of units are accepted. In our work, the minimum number of units is 10. The set is temporarily relaxed (removing one by one the features that seem the less helpful) until a sufficient number of units is found. Though this is done to reduce the size of the selection graph (hence reduce selection time),

it is important also to consider it as part of the selection cost. In fact, it serves as a set of binary target cost functions relying on the assumption that if a unit doesn't respect the required set of features, it can't be used for selection. This means we have an absolute vision of what features units must match. One might argue this is not optimal, but by experience more refined tuning doesn't prove to be better.

In our implementation, the target cost is not directly incorporated in the cost function. Indeed, we consider that there is no need to integrate the nodes failing to show a certain fitting to the target sequence in the graph. As other works showed, the nodes achieving a good target cost are generally equally satisfying. Hence, features used for preselection also stand as binary target sub-costs. This means there is no target cost mark in our implementation, units are processed by pass or reject preselection filters. As the values we use are binary, their integration into the preselection filters is easy. Thus, the units that satisfy a given level of filters are considered equivalent regarding the target cost.

1.4 Perceptual Evaluation of the *baseline* System

The system described here is thereafter called *baseline*.

In order to give an indication of this *baseline* system overall quality, and especially its quality when combined with our own voices, we performed a MOS listening test involving 10 expert listeners. The four following configurations were presented to the listeners:

- Natural samples from *Audiobook* voice;
- Natural samples from *IVS* voice;
- Synthetic samples made with *baseline* system and *Audiobook* voice;
- Synthetic samples made with *baseline* system and *IVS* voice.

Each tester has been presented 10 stimuli for each of the four configurations, this makes 100 test occurrences per configuration overall.

In order to perform a valid comparison, natural stimuli for each voice are taken from *IVS test* or *Audiobook test* and synthetic ones are the artificial versions of the same utterances using the *baseline* system. The results of the evaluation are presented on table 6.2. In order to give an indication of the effect of having a full corpus (respectively 7 and 10 hours for *IVS* and *Audiobook*), the same results for a bi-gram corpus obtained by reducing *IVS learning* and *Audiobook learning* are also provided. Bi-gram corpora were assessed with the same function in an identical but earlier test.

Table 6.2: MOS test results for *Audiobook* and *IVS* voices using the A^* algorithm. The evaluation has been made on the *Audiobook* *test* and *IVS* *test* corpora respectively.

	Audiobook	*IVS*
natural	4.82 ±0.08	4.88 ±0.07
baseline	3.38 ±0.25	3.17 ±0.21
bi-gram corpus (indication)	2.14 ±0.14	1.72 ±0.08

The system is rated between 3.38 and 3.17, depending on the voice, which corresponds to fair quality and is quite representative of the state of the art. Refinements achieved on the unit selection block are discussed further on in this thesis and are therefore not represented in this test. Testers seem to give higher marks to *Audiobook* voice compared to *IVS*, in all cases. In all the experiments carried so far, this better rating and overall preference for *Audiobook* was always present. We assume there are two reasons for this result: First, *Audiobook* has a much greater expressivity, due to its audiobook origin and, second, the sampling frequency for the audio files of *Audiobook* is higher than *IVS*'s (44.1 kHz *vs.* 16 kHz). In consequence, synthesis with *Audiobook* seems substantially more natural.

2 Back to the Unit Selection Pathfinding Problem

We now come back to the unit selection engine described previously and consider whether the usual choice of a Viterbi-like algorithm is the best choice regarding the specifics of the problem.

2.1 Motivations

As we saw in chapter 4, computing the best sequence $\mathcal{U}^*$ leads to a pathfinding problem in a graph. To solve that problem, Viterbi algorithm [Viterbi 1967] (and its derivatives) has been almost the only one employed [Hunt and Black 1996; Conkie 1999; Clark et al. 2007], albeit it is not the only usable one. Over time, several enhancements of the algorithm were proposed, for instance by S. Sakai *et al.* [Sakai et al. 2008] or Tihelka *et al.* [Tihelka et al. 2010]. In particular, the real-time constraint imposed by many TTS application fields led to a broad use of under-optimal beam-search algorithms to solve the problem.

Other path finding algorithms like Bellman-Ford, Dijkstra or A^* [Russell and Norvig 2003; Nilsson 1982; Guennec and Lolive 2014a] are also fitted for the task. Even an exotic attempt to use a genetic algorithm by R. Kumar can be mentioned [Kumar 2004].

Strangely, there is little research in the literature concerning alternatives to Viterbi-like algorithms. There isn't much work giving justification for the choice of that algorithm over other alternatives. Actually, most implementations simply follow unit selection founding articles [Black and Campbell 1995; Hunt and Black 1996]. Most proposals simply try to refine it by adding preselection, clever heuristics or enhancements to the algorithm, but rarely propose to modify core mechanics of the selection process. Many contributions tend

to focus on the concatenation cost complexity, which is indeed the point consuming the greatest part of the computation time: $\mathcal{O}(N * K^2)$ versus $\mathcal{O}(N * K)$ for the target cost, K being the number of candidate phones in the corpus and N being the number of diphonemes in the target sequence. Little work was done, however, to decrease the computational complexity by changing drastically the search strategy (hence completely changing the search algorithm). One of most cited arguments to justify the choice of the Viterbi algorithm is its time-synchronous search in the selection graph, which – thanks to the lattice property of the selection graph – can effectively prove a useful property for on-the-fly synthesis/ASR [2]. For instance, it allows to begin the selection process while the user is still entering text. However, this constraint only exists in some of the TTS applications (and even when applicable, it is often not implemented), and many other cases simply do not require such an ability, *i.e.* full synthesis of an audiobook, resynthesis after translation (where the full sentence is usually needed before translating), simple TTS system, etc. Possibility to prune the graph is also highly cited as an argument for Viterbi, even though pruning is possible in most path finding algorithms.

In our work, we considered that Viterbi might not be the best algorithm for unit selection, and we proposed to investigate the interest of changing the exploration strategy in unit selection. Another inquiry that comes with this one is the following: what is the real importance of getting the optimal unit sequence? As the cost function used by the search algorithm is far from being perfect – it doesn't reproduce exactly the assessment a human would make of a speech stimulus, otherwise we wouldn't need perceptual tests anymore – one might ask whether or not it is really important to get the best sequence according to that cost function. An under-optimal result might do just as well. So the question is: can an under-optimal result to the unit selection problem be used instead of the optimal solution, and if so, how far from the optimum can it be before a substantial degradation is perceived?

In order to answer these questions, we decided to perform a comparison between beam-search Viterbi algorithms and an implementation of the A^* algorithm for unit selection. Several reasons justify our will to test an A^* implementation over other algorithms. First, A^* presents structural advantages for the inclusion of improvements, which is less true for Viterbi. In particular, A^* is better-suited for heuristic introduction to speed up the unit selection step, n-best path generation or preselection: all these options are available without modification. Of course, Viterbi can also do this, but only by introducing non-trivial modifications into the algorithm. Moreover, to enable the exploration of expressive corpus-based synthesis, we have to explore the in-depth behavior of the system and especially cost functions. Having the list of the n-best possible sequences proves useful in that matter. furthermore, the search strategy for A^* in the graph is drastically different to Viterbi: best-first instead of breadth-first for the latter. This is one of the reasons why A^* is regularly

[2] Normally, as Viterbi is a dynamic programming algorithm, computation is made in reverse order, from the end of the target sequence to the beginning. The lattice property of the unit graph allows reversing the algorithm however.

cited as one of the most used algorithms in the world over all applications.

Unit selection algorithms, especially Viterbi, have been presented in detail in section 2. In this section, we first introduce implementation details of the A^* algorithm after a quick reminder concerning Viterbi and its beam-search variant. We show how the algorithm is set in order to respond to the problem of finding the best unit sequence according to the cost function. The goal of this section being to present the specificities and advantages of A^*, we will not come back on these details again in this chapter.

2.2 Beam-search and Viterbi Algorithms

The basic version of Viterbi can be enhanced with several heuristics and improvements ([Sakai et al. 2008; Tihelka et al. 2010]), the simplest and most efficient being stopping the evaluation of a candidate unit when a following candidate has a better way to be accessed (making the algorithm a step closer to A^*). Better computation time can be achieved with harsher pruning.

Beam-search is a breadth-first search algorithm that keeps, for each target unit, only the N best nodes (best cumulated paths). This algorithm is a sub-optimal version of the Viterbi algorithm that avoids to explore the whole lattice of speech units, thus permitting real-time synthesis. This drastically prunes the lattice and breaks the optimality guarantee, but gives a much faster unit selection while generally keeping an under-optimal but still good quality solution. This algorithm is the one classically used to solve the path-finding problem and is implemented in our system.

2.3 A^* Algorithm

The second algorithm implemented in our system is A^* [Guennec and Lolive 2014a]. Contrary to the Viterbi algorithm, which computes a lattice containing all the candidate nodes (or at least M nodes for each time instant), A^* algorithm develops a graph. At each time instant, it explores the best node of the graph using a cost function that depends on both the path from the source node and the estimated cost to the target.

Originally introduced in 1968 by [Hart et al. 1968], the algorithm basically operates by searching for a path in a directed graph, whose nodes only have a finite number of successors, between a start node and a target node. The dedicated start node *init* is used to avoid arbitrary choice of the start node. It has the first candidate units as successors. The unique target node is called *end*. The algorithm uses a cost function of the form $f(n) = g(n) + h(n)$ with $g(n)$ being the cost of the sub-path between *init* and current node n and $h(n)$ being the estimated (heuristic) cost between n and *end*.

At each step, A^* takes the most promising node according to $f(n)$ and expends its successors (computing $f(n)$ by the way) until *end* is reached. $h(n)$ is a heuristic that enables to speed up the algorithm by privileging the nodes that seem to be on an optimal path over those which have a better $g(n)$ cost but may lead to greater costs in the future [Nilsson 1982].

Considering a unique target node, one of the main advantages of A^* is that the algorithm delivers an optimal solution if the heuristic is admissible, *i.e.* if $h(n) \leq h^*(n)$, where $h^*(n)$ is the real minimum value of the distance to the target node. In particular, note that the algorithm is optimal in the trivial case $h(n) = 0$, *i.e.* if there is no heuristic, and turns out to be equivalent to Dijkstra's algorithm.

2.4 Adaptation to the Unit Selection Problem

Algorithm 1 presents the implementation of A^* adapted for unit selection. The main functions that need to be adapted to our problem are (1) the cost function computation and (2) the successor function. In this work, we only consider the $g(n)$ part of the cost function, thus putting $h(n)$ to 0 which insures algorithm optimality. Function $g(n)$ is the regular unit selection cost function local to a node n. In consequence, using the formulation introduced in precedent chapters, the cost function is the following[3]:

$$g(n) = C(\mathcal{U}_{i,j}^{\omega_{i,j}}) = C_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) + C_c(\mathcal{U}_{h,i}^{\omega_{h,i}}, \mathcal{U}_{i,j}^{\omega_{i,j}}). \tag{6.4}$$

Concerning the successor function, it needs to consider domain-based knowledge. During the search process, each phone of the target sequence is considered as the start of a potential unit for developing the graph.

Furthermore, to improve algorithmic performance, the OPEN list is implemented as a binary heap sorted according to the cost function and a joined hash table to get quick membership queries. In addition, all the graph nodes are not computed, only those expanded during the successors search are really created.

In order to explore cost functions behavior, we modified the algorithm to be able to get the N-best paths, and also to get the N-best paths between a minimum and a maximum cost.

3 Evaluation of the Unit Selection Engine

3.1 Experimental Data

For the purposes of our experiments, we use our two voices: *IVS* and *Audiobook*. The sentences used as input of our experimental system, *i.e.* our test corpus, come from 3 different sources: *IVS test*, *Audiobook test* and *Combescure* test corpora.

3.2 Objectives

Our main objectives for this evaluation were the following:

- Assessing the impact of the filters on the selection process;

[3]Reminder: $h, i, j \in [\![1; N]\!]$, $h < i < j$.

Input: Graph **G** to explore, a sorted list ***OPEN*** and a list ***CLOSE***, both empty.

```
// Add start unit init of G into OPEN
add(G, init);

while OPEN ≠ ∅ do
    // Extract the first unit of list OPEN into u
    u = extract_head(OPEN);
    // Insert unit u into list CLOSE
    insert_unit(CLOSE, u);
    if is_target_unit(u) then
        // Add end unit after u in the candidate sequence.
        complete_path(u, end);
        // Exit backtracking the path from init to u
        backtrack(init,u);
    end
    // Function successors() returns the sub-set of G containing all
       the successors of unit u.
    foreach u' among successors(G, u) do
        f(u') = g(u') + h(u');
        if contains(OPEN, u') or
        contains(CLOSE, u') then
            if stored_cost(u') > C(u') then
                // Set a new cost f(u') to unit u'.  Each unit stores its
                   own cost.
                update_cost(u', C(u')) ;
                // Set u as the new parent of u'.  Each unit stores the
                   pointer to its parent in the graph.
                update_parent(u', u );
            end
        else
            insert_unit(OPEN, u');
        end
    end
end
```

Algorithm 1: The A^* algorithm. u (used to lighten notations) is any corpus unit (*i.e.* selection graph node) and u' is a successor unit of u in the selection graph.

- Evaluating the efficiency of the A^* algorithm for corpus-based synthesis and comparing it with the usual beam-search strategy;
- Exploring the best-ranked paths according to the cost function to see if a degradation is quickly perceptible and therefore if getting an optimal solution is necessary (or not).

To achieve these goals, experiments we conducted intend to:

1. Prove that a TTS system using A^* to drive a unit selection process is viable;
2. Assess the overall performance of the system;
3. Compare the performance (time and space usage as well as global synthesis quality) of the system when using the preselection and without;
4. Verify the stability of the cost functions presented above by:
    - Explore the variability & ranking accuracy for the n-best paths found by our system.
    - Assessing effect of reverting the cost function, *i.e.* selecting the worst possible path.

For our experiments on the n-best paths, we decided to fix n to 100, following empirical considerations over the average number of paths available with our voices. The goal here being to look only at a sample of the best paths, the $n = 100$ is a relatively small number in comparison to the usual thousands of paths usually available.

4 General Impact of the Cost Function and Pre-Selection Filters

Preselection has a tremendous impact on unit selection in the sense that harsh preselection will likely cause a unit shortage for the selection algorithm, thus making the cost function mostly ineffective. This is a particularly important problem as the concatenation cost, which is solely present in the cost function – unlike the target cost, which can be replaced by preselection filters, has the heaviest impact on synthetic speech quality. The first task to undertake is therefore to test whether our preselection set does not have a negative effect on the quality of TTS.

To evaluate pre-selection filters impact on quality, we conducted a DMOS subjective evaluation involving our 10 listeners. Each listener assessed 20 stimuli. We also took advantage of this test to verify that our cost function sorts paths effectively by introducing a variant to our cost function that reverts the score awarded to each unit. Therefore, in spite of selecting the best unit sequence, the algorithm selects the worst one.

The following four systems are compared:

- *filter* is the reference system with filters as presented previously (*baseline*);

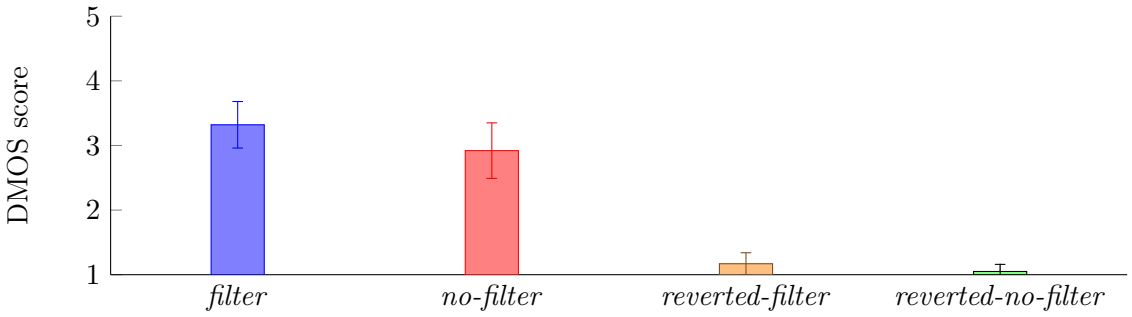

Figure 6.3: DMOS evaluation of the *baseline* cost function and preselection filters. The 4 versions of the system, from left to right: without filters, with filters, with filters and reverted cost function, without filters and with reverted cost function.

- *no-filter* is the reference system without filters (all units are considered based on their label);
- *reverted-filter* is the cost function is reverted to find the worst solution but filters are used;
- *reverted-no-filter* is the cost function is also reverted but no filters are used.

The results are summarized in figure 6.3. First, the scores obtained when reverting the cost function are significantly different from the reference system, actually getting almost the lowest possible score (1) with *reverted-no-filter* (it our aim was to get the worst possible voice, we would be done here). This is interesting because it shows that the cost function works as intended: it sorts quite appropriately unit sequences from unfit ones to better ones. Activating or not the filters for the normal cost function (*i.e.* *filter* and *no-filter* systems) leads to a substantial difference. Filters thus have a significant impact on quality showing that their influence on the cost function is real. This impact is also present, though to a lower scale between *reverted-filter* and *reverted-no-filter*. In this specific case, it avoids a drop in quality.

We can conclude that this result corroborates the fact that filters should be considered fully as part of the cost function. Using filters, as anticipated, also dramatically improves unit selection time (approximately by a factor 100).

5 Comparison of Selection Algorithms

5.1 Objective Evaluation

The main goal of this experiment is to study the behavior of A^* vs. beam-search in terms of performance and quality. First, we evaluate the A^* approach in terms of several objective factors:

- Unit selection execution time (in ms),

- Mean number of nodes passing filters,
- Mean number of nodes expanded,
- Number of concatenations

Data from the synthesis step has been gathered for A^* and three variants of beam-search (BS10, BS100, BS1000 depending on the size of the beam, *i.e.* 10, 100, 1000), with the two aforementioned voices and 3 different corpora. Results are presented in table 6.3, figure 6.4 and figure 6.5.

The number of nodes passing the filters shows that A^* is more restrictive than BS1000, itself being a bit more confining than other beam-search versions. Moreover, this number is quite high (above 100 nodes), even for A^*. The mean number of filters activated is almost constant with a value of 11.83 out of 12 filters, which means that, for all selected units, the 12 filters were activated almost all the time. Consequently, considering these results, more filters may be added to be more selective and thus speed up the selection process.

The number of concatenations that were made to synthesize the *test* corpus is the highest for BS10, drops for BS100 and BS1000 and reaches a minimum with A^*. That means A^* is able to find longer units in the *learning* corpus than other algorithms.

If confronted to mean breath group size, (around 19 for *Audiobook test* and *IVS test*, and 21.5 for *Combescure*, depending on the phonetization tool used) BS10 and BS100 tend to make more than 1 concatenation each 2 units and BS1000 and A^* less.

For all corpora, the number of concatenations is clearly lower for A^* compared to the other algorithms, except for BS1000. These two results show that A^* tends to develop a smaller graph than beam-search versions. Thus, it tends to show that applying A^* is more efficient in the sense that it finds the optimal solution while exploring less candidates. Moreover, the algorithms need almost 5 more units to synthesize *Combescure* breath groups than they need for *Audiobook test* when using *Audiobook* voice, where they need only 2 more for *IVS*. This is an indicator of a greater unit coverage in *IVS* which is linked to the origins of this corpus. Considering time needed to perform unit selection, we see that there is one order difference between BS10 and BS100 (real time) and BS1000 and A^*. The other point is that BS1000, while giving a sub-optimal result, takes more time than A^* (slightly with *IVS* voice and much more with *Audiobook*).

Figure 6.4 and figure 6.5 present the mean selection time and the mean number of nodes selected by target segment. Both figures show that A^* can be considered as a good tradeoff as it develops less nodes than BS1000 and thus gives a solution more quickly (which is optimal). Moreover, we see an order difference between BS10 and BS100 values, and one order between BS100 and the two other systems. Considering A^* and BS1000, the last one explores twice as much nodes while taking more time. This last result depends on the cost function which is guaranteed to be null for units that are consecutive in the corpus. This allows A^* algorithm to process quickly when it finds such consecutive units

Table 6.3: Objective factors for different algorithms on test corpora for *Audiobook* and *IVS* voices. The mean number of concatenations is given by breath group. Mean synthesis time, mean number of explored nodes and mean cost are given by target selection.

