Conservatoire National des Arts et Métiers, Paris
Informatique, Réseaux, Systèmes et Multimédia

Interface utilisant la parole : Apprentissage de la prononciation grâce à la reconnaissance vocale

Projet de valeur C en Conception d'application multimédia

Thomas JGENTI

Voxpel

Septembre 2005

Table de matières

Introduction

Le but de ce projet et de concevoir une application Java capable, à l'aide de l'API JavaSpeech, de traiter la parole en plusieurs langues et d'aider l'utilisateur à améliorer sa prononciation des mots. L'enjeu du projet était de comparer les implémentations de JavaSpeech existantes, d'étudier la possibilité d'intégration de plusieurs langues et de se documenter sur les autres standards, tels que VoiceXML.

La première partie de ce document porte un aspect général. Elle expose les différentes techniques de synthèse et de reconnaissance de la parole en donnant un historique des développements dans le domaine. Les solutions logicielles disponibles sont également passées en revue, ainsi que les standards en matière de développement logiciel utilisant la parole.

La seconde partie concerne le déroulement du projet. Les spécifications fonctionnelles, le choix des technologies employées suite à une étude de faisabilité. La conception et les étapes de réalisation en soulignant les problèmes rencontrés et les solutions apportées.

La troisième partie présente le résultat du projet : une application fonctionnelle, qui porte le nom de code « Voxpel ». Elle apporte également une conclusion sur l'interactivité obtenue grâce au traitement de la parole.

Techniques de la synthèse et la reconnaissance vocale

Synthèse vocale

Définition

La synthèse vocale est le procédé qui permet de transformer un texte écrit en parole générée par la machine. A ce titre elle est également désignée par le terme anglais *Text To Speech* (TTS), car au-delà de la simple synthèse, il s'agit également de l'analyse syntaxique et sémantique du texte en tenant compte des spécificités linguistiques.

Historique

Bien que la première tentative de produire les sons similaires à la voix humaine, grâce à un système acoustique, date du XVIII siècle, la première machine à générer la parole intelligible a vu le jour au début du XX siècle. Il s'agissait du synthétiseur électronique analogique VODER issu de la recherche des laboratoires Bell aux Etats-Unis, destiné à reconstituer la voix véhiculée par le réseau téléphonique. Ce système, plus connu sous le nom de vocodeur (du anglais *Vocoder*, ou *Voice Coder*), est basé sur la reconstitution du spectre de la voix humaine enregistrée par modulation d'un signal sonore synthétique. Il est utilisé de nos jours, principalement comme instrument de musique électronique, très populaire dans les années 1970.

Les synthétiseurs vocaux, tels que nous les connaissons aujourd'hui, ont commencé à voir le jour dans les années 70, avec les progrès de l'électronique numérique et l'arrivée des moyens informatiques performants. Les premières approches consistaient à modéliser l'appareil vocal humain et à synthétiser les *formants* la voix en appliquant les paramètres du modèle, tels que la fréquence fondamentale ou la résonance, à un signal sonore généré, de manière similaire à un vocodeur. Cette méthode permet de produire une voix intelligible, mais aussi très mécanique, ce qui ne lui empêche pas à être utilisée avec succès. Le jeu « Speak & Spell » (« La dictée magique » pour la version française) développé en 1978 par la société américaine Texas Instruments a été le premier produit basé sur la synthèse vocale accueilli avec succès par le grand public.

Dans les années 1980 puis 1990 l'augmentation des capacités des composants mémoire ont permis une autre approche : utilisation d'échantillons enregistrés et numérisés. Comme dans les synthétiseurs musicaux, cette approche permet de rendre les sonorités produites plus naturelles et dans le cas de la voix, plus humaines. Cette méthode de la synthèse vocale consiste dès lors à trouver le jeu d'échantillons élémentaires qui composent la parole et de les concaténer afin de former des mots.

Principe de fonctionnement

La synthèse vocale par échantillons élémentaires, que l'on appelle *phonèmes*, est la plus répandue actuellement. Chaque langue comporte une collection de phonèmes différente. Ainsi en français on trouve 34 phonèmes contre 40 en anglais ou 44 en allemand. De plus chaque langue comporte des combinaisons de phonèmes spécifiques, appelées *diphones*.

