Une synthèse vocale destinée aux déficients visuels

Hélène Collavizza, Jean-Paul Stromboni

Laboratoire I3S, Ecole Polytechnique Universitaire de Nice Sophia Antipolis
helen@polytech.unice.fr , strombon@polytech.unice.fr

ABSTRACT

This paper presents an experiment in developping and testing a text to speech system which is needed for improving some applications dedicated to visually impaired users. When carrying out such applications, it appears that an interface able to speak, i.e. read a text, is mandatory. In order to allow an easy release, this text to speech system should be portable, licence free, using freewares and free solutions.

To fulfill all these needs, a solution was chosen and developped. On the one hand, several experiments were conducted with different such applications. On the other hand, the obtained text to speech system was made available on the Web for evaluation purpose. These two information sources allow to gather a set of advices and knowledge on such tools, and make possible and easier to develop next versions.

1. INTRODUCTION : NOTRE BESOIN

Depuis plusieurs années, l'Ecole Polytechnique Universitaire de Nice Sophia Antipolis organise des journées intitulées Déficients Visuels et Nouvelles Technologies (DeViNT, voir http://www.essi.fr/devint). L'objectif est de mettre en présence des associations, des chercheurs, des entreprises, des organismes de formation spécialisés, et des déficients visuels.

Cette manifestation est l'occasion de développer des applications logicielles dédiées aux déficients visuels à la fois pour les aider à utiliser l'informatique et ses outils, et aussi pour sensibiliser non seulement les élèves mais aussi les enseignants à cette problématique.

Les applications sont diverses, en particulier des jeux, et souvent dédiés aux enfants, d'autant plus que la collaboration avec des établissements spécialisés locaux, tel l'institut d'éducation sensorielle Clément Ader et l'école du Château à Nice est de plus en plus importante. Cette collaboration intervient pour la conception, le développement, le test et la diffusion des résultats.

De cette collaboration comme de nos premières expériences, il résulte que la fonction de synthèse vocale (ou TTS pour text to speech) est essentielle dans ce contexte. L'utilisation de la voix est l'un des moyens de compenser la cécité lorsqu'il faut lire. Par exemple, les jeux de mémoire peuvent être adaptés en demandant de reconnaître des cris d'animaux plutôt que des images. Dans ce cas, il suffit d'enregistrer des sons. Cependant, d'autres applications, par exemple pour l'apprentissage de la lecture, requièrent un TTS pour résoudre le problème d'accessibilité, puisque le texte à lire à voix haute n'est pas prédéfini. La synthèse vocale recherchée doit satisfaire certaines contraintes :

- la langue parlée doit être le français
- la synthèse doit être facilement utilisable sur un maximum d'applications
- les applications doivent être portables
- on doit pouvoir les diffuser gratuitement

Les logiciels les plus utilisés par les aveugles sont les logiciels JAWS et supernova. Ces logiciels intègrent une synthèse vocale qui lit à voix haute le texte de la fenêtre active sur l'environnement de travail. De nombreuses autres synthèses vocales existent comme Acapela, Speechissimo... Or, ces synthèses vocales correspondent mal à l'une ou l'autre des contraintes ci-dessus. D'où la nécessité de construire l'outil adapté pour l'inclure sans obligations ni pénalités dans les interfaces des applications de la journée DeViNT.

Dans les pages suivantes, (1) nous énumérons les choix qui ont dû être faits, (2) nous décrivons la solution obtenue, et (3) nous présentons quelques uns des retours d'expérience des utilisateurs de la synthèse vocale.

2. NOS CHOIX DE DÉVELOPPEMENT

Pour construire un TTS adapté à nos besoins, plusieurs choix doivent être faits, tels le langage de développement, en l'occurence java, pour être utilisé aisément par nos étudiants de première année et pour faciliter la mise en œuvre des interfaces graphiques. En outre, il faut choisir un moteur TTS, capable de lire un texte à voix haute. Ici le moteur MBROLA [1] de l'Université de Mons présente les avantages suivants :

- MBROLA est libre et gratuit
- il peut être piloté en java, sous Windows ou Linux
- il fournit un ensemble de voix françaises
- le projet MBROLA coordonne les expériences et les initiatives d'une communauté scientifique autour de la synthèse vocale, comme le projet EULER [2] ou FipsVox [3], et de l'utilisation du moteur MBROLA.

Nous avons donc choisi d'apporter une contribution à ce projet en développant un TTS français basé sur MBROLA, ce qui n'existait pas alors.