Criterion	BS10	BS100	BS1000	A^*
***Audiobook* voice** - Test Corpus : *Audiobook test*				
Mean number of nodes passing filters	288	290	271	202
Total number of concat.	8 608	7 767	6 741	6 500
Mean number of concat.	11	10	9	8
Mean synthesis time (ms)	11	99	679	206
N. of nodes in final paths	13 493	13 230	11 785	10 270
N. of corpus units in synthesized paths	9 395	8 554	7 528	7 287
N. of corpus units by breath group	12	11	10	9
Mean number of nodes explored	8	79	559	139
Mean cost	572	390	313	309
Mean number of arcs created	41	869	40 330	28 923
Mean number of pre-selection filters passed	10.95	10.95	10.95	10.95
***Audiobook* voice** - Test Corpus : *Combescure*				
Mean number of nodes passing filters	266	266	237	192
Total number of concatenations	1 953	1 842	1 677	1 659
Mean number of concatenations	16	15	14	13
Mean synthesis time (ms)	12	99	634	346
N. of nodes in final paths	2 478	2 442	2 227	2 035
N. of corpus units in synthesized paths	2 074	1 963	1 798	1 780
N. of corpus units by breath group	17	16	15	15
Mean number of nodes explored	8	74	538	198
Mean cost	1 003	787	713	710
Mean number of arcs created	41	855	42 897	37 336
Mean number of pre-selection filters passed	10.68	10.68	10.65	10.61
***IVS* voice** - Test Corpus : *IVS test*				
Mean number of nodes passing filters	218	221	203	126
Total number of concatenations	2 146	1 884	1 681	1 573
Mean number of concatenations	11	10	9	8
Mean synthesis time (ms)	12	73	450	311
N. of nodes in final paths	3 137	3 056	2 695	2 348
N. of corpus units in synthesized paths	2 341	2 079	1 876	1 768
N. of corpus units by breath group	12	11	10	9
Mean number of nodes explored	8	70	422	163
Mean cost	400	316	284	280
Mean number of arcs created	38	761	24 955	48 571
Mean number of pre-selection filters passed	10.87	10.87	10.86	10.83
***IVS* voice** - Test Corpus : *Combescure*				
Mean number of nodes passing filters	229	236	222	123
Total number of concatenations	1 728	1 548	1 368	1 288
Mean number of concatenations	13	12	11	9
Mean synthesis time (ms)	6	64	516	443
N. of nodes in final paths	2 354	2 322	2 057	1 773
N. of corpus units in synthesized paths	1 857	1 677	1 497	1 417
N. of corpus units by breath group	14	13	12	11
Mean number of nodes explored	9	75	492	197
Mean cost	421	330	295	292
Mean number of arcs created	41	867	31 867	66 176
Mean number of pre-selection filters passed	10.86	10.86	10.85	10.83

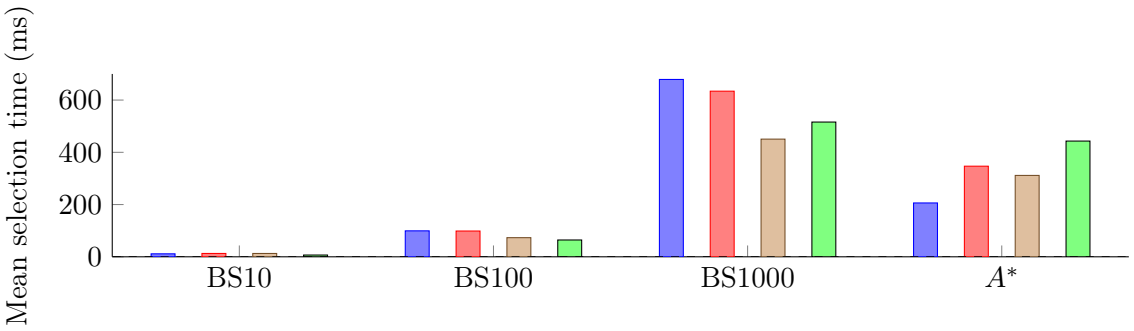

Figure 6.4: Mean selection time by target segment in ms. From left to right, bars represent the following combinations of voices and test corporas: *Audiobook* voice + *Audiobook test* (blue), *Audiobook* voice + *Combescure* (red), *IVS* voice + *IVS test* (brown) and *IVS* voice + *Combescure* (green).

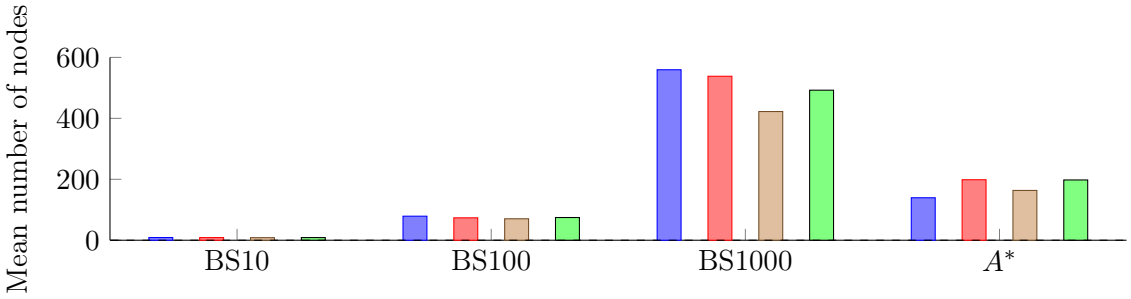

Figure 6.5: Mean number of nodes expanded by target segment. From left to right, bars represent the following combinations of voices and test corporas: *Audiobook* voice + *Audiobook test* (blue), *Audiobook* voice + *Combescure* (red), *IVS* voice + *IVS test* (brown) and *IVS* voice + *Combescure* (green).

while the BS1000 has to explore all the combinations (even if limited by beam size). It is also important to remember that, if real time is involved, implementing a pruned version of A^* is possible. It is possible to do it in many ways, for example by using non-admissible heuristic functions. Note that an admissible heuristic, though harder to find, could as well improve computation time.

A first conclusion about A^* usability, is that it seems to be a valuable approach to insure optimality while maintaining a reasonable search complexity: synthesis is not performing in real time but is much faster than a standard Viterbi.

5.2 Subjective Evaluation

In order to see what these results imply when synthesizing, a number of listening tests was accomplished. This evaluation intends to verify the global quality difference between all four algorithms. To perform this, we followed our two steps AB-based test methodology, presented in the previous chapter. For each of the four tests we made (randomly picked with *IVS* voice, *Audiobook* voice and most different according to a DTW distance with

Table 6.4: Results of the AB listening tests comparing A^* algorithm to 3 beam-search alternatives. The table is split in four distinct parts, depending on the synthetic voice that was used and the method for selecting speech stimuli included in the tests. The two first parts concern randomly picked samples from *IVS test* and *Audiobook test* while the 3rd and 4th are for the most different stimuli, selected via a DTW distance. Algorithms are confronted via two-by-two comparisons. **For each comparison, there are three values. The first (red) is for the algorithm in the corresponding column head (also in red). The second is the number of "Indifferent" answers. The last one (blue) is for the algorithm in the corresponding row head (also in blue).** Values are in vote percentage (40 votes for each configuration).

Randomly picked	*IVS* corpus (%)									
		A^*			BS100			BS1000		
	BS10	37.5	**40**	22.5	35	**42.5**	22.5			
	BS100	27.5	**50**	22.5				35	**37.5**	27.5
	BS1000	22.5	**52.5**	25						
	Audiobook corpus (%)									
	BS10	**52.5**	22.5	25	**47.5**	30	22.5			
	BS100	**37.5**	25	**37.5**				**42.5**	25	32.5
	BS1000	30	25	**45**						
Most different	*IVS* corpus (%)									
	BS10	**65**	17.5	17.5	**52.5**	22.5	25			
	BS100	**35**	**35**	30				30	**45**	25
	BS1000	35	25	**40**						
	Audiobook corpus (%)									
	BS10	**42.5**	27.5	30	**42.5**	20	37.5			
	BS100	**45**	30	25				**57.5**	22.5	20
	BS1000	**37.5**	30	32.5						

IVS, *Audiobook* voice), each one of our 10 listeners evaluated 20 distinct couples of stimuli. Each test confronted 5 different couples of algorithms (A^* *vs.* BS10, BS100, BS1000 and BS100 *vs.* BS10, BS1000), which means all pairs of algorithm were confronted 40 times one with the other. Each algorithm was represented by 20 stimuli for each pair, so that each comparison between two stimuli was done two times, by different testers. Test stimuli are extracted from *IVS test* and *Audiobook test*.

The results of the test are described on table 6.4. For each pair of systems, three values are provided. The red one corresponds to the number of times the system in the first row (the header) was chosen while the blue one corresponds to the system in the column header. The middle number, in black, is the number of "Indifferent" answers. Numbers are in percent of the total amount of votes for each configuration (40 votes). All possible comparisons were not performed. Indeed, the BS1000 *vs.* BS10 confrontation was left on the side, as its outcome – given the data from table 6.3 we discussed previously – was very likely to be extremely close to the results of the BS1000 *vs.* BS10 comparison. This conclusion is reinforced by the results of A^* *vs.* BS10 and BS100 *vs.* BS10 comparisons, which each put the first system (A^* and BS100) ahead of BS10. BS1000, yielding results that are much closer to A^* than BS100, should be very close to A^* result. For this reason,

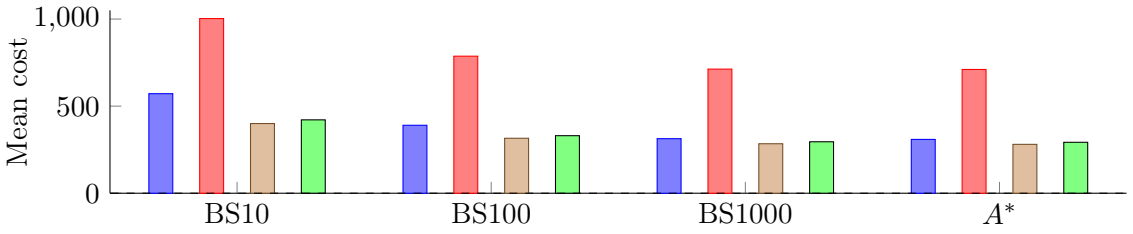

Figure 6.6: Mean cost by target segment for all 4 systems. From left to right, bars represent the following combinations of voices and test corpora: *Audiobook* voice + *Audiobook test* (blue), *Audiobook* voice + *Combescure* (red), *IVS* voice + *IVS test* (brown) and *IVS* voice + *Combescure* (green).

the comparison was omitted, which permitted to lower the complexity of the test.

When looking at the results for randomly picked stimuli, a clear pattern appears indifferently for both voices. All confrontations between BS10 and another configuration ends up with a clear defeat of BS10 (52.5% *versus* 25% for A^* *vs.* BS10 with *Audiobook* voice for instance), meaning this very pruned strategy yields inferior results. Then, all other comparisons are inconclusive, ending up in a draw and most often getting many "Indifferent" answers. A particular point is the A^* *vs.* BS1000 comparison with *Audiobook* voice where BS1000 has a limited lead, but the same comparison results in a very clear draw with *IVS* (22.5% to 25% with 52.5% abstention). This suggest the small lead of BS1000 with *Audiobook* voice is more related to a side effect than to superiority of the BS1000 strategy.

An important difference in the number of "Indifferent" answers between *IVS* and *Audiobook* can also be noted. While these answers count for about 40 to 50% for *IVS* voice when comparisons end up in a draw, *Audiobook* has a lower rate around 25%. There can be two reasons for that. First, variability for *IVS* is lower than for *Audiobook*, as *IVS* is a neutral voice that was recorded with much constraint while *Audiobook* is a very expressive audiobook. Second, segmentation for *IVS* was manually corrected while *Audiobook* ones are fully automatic, hence more subject to substantial variations especially where the concatenation process is concerned.

The result we have here is that speech quality only begins to degrade with very small beam sizes, lower than 100. This can be put in relation with the mean cost by target segment of the selected unit sequence, presented on figure 6.6. Here, the same pattern is observed: A^*, BS1000 and BS100 have very close mean costs, while it is significantly higher for BS10.

Now, in order to see if this result holds, we can look at testers answers for the most different stimuli. For these results, the same trends are observed for *IVS* voice: A^*, BS1000 and BS100 are difficult to order, while BS10 is clearly performing worse.

For *Audiobook*, there is one exception to that rule. Indeed, BS10 in on par with BS100 with a score of 42.5% (BS10) *vs.* 37.5% (BS100) votes and 20% abstention. Furthermore, BS100 is largely dominated by both A^* and BS1000, as for the BS1000 *vs.* BS100 duel for

randomly picked sentences. Another striking result is that the gap between BS10 and A^*, while existing, is quite small (42.5% for A^* against 30% for BS10). These results can be explained by the important variability in *Audiobook* corpus. A very plausible explanation for that phenomenon is the following. Costs along the best candidate sequences are quite homogeneous in terms of magnitude: the best case is that all units have a cost as close to zero as possible. Now, if a sequence is the absolute best but has a few units with substantially higher costs than its own average, these units might not be present in the BS10 graph while they might be present in the case of BS100. So BS10 might end up selecting a sequence with a higher cost in average, but more homogeneously distributed among the sequence units. The result is, for BS100, a stimulus that will perhaps feature an artefact on the part with a higher cost (which often occurs in the listening test) *versus* a stimuli featuring a worse cost in average but with a lesser risk of artefact in the case of BS10. As other results in this chapter indicate though, a *modest* difference of cost between two sequences does not imply that the sequence with the lowest cost is actually better. This reasoning is in fact the foundation of our work on the target duration cost, which is presented in chapter 7. In this work, we try to favor sequences with an homogeneous cost, with very encouraging results.

Overall now, considering the results of all listening tests along with objective measures, we can draw the conclusion that:

1. Speech generated with a beam-search algorithm is perceived as qualitatively equivalent to an optimal algorithm like A^* or non-pruned Viterbi until the size of the beam gets (at least) as little as 100 nodes.

2. Looking for the optimal solution of the unit selection problem is superfluous as a BS100 search yields (except for *Audiobook* most different sentences) a solution in real time, hence much faster.

5.3 Behavior of the Cost Function With the 100-Best Paths

In order to understand this phenomenon, let us look at the (sorted) list of the best possible unit paths in the selection graph[4]. Figure 6.7 shows the evolution of the global cost for the 100 best paths found. The target sentence is *"Car ce n'est pas le chagrin qui la fit partir."* (Because it is not grief that caused her to leave). Due to the differences between selected units among the paths, costs are reported to each phoneme on the sequence (x axis). At first glance, we observe little variability among the paths. Most changes seem to occur on the first/last units (non-speech sounds here actually). The mean number of units passing the filters is approximately 200, with a mean size of 3.8 phones for selected units, which is satisfying. We noted that the relaxation of filters was quite rare, which could signify other filters can be added to refine our target cost.

[4]Obtaining this list is particularly easy with A^* algorithm as the only thing that needs to be done to get it is to continue the search after the optimal solution was found, which leads to the second best, and so on.

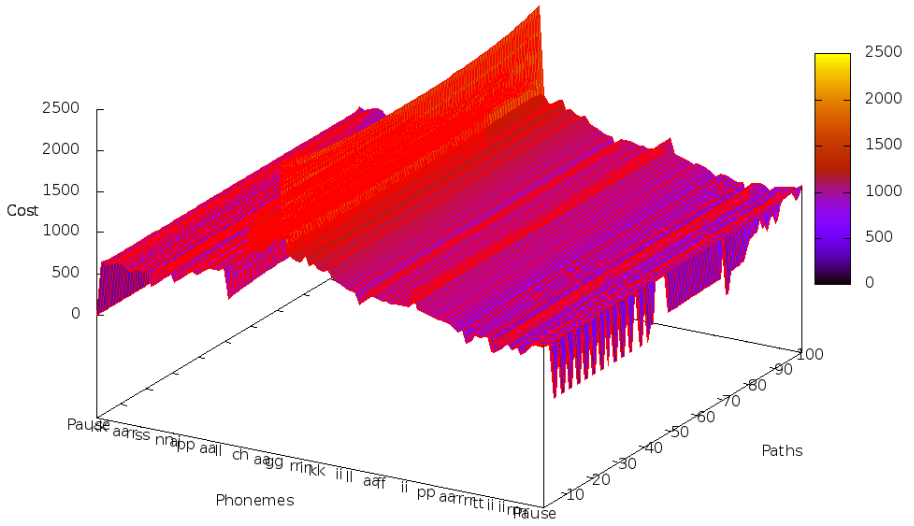

Figure 6.7: Global cost evolution for 100-best paths (French sentence *"Car ce n'est pas le chagrin qui la fit partir."*).

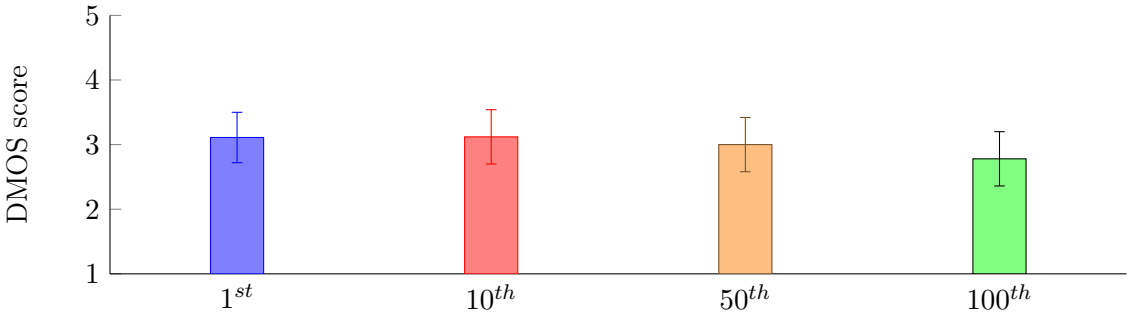

Figure 6.8: DMOS test results for the 1^{st}, 10^{th}, 50^{th} and 100^{th} paths of the cost function. The voice used for this experiment is *IVS*.

Given these results, we have decided to conduct a DMOS test in order to evaluate the loss in performance when selecting a candidate far from the optimal path. Each time, the natural signal is confronted to a synthesized signal corresponding to the 1^{st}, 10^{th}, 50^{th} or 100^{th} path according to unit selection and a duplicated natural reference. Each listener hears the 12 same sentences for each system. Figure 6.8 (right part) shows results of the test.

The results for the 1^{st} and 10^{th} paths are identical at 3.11 ± 0.4 and 3.12 ± 0.4 respectively, meaning slightly annoying degradation in average. The two other configurations get lower marks. The 50^{th} path is rated at 3.00 ± 0.4 and the 100^{th} gets 2.78 ± 0.4. No clear preference can be observed, in particular due to confidence intervals. In fact, it can be put in relation with the AB listening tests on A^* and beam-search algorithms where a similar trend is observed. The performance of BS10 can be compared to the result of the 100^{th} path in the current test, with a comparable increase in selection cost and drop in quality. As for the AB tests also, the first paths get the same marks. Quality then seems to degrade

faster and faster: about 0.10 points between the 10^{th} and 50^{th} path compared to over 0.20 between the 50^{th} and the 100^{th}. Actually, two major cases were encountered in the test. First, many stimuli were very close, if not sounding identical (in particular between the 10^{th} and 50^{th} paths but not only). Second, many others differed in their prosodic contours, often a lot. Many times though, both were correct or as close to be correct but in different contexts (question *vs.* affirmation mostly). As no prosodic target is provided by the high-levels though (except for the F_0 preselection filter), these stimuli were rated the same way. Nonetheless, as these prosodic variations seem (logically) random, a listening test with a particular focus on prosody should not give a different result.

To conclude, this test further reinforces our conclusion that the optimal solution to the unit selection problem is needed at the 10^{th}, for example, will perform just as well. Thankfully though, the selection function works appropriately and the optimal solution fares as well as the 10^{th}.

6 Conclusion

An experimental corpus-based TTS system and a complete evaluation of the algorithmics of its unit selection module were proposed in this chapter. The system, designed as an experimental platform to explore the behavior of concatenative speech synthesis in depth, implements state of the art mechanisms to perform the unit selection. A comparison was made between two approaches for performing the unit selection part in a corpus based TTS system: the usual beam-search strategy and a new A^*-based algorithm. The comparison, considering objective measures as well as subjective assessment by listeners, was made using the same pre-selection filters, cost function and corpora for all unit selection algorithms tested.