La chaîne du procédé TTS par phonèmes est constituée d'étapes suivantes :

1. Analyse du texte :
    - Analyse sémantique : décodage des abréviations en fonction du contexte, recherche des mots *homographes hétérophones* (mots qui s'écrivent de la même manière, mais se prononcent différemment selon le contexte) ;
    - Analyse syntaxique : construction de suites de phonèmes à partir de la syntaxe des mots, attribution des accents et de l'intonation (*la prosodie*) ;
2. Préparation des échantillons : choix des diphones en application des paramètres de la prosodie ;
3. Synthèse : concaténation d'échantillons de phonèmes (ou de diphones) avec lissage et filtrage suivant la prosodie ;

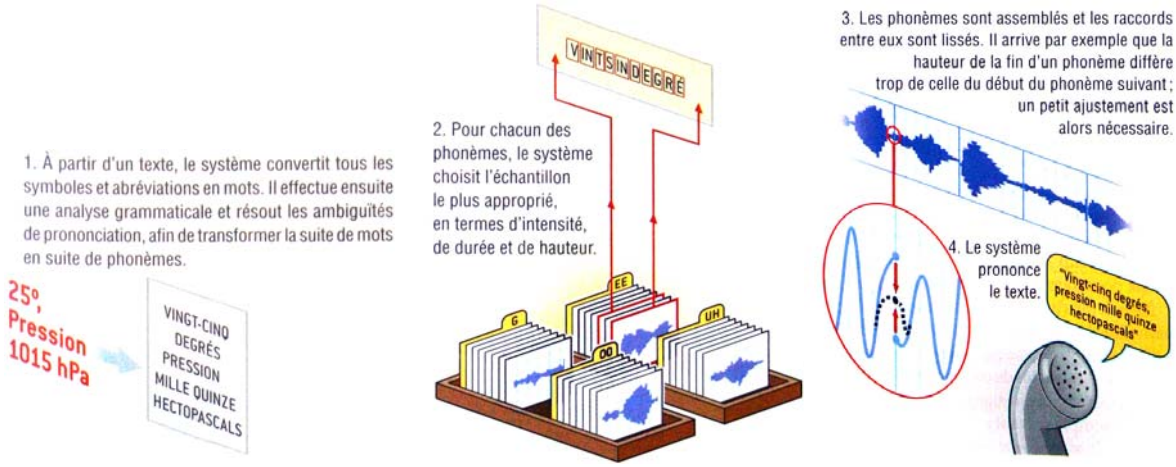

Figure 1 - Les étapes de la synthèse vocale

Bien que cette approche soit destinée à générer une voix plus naturelle, celle-ci reste néanmoins encore sensiblement mécanique et les progrès à faire dans l'application de la prosodie restent primordiaux afin de lui donner toutes les nuances de la vraie parole humaine.

Reconnaissance vocale

Définition

La reconnaissance vocale est le procédé qui permet d'analyser la parole humaine en identifiant automatiquement les mots et les phrases. Par opposition à la synthèse vocale, celle-ci doit permettre de transformer la parole en texte écrit. Le terme consacré à la reconnaissance vocale en anglais est généralement *Automatic Speech Recognition* (ASR).

Historique

Les développements des méthodes de la reconnaissance vocale ont été effectués dans le sillage de ceux de la synthèse. De nombreux aspects théoriques, tels que les formants ou les phonèmes, se retrouvent dans les deux domaines. Mais si les résultats de la synthèse furent encourageants dès les débuts, la mise en place de la reconnaissance, fut quant à elle, plus longue à aboutir.

A la fin des années 1940 un projet militaire américain constitue une première tentative de reconnaissance automatique de la parole, qui n'a toutefois pas donné de résultats probants. D'autres systèmes ont suivi dans les années 1950 avec des systèmes analogiques, puis numériques des 1960. Deux approches étaient déjà employées : reconnaissance mot à mot et reconnaissance des phonèmes.