2.1 La chaîne de synthèse MBROLA

Pour créer une synthèse vocale en utilisant MBROLA, il suffit en fait d'écrire un module de transcription texte vers phonèmes. En effet, le processus de synthèse vocale est

scindé en deux blocs, nommés « Natural Language Processing » et « Digital Signal Processing » [4]. Le premier bloc est chargé du traitement d'un texte qu'il faut traduire en une séquence de phonèmes. Le deuxième bloc est l'outil MBROLA lui-même qui prend les phonèmes et d'autres paramètres comme la voix à utiliser, pour lire le texte à voix haute et l'enregistrer dans un fichier audio (au format wav). Un constat important est que MBROLA rassemble et résout les problèmes de Traitement du Signal. Le travail restant, la construction du bloc « Natural Language Processing », est plus motivant pour nos étudiants en sciences informatiques.

2.2 Le bloc « Natural Language Processing»

Le bloc « Natural Language Processing » prend un texte et construit la séquence de phonèmes adéquate à prononcer avec une certaine prosodie. En général, ce bloc est décomposé en trois traitements :

1. le prétraitement qui remplace des éléments de texte particuliers, tels les acronymes, les nombres, les abréviations par un texte in extenso avant la lecture
2. la conversion de texte en phonèmes qui détermine les phonèmes à associer aux groupes de lettres
3. le générateur de prosodie en charge de l'intonation

Les résultats de ce traitement sont livrés dans un fichier de phonèmes au format reconnu par MBROLA.

Il existe différentes implémentations de ce traitement. En particulier, le package FreeTTS [5] qui est écrit en java et en libre distribution. Malheureusement, il ne prononce que l'anglais et n'utilise pas de règles de prononciation paramétrées. Il faut donc réécrire la plus grande part des classes java pour créer un TTS en français. De plus, l'architecture logicielle de FreeTTS est très complexe, ce qui a découragé nos étudiants. Une autre implémentation bien connue est fournie par le projet EULER [2]. Cependant, le logiciel est écrit en C++ ce qui n'est pas adapté à notre première année de cursus ingénieur.

Pour développer notre propre module de traduction en java nous avons décidé d'utiliser les règles de transcription en phonèmes de l'implémentation en PERL proposée par D. Haubensack [1]. Notre objectif étant de compléter cet ensemble de règles et d'améliorer la prosodie jugée insuffisante pour nos desseins.

3. Programmer la synthèse vocale

Nous présentons d'abord le convertisseur de texte en phonèmes puis la façon de générer la prosodie.

3.1 Conversion du texte en phonèmes

Règles de prononciations

Pour traduire un texte en une liste de phonèmes, au moins deux approches peuvent être choisies. Tout d'abord, donner un ensemble de règles de prononciations comme dans [6]. L'avantage est que peu de règles peuvent suffire, mais l'inconvénient est que ces règles peuvent comporter des exceptions. La deuxième approche est de stocker la prononciation de tous les mots français (lemmes et règles d'inflexion) dans un dictionnaire [7]. Le problème est alors de se procurer un tel dictionnaire.

Nous avons décidé de fusionner ces deux approches : nous sommes partis de l'ensemble de règles de prononciation données dans le TTS en PERL [1] et l'avons enrichi en ajoutant un ensemble d'exceptions à ces règles. Règles de prononciation et exceptions sont décrites dans des fichiers de textes à l'aide d'une syntaxe simple et sont aisément modifiables par l'utilisateur qui peut ainsi améliorer la prononciation. Nous utilisons la syntaxe de [8] pour exprimer les règles et les exceptions:

prefixe [[*racine*]] *suffixe -> liste_phonème*

où *prefixe* et *suffixe* sont des expressions régulières. *Prefixe* est le contexte qui a été analysé avant la racine, *suffixe* est le contexte qu'il reste à analyser.

Par exemple, certaines règles de prononciation de “am” sont présentées ci-dessous :

1 [[am]] n -> a m ## amnistie
2 [[am]] m -> a ## programmation
3 [[am]] C -> a~ ## camp

Dans ces règles, C est la classe des consonnes, ## est suivi d'un commentaire qui illustre la règle. Règle 1: *am* suivi de n est prononcé “a m”. Règle 2: *am* suivi de m est prononcé “a” (on enlève simplement un m). Règle 3: *am* suivi d'une consonne est prononcé “an”.