The results showed that A^* in its admissible version performed faster and better than a beam-search algorithm with a huge beam size. However, beam-search with a tiny beam, far from the optimal solution but running in real time, is actually perceived as good as other algorithms results. Even if A^* achieves a lower number of concatenations, compared to the other algorithms, subjective evaluations show that it does not imply better speech quality. The results seem independent both of the voice style used and the target corpus used. This leads us to two conclusions: first, finding the optimal solution to the unit selection problem seems of little use, as heavily pruned algorithms fare as well as admissible ones. This is because the variability between the best paths is very low. Second, A^* is indeed better suited than Viterbi for unit selection. In particular, even though this was not demonstrated here, it can also be pruned in order to get an under-optimal solution faster. However, in the rest of this document, we will always use the optimal unit sequence with A^* algorithm when generating test stimuli. The reason is that pruned search could be a side effect adding a bias in our further results.

Furthermore, as unit selection is highly dependent of the preselection strategy as this puts a severe constraint on the engine, we evaluated the impact of filters on quality of the syn-

thesis. We showed that our current set of filters did not degrade synthesis while saving a considerable amount of computation time.

Part III

Work on the Unit Ranking

Chapter 7

Work on the Duration Target Cost

"Oh time, you devourer of things,
and you, envious old age,
Together you destroy everything.
And slowly gnawing at them with your teeth,
You consume all things, little by little, in lingering death!"

Publius Ovidius Naso (Ovid)
(43 BC–17/18 AD)
Metamorphoses, volume 15, 234-236

In this chapter, and the next one, we focus on the target cost; formulating new propositions to enhance it. Here, we describe a new duration target cost that takes a whole sequence into account [Guennec et al. 2015]. It aims at selecting a sequence globally good, instead of a very good sequence almost everywhere but having a few local duration cost leaps that are counter-balanced by other units. The problem of weighting this new duration cost with other sub-costs is also investigated. Experiments showed this new measure performed well on sentences featuring duration artefacts, while not deteriorating others.
The proposed target cost and the underlying duration model are presented in section 2. Experimental evaluation on French corpora including objective assessments of both the model and the target cost (3.2) and subjective evaluation by listeners (3.3) are presented in section 3. Conclusions and future work are presented in section 4.

1 Motivation

Speech created using unit selection features naturalness and prosodic quality unmatched by other methods, as it basically concatenates speech actually produced by a human being. For this reason, most industrial TTS systems mainly use either pure unit selection approaches or hybrid ones. However, unit selection offers less control than statistical parametric methods, especially over prosody. Moreover, artefacts may appear in the synthesized signal and penalize intelligibility. While obtaining good speech output with neutral voice is (almost) a solved problem with unit selection, getting prosody right for natural and expressiveness is entirely another matter. Prosody modification methods after selection - like TD-PSOLA for adapting duration - are an option, but for now none has been convincing. The possibility of influencing selection to choose units that are the closest to the required prosody remains. A good state of the art for expressive speech synthesis is made in [Govind and Prasanna 2012].

As phonetic durations are subject to a lot of changes when considering voices with different levels of expressiveness, controlling duration gets particularly important. Lastly, decision trees have been the most widely used method to predict duration, for instance, in systems like HTS, with only a few mentions to using a target duration cost (*e.g.*, in [Alías et al. 2011]) within unit selection cost function. Recent approaches where DNNs replace HTS decision tree can also be mentioned [Hashimoto et al. 2015].

In this chapter, we propose a new way of computing duration target cost, not only based on the assumption that we want to get units as close as possible to a predicted duration. Thus, we try to find the units that stay the closest to requested duration by optimizing the mean duration error with respect to the previous units. Hence, it prevents inadequate units in terms of duration from being selected if other units are available while not forcing a path with homogeneous durations. The main idea is that it is better to have units globally longer or shorter than to have only one or two units with a big duration error in the synthesized speech.

2 An Adaptive Duration Target Cost

2.1 Neural Network

Prediction of phoneme duration has a long history in the TTS field. It was first performed by creating expert hand-made rules that were integrated in rules-based (formant synthesis) and concatenation synthesizers. Over last years, decision trees have been the most widely used method to predict duration, for instance, in systems like HTS. In particular, the use of neural networks for phoneme duration prediction starts in the early '90s. A TTS system using a set of ANNs (one for each phoneme) trained on cepstral coefficients can be cited [Tuerk and Robinson 1993]. A TDNN (Time Delay Neural Network) has also proven to be very efficient for predicting duration, though the learning set was small [Karaali et al. 1996]. In following years, major improvements in the technique were obtained mainly by

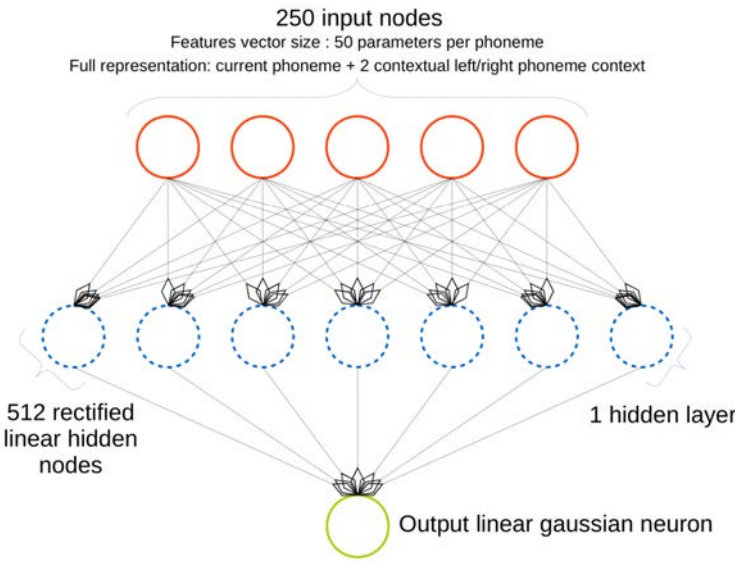

Figure 7.1: The neural network used for the prediction phonemic durations, composed of 250 input neurons for as much input features, 1 rectified linear hidden layer of 512 neurons and one output linear Gaussian neuron (the predicted duration).

increasing the number of input features and the size of the learning corpus. The advantage of neural networks is that, contrary to decision trees, they do not cluster predicted values (at least when properly trained). When the network faces an unknown set of features, the predicted value is different from an assimilated result for the closest feature set, which can result in much better results [Taylor 2006]. Recent work in speech synthesis is now focusing on deep approaches (DNNs, DBNs, DRNs). For duration prediction, we did not think such deep approaches were necessary. Thus, we use a MLP (Multi-Layer Perceptron) with batch gradient descent. Input data is composed of a set of 50 features by phoneme, mainly phonetic and linguistic parameters. We also take into account the contextual information for the two preceding and following phonemes. Thus, the network has a topology of 250 input neurons, 1 rectified linear hidden layer of 512 neurons and one output linear Gaussian neuron (directly predicting durations in *ms* as other measures like *log ms* were not performing better). This layout is summarized in figure 7.1. These parameters were the best among the different configurations tested.

2.2 Duration Target Cost

The proposed duration target cost aims at influencing selection so that selected units are, on average, at the same distance of the predicted unit durations. Defining the cost that way means we prefer a sequence moderately close to predicted values, but homogeneous in the repartition of the duration distance among units, to a sequence of perfect elements featuring one unit with dramatic cost. The cost for the n^{th} candidate unit $\mathcal{U}_{i,j}^{\omega_{i,j}}$ ($\mathcal{U}_{h,i}^{\omega_{h,i}}$

being the $(n-1)^{th}$) in the sequence $\mathcal{U}^*$ (see eq. 4.3, chapter 4) is as follows:

$$D_e = |D_t(\mathcal{U}_{i,j}^{\omega_{i,j}}) - D(\mathcal{U}_{i,j}^{\omega_{i,j}})| \tag{7.1}$$

$$C_d(\mathcal{U}_{i,j}^{\omega_{i,j}}) = |\Delta(\mathcal{U}_{h,i}^{\omega_{h,i}}) - D_e| \tag{7.2}$$

$$\Delta(\mathcal{U}_{i,j}^{\omega_{i,j}}) = \frac{\Delta(\mathcal{U}_{h,i}^{\omega_{h,i}}) * (n-1) + D_e}{n} \tag{7.3}$$

with $\Delta_{\mathcal{U}_{i,j}^{\omega_{i,j}}}$ being the mean distance to predicted duration for previous target units in the sequence (from the first selected unit to $\mathcal{U}_{i,j}^{\omega_{i,j}}$), $D_t(\mathcal{U}_{i,j}^{\omega_{i,j}})$ the target duration for unit $\mathcal{U}_{i,j}^{\omega_{i,j}}$, $D(\mathcal{U}_{i,j}^{\omega_{i,j}})$ the duration of $\mathcal{U}_{i,j}^{\omega_{i,j}}$ and $C_d(\mathcal{U}_{i,j}^{\omega_{i,j}})$ the target duration cost for unit $\mathcal{U}_{i,j}^{\omega_{i,j}}$.

Equation (7.1) computes the local cost between the target duration and the current unit. This cost is then used to compute the duration target cost in equation (7.2), which takes into account the mean distance to predicted duration for all the previous units. Finally, the mean duration error is updated using equation (7.3). Thus, the quality of the current unit depends on the quality of previous units. In other words, it means that if $\mathcal{U}_{i,j}^{\omega_{i,j}}$ is longer (resp. shorter) that desired, the target cost will be low if the previous units are also longer (resp. shorter). This way, we want to keep the consistency between the different units which might be better than inconsistency and perhaps produce a credible speaking rate slow-down or speed-up.

3 Experiments

We have conducted experiments aiming at (i) testing the accuracy of our ANN, (ii) measuring the impact of the new target cost on the unit selection algorithm and (iii) subjectively assessing the improvement in produced speech.

3.1 Experimental Data

Our two voice corpora, *IVS learning* and *Audiobook learning*, were used both as learning sets for ANNs and TTS voices. Test and validation corpora for these voices were also used for ANN learning rate evaluation. For the purpose of the tests, we use a sub-set of 100 sentences from *Various Styles*, insuring that a wide variety of styles, from different audiobooks, were represented.

3.2 Objective Analysis

Neural Network

The mean RMS error for *IVS* voice is slightly better (RMS=24.24, std=9.07) than for *Audiobook* (RMS=26.58, std=6.61). Pearson scores show that predictions are strongly correlated to real values, and the probability of error on the Pearson score is extremely weak. A detailed analysis on a per phoneme basis shows that the worst phonemes are those having very few representations in the learning corpus, for each voice. For instance,

/n/ has only 2 realizations in the *Audiobook learning* corpus, and only one in *Audiobook validation* (*cf.* appendix B). Finally, when looking at real and predicted centroids for each phoneme, most of them are very close, if not identical. Given these results, which we consider as fair, and knowing we do not need extremely accurate predictions as they are solely used to influence selection, these models have been kept as is.

Behavior of the Cost Function

To evaluate the impact of duration cost and its interactions with concatenation costs, we considered all $\{W_{tc}, W_{cc}\}$ couples in the $[0, 100]$ interval with a pace of 10. For each weight configuration, we generated the 100 sentences in our *Various Styles* corpus. Sentence (not utterance) based measures were extracted for each configuration. In this section, we will only discuss these measures on *IVS* voice, but exactly the same patterns are observed on *Audiobook* voice. Only small variations in magnitudes are observed between the two voices. It is important to point out that costs presented here are obtained *without* applying W_{tc} and W_{cc} weights. Magnitudes due to these weights have been removed to get raw costs.

Figure 7.2 shows the evolution of the mean delta per phoneme in *ms* between predictions by the network and final produced durations in relation to target and concatenation costs magnitudes for *IVS* voice. The same for *Audiobook* voice is shown on the bottom part of the figure. As it can be seen, the general trend is that distance increases when the target cost increases, which shows a good functioning of our target cost. Moreover, when getting the worst target cost, the delta largely increases. An unexpected result is the relation between the delta and concatenation cost when target cost is high which seems to suggest that concatenation cost excludes units with worst duration, improving the delta. When concatenation cost increases again, the delta dramatically increases again too. We can further note that duration delta at high target costs and low concatenation costs, while being good, remains much higher than the delta we get at lower target costs (this time independently of concatenation cost).

This result led us to think it would be worth investigating the behavior of a system where the duration target cost would be activated only on certain conditions, like for high concatenation cost or when confronted to a drastic relaxation of preselection filters.

3.3 Subjective Evaluation

Based on precedent measures, we selected configuration $\{W_{tc} = 30, W_{cc} = 70\}$ for listening tests. This choice was motivated by the low variability in terms of duration costs when getting over $W_{tc} = 30$ and the fact that concatenation cost alteration at this level is low. The same reasoning led us to $W_{cc} = 70$. In consequence, listening tests were performed using two system configurations: *baseline* system, which logically corresponds to configuration $\{W_{tc} = 0, W_{cc} = 100\}$; and the configuration incorporating our duration distance, $\{W_{tc} = 30, W_{cc} = 70\}$, called *Controlled*.

We performed two AB tests involving 13 testers for the first and 11 for the second (a

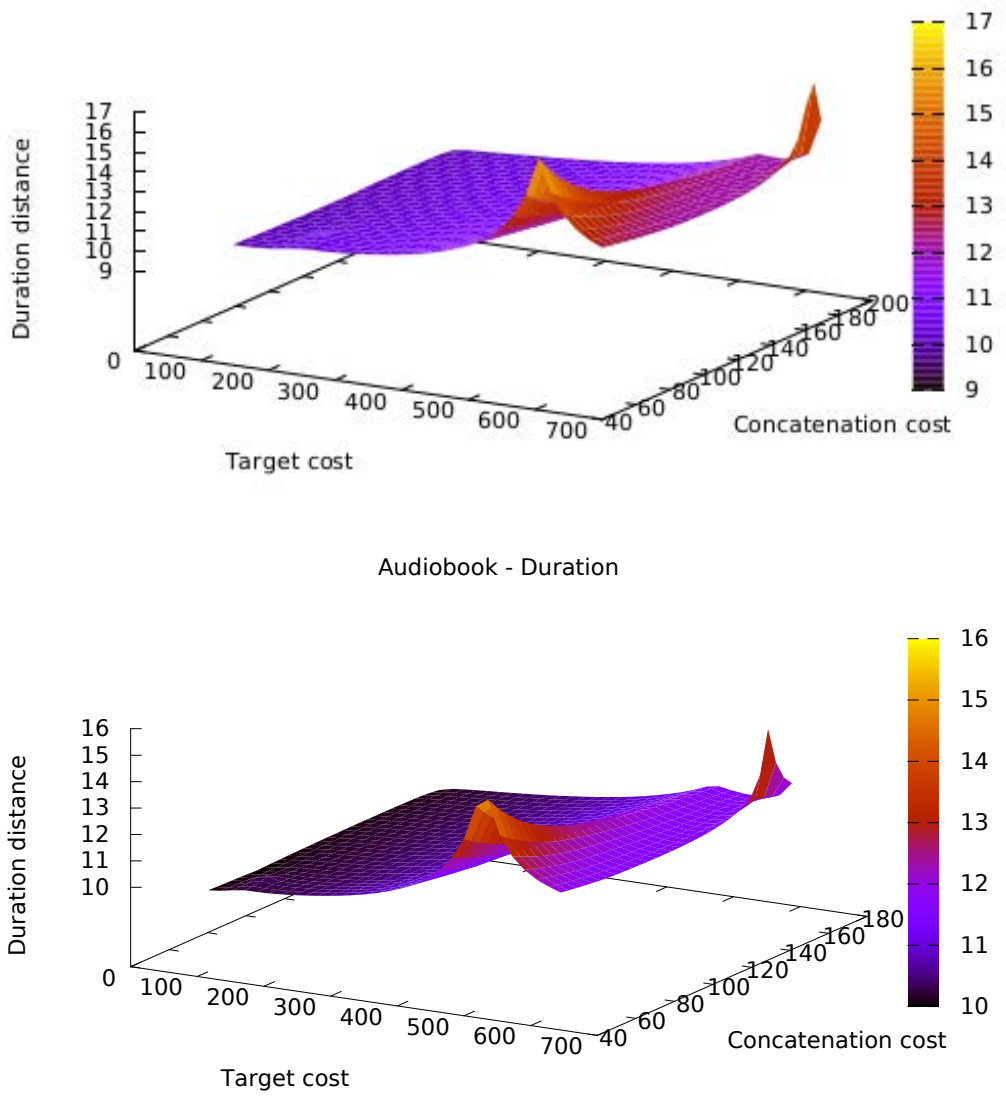

Figure 7.2: Duration delta between model predictions and synthesized durations evolution when target and concatenation costs vary for *IVS* voice (top) and *Audiobook* (bottom). Distance, per phoneme, is given in ms. Data computed using synthesis from *Various Styles* corpus.

few more listeners than our pool of ten people was available) on the *baseline* and *Controlled* systems. Both *Audiobook* and *IVS* voices were mixed in each test. The first test was based on a random selection of speech stimuli to present to the testers, while the second one was based on a random selection within the sub-set of synthesized sentences that featured audible duration artefacts with *baseline*. In order not to bias this second test, the stimuli produced by *Controlled* are not involved (and especially not listened to) in the process of selection for the listening test. The second test is of great importance here. Indeed, though duration issues in the synthesis remains a major problem in synthesized speech, the number of speech stimuli presenting significant duration incoherencies is somewhat small

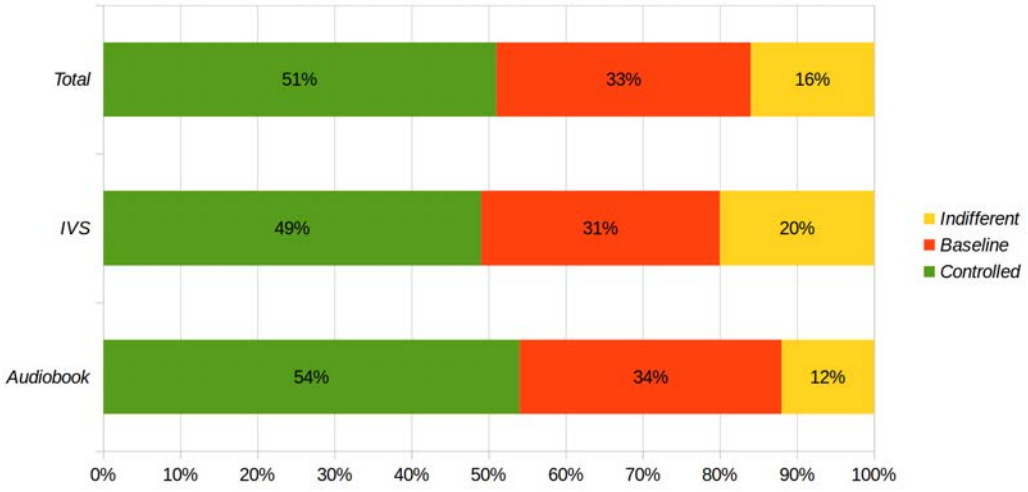

Figure 7.3: AB test results. *Uncontrolled* featuring duration artefacts is opposed to *Controlled* system. First and second row are a decomposition of the third one. *Controlled* is clearly preferred.

with large voices like *IVS* or *Audiobook*, even without duration control; therefore, there is a good chance that stimuli selected for the first listening test mostly present minor or even inaudible duration artefacts. In such cases, listeners might be influenced in their judgment by other factors, even though the question asked clearly states that they should focus on duration (which is a difficult exercise). The second test aims at preventing this problem.

The first test presented 20 stimuli for each voice, **taken randomly in the TTS test set**. The testers were asked to assess the rhythm of speech and select the best system. On raw results, systems were getting almost as much votes (43% for *baseline* and 38% for *Controlled* with overlapping confidence intervals). We spotted extremely different scales of notation among testers, with none seeming to have the same way of performing the test. Thus, no hard conclusion can be derived from this test. Nonetheless, it suggests the two systems are on par. It is important to underline that post-analysis of the stimuli presented for this test showed that very few samples had strong duration incoherencies.

An important point is that *IVS* corpus featuring only neutral voice, duration artefacts are less serious and less frequent. On the contrary, *Audiobook*, being very expressive, features much more minor duration issues. Major duration problems are also much more frequent.