Les premières applications de reconnaissance vocale, très rudimentaires, ont été commercialisées dans les années 1970. Ces applications étaient très limitées dans le nombre des mots reconnus et nécessitaient des équipements informatiques conséquents. Dans les années 1980, des circuits intégrés spécialisés ont vu le jour, mais ce n'est que dans les années 1990 que les applications de reconnaissance vocale ont réellement atteint le grand public grâce notamment à des procédés de reconnaissance de parole continue qui pouvaient être utilisés sur des ordinateurs personnels.

Principe de fonctionnement

Les méthodes de reconnaissance vocale se situent à plusieurs niveaux : détection de mots-clés, de syllabes ou de phonèmes.

La première méthode, qui est également la plus ancienne, permet d'identifier un nombre très restreint de mots-clés ou de commandes. Elle est basée sur la comparaison des signatures spectrales du mot prononcé avec les mots de référence. Par conséquent cette méthode nécessite la constitution d'un vocabulaire préenregistré. De plus elle est très sensible au changement du locuteur et nécessite une phase d'apprentissage d'autant plus longue que le vocabulaire est étendu. Cette méthode est néanmoins utilisée de nos jours pour les applications telles que les répertoires à commande vocale des téléphones portables, pour lesquelles elle reste suffisamment simple et efficace.

Les méthodes suivantes sont plus flexibles, mais également plus complexes. Comme pour la synthèse, de nos jours la plupart des systèmes de reconnaissance vocale utilisent les phonèmes. L'analyse spectrale permet d'identifier les formants qui composent les phonèmes et, ainsi, à interpréter les sons des mots prononcés. Cependant, comme dans les systèmes de TTS, le passage entre les phonèmes et les lettres d'un texte est une tâche complexe. Un

système de reconnaissance vocale doit tenir compte des spécificités de la langue pour laquelle il est conçu. Il doit posséder un dictionnaire de cette langue, ainsi que les règles grammaticales afin de retranscrire un texte qui est prononcé. De plus une analyse sémantique doit permettre de corriger des mots déjà traités en fonction du contexte de la suite de la phrase. Enfin une difficulté supplémentaire consiste à séparer les mots dans la reconnaissance de la parole continue.

Toutes ces considérations rendent la mise en œuvre des systèmes de reconnaissance vocale bien plus compliquée que celle de la synthèse. Mais dans les deux cas ce sont le « naturel humain » de la parole qui pose le plus grand problème à résoudre.

Technologies actuelles

Systèmes commercialisés

Aujourd'hui de nombreuses sociétés proposent une large panoplie des systèmes de traitement de la parole. Aussi les acteurs majeurs du domaine, tels que IBM, offrent des systèmes qui, pour la plupart, combinent la synthèse et la reconnaissance vocale. Cependant il existe de très nombreuses sociétés qui proposent des systèmes de synthèse vocale seule.

Ainsi les moteurs de traitement de la parole les plus connus et répandus sont *ViaVoice* d'IBM et *Dragon Naturally Speaking* développé par *Dragon Systems*. Ces moteurs s'accompagnent de logiciels de dictée continue destinés au grand public, mais il existe d'autres moteurs, ceux issu de la recherche et développement de *AT&T Bell Labs, SpeechWorks, Lernout & Hauspie* et *Babel Technologies*. A noter que Bell Labs fait figure de pionnier en la matière, car c'est à eux que l'on doit le premier synthétiseur vocal.

Suite à de récents bouleversements économiques parmi les sociétés éditeurs de ces solutions, les majors du domaine sont désormais les compagnies suivantes : IBM, *ScanSoft*, *Lucent Technologies* et *AT&T*. Ainsi *Lucent Bell Labs* et AT&T qui sont tous deux des sociétés de télécommunications, ce qui n'est pas un hasard, étant donné l'importance du traitement de la parole dans ce domaine, profitent-ils naturellement des recherches de AT&T Bell Labs. Alors que la société ScanSoft (anciennement SpeechWorks) se positionne en tant que fournisseur majeur de solutions du traitement vocal. A ce titre elle propose notamment dans son catalogue la solution ViaVoice d'IBM, mais également les solutions des sociétés absorbées au début des années 2000. Ainsi le logiciel de reconnaissance vocale *Dragon Naturally Speaking* de *Dragon Systems*, bien connu du grand public, est désormais propriété de ScanSoft. De même la société belge *Lernout & Hauspie*, un des leaders en reconnaissance vocale des années 1990, est récupérée par ScanSoft après avoir été déclarée en faillite en 2001. A noter également que la société américaine Babel Technologies est devenu *Acapela*.