Création de la liste de phonèmes

Pour générer les phonèmes, nous cherchons dans l'ensemble des règles et des exceptions la règle dont la racine a le plus grand préfixe commun avec le mot à transcrire. Ensuite, nous vérifions si les préfixe et suffixe concordent. Par exemple, pour générer les phonèmes du mot “camp”, deux règles ont pour racine “c” qui est un préfixe de “camp” :

6 [[c]] (e|é|è|ê|i) → s ## cesse
7 [[c]] → k

La règle 6 ne peut pas être appliquée car dans “camp” “c” est suivi de “a” (pas e ni é ni è …). Nous appliquons donc la règle 7 et générons le premier phonème **“k”**. Il faut ensuite analyser “amp”. La plus longue règle qui s'applique est la règle 3 de l'exemple précédent et nous générons le phonème **“a~”**. Il faut maintenant traiter la dernière lettre “p” en appliquant la règle 8:

8 [[p]] T →

où T signifie “Terminal”: un “p” à la fin d'un mot n'est pas prononcé. Nous obtenons enfin la liste: “**k a~”**.

Les règles de prononciation et leurs exceptions sont stockées dans la même structure de données et sont traitées en même temps. Voici par exemple des règles et exceptions pour les mots qui commencent par “qua”:

 # règles générales
9 T [[quadr]] → k w a d R ## quadrature

10 T [[quadr]] il → k a d R ## quadrillage

11 [[qu]] → k

 # exceptions

12 T [[quantum]] T → k w a~ t o m

13 T [[quantifier]] T → k w a~ t i f i e

"quadr" est prononcé "k w a" (règle 9) excepté dans les mots de la famille de "quadrillage" (règle 10) comme "quadriller" ou "quadrille". Les autres mots comme "quand", "qualité", sont prononcés "k" avec la règle générale 11. Les règles 12 et 13 sont des exceptions pour les mots de la famille de "quantum" comme "quantifier" qui sont prononcés "k w a" (règles 12, 13).

Quand on analyse le mot "quantum" la règle 12 est utilisée (et pas la règle 11), car c'est celle dont la racine a le plus grand préfixe commun avec "quantum". Les règles de prononciation et les exceptions sont stockées dans des arbres lexicaux qui partagent les préfixes communs. Cette structure de données est peu coûteuse en complexité temporelle et spatiale. Les préfixes partagés sont les racines des règles et les noeuds terminaux contiennent la liste des règles associées à la racine (voir figure 2).

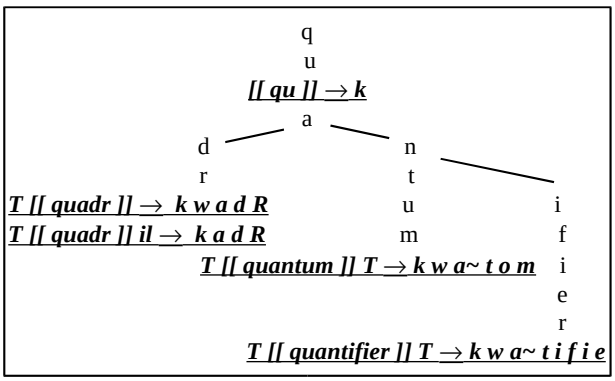

Figure 2: arbre lexical des règles 8, 9, 10, 11, et 12

En utilisant de tels arbres lexicaux, l'algorithme de transcription en phonèmes est tout simplement un parcours en profondeur de l'arbre, avec retour arrière si la règle ne peut pas s'appliquer à cause du suffixe. Les autres paramètres du fichier d'entrée de MBROLA sont générés par le module "prosodie".

3.2 Prosodie

La génération de la prosodie est un problème complexe. Notre objectif étant de concevoir un TTS utilisable dans des applications pour déficients visuels où les textes lus sont courts, nous avons adopté des principes simples. Nous avons affiné la micro prosodie qui est associée aux phonèmes dans l'implémentation en PERL. Par exemple, les phonèmes comme "a~" ou "o~" sont longs tandis que les phonèmes "R", "l", "H" sont courts. Le pitch des occlusives voisées, par exemple les lettres "b", "d" et "g" est choisi grave et la durée des derniers phonèmes des phrases est allongée.