The second test **focused on sentences having audible duration artefacts**. 22 different sentences featuring duration artefacts (of various amplitudes but all being audible) were extracted from *baseline* synthesis (11 for each voice). They were confronted to their equivalent with *Controlled* system. The testers were asked to say which system has the most natural voice. The testers were also asked to pay particular attention to rhythm (but not exclusively).

Results for this second test are presented on figure 7.3. First row shows results for *Audiobook* voice only, second for *IVS* only while the third one is the global result. In this

test, *Controlled* is strongly preferred by testers, especially for *Audiobook* voice which is normal as it is the voice the most likely to generate artefacts. It was also interesting to see that testers all followed the same trend, placing *Controlled* ahead with different levels of preference. Experts especially had a strong preference for *Controlled* when using expressive voice *Audiobook*, and less for *IVS*.

Given these results, it can be derived that our target costs behave well in enhancing durations when needed and only when needed, while not deteriorating synthesis on other aspects.

4 Conclusion

We presented a new duration target cost for unit selection. This cost aims at selecting the whole unit sequence that best minimizes duration distance with predicted values rather than choosing the sequence containing units that individually minimize a duration distance. This is intended to avoid cases like excellent synthesis penalized by few very bad units. Experiments showed that this new measure performs well on speech samples that feature durations issues, especially on our expressive voice. Furthermore, the new measure does not seem to affect synthesized samples that have good durations from the beginning. While the new cost is here used only for phone durations, it is extendable to all target costs. It could even be extended to the concatenation cost the following way. Some concatenation sub-costs are not adapted (MFCC for instance), but others are (e.g. the F_0 sub-cost). Each adapted concatenation sub-cost, operating on some parameter (like F_0), could be mixed with a second sub-cost trying to minimize the distance, on the same parameter, between the unit and the mean in the candidate sequence. This extension to all applicable distances in both target and concatenation costs should be tested.

Chapter 8

Work on the Pitch Target Cost

"The cello is a hero because of its register - its tenor voice. It is a masculine instrument, whereas the violin is feminine because of its soprano pitch."

Mstislav Rostropovich (1927–2007)

The *study introduced in this chapter presents two unit selection target costs aiming at controlling candidate units F_0 contours. It uses an atom-based decomposition method to decompose F_0 into a breath group-wide gamma shape called a phrase atom and smaller syllable/segment-level gamma shapes called local atoms. Based on the belief that unit level pitch contour is governed by physiologically pertinent local atoms, which cause small but noticeable variations to F_0, these target costs use atom parameters for one and reconstructed variations (induced by atoms) of F_0 for the other to impact on unit selection. In that work, the TTS system is an oracle. It uses annotations from real speech stimuli for the target sequence, instead of predictions (preventing evaluation error that would result from predictions). Particular attention was paid to evaluating the impact of the method on the prosody of synthetic sentences during experiments. First results proved both costs were more efficient than a traditional F_0 distance in listening tests.*

The chapter is organized as follows. In section 2, a presentation of atom-based intonation modeling is provided. Section 3 describes how atom-based unit selection target costs were designed and implemented in the TTS system. Finally, sections 4 and 5 include an evaluation of these new costs and give some insight on the perspectives.

1 Motivation

As TTS systems get better and better in quality when synthesizing neutral speech, focus is more and more centered on expressiveness; a number of companies now sell what they call expressive voices, generally featuring the 6 main emotional states [Ekman and Friesen 1969]. However, they fail at providing real naturalness, not to mention expressivity (which is much more complex than 6 emotional states [Cowie and Cornelius 2003]). One can work on applying expressiveness to the voice only if prosody for the neutral voice is adequately managed, which is not the case of all TTS systems. For unit selection especially, prosody is difficult to model and control. In particular, strict recording conditions, aiming at getting the best concatenation experience, strip corpora of any expressiveness. This is why it is interesting to search for the ability to control prosody for voices that feature many different styles, like in audiobooks. In this chapter, we use a set of selection costs to constrain target speech prosody by using atom decomposition-based intonation modeling [Honnet et al. 2015]. Atom-based intonation modeling is a generalization of the command-response model. In that representation, intonation is decomposed into an utterance-size gamma function (phrase atom) and a set of phoneme/syllable-size local gamma shapes (local atoms). These local atoms, having a time span that's similar to most corpus units that are selected, seem well-suited to feed a target cost on intonation. In particular, we assume local atoms are correlated to events at the origin of intonation in the vocal apparatus, which would reinforce their interest in a target cost.

Atom-based F_0 decomposition is explained in the next section. For a complete description of the method, Honnet *et. al.* can be cited [Honnet et al. 2015]. The work presented in this chapter was originally performed during a 5 month stay at IDIAP, Martigny, Switzerland in 2015.

2 Atom-Based F_0 Decomposition

Among the many models aiming at representing intonation, Fujisaki's command-response (CR) model has gained a very high credit [Fujisaki and Nagashima 1969; Hirose and Fujisaki 1982]. Command-Response model assumes *log* F_0 is the sum of a base sound level, a breath-group level phrase component and local accent components. Whilst the phrase component can be explained directly in relation with subglottal pressure evolution (which evolves slowly within a range of about 15% during phrase phonation, and drastically drops at the end of the breath group), local commands are related to voice muscle activity (cricothyroid, vocalis and sternohyoid muscles). The CR model decomposes the signal in impulses, corresponding to the phrase component, and step functions, corresponding to the accent components. The atom-based intonation decomposition model (Generalized Command-Response model, or GCR) is a generalization of the step into a sequence of

impulses that can be expressed as:

$$G_\theta(t) = \begin{cases} \frac{1}{\theta^2} t e^{-1/\theta} & \text{for t} \geq 0, \\ 0 & \text{for t} < 0. \end{cases} \tag{8.1}$$

This definition is an order $k = 2$ gamma function which higher order form is:

$$G_{k,\theta}(t) = \frac{1}{\theta^k \Gamma(k)} t^{k-1} e^{-1/\theta} \quad for\ t \geq 0 \tag{8.2}$$

Basically, the atom prosody modeling decomposes F_0 into a set of gamma shaped kernel functions where every atom is expressed according to the preceding function. The breath-group sized atom, representing the phrase component, is also defined through a gamma form. Atom annotations are computed from the $logF_0$ contour using a matching pursuit algorithm. Basically, the algorithm tries to approximate the contour by finding the best linear combination of a predefined set of kernel functions of size M:

$$f_0(t) = \sum_{m=1}^{M} \sum_{i=1}^{I_m} \alpha_{m,i} \Phi_m(t - \tau_{m,i}) + \epsilon(t) \quad t \geq 0 \tag{8.3}$$

In the above formula, the gamma function amplitude $\alpha_{m,i}$ may be positive or negative. Φ_m is the m$^{\text{th}}$ kernel function. I_m instances of the m$^{\text{th}}$ kernel function are evaluated. $\tau_{m,i}$ is the temporal position of atom impulse. In order to make the building of the cost function easier, the temporal position τ of the associated gamma function maximum and the temporal width w of the function that is considered in the computation are added into the atom definition. Finally, the atom set is the following:

$$\begin{aligned} \Psi = \{(k, \alpha, \theta, \tau, max, w)/k \in \mathbb{N}, \alpha \in \mathbb{R}, \\ \theta, \tau, max, w \in \mathbb{R}^+\} \end{aligned} \tag{8.4}$$

where an atom is thus defined with k and θ that are the parameters of $G_{k,\theta}$, amplitude (α), initial temporal position (τ), temporal position of the function maximum (max) and temporal width (w). In this work, following a recent investigation [Honnet et al. 2015], only order $k = 6$ atoms are used in the dictionary as they give the best results.

3 Atoms for Driving a Unit Selection Target Cost

Most contributions in the literature making use of Fujisaki's model are related to SPSS (mostly HMM-based) frameworks, for example [Hirose et al. 2005]. In this work, we take a very different approach: we focus on how to use atom annotations for building a unit selection target cost.

3.1 Defining New Prosody Target Costs

The motivation for using atoms in this particular task relies on two fundamental hypotheses. First, we assume that the decomposition into phrase and local atoms is not only virtual but is related to the muscle activity in the vocal apparatus (hyp. 1); *i.e.* atoms are the responses to muscle impulses. Second, we make the hypothesis that perceptual inconsistencies in synthesized prosody are due to the fact that the underlying speech production mechanisms are not taken into account (hyp. 2). For instance, a pitch distance functioning as a target cost only focuses on elements resulting from an analysis of produced speech (or predicted), and not on analysis of the mechanisms that produced that speech. The problem is that the pitch contour itself is altered by microprosody acting as noise. In those conditions, a basic F_0 distance is definitely biased. As atoms (especially higher amplitude ones) seem to be correlated to the muscular work in question, a target cost exploiting atom decomposition might prove efficient in selecting units with the right prosody. In addition, an F_0 regenerated using atoms only may be much more suitable as it removes microprosody. Taking these hypotheses into account, two main ideas have been considered for a target cost.

Target Cost with Atom Properties

The first idea, called *Atom-Param*, is to directly use atom parameters to compute a cost between a candidate and a target unit. First, we define the cost between two atoms ψ and φ as:

$$C(\psi, \varphi) = W_{max}|\psi_{max} - \varphi_{max}| + W_w|\psi_w - \varphi_w| + W_\theta|\psi_\theta - \varphi_\theta| \tag{8.5}$$

where the notation ψ_X, φ_X is used to denote the parameter X of atoms ψ and φ as defined in (8.4). Magnitudes of the sub-costs are homogenized with manually set weights.

Let S_T and S_C be the atom sets in the target and candidate unit respectively. Let S_{min} be the smallest of them, and S_{max} the biggest, in terms of cardinal. We construct the set L which pairs each element of S_{min} with the closest atom in S_{max} according to the following:

$$(\psi, \varphi) = \underset{\psi \in S_T, \varphi \in S_C}{\arg\min} \; (W_{max}|\psi_{max} - \varphi_{max}| + W_\theta|\psi_\alpha - \varphi_\alpha|) \tag{8.6}$$

We also define a multiplicative penalty K as:

$$K(\psi, \varphi) = \begin{cases} K_{max} & \text{if } \frac{\psi_\alpha}{\varphi_\alpha} \leq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{8.7}$$

where ψ and φ are two atoms and K_{max} is the highest possible cost $C(\psi, \varphi)$ between two atoms measured on the corpus.

The final target cost for a unit is the sum of this atom cost for each pair in L. We add

a full penalty for each atom that is part of S_{max} but not of any pair in L, *i.e.* atoms of S_{max} that cannot be aligned with atoms in S_{min}. We note $|L|$ the cardinal of set L.

$$TCost = \sum_{(\psi,\varphi)\in L} K(\psi,\varphi) * C(\psi,\varphi) + (max(|S_T|,|S_C|) - |L|) * K_{max} \tag{8.8}$$

Target Cost Exploiting Atom-reconstructed F_0

Having obtained atom annotations, the next step is to regenerate F_0 using atoms only. Then, the idea is to see whether the regenerated F_0 might prove better than original F_0. The second cost, called *Atom-Pred*, aims at checking whether this is true or not. It reconstructs F_0, for each unit, from parameters of the atoms linked to that unit using function 8.2. The sampled reconstructed signal S is then used for a Euclidean cost:

$$TCost = \sum_{s=1}^{card(S)} |f_0^t(s) - f_0^c(s)| \tag{8.9}$$

where f_0^t and $f_0^c(s)$ are F_0 sequences for target and candidate units resp.

4 Experiments

4.1 Experimental Process

While the objective is evidently to start from text and predict atoms, our work is divided into 2 distinct steps. The aim, in a first step, is to determine whether atom decomposition can help enhance F_0 contours in synthesized speech by first looking at what happens when using only atom annotations (see 3.1) and then to investigate if F_0 resynthesized with atoms performs better than original in a target cost. For that first step, we synthesize texts for which real speech exists, hence real atom annotations and real original F_0 measures are used. This spoken version might come either from the same speaker as the speech corpus or another one. So in this first case, the target cost data comes from an oracle. Then, in a second step, models may be used to predict atoms or atom-based reconstructed F_0, at least if target costs prove to be efficient enough. In this work, we focus on the first part and we use 4 system variants:

- The ***baseline* system**: filters and concatenation cost only.
- **F_0-*sys* system**: *baseline* + target cost using F_0 (from real speech stimuli) and making a Euclidean distance with corpus units. F_0 annotations were obtained with the Kaldi pitch extractor [Ghahremani et al. 2014].
- ***Atom-Pred-sys* system**: *baseline* + *Atom-Pred* target cost.
- ***Atom-Param-sys* system**: *baseline* + *Atom-Param* target cost.

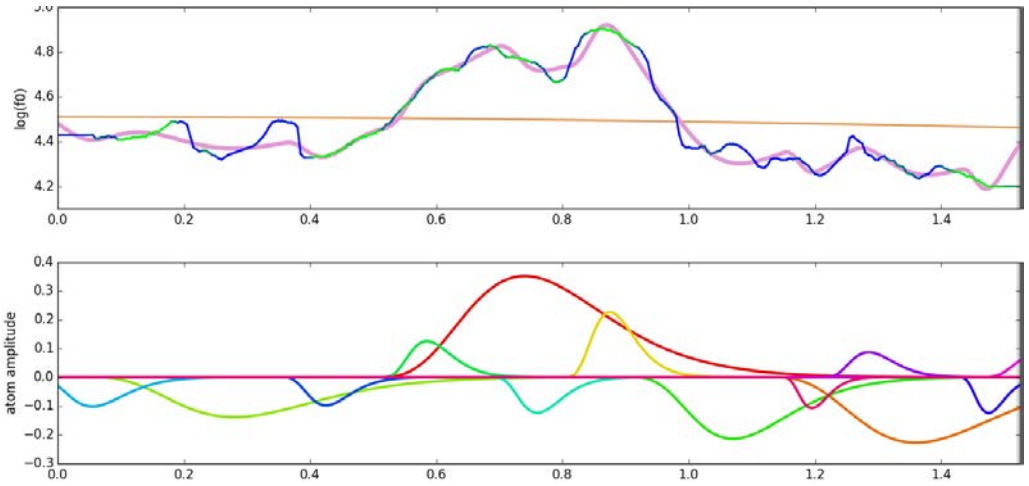

Figure 8.1: Example of atom-based F_0 decomposition. The first frame shows *log* F_0 extracted with Kaldi, reconstructed F_0 and the phrase atom. In this graph, the phrase atom is shown in brown, Kaldi F_0 in blue and green and reconstructed F_0 in pink. Second frame shows a view of local atoms. Each atom is given a single color in that frame.

4.2 Experimental Data

For the experimental evaluation of this contribution, technical constraints concerning the sampling frequency of *IVS* voice (too low for one of our analysis tools) made it impossible to use. We will therefore use *Audiobook* voice only. For the tests, we will use two corpora. The first one is *Audiobook test*. In order to reduce the bias caused by having the same speaker for *Audiobook* voice and the test corpus, we will also use another test corpus for which the speaker is different. This corpus consists of the same 100 sentences from *Various Styles* as in chapter 7. In earlier chapter, only the text versions of the sentences in *Various Styles* were used. In this chapter, we also use the related speech recording to get F_0 annotations and atoms. Each unit in both test corpora and *Audiobook* voice is annotated with all atoms belonging to the unit, even if the impulse is located in an earlier unit. In order to do this, we use the recordings made on *Various Styles* (made by a different speaker than the one that recorded *Audiobook*). All atoms above an amplitude of 0.03 are used for the cost function. It corresponds to the smallest amplitude threshold that is given to the atom extraction tool while annotating the corpus.

4.3 Atom Decomposition

Atom decomposition is illustrated in figure 8.1. The first frame shows *log* F_0 extracted with Kaldi tools (blue and green), reconstructed F_0 (pink) and the phrase atom (line in the middle, in brown). The second frame is a visualization of local atom positioning in the sentence. We can clearly see in that frame that atoms that have a stronger amplitude (positive or negative) are to be given a greater importance, which is the case in cost *Atom-Param-sys*. The reconstructed F_0, here using only atoms that have an amplitude bigger than 0.03, is actually very good. The main effect when comparing reconstructed F_0 to

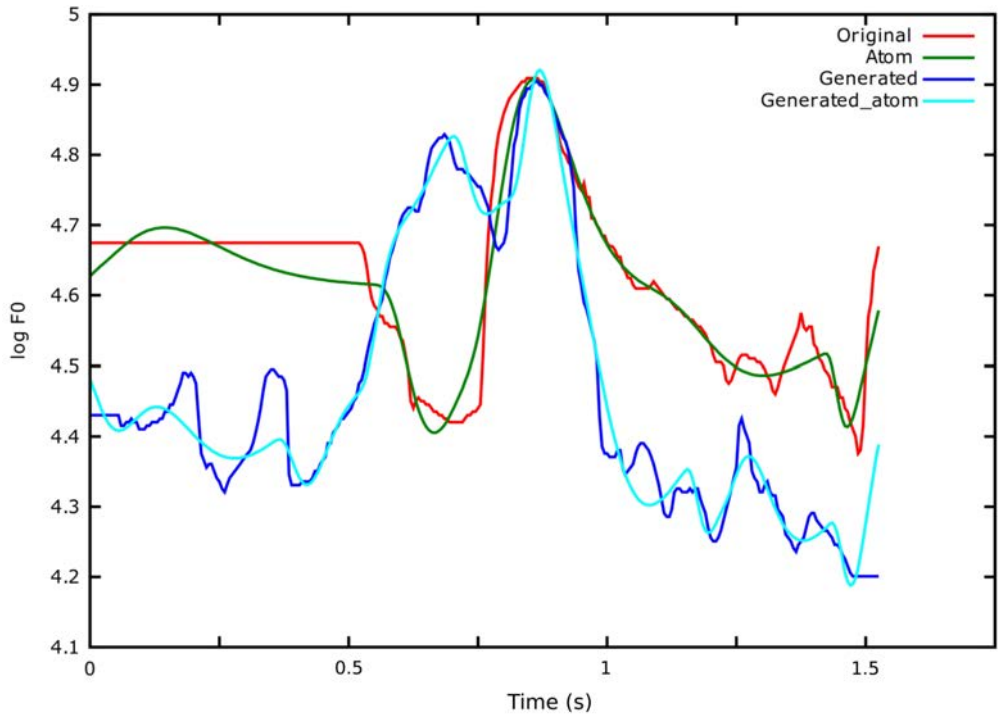

Figure 8.2: In dark, the curve shows pitch contour and atom reconstruction of that contour for the original sentence. Light gray ones show the same data for the synthesized version of the sentence (with *Atom-Param-sys*).

original data as extracted by Kaldi is a smoothing of the curve, stripping it from smaller variations. This is fortunate in our case as those smaller variations might indeed act as noise in the cost function.

Figure 8.2 shows the same test stimulus synthesized with *Atom-Param-sys*. There, dark curves represent *log* F_0 for the original stimulus (from a different speaker to the one who recorded the corpus). Light curves represent *log* F_0 for the generated sentence. The two smooth curves show the atom-reconstructed contour while the other two give the real one. Though the gap before the spike in the middle of the sentence does not exist in the generated version, the rest of the curve seems to match correctly (for unit selection). It has to be noted that even if synthesized samples appear to show better correlation for *Atom-Pred-sys* and *Atom-Param-sys*, the overall distance between generated and original F_0 is similar for all 4 systems. Initial work shows there is no strong linear correlation between generated and original contours but we assume a more complex form exists. No particularly annoying phenomenon (beyond the forecastable mean F_0 magnitude change) to a difference of speaker was observed between sentences using atom annotations.

4.4 Subjective Evaluation

As an evaluation of the target costs, we performed a MUSHRA test involving 7 native French speakers. One of the main advantages of MUSHRA tests is that they do not require as much

System	Mean mark	Centered mean
baseline	56.4 ±4.3	47.2 ±3.9
F_0-*sys*	57.2 ±3.6	48.6 ±3.2
Atom-Param-sys	**60.4** ±4.3	**52.9** ±3.9
Atom-Pred-sys	60 ±3.6	51.3 ±3.3

Figure 8.3: Results of the MUSHRA test with 7 listeners. Column mean mark shows raw results. Centered mean is the same but centered to 50. Confidence intervals are at 95%.

testers as other tests , so a capital of 7 testers is here better used than with a MOS or an AB. The question proposed to testers was not targeted at general quality. Instead, the question was the following: "How do you assess the quality of the PROSODY (intonation, rhythm), and only prosody, of this sample?". As assessing only prosody is difficult, only experts used to listen to unit selection-based speech synthesis performed the test. Each of the 12 steps of the test consists of a set of sentences (for each system tested), generated from the same text, that the listener has to mark from 0 to 100 in relation to the question above. 6 of the sentences came from *Various Styles* and 6 from *Audiobook test*. Again, no real difference was observed in test results between sentences from *Various Styles* and *Audiobook test*. Test conditions conformed to ITU-T recommendations [ITU-R 2015].