Un autre acteur majeur à part est la société Microsoft, qui fournit SAPI (Speech API) - l'outil de développement de traitement de la parole qui se veut être un standard sur la plateforme Windows. Enfin, la société Philips commercialise également des outils de développement de reconnaissance vocale – Speech SDK et SpeechMagic SDK.

Systèmes ouverts et logiciels libres

A coté des sociétés qui commercialisent leurs solutions, il existe de nombreux outils issus de la recherche universitaire. Les plus connus sont des moteurs de synthèse *Festival* de l'université d'Edinburgh et *MBROLA* de la faculté polytechnique de Mons en Belgique. Ce dernier propose un outil libre et performant de synthèse par diphones en plusieurs langues, mais ne dispose pas d'outil de TTS complet pour ces langues. Multitel – une société *spin-off* de cette faculté poursuit les recherches dans ce domaine et commercialise des outils de TTS adéquats.

Aux Etats-Unis l'université Carnegie Mellon (CMU) développe également des projets open source, dont *CMU Sphinx* – moteur de reconnaissance vocale, une API en langage C et *FestVox* – logiciel de création de voix artificielles, utilisées notamment par Festival.

Récapitulatif

La liste ci-dessous présente un récapitulatif des solutions logicielles citées. Cette liste n'est absolument pas exhaustive.

Editeur	***Nom du système***	***TTS***	***ASR***[1]	***Langues***	***Commentaires***
IBM	ViaVoice	OUI	OUI	Multiples	
ScanSoft	Dragon Naturally Speaking	-	OUI	Multiples	Logiciel grand public, anciennement produit de Dragon Systems
ScanSoft	Open Speech	-	OUI	Très grand nombre	Solution industrielle issue de SpeechWorks
Lucent		OUI	OUI	Multiples	Solution industrielle pour équipements téléphoniques
AT&T		OUI	OUI	Multiples	Idem
Acapela		OUI	-	Multiples	Solutions industrielles et grand public
ScanSoft	VoCon	-	OUI	Multiples	API de reconnaissance vocale embarquée ou destinée aux jeux vidéos. Technologie SpeechWorks.
Philips	Speech SDK	-	OUI	Multiples	API de reconnaissance vocale
Microsoft	SAPI 4 et 5	OUI	OUI	Multiples	API C++ standard pour la plateforme Windows, utilisée par l'API MSAgent. Le moteur TTS anglais est présent par défaut sur les systèmes Windows.
Carnegie Mellon University	Sphinx Flite	OUI Flite	OUI	Anglais	API de ASR, libre et multiplateforme en langage C. Flite (Festival Lite) est un moteur de TTS léger.
Université d'Edinburgh	Festival	OUI	-	Anglais / Espagnol	API C++ de TTS. Initialement développée pour les plateformes UNIX
Faculté Polytechnique de Mons	MBROLA	Synth. uniquement	-	Multiples	Moteur multiplateforme de la synthèse vocale par diphones..

[1] ASR – Automatic Speech Recognition, ou reconnaissance de la parole automatique

	FreeTTS	OUI	-	Anglais US	Implémentation open source de JSAPI, entièrement écrit en Java.

Interfaces de programmation standard

Speech API

Speech API, ou SAPI, est le SDK de traitement de la parole de Microsoft. En plus d'un moteur propriétaire, SAPI propose une interface C++/C# standardisée qui permet d'utiliser les moteurs d'autres éditeurs compatibles avec cette interface. Il existe deux versions d'interface SAPI : 4 et 5.1. De nombreux logiciels, tels que ViaVoice ou Dragon Naturally Speaking sont compatibles avec SAPI. Il est par conséquent possible d'utiliser leurs moteurs par un programme C++/C# prévu pour la plateforme cible Windows.