Nous avons aussi ajouté une macro prosodie, grâce à une approche donnée dans [9] (ce choix pragmatique est issu d'anciens travaux d'un collègue enseignant chercheur en traitement du signal). Nous considérons cinq types d'intonation : (A) continuation mineure, (B) continuation majeure, (C) interrogatif, (D) fin de phrase, et (E) mode exclamatif ou impératif. Ces mouvements intonatifs sont définis par des courbes qui s'échelonnent sur quatre niveaux. La distance entre ces niveaux ainsi que les pentes des courbes sont paramétrables. Par exemple, l'équation du type (A) est :

$$y_i = y_1 + 2d + k(i - n)^2$$

Ainsi nous calculons le pitch du $i^{ème}$ phonème (y_i) en fonction de celui du 1^{er} phonème (y_1), du nombre de phonèmes total n et des paramètres d et k.

Pour appliquer ces schémas intonatifs, le texte est découpé en syntagmes qui sont identifiés soit par la ponctuation, soit par les conjonctions. Nous associons un schéma intonatif selon le type de syntagme : le début d'une phrase est de type (A), le milieu de la phrase est de type (B), la fin des phrases est de type (C), (D) ou (E). La durée de pause entre les syntagmes dépend de la ponctuation ou du type de conjonction. Par exemple, les conjonctions de subordination comme "bien que" sont traitées avec des pauses longues. La figure 3 donne un exemple d'une phrase et de sa prosodie.

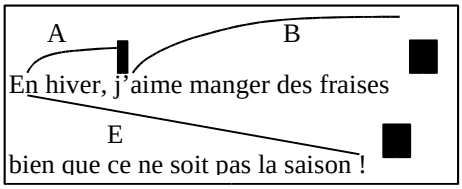

Figure 3: exemple de prosodie (rectangle égale pause)

Malgré la simplicité des principes mis en œuvre pour la prosodie, les aveugles qui ont testé nos logiciels ont trouvé la qualité du TTS suffisante.

4. APPLICATIONS

Cette synthèse vocale est expérimentée depuis deux ans pour les applications de la journée DeViNT. D'autre part, elle est accessible sur http://kaivalyam.essi.fr (et référencée sur le site du projet MBROLA [1]) où elle a donné lieu à un ensemble de contacts, de suggestions et de retours de la part d'internautes et utilisateurs potentiels.

4.1. Les applications de la journée DeViNT

Les applications développées pour la journée DeViNT sont diverses, comme on le devine dans la liste non exhaustive présentée ci-dessous. Il s'agit souvent de jeux, destinés à des enfants de l'école primaire ou du collège. La synthèse vocale y joue un rôle fondamental pour l'accessibilité :

- CB2S : un TTS est associé à un lecteur de code barre. Ainsi les déficients visuels peuvent se faire lire toutes les informations sur les produits à partir d'un dispositif portable (tel PDA) qui interroge le serveur de synthèse vocale cité ci-dessus.
- Quinze questions : il s'agit d'un jeu de questions à

choix multiples, inspiré de « Qui veut gagner des millions ». Le TTS lit les questions et les réponses.

- Lecteur MP3 : l'objectif est de rendre accessible la musique MP3. Le TTS lit le titre des chansons.
- SI boud'chou : s'inspire du jeu ADIBOU pour apprendre le calcul et l'orthographe. Le TTS doit y lire des nombres (jeux de calcul) ou des mots (dictée).
- Installor : cet outil indispensable guide les déficients visuels lors de l'installation des logiciels DeViNT sur leurs propres ordinateurs.

4.2 Quelques retours d'expérience de DeViNT

Afin d'évaluer l'ergonomie de leurs logiciels, nos étudiants se sont rendus à l'institut Clément Ader à Nice présenter leurs logiciels en notant toutes les réactions des enfants. Ils ont ainsi récolté un ensemble d'informations pertinentes sur l'utilisation du son :

- il est souhaitable de développer une utilisation partagée de sons enregistrés et de synthèse vocale. Les enregistrements plus vivants sont utilisés pour les textes courts et répétitifs connus d'avance (par exemple, le texte des menus, les messages d'aide). Le TTS est utilisé pour les messages dépendants du contexte, ou qui sont inconnus a priori
- L'intonation monotone et machinale de la synthèse est jugée irritante à la longue, par les utilisateurs voyants. Mais il s'avère d'après certains utilisateurs non voyants que les voix plus performantes, telles 'Claire' ou 'Julie' ne sont pas toujours bien reçues pour d'autres raisons, en particulier le ton trop chantant qui est fatiguant. D'ailleurs, les aveugles choisissent souvent une grande rapidité d'élocution pour briser la monotonie de ces voix synthétiques quand ils utilisent JAWS
- La clarté de la prononciation laisse parfois à désirer dans notre synthèse, sans pouvoir trancher s'il s'agit de la qualité de la base de diphones utilisée, ou la technique 'overlap and add' utilisée dans MBROLA pour accoler les diphones et synthétiser la voix
- Les règles de prononciation utilisées pour la traduction en sont parfois insuffisantes. Il ne suffit pas toujours de compléter le fichier de règles : il faudrait dans certains cas utiliser un algorithme d'analyse syntaxique et pas seulement lexicale [3]