Table 8.3 shows the results for the test. Basically, the first column shows raw results for the test. The second column shows the same results when centered to 50. What we can observe is a distinct preference for atom-based costs, with *Atom-Param-sys* being more than 5 points better than *baseline* on the centered data. While there is a degradation with *Atom-Pred-sys* (regenerated F_0), it still outperforms both *baseline* and F_0-*sys*. This last one is performing particularly poorly, as it doesn't show any real improvement over *baseline*. This is striking as the only real difference between F_0-*sys* and *Atom-Pred-sys* is the use of resynthesized F_0 in *Atom-Pred-sys*. More careful analysis, isolating trends for each tester shows a quasi-unanimous ranking: *Atom-Param-sys* $>$ *Atom-Pred-sys* $>$ F_0-*sys* $\geq$ *baseline*.

As for now, more testers will be needed to refine results. Proof of a strong (non-linear) correlation has also to be discovered. Nonetheless, atom-based target cost has proven to be efficient in listening tests. Moreover, resynthesized F_0 proves to be better fitted for unit selection than original F_0.

5 Conclusion

In this chapter, we described a new F_0 target cost, which we elaborated in order to constrain the F_0 contour, based on atom decomposition of pitch. We assume the unit selection process naturally chooses units that feature a homogeneous F_0 (F_0 is part of the concatenation cost), most issues come from segments that have a very different local contour than what is expected. As local atoms model that segment-level/syllable-level information, and as there are elements suggesting local atoms are related to physiological work, we made the hypothesis

that using this data enhances synthesized prosody. A first target cost using parameters of atoms was created and performed much better than a target cost based on real F_0 in listening tests. Regenerating the F_0 from atoms and using this new contour in the target cost also performed much better than original F_0. We can therefore validate our hypothesis. This allows us to think that a target cost based on predictions of atom-reconstructed F_0 (or directly using predicted atoms, though it might be harder) will perform better than direct prediction of F_0, as it gets rid of microprosodic and estimation errors.

Chapter 9

Work on the Concatenation Cost

"Enjoy every sandwich."

Warren Zevon (1947–2003)

The *role of the concatenation cost is to insure that joining two voice segments will not cause any acoustic artefact to appear. For this task, acoustic distances (MFCC, F_0) are typically used but in many cases, this is not enough to prevent concatenation artefacts. Among other strategies, the improvement of corpus covering by favoring units that naturally support well the joining process (vocalic sandwiches) seems to be effective on TTS.*
In this chapter, we investigate if vocalic sandwiches can be used directly in the unit selection engine when the corpus was not created using that principle. First, the sandwich approach is directly transposed in the unit selection engine with a penalty that greatly favors concatenation on sandwich boundaries. Second, a derived fuzzy version is proposed to relax the penalty based on the concatenation cost, with respect to the cost distribution. We show that the sandwich approach, very efficient at the corpus creation step, seems to be inefficient when directly transposed in the unit selection engine. However, we observe that the fuzzy approach enhances synthesis quality, especially on sentences with high concatenation costs.

1 Motivation

Discriminating the segments coming from the corpus that fit the requirements expressed via the target sequence is usually done by ranking the units with an evaluation of the context matching degree (target cost) and the risk of creating an artefact if concatenating the unit (concatenation cost) via balanced cost functions. The concatenation cost typically

relies mainly on acoustic features (MFCC, F_0) [Stylianou and Syrdal 2001; Tihelka et al. 2014] to evaluate the level of spectral resemblance between two voice stimuli on and around the concatenation point. As for now, concatenation costs are far from being perfect and audible artefacts appear both in commercial and research TTS systems, even after post-concatenation processing. A few analyses, for example [Yi 1998], showed that these artefacts occur more often on some phoneme than others. For instance, phonemes with high context-dependency (e.g. liquids) might show substantial inter-occurrence spectral variability [Lindblom 1963], which is particularly dangerous for unit selection, especially because joining is usually done on phone centers (*i.e.*, diphone boundaries). This being considered, some authors tried to use phonologically motivated rules to prevent joining on “risky” phonemes. For instance, in [Yi 1998] the authors successfully tested a penalty system based on the phonological class of candidates to concatenation. A refined version of this idea was used by D. Cadic in the context of recording-script construction in [Cadic et al. 2009] to favor covering of what has been called "vocalic sandwiches", also with success.

Based on these considerations, we decided to assess the impact of vocalic sandwiches back in the concatenation cost of a modern unit selection system, when a corpus was not created using the “sandwich” process described in [Cadic et al. 2010]. We use the 3 phonologically-based phoneme clusters defined by [Cadic et al. 2009] to forbid concatenations on phones believed to often cause joining artefacts. Believing this direct transposition marginalizes acoustic concatenation costs, we develop an enhanced version that softens penalties. This is done through the use of a fuzzy function that relaxes the penalty based on the acoustic concatenation cost distribution. It allows to smoothen the constraints imposed by sandwich penalties.

The main impact of this study is to improve TTS in the case of less controlled data, such as audiobooks, by transposing a constraint originally proposed for the corpus creation step directly into the TTS engine [Guennec and Lolive 2016]. The challenge is to know if the efficiency obtained at corpus building level can be found also at unit selection level. To that respect, unit selection makes it much simpler than SPSS to add the sandwich feature and test its efficiency. Experiments show the efficiency of the proposed approach and its suitability for corpus-based approaches at a low cost.

The remainder of the chapter is organized as follows. In section 2, we will present the concept of vocalic sandwich as it is implemented in our work. Section 3 first presents the integration of sandwiches into the system as a simple penalty constraint system. Then a fuzzy enhancement of the sandwich system, much more adaptive, is introduced. In section 4, we first describe our test data and then our experimental protocol. The experiments and their results are then presented and discussed.

2 Enhancing Speech Corpora With Vocalic Sandwiches

Analysis of sentences containing artefacts shows that concatenation on some phonemes, especially vowels and semi-vowels, is more likely to engender artefacts than others (plo-

sives and fricatives for example, especially unvoiced ones) [Yi 1998]. Phonemes featuring voicing, high acoustic energy or important context dependency are generally subject to more distortions. Based on this claim, [Cadic et al. 2009; Cadic and D'Alessandro 2010] proposed a corpus covering criterion where the objective is to get a maximum covering of "sandwich units". A sandwich unit is a sequence of phonemes where one or several syllabic nuclei are surrounded by two phonemes considered as not likely to cause artefacts (we call it "resistant" to concatenation artefacts). A sandwich can therefore be formally defined as:

$$R(A^*VA^*)^+R \tag{9.1}$$

where + means 1 or more occurrences, * means 0 or more occurrences and R, A and V are the three following phonetic clusters, which Cadic *et al.* justifies in [Cadic et al. 2009]:

V (vowel) : Vowels, on which concatenation is hardly acceptable.

A (acceptable) : Semi-vowels, liquids, nasals, voiced fricatives and schwa. These units are viewed as acceptable concatenation points, but still precarious.

R (resistant) : the remaining phonemes (unvoiced consonants, voiced plosives), where concatenation is definitely possible. The word "Resistant" is used in the following to describe units of this class.

3 Sandwiches in a Unit Selection Engine

In this section, we describe how we integrate sandwich clusters into the unit selection concatenation cost, first with simple penalties and then with a much more refined fuzzy version.

3.1 Phonologically Motivated Penalty Based on Sandwich Classes

For the purpose of our study, we defined two penalization methods based on the three phonetic clusters defined in section 2.

We chose these clusters specifically because they are the same as those presented and justified in [Cadic et al. 2009], though the choice of elements put inside each cluster is arguable, for example the choice of considering all vowels dangerous areas for joining.

As said earlier, using the phonetic class to constrain or penalize phonemes considered as problematic for concatenation is not a novel idea, and a few works can be cited, for example [Donovan 2001; Yi 1998]. However, in these works, costs and penalties are very constraining, always trying to find the perfect unit (which may not exist in the corpus).

A key point of the idea we investigate here is that, because we do not want to add too many constraints in the cost function, we only defined 3 subsets of phonemes. The purpose of the penalty is not to act as a standalone cost, but simply to introduce knowledge that is not captured by the concatenation cost and then help achieve a finer ranking of units.

Moreover, the proposed classes are based simply on basic linguistic/phonological knowledge and it may be necessary to adapt them depending on the language.

The first method for applying the penalty, called *pho-class*, is to give a fixed penalty $p(v)$ to each phoneme class: 0 for phonemes in R, a penalty slightly higher than the highest value of C_c observed in the corpus for all phonemes in A. Vowels (V) are given a huge penalty, big enough to prevent compensation by other costs in the candidate sequence. It corresponds to a penalization of candidate units based on the phonemes on which concatenation may be performed if choosing this unit. In this case, a new concatenation cost function C'_c is formulated as:

$$C'_c(u,v) = C_c(u,v) + K(u,v) \tag{9.2}$$

where $K(u,v) = p(v)$ is the penalty depending on the phoneme that begins the unit v as described before, which is the same as the phoneme ending u as we perform joining on diphone boundaries.

3.2 Fuzzy Penalty System

The second method, called *fuzzy-pho-class*, is to relax the penalty in certain cases. Thus, we introduce a fuzzy weighting function giving to each penalty a weight ranging between 0 and 1 as shown on figure 9.1. It describes how satisfying the candidate unit is with respect to its concatenation quality. Assuming MFCC, Amplitude and F_0 cost distributions follow normal distributions, we define two thresholds for each sub-cost. For instance, the two thresholds $T^1_{F_0}$ and $T^2_{F_0}$ for the F_0 sub-cost may be defined as:

$$T^1_{F_0} = \mu_{C_{F_0}} - \sigma_{C_{F_0}} \tag{9.3}$$

$$T^2_{F_0} = \mu_{C_{F_0}} + \sigma_{C_{F_0}} \tag{9.4}$$

Formally, the fuzzy function is defined, for the F_0 sub-cost, as:

$$f_{F_0}(u,v) = \begin{cases} 0 & \text{if } C_{F_0}(u,v) < T^1_{F_0}, \\ 1 & \text{if } C_{F_0}(u,v) > T^2_{F_0}, \\ 1.0 - \frac{(T^2_{F_0} - C_{F_0}(u,v))}{(T^2_{F_0} - T^1_{F_0})} & \text{otherwise.} \end{cases} \tag{9.5}$$

The same is done for $f_{MFCC}(u,v)$ and $f_{amp}(u,v)$. This process sets thresholds at $\mu-\sigma$ and $\mu+\sigma$ for each distribution. If a unit's sub-cost values less than $\mu-\sigma$, it will get no penalty. if it is more than $\mu+\sigma$, a full penalty is applied. Finally, if it is in the between (corresponding to about 70% of the distribution), a linear function is used to apply a weight between 0 and 1 to the penalty.

The choice for that tolerance interval is motivated by the observation of real cost distributions. Indeed, we observed that the 30% of the distributions that are under $\mu-\sigma$ and over $\mu+\sigma$ are respectively much lower and much bigger than costs between

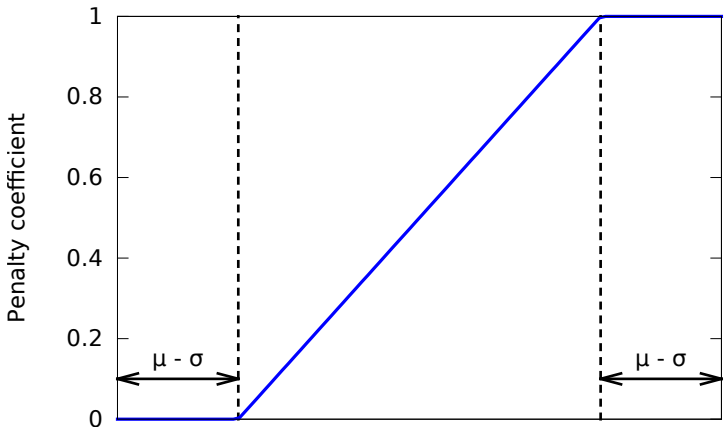

Figure 9.1: Fuzzy function over the distribution of sub-costs. The weight 0 (resp. 1) is given to units that have a concatenation costs approximately among the 15% lowest (resp. highest) costs. Between these thresholds, the weight increases linearly.

these thresholds. To be complete, the choice of the thresholds should be differentiated depending on the type of sub-cost and optimized separately.

Finally, the penalty is modified in the following way:

$$K(u,v) = (f_{mfcc}(u,v) + f_{amp}(u,v) + f_{F_0}(u,v)) * p(v) \tag{9.6}$$

where $f_{mfcc}(u,v)$, $f_{amp}(u,v)$ and $f_{F_0}(u,v)$ correspond to the fuzzy functions of the form described in figure 9.1 respectively for MFCC, amplitude and F_0. With those functions, the main idea is to decrease the penalty when the unit has a concatenation sub-cost value which is statistically among the best ones. These distributions are estimated using the voice corpus by computing concatenation sub-costs for F_0, amplitude and MFCC using all units present in the corpus.

To sum up, if concatenation cost is above the higher threshold then we definitely have to apply the full penalty as the unit considered is among worst possible units. Between the two thresholds, we augment progressively the penalty as the concatenation cost increases.

4 Experimental Evaluation

In this section, we first analyze the behavior of our three methods (*baseline*, *pho-class* and *fuzzy-pho-class*) in terms of concatenation costs, then our experimental protocol for the perceptual evaluation is presented and finally the results of our experiments.

4.1 Concatenation Costs Analysis

First we studied the evolution of costs using the three systems by comparing mean concatenation costs for resistant units only (class R), non-resistant units only (classes A, V)

Table 9.1: Concatenation costs without penalties following the three strategies on 100 sentences of *Various Styles*. Note that penalties have been subtracted *a posteriori*. R, A and V refers to classes introduced in section 3.1.

IVS	Resistant units (R)		Nonresistant Units (A, V)		Both	
	μ (std)	N	μ (std)	N	μ (std)	N
baseline	2.90 (0.69)	582	3.14 (0.70)	1249	3.06 (0.71)	1831
pho-class	3.28 (0.92)	1025	3.35 (0.88)	813	3.31 (0.90)	1838
fuzzy-pho-class	3.35 (0.92)	**1095**	**2.58** (0.42)	1169	**2.95** (0.80)	2264
Audiobook	Resistant units (R)		Nonresistant Units (A, V)		Both	
	μ (std)	N	μ (std)	N	μ (std)	N
baseline	2.44 (0.52)	606	2.90 (0.60)	1057	2.74 (0.61)	1663
pho-class	2.65 (0.71)	865	3.14 (0.78)	785	2.88 (0.78)	1650
fuzzy-pho-class	2.65 (0.64)	**907**	**2.47** (0.38)	1139	**2.55** (0.52)	2046

and both together, each time excluding contiguous diphonemes. All these results are presented in table 9.1 for the same 100 sentences from *Various Styles* as in chapter 7. As we can notice, the *baseline* system has lower costs for both resistant and nonresistant units compared to *pho-class* system. An explanation is that the *pho-class* system, by penalizing non-resistant units, favors resistant units even if their concatenation cost is higher. The number of concatenations made on resistant units (1025 for *IVS*) is then significantly higher compared to *baseline* system (582 for *IVS*). As for *fuzzy-pho-class*, the results in terms of number of concatenations are more balanced. Indeed, as good concatenations on nonresistant units have no or low penalization, the *fuzzy-pho-class* system achieves the lowest cost for nonresistant units. Introduction of variable penalties enables to evaluate units more finely than with *pho-class* system, for which all penalized units are equivalent. A counterpart of this is that the number of concatenations globally increases for the *fuzzy-pho-class* system, which is not a problem since they are better controlled. It is worth to mention that these results are equivalent on both voices. We can then consider them to be fairly independent from the voice type.

To sum up, the penalty seems to behave well as it enables to favor unit sequences with a lower cost on sensible units and more concatenations on resistant units.

4.2 Subjective Evaluation Process

For test purposes, we used our two voices *Audiobook* and *IVS*. The evaluation corpus is the full test corpus *Various Styles*. The 27 141 test sentences were synthesised for our 3 systems (*baseline*, *pho-class* and *fuzzy-pho-class*). In order to evaluate the two sandwich concatenation cost adaptations presented earlier, we carried out a total of 12 AB listening tests split in 3 groups of four tests:

Random sentences: 4 tests where the sentences are picked up randomly among those generated in our test set. This serves as a baseline evaluation which aims at studying if the sandwich systems are, in average (*i.e.*, in general), an enhancement over *baseline*.

Most different sentences: 4 tests where the most different synthesized stimuli pairs are chosen. Choice of the most different stimuli is made using DTW, as we presented in [Chevelu et al. 2015]. It aims at revealing differences that might have been obscured by the first set of tests by comparing the stimuli that are the most impacted by sandwich methods. If tested methods are worse than *baseline* in these tests, this methodology allows us to say sandwiches have mostly a negative impact on TTS, or the reverse if results are in favor of sandwiches.

Sentences with highest concatenation cost: 4 tests where the sentences are the ones that feature the biggest concatenation costs for the *baseline* system. They correspond to the sentences that most need improvement, and thus the primarily target we wish to enhance with the sandwich costs. If these sentences are not enhanced, this most likely means that sandwiches are inefficient as their purpose is to prevent disastrous concatenations more than enhancing joining quality.

Each test was made by 10 expert testers, each one evaluating 10 distinct stimuli pairs. 100 stimuli pairs are evaluated in total (all 100 synthesized from distinct sentences), each tester evaluating his own set of stimuli. In every test, the standard question concerning overall quality is asked. For the last set of tests however, (high concatenation costs), a second question is asked along with the first one, this time over concatenation quality. Because of the difficulty to answer this question, the choice of expert testers is here particularly justified. For each set of 4 tests, we carry out two tests using *IVS* voice and two with *Audiobook*. For each voice, one test compares system *baseline* with *pho-class*, the other with *fuzzy-pho-class*. Test conditions are studio-like and follow ITU-T recommendations.

4.3 Results

Table 9.2 presents the results of the tests for the comparison *baseline versus pho-class*. Each line corresponds to one AB test. Column 1 indicates the voice used for the test and column 2 the selection method for the test sentences ("R." for random, "DTW" for most different and "C. C." for highest concatenation cost). Column 3 refers to the question asked during the test: either "C. Q." for the question on concatenation quality or "G. Q." for the assessment of global quality. Using the same representation, results for the *fuzzy-pho-class* method are presented on table 9.3.

First, if we compare the behavior of the systems regarding the number of concatenations (table 9.1), we find that both *pho-class* and *fuzzy-pho-class* lead to a larger number of concatenations than *baseline*. This situation is completely normal as the acoustic concatenation cost aims at minimizing the number of concatenations as selecting a long unit means putting one or several diphone costs to 0. The sandwich cost aims at forbidding concatenations on some phonemes, some of them being long unit boundaries. This is not a problem: two well made (and therefore inaudible) concatenations are worth much more than one failed joining. When looking at the phoneme classes on which concatenations are

Table 9.2: Results for the AB listening tests for the *pho-class* system. Lines concerning tests on random sentences have the mention "R." in the second column. "DTW" is for tests with most different sentences and "C. C." for tests on sentences of *baseline* with the highest concatenation costs. Column 3 displays "G. Q." when the question was on global quality and "C. Q." when it was on concatenation quality only.

			Answers		
			Base	*pho-class*	Indifferent
IVS	R.	G. Q.	**45%**	34%	21%
	DTW	G. Q.	31%	34%	**35%**
	C. C.	C. Q.	33%	30%	**37%**
		G. Q.	30%	**35%**	**35%**
Audiobook	R.	G. Q.	38%	**39%**	23%
	DTW	G. Q.	**47%**	32%	21%
	C. C.	C. Q.	**38%**	31%	31%
		G. Q.	**39%**	30%	31%

Table 9.3: Results for the AB listening tests for the *fuzzy-pho-class* system. Please refer to table 9.2 caption for explanation of the table.