Voici une liste de principales implémentations de SAPI :

- Microsoft SAPI 5.1 ;
- IBM ViaVoice SDK (version 9) ;
- Dragon Naturally Speaking ;
- Infovox TTS (anciennement filiale de Babel Technologies) ;
- Philips Speech SDK (version < 3) ;

JavaSpeech API

Le standard JavaSpeech, dont la première spécification date de 1998, est une initiative de Sun Microsystems afin de doter le langage Java d'une interface universelle de programmation du traitement de la parole. Sun Microsystems ne fournit pas d'implémentation de ce standard, mais s'appuie sur les moteurs tiers. En effet la norme fut élaborée avec la participation de *Apple Computer, Inc., AT&T, Dragon Systems, Inc., IBM Corporation, Novell, Inc., Philips Speech Processing, and Texas Instruments Inc.*

Après la sortie de la spécification 1.0 de JavaSpeech, plusieurs implémentations ont vu le jour, dont celle d'IBM (IBMJS) fonctionnant avec les bibliothèques de ViaVoice, mais également l'implémentation de TTS de Lernout & Hauspie, celle de Festival et enfin FreeTTS, une implémentation du moteur de synthèse entièrement écrite en Java, projet open source initié par les développeur de Sun Microsystems et se basant sur le logiciel Flite de Carnegie Mellon University. La liste suivante donne les principales implémentations de JavaSpeech :

- FreeTTS ;
- IBM *IBMJS* (n'est plus disponible);
- Lernout & Hauspie TTS (n'est plus disponible) ;
- Festival ;
- Cloud Garden *TalkingJava SDK* (n'est pas une vraie implémentation, mais interface entre JSAPI et MS SAPI) ;
- Conversay *Conversa Web* (n'est plus disponible) ;

On remarque que trois des six implémentations ci-dessus ne sont plus disponibles. Parmi celles qui existent actuellement, 2 sont des moteurs de TTS uniquement et TalkingJava n'est pas réellement une implémentation, mais une adaptation des moteurs compatibles SAPI et est donc très dépendante des volontés des éditeurs. Lorsqu'on sait que Philips Speech SDK ne supporte plus SAPI, on peut redouter la même chose chez IBM et ScanSoft. De plus CloudGarden (une petite société américaine – une initiative individuelle) ne fait plus évoluer la bibliothèque depuis deux ans, les sources restant non disponibles.

On peut expliquer cet abandon massif par les éditeurs logiciels, d'une part par la disparition des sociétés L&H et Dragon Systems, et d'autre part par une brouille entre Sun et IBM (fait relaté). Une spécification 2 de JavaSpeech devait démarrer en 2001 avec l'initiative de Sun et de la société américaine Conversay (comme en témoigne un communiqué de presse publié par Conversay). Cependant ces efforts ne semblent pas avoir poursuivi. Mais malgré cet état de fait, JavaSpeech API 1.0 reste un standard complet et abouti.

VoiceXML

VoiceXML est une définition de document (DTD) au format XML destiné à la création d'interfaces utilisateur vocales. Standard initié par AT&T, IBM, Lucent et Motorola, la première version de VoiceXML a été approuvée par le consortium W3C en 2000. Aujourd'hui la version 2.1 de cette norme est publiée et la version 3 est en préparation. Orienté avant tout vers le domaine de téléphonie, VoiceXML est supporté par un très grand nombre d'industriels d'informations et télécommunications, parmi lesquels on trouve France Telecom, Nortel Networks et Cisco Systems, aussi bien que la branche SpeechWorks de ScanSoft.

Le domaine visé par VoiceXML porte le nom de *Interactive Voice Response* (IVR) ou de *Serveur Vocal Interactif* en français. Pour simplifier, il s'agit d'un système de dialogue automatisé entre un humain (appelant) et un centre d'appel, par exemple. VoiceXML, à l'instar d'une page HTML, définit des menus qui, au lieu d'être visualisés, sont prononcés via un système de TTS. L'appelant a la possibilité de faire une sélection dans ce menu soit par la parole, soit par une touche à tonalité du téléphone. Le moteur de reconnaissance vocale associé au browser VXML interprète la sélection et transfère l'appelant soit vers un autre document VXML, soit vers un numéro donné, à l'image des liens hypertexte de HTML. La figure 2 illustre la mise en place d'un portail VXML entre le réseau téléphonique commuté (PSTN) et Internet.