4.3 Les retours de : http://kaivalyam.essi.fr

Le TTS est accessible sur l'URL ci-dessus où il est possible de proposer un texte écrit et de récupérer le fichier audio correspondant. Le site permet aussi de contacter les auteurs pour faire des suggestions ou demander les sources java de la synthèse. Ce site a été maintes fois consulté, utilisé, et la synthèse demandée par des utilisateurs handicapés ou non, par des chercheurs, des internautes … et pas toujours pour résoudre la déficience visuelle. Citons par exemple un internaute privé temporairement de voix qui a utilisé notre synthèse pour communiquer par mail avec un enfant qui ne sait pas lire. Un autre exemple est l'utilisation dans le cadre du développement de sites citoyens pour le département du Val d'Oise, où une fonctionnalité de synthèse de la parole a été ajoutée pour donner un meilleur accès aux mal-voyants. Une autre demande a été faite par un élève de l'ENAC (Ecole Nationale de l'Aviation Civile à Toulouse) pour développer en java un simulateur de la gestion du traffic aérien, afin de préparer les étudiants aux épreuves pratiques.

5. Conclusion et perspectives

La synthèse vocale que nous avons présentée répond à nos objectifs initiaux : elle est gratuite, ouverte, portable, écrite en java, elle parle français. Elle a été intégrée dans plusieurs logiciels de jeux diffusés sur CDrom lors de la journée DeViNT. Bien sûr la qualité de cette synthèse vocale pourrait être améliorée, en particulier la prosodie. N'étant pas spécialistes en la matière, nos efforts vont plutôt porter vers l'amélioration de l'ergonomie des applications dédiées aux déficients visuels. Tout d'abord en améliorant la collaboration entre les sons enregistrés et l'utilisation de synthèses vocales. Nous aimerions également réaliser une API permettant d'utiliser plusieurs synthèses vocales. L'idée est que nos étudiants puissent utiliser de façon transparente notre propre TTS (pour pouvoir le distribuer avec les jeux) mais également une ou des synthèses vocales commerciales de qualité qui seraient utilisées pour les démonstrations des journées DeViNT. Il nous faut aussi affiner l'analyse des retours d'expérience des utilisateurs déficients visuels.

Bibliographie

[1] T. Dutoit & all, "The MBROLA project: Towards a Freely Available Multilingual Speech Synthesizer"
http://tcts.fpms.ac.be/synthesis/mbrola.html

[2] M. Bagein, T. Dutoit, F. Malfrère, A. Ruelle, N. Tounsi, D. Wynsberghe, *« The EULER Project, An Open, Generic, Multi-lingual and Multi-platform Text-To-Speech system", ProRISC'2000*

[3] J. P. Goldman, A. Gaudinat, L. Nerima, E. Wehrli, "FipsVox : a french TTS based on a syntactic parser", *4th ISCA int. Workshop on speech synthesis, 2001*

[4] T. Dutoit, *"An Introduction to Text-To-Speech Synthesis"*, Kluwer Academic Publishers, Dordrecht, 320 pp., ISBN 0-7923-4498-7, 1997

[5] W. Walker, P. Lamere, P. Kwok, "FreeTTS 1.2 - A speech synthesizer written entirely in the Java™ programming language", *Sun Microsystems Laboratories, http://freetts.sourceforge.net*

[6] E. Keller, "Simplification of TTS architecture vs operational quality", *EUROSPEECH 1997*.

[7] F. Yvon, C. D'Alessandro, V. Aubergé, P. Boula de Mareüil, "Ressource standard pour le français : un large lexique orthographique-phonétique", *XXIIIèmes journées d'Etudes sur la parole, Aussois 2000.*

[8] N. Tounsi, R. Beaufort, « MLRR v1.1, The multi-Layer Rewriting rules parser », *TCTS, Faculté poly. de Mons http://tcts.fpms.ac.be/synthesis/mlrr/mlrr.html, Jul. 2001.*

[9] H. Fervers, J. Leroux, L. Miclet, « Programme de transcription phonémique en langue française », *Laboratoire de théorie des systèmes, Rapport ENST D-76003, 1977*