			Answers		
			Base	*fuzzy-pho-class*	Indifferent
IVS	R.	G. Q.	35%	**40%**	25%
	DTW	G. Q.	31%	**48%**	21%
	C. C.	C. Q.	20%	**59%**	21%
		G. Q.	27%	**49%**	24%
Audiobook	R.	G. Q.	**43%**	42%	15%
	DTW	G. Q.	42%	**46%**	12%
	C. C.	C. Q.	33%	**38%**	29%
		G. Q.	36%	**43%**	21%

made, we see that *pho-class* and *fuzzy-pho-class* produce much more (about twice more) concatenations on robust (cluster R) phonemes. This is the proof the two methods work as expected. In addition, *fuzzy-pho-class* also causes substantially more concatenation on A and V clusters, as intended. DTW score distributions are similar in terms of shape (Gaussian) and magnitude.

Second, from the listening tests results, we observe that the *pho-class* approach seems largely inefficient (at best) when integrated directly in the unit selection engine, as shown in every test made with the method. In some cases, it was even counter-productive: the AB test on random sentences for *IVS* clearly show it, and the same conclusion can be observed on 3 tests out of 4 concerning *Audiobook* voice. What is also noticeable is the high quantity of "indifferent" ratings, proof that the difference between the two systems isn't very clear. So we can say that, if sandwiches proved useful for the construction of a recording script (*cf.* Cadic et al. 2009), they prove inefficient, or even counter-productive when directly integrated into the concatenation cost.

On the contrary, for almost every test with *IVS* voice, a clear superiority of the *fuzzy-pho-class* approach can be observed. The result is also observable for *Audiobook* voice, though with a smaller gap. *Audiobook* voice faring better than *IVS*, the lesser difference for the first one seems logical. Concatenation quality is perceived better with *fuzzy-pho-class*. The number of "Indifferent" answers is also consistently lower for *fuzzy-pho-class*, meaning that differences are more easily felt. In conclusion, *fuzzy-pho-class* approach proves to be effective thanks to the degree of flexibility it adds in regard to *pho-class* method. In particular, we observe that *fuzzy-pho-class* ranking is between *pho-class* and *baseline*, which means it alters *baseline* ranking, based solely on acoustic measures that we know are imperfect, but not as much as *pho-class* (which completely changes the ranking and loses the information of acoustic measures). It is also interesting to see that the controlled corpus *IVS*, was more affected by sandwiches than *Audiobook*, which is completely uncontrolled. The question this raises is the following: is it the quality of *Audiobook* voice or its uncontrolled nature that causes the observed lower performance of sandwiches for that voice?

We believe that the key to the success of all these measures (including *fuzzy-pho-class*), is a close integration in the concatenation cost. The penalty cannot hide the ranking provided by acoustic costs, and this for a good reason: these penalties aim at correcting acoustic rankings on key points, using expert knowledge (and it is exactly what sandwiches at corpus building level does). But a too constraining penalty system (*i.e.*, *pho-class*), which is not a good concatenation cost on its own, causes a complete re-ranking of the system, hence the drop in quality.

Though this contribution was made with a particular focus on French language but our conclusions should apply to other language, especially syllable-timed languages. Indeed, besides French, *fuzzy-pho-class* method was also applied on Indian languages for the Blizzard Challenge 2015 [Alain et al. 2015]. Though no dedicated listening test was carried out to compare *fuzzy-pho-class* to *baseline* on these languages, the former seemed to yield better results.

5 Conclusion

In this chapter, we have presented a study of the impact of vocalic sandwiches back in the concatenation cost of a modern unit selection system, through two penalty-based systems. This penalty enables to avoid some artefacts during synthesis and its fuzzy version preserves the ranking made by acoustic components of the concatenation cost. The subjective experiments we conducted show a better performance for fuzzy version both for a neutral and an expressive voice. It shows that the concatenation cost does not capture all the perceptual information and that adding some preferences over the type of units to concatenate improves the synthesized speech quality. On the contrary, pho-class method, which fares well at script construction level, seems largely inefficient when integrated directly in the unit selection engine. Along with other elements, this leads us to think that penalty sys-

tems are not a form to privilege in the cost function. The impact of the fuzzy method is particularly clear with IVS voice, which we can explain by the fact that its quality is lower than Audiobook (so that there is more room for improvement).

General Conclusion

Work during this thesis has been centered around unit selection algorithms and cost functions for corpus-based speech synthesis systems. As part of this effort, our first task was to participate in the creation and sustained development of a completely new TTS engine, the IRISA Text-To-Speech system. This task has taken considerable development and maintenance time. Nonetheless, it allowed us to conceive the system, from the very beginning, so that several search algorithms and modular cost function could be easily implemented.

In chapter 5, we presented the tools that were used to feed the TTS system with speech data, which we also worked on intensively: the ROOTS toolkit and then the lighter TTS-corpus format were used to allow synthesis with an expressive male voice, *Audiobook*, and a neutral voice, *IVS*. We saw that *IVS* and *Audiobook* voices were close to average French corpora, with slight variations for *Audiobook* due to its literary origins. Working on two French speaking voices, our work is consequently to be put in the context of that language, even though we occasionally synthesized English and several Indian languages for the Blizzard challenge.

Our research work was done in two successive parts: first, we focused on the selection algorithm and the constraints posed by the preselection filters. Then, our work focused on the selection cost, where we made three propositions: a proposition to use atoms – a generalized version of Fujisaki's command-response model – to control F_0 (and possibly more general prosodic aspects), a proposition of cost integrating long-term constraints in the usually short term cost function – used to constrain duration but transposable to other measures – and finally a proposition of fuzzy penalty system for the concatenation cost.

Most of our work was achieved at ENSSAT, France. The work on atoms was performed during the 5 (exciting) month we spent at IDIAP, Switzerland, in our 3$^{\text{rd}}$ year of PhD though.

Summary of the Contributions

The first – and consequent – part of our research work focused on the unit selection algorithm and the unit selection graph (therefore including preselection filters), in particular the consequences of preselection on the richness of the selection graph. In our design, at least for the *baseline* system we presented in chapter 6, preselection filters act as a binary target cost, considering that if a unit does not fit a minimum number of linguistic and phonological criteria it has no place in the selection graph. Now, this is a double edged sword: the least there is nodes in the selection graph, the fastest selection will be; but the fewest the nodes, the least effective is the selection function (as it will then lack enough choice to be efficient). The join cost is particularly affected by the lack of choice as preselection filters implement target cost features. Our first experiment was therefore to assess the impact of our filters on the synthesis. The experiments we conducted proved our final set of filters, presented in section 1.3, to be efficient in reducing selection time without causing harm to the selection algorithm. Furthermore, this experiment allowed us to perform a first verification of the effectiveness of our cost function: confronting the best possible path with the worst one showed a very consequent gap, as expected. In the same experiment, we tested the impact of the preselection step on synthesis quality. We saw that filters did bring a consequent increase in speech quality, validating our point of view that it should be considered as part of the cost function, even though it is acting before the insertion of a node in the unit graph. This last point means that the more preselection filters are used, the less choice is given to the actual cost function as less nodes are added to the graph. This means a subtle balance between the number of filters and the number of nodes included in the graph has to be respected to yield best result. During the development of our system, we therefore added a mechanism relaxing the filter set when a minimal constraint on the number of nodes selected by the filters was not met (10 nodes in our experiments). Nonetheless, even with this mechanism, adding too many filters may result in a quality drop as the minimal number of nodes would rarely be exceeded.

Once these preliminary – but fundamental – results were obtained, we investigated the impact of the selection algorithm on the unit selection process; this is our first contribution. We implemented a unit selection block where the selection process searching the best unit sequence relied on an A^* search algorithm, which is more generalist that the usual Viterbi. Yet, we showed memory usage and ultimately computation time was more interesting with A^* than Viterbi and even some under-optimal Viterbi-based algorithms, like the beam-search algorithm we implemented (when the beam size in sufficiently consequent of course). For this experiment, we implemented 3 variants of the beam-search algorithm, with beam sizes of 10, 100 and 1 000 units (*i.e.* nodes).

This result is nuanced by the overall speech quality obtained with the 4 algorithms, which listening tests showed as roughly equivalent for A^* and beam-search with a beam

size of a few hundred. The results of the beam-search begin to degrade only when pruning gets consequent (*i.e.* with a beam size between 100 and 10 units). This result, which may sound astonishing at a glance, is easier to understand with the last experiment we performed. We explored the – sorted – list of the 100-best paths found by our optimal A^* algorithm and saw there was quite few variabilities in the selected sequences. A listening test comparing the 1^{st}, 10^{th}, 50^{th} and 100^{th} paths showed some degradation of the signal over the rank of the selected path, especially for the 100^{th}, but not as big as one could have thought.

After this work, we focused on the target cost and proposed two enhancements. The first one is a new way to compute a cost which aims at selecting the whole unit sequence that best minimizes a distance rather than choosing the sequence containing units that individually minimize the same distance. The objective is to include long term dependencies directly in the cost function, without any modification anywhere else. This new cost computation technique was tested on a phonemic duration distance, with the intent to avoid cases like excellent synthesis penalized by few very bad units. As the computation method is independent from the distance used within, it can also be used for the concatenation cost. Experiments showed that the new cost yielded to better synthesis, especially with an expressive voice.

The second proposition on the target cost we presented is a F_0 cost, aiming at constraining the F_0 contour via a generalized version of Fujisaki's Command-Response model called "atom decomposition of pitch". This cost is based on the assumption that the unit selection process naturally chooses units that feature a homogeneous F_0, a consideration reinforced by the fact F_0 is part of the concatenation cost. As most issues come from segments that have a very different local contour than what is expected, we believe that local atoms – modelling segment-level/syllable-level information by Gaussian functions – can enhance synthesized prosody if integrated in an appropriate target cost. We built a prototype which used parameters of atoms and compared it to a system using a regular F_0 target cost (in addition to the concatenation F_0 cost, present in both systems). Both systems were oracles, meaning they used real F_0 and real atom parameters extracted from it instead of predicted values. In that perceptive test, we saw our new cost performing much better than the regular F_0 target cost.

Finally, leaving the target cost, we developed a penalty system based on the notion of vocalic sandwich [Cadic et al. 2009]. Following an ordering of phonemes into 3 phonologically motivated clusters – the same as D. Cadic's [Cadic et al. 2009], penalties are applied in order to favor concatenations on phonemes known for their ability to support concatenations well. That kind of penalty has been the subject of several contributions for unit selection. Most propositions focused on two aspects: either preventing concatenations on vowels, known to be difficult to join without causing the appearance of artefacts in the

signal, or favoring the concatenation on syllable borders; which is almost equivalent. The difference with the approach we explored lies in the fact that we consider some phonemes that are not vowels can potentially prove as problematic (perhaps even more) than vowels (this is the case of liquids), and we also tend to prevent concatenations on these phonemes. The second and biggest difference lies in the fuzzy function we defined to smoothen the penalty based on the relative magnitude of the concatenation sub-costs in regard to their distributions. This fuzzy function is the key point of the technique we presented. Effectively, experiments did show a clear superiority of that approach over our *baseline* system and the result stands for all tested voices, expressive or not.

Perspectives

During this thesis, I first reviewed the standard unit selection mechanics and proposed an updated search architecture based on the A^* algorithm. In a second part, I focused on the component driving the search algorithm, the selection cost, and proposed enhancements to its sub-parts that enhanced synthesis quality.

My contributions helped identify and understand clearly the impact on synthesis of all major components of the unit selection search engine – preselection filters and subsequently the unit graph (and graph size), search strategy, formulation of the cost and sub-costs – as well as the interactions existing between these components. Even though, the work I have been undertaking could benefit several extensions, some planned but not realized by lack of time, others simply diverging too much from my research topic. The most important extensions are detailed hereafter:

On the selection algorithms work: For the work on the selection costs, we have focused on a very precise comparison, A^* *versus* Beam-search (and Viterbi by extension). Albeit this comparison probably is the one that makes most sense, it could be extended to more algorithms, including some exotic algorithms already experimented (R. Kumar's genetic implementation for instance). The experiment we accomplished concerning the 100-best paths is a punctual – yet embryonic – work. The aim of this analysis was never to complete an extensive study of the variability of unit sequences in all possible paths as it is a full study subject on its own. Such a subject would especially require an experimentation on a consequent quantity of data. Rather, our aim was to give some insight on what level of variability actually existed on the supposed range of paths that could be selected by the algorithms we have been taking into consideration. Even though, extending our study to a larger scale would certainly yield interesting conclusions. Concerning preselection filters, a full experiment could be made with very different sets of filters. Optimization methods, allowing to find the best set among a range of filters would also prove interesting. Indeed, presently,

preselection filters are often – if not always – selected by hand, at best with a small empirical evaluation (as for target and concatenation costs weights and sub-weights).

On the duration target cost: An interesting extension of our work on the "adaptive" duration target cost is to test activating the duration cost only on some sub-parts of the target sequence, when particular conditions are met. For instance, it could prove particularly useful when strong relaxation of preselection filters happened or simply when there is a high concatenation cost. A distinct pause duration model, which could use the same specifications as the duration target cost could and even should also be added. Implementing an intonation target cost relying on a F_0 contour prediction model is also part of our next work. However, the most important extension to this work should certainly be to test the generalization of the new "adaptive" distance to more costs (target and concatenation) of the cost function. This would require some deep changes inside concatenation sub-costs though, as the "adaptive" cost is based on a distance with a target value. However, some concatenation sub-cost (the F_0 sub-cost in particular), operating on some parameter, could be mixed with a second sub-cost trying to minimize the distance, on the same parameter, between a candidate unit and the mean in the candidate sequence. All sub-costs are not fit for this (our MFCC sub-cost for instance), but the problems the "adaptive" cost tries to fix are present for most sub-costs of the cost function.

On the atom target cost: In our work on atoms, we tested an "oracle". The TTS system, for the atom cost, relies on atom components extracted from actual annotations made on real speech data. Therefore, this work is not absolutely complete. Prediction of atom components and atom-reconstructed contours must be done, and evaluation of the resulting system has to be performed to get a reliable indicator of gains using atoms in a target cost. We assume directly predicting reconstructed contour is easier than predicting discrete events or parameters like atom impulses or amplitude. As it is smoother than real F_0, and especially as it is a sum of Gaussian functions, the reconstructed contour is easier to predict than real F_0. Results (especially from the listening test) are good, but the gap between atom-based costs and others is not as wide as expected. As the predictions given by a model may degrade the quality of atom annotations, final results using a predictor might not be strongly better than the current ones.

On the sandwich penalty system: Further improvement of the fuzzy method can be made though. In particular, more advanced fuzzy patterns might be investigated. Further work should be conducted about classification of phonemes in sets R, A and V. These subsets shouldn't be considered fixed and investigation on how they compare with other classifications should be done. In particular, liquids and glides could be added to V as they are usually problematic. Investigating language dependence of those classes is another important path. Finally, it would be particularly interesting

to activate the fuzzy penalty only when the concatenation cost magnitude becomes considerable. Moreover, the effectiveness of the fuzzy approach may be evaluated on corpora built using a recording script optimizing vocalic sandwich covering (following methodology in [Cadic et al. 2009]).

Finally, considering the fact that all our experiments were made on French corpora, internationalization of our conclusions is another challenge that takes sense. Since recently, this task is possible with the integration of new multi-lingual tools in the frontend of the IRISA TTS system. Some of our results are certainly directly transposable in most languages (the work on selection algorithms for instance) while others might share a bound with the French language, for example the duration cost (even though the cost mechanism itself – the most important part actually – seems directly transposable). In particular, our work on the fuzzy sandwich penalty should be tested on other languages, as it shows great promises.

Appendices

Appendix A

TTS Corpus Key Content

Each item in the TTS corpus has a key that is loaded on runtime (see section 1.3). The list of the 69 subparts of the key is given on the following figure:

Table A.1: List of the 69 subparts of the TTS corpus key defining each phone and NSS.

subkey	bit sta.	bit len.	value	m. code	meaning
0	0	6	34	0	segmental label
1	6	1	0	1	phone is in the onset of the syllable
2	7	1	0	2	phone is in the coda of the syllable
3	8	1	0	3	item is a non-speech sound
4	9	1	0	4	phone ends a breath group
5	10	1	0	5	phone ends a word
6	11	1	0	6	phone ends a sentence
7	12	1	0	7	phone is in the last syllable before NSS[1]
8	13	1	0	8	syllable is at the beginning of the word
9	14	1	0	9	syllable is at the end of the word
10	15	1	0	10	sandwich robustness class C
11	16	1	0	11	sandwich robustness class W
12	17	1	0	12	sandwich robustness class V
13	18	1	0	14	has to be ignored during selection process
14	19	1	0	15	syllable has an onset
15	20	1	0	16	syllable has a coda
16	21	1	0	17	phone position in the syllable
17	22	1	0	19	item is the first phone of the syllable
18	23	1	0	20	item is the last phone of the syllable

Continued on next page

[1] *i.e.*; at the end of the end of the breath group.

subkey	bit sta.	bit len.	value	m. code	meaning
19	24	1	0	21	syllable is the last one in the sentence
20	25	1	0	22	phone is rising
21	26	1	0	23	phone is descending
22	27	1	0	24	syllable is rising
23	28	1	0	25	syllable is descending
24	29	1	0	26	phone has diacritic long
25	30	1	0	27	phone is nasalized
26	31	1	0	28	phone has a low stress
27	32	1	0	29	phone has a high stress
28	33	1	0	30	phone is vowel
29	34	1	0	31	phone is liquid
30	35	1	0	32	phone is pulmonic
31	36	1	0	33	phone is plosive
32	37	1	0	34	phone is fricative
33	38	1	0	35	phone is approximant
34	39	1	0	36	phone is trill
35	40	1	0	37	phone is lateral
36	41	1	0	38	phone is flap
37	42	1	0	39	phone is dental
38	43	1	0	40	phone is alveolar
39	44	1	0	41	phone is velar
40	45	1	0	42	phone is glottal
41	46	1	0	43	phone is front
42	47	1	0	44	phone is back
43	48	1	0	45	phone is palatoalveolar
44	49	1	0	46	phone is retroflex
45	50	1	0	47	phone is palatal
46	51	1	0	48	phone is uvular
47	52	1	0	49	phone is pharyngeal
48	53	1	0	50	phone is epiglottal
49	54	1	0	51	phone is near front
50	55	1	0	52	phone is central
51	56	1	0	53	phone is near back
52	57	1	0	54	phone is close
53	58	1	0	55	phone is near close
54	59	1	0	56	phone is near open
55	60	1	0	57	phone is mid
56	61	1	0	58	phone is close mid

Continued on next page

subkey	bit sta.	bit len.	value	m. code	meaning
57	62	1	0	59	phone is open mid
58	63	1	0	60	phone is click
59	64	1	0	61	phone is voiced implosive
60	65	1	0	62	phone is ejective
61	66	1	0	63	phone is bilabial
62	67	1	0	64	phone is labiodental
63	68	1	0	65	phone is rounded
64	69	1	0	66	phone is double
65	70	1	0	67	phone is affricate
66	71	1	0	68	phone is voiced

Appendix B

Phonemic Alphabets and Appearance Frequencies

Concerning *IVS* corpus, distributions are very close to those F. Wioland observed [Wioland 1985], except for phonemes [ø] and [ə], which are inverted. The reason is the low difference between the two phonemes in French. Some phonemes [ø] can therefore be noted [ə] in the database. In particular, [ə] is often pronounced when it could be omitted. It is introduced by epenthesis, that is, it serves to ease the pronunciation of neighboring phones and is usually equivalent to a schwa (which is not a phoneme). Phoneme [ɲ] is assimilated to the sequence [nj] and is therefore absent form *IVS*. [ŋ], used quasi-exclusively for words imported from English, is also omitted. [ɲ] and [ŋ] are present in *Audiobook*.

In both corpora, liquid consonant [ʁ] is grouped with its regional variants [r] and [ʀ]. All three are noted [ʁ]. Phoneme [œ̃] is not present in *Audiobook*. It is assimilated to phoneme [ɛ̃]. Appearance frequencies for *Audiobook* are compatible with Wioland's, with sensible variations due to the literary nature of the corpus.

Phonemes representation				Frequency of the phoneme (%)		
IPA	*IVS*	*Audiobook*	Example	Wioland [Wioland 1985]	*IVS*	*Audiobook*
Consonants						
ʁ	ʁ	rr	**r**at	7,5	8,23	7,89
l	l	ll	**l**ait	6	6,64	6,21
s	s	ss	**s**ac	6,5	6,04	5,52
t	t	tt	**t**as	5,5	5,74	5,43
d	d	dd	**d**os	4,5	4,45	4,42
p	p	pp	**p**as	4	3,47	3,59
k	k	kk	**c**as	4,5	3,37	4,09
n	n	nn	**n**id	3	3,26	3,15
m	m	mm	**m**ot	3	2,83	3,92
v	v	vv	**v**ie	2,5	2,27	2,46
z	z	zz	**z**èbre	1,5	1,87	1,71
f	f	ff	**f**in	1,5	1,53	1,32
b	b	bb	**b**ien	1	1,22	1,22
ʒ	ʒ	jj	**j**oue	1,5	1,13	1,52
g	g	gg	**g**are	0,5	0,77	0,50
ʃ	ʃ	ch	va**ch**e	0,5	0,62	0,55
ɲ	-	gn	a**gn**eau	-	-	0,08
ŋ	-	ng	parki**ng**	-	-	0,00
Semi-vowels						
j	j	yy	tai**ll**e	2	2,17	1,54
w	w	ww	**ou**i	1	0,91	0,95
ɥ	ɥi	uy	p**ui**s	0,5	0,40	0,43
Vowels						
a	a	aa	pl**a**t	8	7,46	7,54
ø	ø	eu	j**eu**	0,5	4,29	3,85
ɛ	ɛ	ai	f**ai**t	5	4,20	6,57
i	i	ii	l**i**t	5,5	5,23	4,89
o	o	au	m**o**t	2	2,13	1,11
u	u	ou	c**ou**r	2,5	1,79	2,16
y	y	uu	r**u**e	2	2,21	2,14
e	e	ei	d**é**	5,5	5,84	4,95
ɔ	ɔ	oo	b**o**sse	1,5	1,46	1,68
œ	œ	oe	c**oeu**r	0,5	0,68	0,53
ə	ə	ee	ch**e**val	3,5	0,64	1,70
ɑ̃	ɑ̃	an	bl**an**c	3,5	3,34	3,37
ɔ̃	ɔ̃	on	t**on**	2	2,09	1,61
ɛ̃	ɛ̃	in	br**in**	1	1,21	1,41
œ̃	œ̃	-	br**un**	0,5	0,51	-

Figure B.1: Phonemes used in the thesis voice corpora and their appearing frequencies. Equivalents to IPA standard notations (used throughout the thesis) are given for *IVS* and *Audiobook* (full corpora). Symbol "-" means the phoneme is not present in the given corpus (never realized or merged with another phoneme). Appearance frequency for these phonemes are given for the two corpora, along with the frequencies observed by F. Wioland [Wioland 1985].

Appendix C

Example of Sentences Used in the Listening Tests

The following text is an extract of the 27141 sentences synthesized for each listening test:

Ils sont absolument privés de tous leurs droits civiques ;
Mais je ne dois pas laisser les questions de politique scolaire me détourner de mon sujet.
Reste, comme je l'ai signalé plus haut, une objection :
Mes lecteurs vont penser à présent que je ne suis guère logique avec moi-même.
Mais bornons là notre éloge de cet élément bénéfique et expliquons-nous.
Un exemple fera plus pour éclairer ma pensée que tout un volume de généralités.
Supposons que je voie approcher deux individus dont je désire déterminer le rang.
Mais je ne dois pas céder à la tentation de m'étendre sur ce sujet.
Telle est du moins la pénible leçon que l'expérience m'a enseignée.
La situation de cette minorité qui n'a pas réussi est réellement pitoyable.
Rejetés par les classes supérieures, ces gens sont aussi méprisés par leurs inférieurs.
Les professions libérales, les services publics leur sont fermés ;
La dimension des côtés dépendra, bien entendu, de l'âge de l'individu.
Mais la dimension de nos côtés n'est pas notre propos.
Si nos côtés étaient inégaux, nos angles pourraient l'être aussi.
Mais la vie serait trop brève pour ces tâtonnements monotones.
Et sinon, comment l'empêcher de semer la désolation dans les rangs de ses camarades ?
La mode se répandit comme une traînée de poudre.
Inutile de dire qu'elle ne tarda pas à s'étendre aux régions voisines ;
En ce temps-là, vivre était un délice en soi, car vivre, c'était voir.
Toute assemblée, même réduite, réjouissait le regard ;
Il suffira d'une brève explication pour le leur faire comprendre.

Vous verrez, bien entendu, une ligne droite, moitié rouge, moitié verte.
La malheureuse épousée se suicida en découvrant la fraude dont elle avait été victime.
Un certain nombre d'entre elles s'y avouèrent ouvertement opposées ;
Il leur faudrait à présent sacrifier cette ambition honorable.
La bataille, ou plutôt le carnage, fut de courte durée.
Je devrais plutôt dire qu'il aurait beaucoup de mal à le faire ;
Plus d'un enfant plein de promesses a été sacrifié de cette manière.

List of Figures

List of Tables

Publications During the Thesis

1 International Conferences with a Reading Comitee

(1) D. Guennec and D. Lolive, "Unit Selection Cost Function Exploration Using an A* based Text-to-Speech System", *17th International Conference on Text, Speech and Dialogue*, 2014.

(2) D. Guennec, J. Chevelu and D. Lolive, "Defining a Global Adaptive Duration Target Cost for Unit Selection Speech Synthesis", *18th International Conference on Text, Speech and Dialogue*, 2015.

(3) J. Chevelu, D. Lolive, S. Le Maguer and D. Guennec, "How to Compare TTS Systems: A New Subjective Evaluation Methodology Focused on Differences", *16th Interspeech Conference*, 2015.

(4) P. Alain, J. Chevelu, D. Guennec, G. Lecorvé and D. Lolive, "The IRISA Text-To-Speech System for the Blizzard Challenge 2015", *Blizzard Challenge workshop*, 2015.

(5) E. Delais-Roussarie, D. Lolive, H. Yoo and D. Guennec, "How to improve rhythmic patterns according to literary genre in synthesized speech", *8th Speech Prosody Conference*, 2016.

(6) M. Sečujski, B. Gerazov, T. G. Csapó, V. Delić, P. N. Garner, A. Gjoreski, D. Guennec, Z. Ivanovski, A. Melov, G. Németh, A. Stojković and G. Szaszák, "Design of a Speech Corpus for Research on Cross-Lingual Prosody Transfer", *18th International Conference on Speech and Computer*, 2016.

(7) D. Guennec and D. Lolive, "On the suitability of vocalic sandwiches, a corpus-based TTS engine", *17th Interspeech Conference*, 2016.

2 International Conferences with a Reading Comitee in French Language

(1) D. Guennec and D. Lolive, "Utilisation d'un algorithme A* pour l'analyse de la sélection d'unités en synthèse de la parole", *30th Journées d'Études sur la Parole*, 2014.

(2) D. Guennec and D. Lolive, "Une pénalité floue fondée phonologiquement pour améliorer la Sélection d'Unité", *31th Journées d'Études sur la Parole*, 2016.

(3) E. Delais-Roussarie, D. Lolive, H. Yoo and D. Guennec, "Patrons Rythmiques et Genres Littéraires en Synthèse de la Parole", *31th Journées d'Études sur la Parole*, 2016.

(4) J. Chevelu, D. Lolive, S. Le Maguer and D. Guennec, "Se concentrer sur les différences : une méthode d'évaluation subjective efficace pour la comparaison de systèmes de synthèse", *31th Journées d'Études sur la Parole*, 2016.

Bibliography

[Alain et al. 2015]

Alain, P., J. Chevelu, D. Guennec, G. Lecorvé, and D. Lolive (2015). "The IRISA Text-To-Speech System for the Blizzard Challenge 2015." *Blizzard Challenge workshop* (cit. on p. 139).

[Alias et al. 2003]

Alias, F. and X. Llorà (2003). "Evolutionary weight tuning based on diphone pairs for unit selection speech synthesis." *Proceedings of the 8th European Conference on Speech Communication and Technology (EuroSpeech).* 1, pp. 1333–1336 (cit. on p. 63).

[Alías et al. 2011]

Alías, F., L. Formiga, and X. Llorá (2011). "Efficient and reliable perceptual weight tuning for unit-selection text-to-speech synthesis based on active interactive genetic algorithms: A proof-of-concept." *Speech Communication,* 53 (5), pp. 786–800 (cit. on pp. 60, 61, 63, 114).

[Atal 1971]

Atal, B. S. (1971). "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave." *The Journal of the Acoustical Society of America,* 50 (2B), p. 637 (cit. on p. 25).

[Barbot et al. 2012]

Barbot, N., O. Boeffard, and A. Delhay (2012). "Comparing performance of different set-covering strategies for linguistic content optimization in speech corpora." *International Conference on Language Resources and Evaluation (LREC'12),* (cit. on p. 42).

[Bartkova et al. 1987]

Bartkova, K. and C. Sorin (1987). "A model of segmental duration for speech synthesis in French." *Speech Communication,* 6 (3), pp. 245–260 (cit. on p. 24).

[Bechet 2001]

Bechet, F (2001). "Liaphon - Un systeme complet de phonetisation de textes." *Traitement Automatique des Langues (T.A.L.) edition Hermes,* 42 (1) (cit. on p. 89).

[Benesty et al. 2008]

Benesty, J., M. M. Sondhi, and Y. Huang, eds. (2008). *Springer Handbook of Speech Processing.* Springer-Verlag Berlin Heidelberg, pp. XXXVI, 1176 (cit. on pp. 7, 8, 11–14).

[Beutnagel et al. 1998]

Beutnagel, M., A. Conkie, and A. K. Syrdal (1998). "Diphone synthesis using unit selection." *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis* (cit. on pp. 55, 92).

[Beutnagel et al. 1999]

Beutnagel, M., M. Mohri, and M. Riley (1999). "Rapid unit selection from a large speech corpus for concatenative speech synthesis." *Proceedings of the European Conference on Speech Communication and Technology (Budapest, Hungary).* Vol. 2, pp. 607–610 (cit. on p. 61).

[Black et al. 1995]

Black, A. W. and N. Campbell (1995). "Optimising selection of units from speech databases for concatenative synthesis." *Proc. Eurospeech,* pp. 581–584 (cit. on pp. vi, vii, 3, 60, 61, 95).

[Black et al. 1994]

Black, A. W. and P. Taylor (1994). "CHATR: a generic speech synthesis system." *15th conference on Computational linguistics.* Association for Computational Linguistics, pp. 983–986 (cit. on p. 26).

[Black et al. 2007]

Black, A. W., H. Zen, and K. Tokuda (2007). "Statistical Parametric Speech Synthesis." *IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07,* 4 (cit. on pp. vi, 29).

[Blouin et al. 2002]

Blouin, C., O. Rosec, P. Bagshaw, and C. D'Alessandro (2002). "Concatenation cost calculation and optimisation for unit selection in TTS." *IEEE Workshop on Speech Synthesis,* pp. 0–3 (cit. on pp. 55, 61, 62).

[Boë 1990]

Boë, L.-J. (1990). "Éléments d'unification pour les Sciences de la Parole." Habilitation à Diriger des Recherches. Institut National Polytechnique de Grenoble, INPG. (cit. on pp. iii, 7).

[Boeffard 2004]

Boeffard, O. (2004). "Contributions à la synthèse de la parole." Habilitation à Diriger des Recherches. Rennes 1 / ENSSAT (cit. on pp. iii, 7).

[Boeffard et al. 2012]

Boeffard, O., L. Charonnat, S. Le Maguer, D. Lolive, and G. Vidal (2012). "Towards Fully Automatic Annotation of Audio Books for TTS." *LREC,* pp. 975–980 (cit. on pp. 73, 74).

[Breen et al. 1998]

Breen, A. and P Jackson (1998). "Non-uniform unit selection and the similarity metric within BT's Laureate TTS system." *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis* (cit. on p. vi).

[Buchholz et al. 2013]

Buchholz, S., J. Latorre, and K. Yanagisawa (2013). "Crowdsourced assessment of speech synthesis." *Crowdsourcing for Speech Processing,* pp. 173–216 (cit. on p. 83).

[Cadic 2011]

Cadic, D. (2011). "Optimisation du procédé de création de voix en synthèse par sélection." PhD thesis. Université Paris-sud 11 (cit. on pp. 27, 42, 43).

[Cadic et al. 2010]

Cadic, D. and C. D'Alessandro (2010). "High Quality TTS Voices Within One Day." *Seventh ISCA Workshop on Speech Synthesis* (cit. on p. 133).

[Cadic et al. 2009]

Cadic, D., C. Boidin, and C. D'Alessandro (2009). "Vocalic sandwich, a unit designed for unit selection TTS." *Tenth Annual Conference of the International Speech Communication Association.* 1, pp. 2079–2082 (cit. on pp. x, 3, 132, 133, 138, 143, 146).

[Cadic et al. 2010]

Cadic, D., C. Boidin, and C. D'Alessandro (2010). "Towards optimal TTS corpora." *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010), Valetta, Malta,* pp. 99–104 (cit. on pp. 42, 132).

[Calliope 1989]

Calliope (1989). *La parole et son traitement automatique.* Éditions Masson (cit. on pp. 7, 8, 14, 15, 24, 25).

[Cernak et al. 2005]

Cernak, M. and M Rusko (2005). "An evaluation of synthetic speech using the PESQ measure." *Proc. of European Congress on Acoustics,* pp. 1–4 (cit. on p. 82).

[Chevelu et al. 2007]

Chevelu, J., N. Barbot, O. Boëffard, and A. Delhay (2007). "Lagrangian relaxation for

optimal corpus design." *Proceedings of the 6th ISCA Tutorial and Research,* pp. 211–216 (cit. on p. 42).

[Chevelu et al. 2014]
Chevelu, J., G. Lecorvé, D. Lolive, G. L. Jonathan Chevelu, and D. Lolive (2014). "ROOTS: a toolkit for easy, fast and consistent processing of large sequential annotated data collections." *LREC,* pp. 619–626 (cit. on p. 69).

[Chevelu et al. 2015]
Chevelu, J., D. Lolive, S. Le Maguer, and D. Guennec (2015). "How to Compare TTS Systems: A New Subjective Evaluation Methodology Focused on Differences." *Interspeech,* (cit. on pp. viii, 69, 83, 85, 137).

[Clark et al. 2007]
Clark, R. A., K. Richmond, and S. King (2007). "Multisyn: Open-domain unit selection for the Festival speech synthesis system." *Speech Communication,* 49 (4), pp. 317–330 (cit. on pp. vi, 95).

[Conkie 1999]
Conkie, A. (1999). "A robust unit selection system for speech synthesis." *The Journal of the Acoustical Society of America,* 105 (2), p. 978 (cit. on pp. 25, 95).

[Conkie et al. 1994]
Conkie, A. and S. Isard (1994). "Optimal coupling of diphones." *Second ESCA/IEEE Workshop on Speech Synthesis,* (September), pp. 293–304 (cit. on p. 65).

[Conkie et al. 2000]
Conkie, A., M. C. Beutnagel, A. K. Syrdal, and P. E. Brown (2000). "Preselection of candidate units in a unit selection-based text-to-speech synthesis system." *International Conference on Spoken Language Processing - ICSLP.* Vol. 3. Icslp, pp. 314–317 (cit. on p. 63).

[Conkie et al. 2008]
Conkie, A., A. Syrdal, Y. J. Kim, and M. Beutnagel (2008). "Improving preselection in unit selection synthesis." *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH,* pp. 585–588 (cit. on p. 63).

[Cooper et al. 1951]
Cooper, F. S., A. M. Liberman, and J. M. Borst (1951). "The Interconversion Of Audible And Visible Patterns As A Basis For Research In The Perception Of Speech." *Psychologie,* 37, pp. 318–325 (cit. on p. 24).

[Cowie et al. 2003]
Cowie, R. and R. R. Cornelius (2003). "Describing the emotional states that are expressed in speech." *Speech Communication,* 40 (1-2), pp. 5–32 (cit. on p. 122).

[Delattre 1966]
Delattre, P. (1966). "Les Dix Intonations de base du francais." *The French Review,* 40 (1), pp. 1–14 (cit. on p. 14).

[Di Cristo 2000]
Di Cristo, A. (2000). "Interpréter la prosodie." *Actes des XXIIIèmes Journées d'Etude sur la Parole,* pp. 13–29 (cit. on pp. v, 13).

[Dixon et al. 1968]
Dixon, N and H Maxey (1968). "Terminal analog synthesis of continuous speech using the diphone method of segment assembly." *IEEE Transactions on Audio and Electroacoustics,* AU-16 (1), pp. 39–50 (cit. on pp. iv, 10, 25).

[Donovan 1996]
Donovan, R. E. (1996). "Trainable speech synthesis." PhD thesis. Cambridge (cit. on p. 51).

[Donovan 2001]
Donovan, R. E. (2001). "A new distance measure for costing spectral discontinuities in concatenative speech synthesizers." *ITRW* (cit. on pp. 92, 133).

[Ekman et al. 1969]
Ekman, P. and W. V. Friesen (1969). "The repertoire of nonverbal behavior: Categories, origins, usage, and coding." *Semiotica,* 1, pp. 49–98 (cit. on p. 122).

[Estes et al. 1964]
Estes, S., H. Kerby, H. Maxey, and R. Walker (1964). "Speech synthesis from stored data." *IBM Journal of Research and Development,* (cit. on p. 25).

[Fan et al. 2015]
Fan, Y., Y. Qian, F. K. Soong, and L. He (2015). "Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis." *IEEE International Conference on Acoustics, Speech and Signal Processing,* (cit. on pp. 28, 31).

[Flanagan 1972]
Flanagan, J. L. (1972). "Voices of men and machines." *The Journal of the Acoustical Society of America,* 51 (5), pp. 1375–1386 (cit. on pp. 17, 18, 21, 22).

[Francois 2002]
Francois, H. (2002). "Synthèse de la parole par concaténation d'unités acoustiques :

construction et exploitation d'une base de parole continue." PhD thesis. Université de Rennes 1, pp. 1–204 (cit. on pp. 48, 53, 54).

[Francois et al. 2002]
Francois, H. and O. Boeffard (2002). "The greedy algorithm and its application to the construction of a continuous speech database." *Proceedings of LREC.* Vol. 5, pp. 1420–1426 (cit. on p. 41).

[Fujimura 1976]
Fujimura, O (1976). "Syllables as concatenated demisyllables and affixes." *The Journal of the Acoustical Society of America.* Vol. 59. 1 (cit. on p. 25).

[Fujisaki et al. 1969]
Fujisaki, H and S Nagashima (1969). "A model for the synthesis of pitch contours of connected speech." *Annual Report of the Engineering Research Institute, University of Tokyo,* 28, pp. 53–60 (cit. on p. 122).

[Fukada et al. 1992]
Fukada, T., K. Tokuda, T. Kobayashi, and S. Imai (1992). "An adaptive algorithm for mel-cepstral analysis of speech." *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 1992,* 1, pp. 137–140 (cit. on p. 30).

[Garcia et al. 2006]
Garcia, M.-n., C. D'Alessandro, G. Bailly, P. Boula De Mareüil, and M. Morel (2006). "A joint prosody evaluation of French text-to-speech synthesis systems." *LREC,* pp. 55–57 (cit. on p. 83).

[Gauvain et al. 1990]
Gauvain, J.-L. L, L. F. Lamel, and M. Eskenazi (1990). "Design considerations and text selection for BREF, a large French readspeech corpus." *Proc. of ICSLP,* pp. 1097–1100 (cit. on p. 41).

[Ghahremani et al. 2014]
Ghahremani, P. et al. (2014). "A pitch extraction algorithm tuned for automatic speech recognition." *IEEE International Conference on Acoustics, Speech and Signal Processing* (cit. on p. 125).

[Govind et al. 2012]
Govind, D. and S. R. M. Prasanna (2012). "Expressive speech synthesis: a review." *International Journal of Speech Technology,* 16 (2), pp. 237–260 (cit. on p. 114).

[Guennec et al. 2014a]
Guennec, D. and D. Lolive (2014a). "Unit Selection Cost Function Exploration Using

an A* based Text-to-Speech System." *17th International Conference on Text, Speech and Dialogue,* pp. 449–457 (cit. on pp. 58, 95, 97).

[Guennec et al. 2014b]
Guennec, D. and D. Lolive (2014b). "Utilisation d'un algorithme A* pour l'analyse de la sélection d'unités en synthèse de la parole." *XXXèmes journées d'études sur la parole* (cit. on p. 58).

[Guennec et al. 2016]
Guennec, D. and D. Lolive (2016). "On the suitability of vocalic sandwiches in a corpus-based TTS engine." *17th Interspeech conference* (cit. on p. 132).

[Guennec et al. 2015]
Guennec, D., J. Chevelu, and D. Lolive (2015). "Defining a Global Adaptive Duration Target Cost for Unit Selection Speech Synthesis." *18th International Conference on Text, Speech and Dialogue.* Plzen, pp. 149–157 (cit. on p. 113).

[Harris 1953]
Harris, C. (1953). "A study of the building blocks in speech." *The Journal of the Acoustical Society of America,* 25, p. 183 (cit. on p. 25).

[Hart et al. 1968]
Hart, P., N. Nilsson, and B Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths." *IEEE Transactions of Systems Science and Cybernetics,* 4 (2), pp. 100–107 (cit. on pp. 57, 97).

[Hashimoto et al. 2015]
Hashimoto, K., K. Oura, Y. Nankaku, and K. Tokuda (2015). "The Effect Of Neural Networks In Statistical Parametric Speech Synthesis." *IEEE International Conference on Acoustics, Speech and Signal Processing.* Melbourne, pp. 4455–4459 (cit. on pp. vi, 31, 114).

[Hinterleitner et al. 2011a]
Hinterleitner, F, S Zabel, S Möller, L Leutelt, and C Norrenbrock (2011a). "Predicting the Quality of Synthesized Speech Using Reference-Based Prediction Measures." *Proc. of the 22th Konferenz Elektronische Sprachsignalverarbeitung (ESSV), Aachen, Germany,* pp. 99–106 (cit. on p. 82).

[Hinterleitner et al. 2011b]
Hinterleitner, F., G. Neitzel, S. Moller, and C. Norrenbrock (2011b). "An Evaluation Protocol for the Subjective Assessment of Text-to-Speech in Audiobook Reading Tasks." *Proc. Blizzard Challenge Workshop* (cit. on p. 83).

[Hirose et al. 1982]

Hirose, K and H Fujisaki (1982). "Analysis and synthesis of voice fundamental frequency contours of spoken sentences." *Acoustics Speech and Signal Processing IEEE International Conference on ICASSP 82,* 7, pp. 950–953 (cit. on p. 122).

[Hirose et al. 2005]

Hirose, K., K. Sato, Y. Asano, and N. Minematsu (2005). "Synthesis of F0 contours using generation process model parameters predicted from unlabeled corpora: Application to emotional speech synthesis." *Speech Communication,* 46 (3-4), pp. 385–404 (cit. on p. 123).

[Holmes 1983]

Holmes, J. (1983). "Formant synthesizers: Cascade or parallel?" *Speech communication,* 2 (4), pp. 251–273 (cit. on p. 24).

[Honnet et al. 2015]

Honnet, P.-e., B. Gerazov, and P. N. Garner (2015). "Atom decomposition-based intonation modelling." *IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 1–5 (cit. on pp. 122, 123).

[Hunt et al. 1996]

Hunt, A. J. and A. W. Black (1996). "Unit selection in a concatenative speech synthesis system using a large speech database." *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings.* Vol. 1. Ieee, pp. 373–376 (cit. on pp. vi, 2, 26, 52, 53, 58, 61, 95).

[ITU-R 2015]

ITU-R (2015). *Method for the subjective assessment of intermediate quality level of audio systems.* Tech. rep. International Telecommunication Union, pp. 1–18 (cit. on pp. 84, 128).

[ITU-T 2001]

ITU-T (2001). *ITU-T Recommendation P.862 - PESQ measure.* Tech. rep. International Telecommunication Union (cit. on p. 82).

[Karaali et al. 1996]

Karaali, O., G. Corrigan, and I. Gerson (1996). "Speech synthesis with neural networks." *World Congress on Neural Networks,* (September), pp. 45–50 (cit. on p. 114).

[Kawahara et al. 2008]

Kawahara, H et al. (2008). "Tandem-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, F0, and aperiodicity estimation." *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on,* pp. 3933–3936 (cit. on pp. 30, 66).

[Kawai et al. 2000]

Kawai, H., S. Yamamoto, N. Higuchi, and T. Shimizu (2000). "A Design Method of Speech Corpus for Text-To-Speech Synthesis Taking Account of Prosody." *Sixth International Conference on Spoken Language Processing (ICSLP 2000),* pp. 420–425 (cit. on p. 41).

[Kempelen 1791]

Kempelen, W. von (1791). *Mechanismus der meschlichen Sprache.* Bauer, B. (cit. on pp. 17, 18, 20).

[King 2010]

King, S. (2010). "A beginners' guide to statistical parametric speech synthesis" (cit. on p. 30).

[King et al. 2012]

King, S. and V. Karaiskos (2012). "The blizzard challenge 2012." *Proc. Blizzard Challenge workshop 2012* (cit. on p. 82).

[Klatt 1982]

Klatt, D. H. (1982). "The Klattalk text-to-speech conversion system." *Acoustics, Speech, and Signal Processing, IEEE,* 02139 (1), pp. 1589–1592 (cit. on p. 24).

[Klatt 1987]

Klatt, D. H. (1987). "Review of text-to-speech conversion for English." *The Journal of the Acoustical Society of America,* 82 (3), pp. 737–793 (cit. on pp. 17, 24).

[Klein et al. 2003]

Klein, D. and C. D. Manning (2003). "A* parsing: fast exact Viterbi parse selection." *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03.* June, pp. 40–47 (cit. on p. 51).

[Krul et al. 2006]

Krul, A., T. Moudenc, G Damnati, F Yvon, and T. Moudenc (2006). "Corpus design based on the Kullback-Leibler divergence for Text-To-Speech synthesis application." *Proc. of ICSLP,* pp. 2030–2033 (cit. on p. 40).

[Kumar 2004]

Kumar, R. (2004). "A genetic algorithm for unit selection based speech synthesis." *Eighth International Conference on Spoken Language Processing* (cit. on pp. 58, 95).

[Lambert et al. 2007]

Lambert, T., N. Braunschweiler, and S. Buchholz (2007). "How (Not) to Select Your

Voice Corpus: Random Selection vs. PhonologicallyBalanced." *SSW6,* pp. 264–269 (cit. on p. 43).

[Laprie et al. 1998]
Laprie, Y. and V. Colotte (1998). "Automatic pitch marking for speech transformations via TD-PSOLA." *Proceeding of the European Signal Processing Conference,* pp. 1133–1136 (cit. on p. 65).

[Latorre et al. 2014]
Latorre, J., K. Yanagisawa, V. Wan, B. Kolluru, and M. J. F. Gales (2014). "Speech intonation for TTS: Study on evaluation methodology." *Proceedings of Interspeech* (cit. on p. 83).

[Le Maguer 2013]
Le Maguer, S. (2013). "Evaluation expérimentale d'un système statistique de synthèse de la parole, HTS, pour la langue française." PhD thesis. Université de Rennes 1 (cit. on p. 29).

[Lindblom 1963]
Lindblom, B (1963). "Spectrographic study of vowel reduction." *The Journal of the Acoustical Society of America,* 35 (November 1963), pp. 1773–1781 (cit. on p. 132).

[Lolive 2008]
Lolive, D. (2008). "Transformation de l'intonation." PhD thesis. Université de Rennes 1 (cit. on p. 14).

[Macon et al. 1998]
Macon, M., A. Cronk, and J. Wouters (1998). "Generalization and discrimination in tree-structured unit selection." *Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis,* pp. 1–6 (cit. on p. 61).

[Maeda 1979]
Maeda, S. (1979). "An Articulatory Model of the Tongue Based On a Statistical Analysis." *The Journal of the Acoustical Society of America,* 65 (1), S22–S22 (cit. on p. 23).

[Makhoul 1975]
Makhoul, J. (1975). "Linear Prediction: a Tutorial Review." *Proceedings of the IEEE,* 63 (4), pp. 561–580 (cit. on p. 25).

[Mermelstein 1973]
Mermelstein, P (1973). "Articulatory model for the study of speech production." *Journal of the Acoustical Society of America,* 53 (4), pp. 1070–1082 (cit. on p. 21).

[Moulines et al. 1990]

Moulines, E. and F. Charpentier (1990). "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones." eng. *Speech communication,* 9 (5-6), pp. 453–467 (cit. on pp. 26, 65).

[Nilsson 1982]

Nilsson, N. J. (1982). *Principles of Artificial Intelligence.* Springer-Verlag (cit. on pp. 49, 57, 95, 97).

[Peterson et al. 1958]

Peterson, G. E., W. S. Y. Wang, and E. Sivertsen (1958). "Segmentation techniques in speech synthesis." *The Journal of the Acoustical Society of America,* 30 (8), pp. 739–742 (cit. on pp. iv, 10, 25).

[Pfitzinger 2004]

Pfitzinger, H. R. (2004). "DFW-based spectral smoothing for concatenative speech synthesis." *Interspeech* (cit. on p. 65).

[Pols et al. 1987]

Pols, L., J. Lefevre, G. Boxelaar, and N. Son (1987). "Word intelligibility of a rule synthesis system for French." *European Conference on Speech Technology.* September, pp. 1179–1182 (cit. on p. 25).

[Popescu et al. 2006]

Popescu, A., C. Boidin, and D. Cadic (2006). "Contraintes globales pour la sélection des unités en synthèse vocale." *XVIèmes Journées d'Etude sur la Parole.* Vol. 0, p. 38 (cit. on p. 64).

[Prahallad et al. 2014]

Prahallad, K. et al. (2014). "The blizzard challenge 2014." *Proc. Blizzard Challenge workshop 2014* (cit. on p. 82).

[Přibil et al. 2015]

Přibil, J., A. Přibilová, and J. Matoušek (2015). "Experiment with GMM-Based Artefact Localization in Czech Synthetic Speech." *Text, Speech, and Dialogue: 18th International Conference.* Ed. by P. Král and V. Matoušek. Pilsen: Springer International Publishing, pp. 23–31 (cit. on p. 82).

[Rubin et al. 1981]

Rubin, P. and T. Baer (1981). "An articulatory synthesizer for perceptual research." *The Journal of the Acoustical Society of America,* 70 (2), p. 321 (cit. on p. 21).

[Russell et al. 2003]

Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach.* Prentice H. Prentice Hall (cit. on p. 95).

[Sagisaka 1988]

Sagisaka, Y. (1988). "Speech synthesis by rule using an optimal selection of non-uniform synthesis units." *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing.* Ieee, pp. 679–682 (cit. on pp. vi, 2, 26, 47).

[Saha Ray 2013]

Saha Ray, S. (2013). *Graph Theory with Algorithms and its Applications.* Springer India (cit. on p. 57).

[Sainz et al. 2014]

Sainz, I., E. Navas, I. Hernaez, A. Bonafonte, and F. Campillo (2014). "TTS evaluation campaign with a common Spanish database." *LREC,* pp. 2155–2160 (cit. on p. 83).

[Sakai et al. 2008]

Sakai, S., T. Kawahara, and S. Nakamura (2008). "Admissible stopping in Viterbi beam search for unit selection in concatenative speech synthesis." *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* (January 2016), pp. 4613–4616 (cit. on pp. 55, 95, 97).

[Schroeder 1993]

Schroeder, M. R. (1993). "A brief history of synthetic speech." *Speech Communication,* 13 (1-2), pp. 231–237 (cit. on p. 17).

[Schwarz 2007]

Schwarz, D. (2007). "Corpus-Based Concatenative Synthesis." *IEEE signal processing magazine,* (March), pp. 92–104 (cit. on pp. 55, 56).

[Story 2009]

Story, B. H. (2009). "Simulation of sentence-level speech with kinematic models of the vocal tract shape and vocal folds." *third international symposium on biomechanics, human function and information science.* Kanazawa, pp. 55–61 (cit. on p. 21).

[Story 2011]

Story, B. H. (2011). "TubeTalker: An airway modulation model of human sound production." *First International Workshop on Performative Speech and Singing Synthesis,* 8 (cit. on p. 23).

[Stylianou et al. 2001]

Stylianou, Y and A. K. Syrdal (2001). "Perceptual and objective detection of discon-

tinuities in concatenative speech synthesis." *International Conference on Acoustics, Speech, and Signal Processing.* Vol. 2, pp. 837–840 (cit. on pp. x, 61, 132).

[Stylianou 2001]
Stylianou, Y. (2001). "Applying the harmonic plus noise model in concatenative speech synthesis." *IEEE Transactions on Speech and Audio Processing,* 9 (1), pp. 21–29 (cit. on p. 65).

[Stylianou et al. 1997]
Stylianou, Y., T. Dutoit, and J. Schroeter (1997). "Diphone Concatenation using a Harmonic plus Noise Model of Speech." *Interspeech,* pp. 613–616 (cit. on p. 65).

[Syrdal et al. 1998]
Syrdal, A. K., Y. Stylianou, L. Garrison, A. Conkie, and J. Schroeter (1998). "TD-PSOLA versus Harmonic Plus Noise Model in Diphone Based Speech Synthesis." *International Conference on Acoustics, Speech, and Signal Processing,* pp. 273–276 (cit. on p. 65).

[Taylor 2006]
Taylor, P. (2006). "The Target Cost Formulation in Unit Selection Speech Synthesis." *Stress: The International Journal on the Biology of Stress,* pp. 2038–2041 (cit. on p. 115).

[Taylor et al. 1999]
Taylor, P. and A. W. Black (1999). "Speech synthesis by phonological structure matching." *Proceedings of EuroSpeech,* 4, pp. 1531–1534 (cit. on p. 51).

[Taylor et al. 1998]
Taylor, P., A. W. Black, and R. Caley (1998). "The architecture of the Festival speech synthesis system." *Proc. of the ESCA Workshop in Speech Synthesis,* pp. 147–151 (cit. on p. vi).

[Tihelka et al. 2010]
Tihelka, D., J. Kala, and J. Matoušek (2010). "Enhancements of Viterbi search for fast unit selection synthesis." *INTERSPEECH,* pp. 174–177 (cit. on pp. 55, 95, 97).

[Tihelka et al. 2014]
Tihelka, D., J. Matoušek, and Z. Hanzlíček (2014). "Modelling F0 Dynamics in Unit Selection Based Speech Synthesis." *Text, Speech and Dialogue,* 1 (Springer), pp. 457–464 (cit. on pp. x, 60, 132).

[Tokuda et al. 1995]
Tokuda, K., T. Kobayashi, and S. Imai (1995). "Speech parameter generation from

HMM using dynamic features." *Acoustics, Speech and Signal Processing.* Vol. 1. 5. IEEE, pp. 660–663 (cit. on p. 28).

[Tuerk et al. 1993]
Tuerk, C and T Robinson (1993). "Speech synthesis using artificial neural networks trained on cepstral coefficients." *EUROSPEECH.* Vol. 1, pp. 4–7 (cit. on p. 114).

[Vepa 2004]
Vepa, J. (2004). "Join cost for unit selection speech synthesis." *Speech Synthesis.* The University of Edinburgh. College of Science and Engineering. School of Informatics, pp. 35–62 (cit. on p. 51).

[Viswanathan et al. 2005]
Viswanathan, M. and M. Viswanathan (2005). "Measuring speech quality for text-to-speech systems: development and assessment of a modified mean opinion score (MOS) scale." *Computer Speech & Language,* 19 (1), pp. 55–83 (cit. on p. 83).

[Viterbi 1967]
Viterbi, A. J. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Transactions on Information Theory,* 13 (2), pp. 260–269 (cit. on pp. 56, 95).

[Viterbi 2006]
Viterbi, A. J. (2006). "A personal history of the Viterbi algorithm." *IEEE Signal Processing Magazine,* 23 (4), pp. 120–142 (cit. on p. 56).

[Wioland 1985]
Wioland, F. (1985). *Les Structures syllabiques du français : fréquence et distribution des phonèmes consonantiques : Contraintes idiomatiques dans les séquences consonantiques.* Genève : S (cit. on pp. 77, 79, 80, 153, 154).

[Wouters et al. 2001]
Wouters, J. and M. W. Macon (2001). "Control of Speech Dynamics in Concatenative Speech Synthesis." *IEEE Transactions on Speech and Audio Processing,* 9 (1), pp. 30–38 (cit. on p. 66).

[Wouters et al. 1998]
Wouters, J. and M. Macon (1998). "A perceptual evaluation of distance measures for concatenative speech synthesis." *ICSLP,* pp. 5–8 (cit. on p. 61).

[Yamagishi et al. 2008]
Yamagishi, J., Z. Ling, and S. King (2008). "Robustness of HMM-based speech synthesis." *Science And Technology,* pp. 2–5 (cit. on p. vi).

[Yamagishi et al. 2009]

Yamagishi, J. et al. (2009). "Robust Speaker-Adaptive HMM-Based Text-to-Speech Synthesis." *IEEE Transactions on Audio, Speech, and Language Processing,* 17 (6), pp. 1208–1230 (cit. on p. 30).

[Yi 1998]

Yi, J. (1998). *Natural-sounding speech synthesis using variable-length units.* Tech. rep. Massachusetts Institute of Technology (cit. on pp. x, 51, 132, 133).

[Yoshimura et al. 1999]

Yoshimura, T., K. Tokuda, and T. Masuko (1999). "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis." *On Speech,* (cit. on p. 29).

[Young 1979]

Young, S. (1979). "Speech synthesis from concept: a method for speech output from information systems." *The Journal of the Acoustical Society of America,* 66 (3), pp. 685–695 (cit. on p. 36).

[Zen et al. 2015]

Zen, H. and H. Sak (2015). "Unidirectional Long Short-term Memory Recurrent Neural Network With Recurrent Output Layer For Low-latency Speech Synthesis." *IEEE International Conference on Acoustics, Speech and Signal Processing.* Melbourne, pp. 4470–4474 (cit. on p. 31).

[Zen et al. 2007]

Zen, H., T. Nose, J. Yamagishi, and S. Sako (2007). "The HMM-based speech synthesis system (HTS) version 2.0." *SSW6,* pp. 294–299 (cit. on pp. 28, 30).

Résumé : La synthèse de la parole par corpus (sélection d'unités) est le sujet principal de cette thèse. Tout d'abord, une analyse approfondie et un diagnostic de l'algorithme de sélection d'unités (algorithme de recherche dans le treillis d'unités) sont présentés. L'importance de l'optimalité de la solution est discutée et une nouvelle mise en oeuvre de la sélection basée sur un algorithme A^* est présenté. Trois améliorations de la fonction de coût sont également présentées. La première est une nouvelle façon – dans le coût cible – de minimiser les différences spectrales en sélectionnant des séquences d'unités minimisant un coût moyen au lieu d'unités minimisant chacune un coût cible de manière absolue. Ce coût est testé pour une distance sur la durée phonémique mais peut être appliqué à d'autres distances. Notre deuxième proposition est une fonction de coût cible visant à améliorer l'intonation en se basant sur des coefficients extraits à travers une version généralisée du modèle de Fujisaki. Les paramètres de ces fonctions sont utilisés au sein d'un coût cible. Enfin, notre troisième contribution concerne un système de pénalités visant à améliorer le coût de concaténation. Il pénalise les unités en fonction de classes reposant sur une hiérarchie du degré de risque qu'un artefact de concaténation se produise lors de la concaténation sur un phone de cette classe. Ce système est différent des autres dans la littérature en cela qu'il est tempéré par une fonction floue capable d'adoucir le système de pénalités pour les unités présentant des coûts de concaténation parmi les plus bas de leur distribution.

Mots clés: Synthèse de la parole ; synthèse par corpus ; sélection d'unités ; algorithme de recherche de chemin dans un graphe ; coût cible ; coût de concaténation.

Summary: This PhD thesis focuses on the automatic speech synthesis field, and more specifically on unit selection. A deep analysis and a diagnosis of the unit selection algorithm (lattice search algorithm) is provided. The importance of the solution optimality is discussed and a new unit selection implementation based on a A^* algorithm is presented. Three cost function enhancements are also presented. The first one is a new way – in the target cost – to minimize important spectral differences by selecting sequences of candidate units that minimize a mean cost instead of an absolute one. This cost is tested on a phonemic duration distance but can be applied to others. Our second proposition is a target sub-cost addressing intonation that is based on coefficients extracted through a generalized version of Fujisaki's command-response model. This model features gamma functions modeling F_0 called atoms. Finally, our third contribution concerns a penalty system that aims at enhancing the concatenation cost. It penalizes units in function of classes defining the risk a concatenation artefact occurs when concatenating on a phone of this class. This system is different to others in the literature in that it is tempered by a fuzzy function that allows to soften penalties for units presenting low concatenation costs.

Keywords: Speech synthesis; corpus-based speech synthesis; unit selection; graph pathfinding algorithm; target cost; concatenation cost.