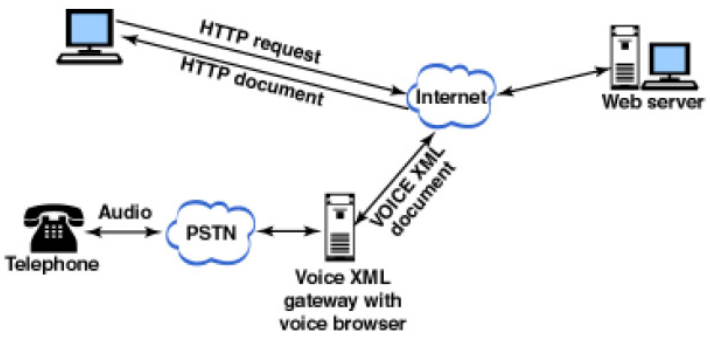

Figure 2 - Exemple de mise en oeuvre de portail VoiceXML

L'exemple suivant est extrait d'un tutorial VoiceXML d'IBM. Il a été simplifié et illustre la syntaxe d'un fichier VoiceXML. Dans cet exemple l'utilisateur entend une demande de prononcer le mot d'un destinataire de son email.

```
<?xml version="1.0"?>
<vxml version="1.0">
<form id="email" >
      <block>
            <field name="sendto">
            <grammar>
                  <![CDATA[
                  [
                        john {<sendto "john">}
                        vivek {<sendto "vivek">}
                  ]
                  ]]>
            </grammar>
            <prompt>
                  <audio>Who do you want to send an email to?</audio>
            </prompt>
            <help>
                  Say the name of the person you want to send an email to.
            </help>
      <filled>
            <if cond="sendto == 'john'">
                  <assign name="emailid" expr="'john@yahoo.com'"/>
            <else/>
                  <assign name="emailid" expr="'vmalhot@yahoo.com'"/>
            </if>
      </filled>
      </field>
      </block>
</form>
</vxml>
```

Le standard VoiceXML ne propose de pas de SDK classique, comme c'est le cas de JSAPI, mais fonctionne avec une approche distribuée, où le client se trouve à distance du serveur vocal qui traite la parole (figure ci-dessous).

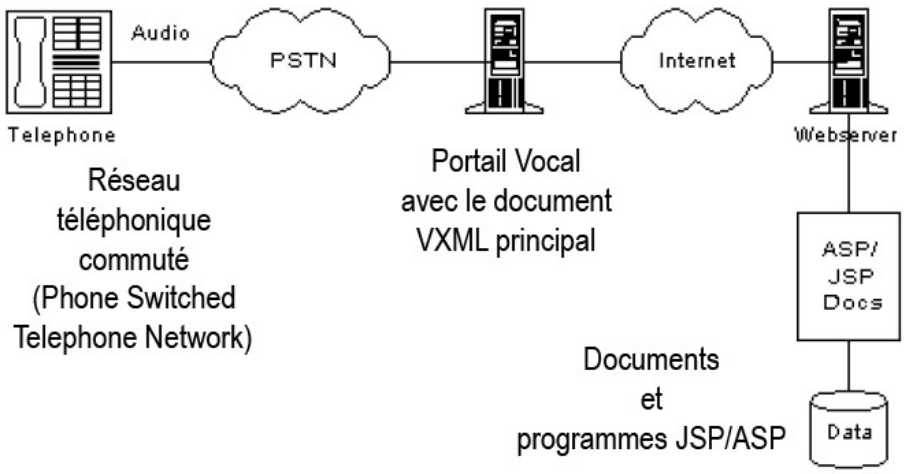

Figure 3 - Architecture d'une application VoiceXML

Il existe de nombreuses implémentations de serveurs (browsers) vocaux compatibles VoiceXML, qui, en majeure partie, restent des technologies fermées. Il existe cependant un

interpréteur VXML open source : *OpenVXI*, proposé par la société américaine *Vocalocity* sous licence GPL. Cependant il ne s'agit pas d'un browser VXML complet. En général la mise en place d'une application de traitement de la parole exige une infrastructure complexe impliquant au minimum : un browser VXML, un serveur HTTP avec JSP ou ASP. Une base de données est la plupart du temps indispensable. Ainsi l'application doit être écrite en VoiceXML avec des commandes JSP ou ASP et repartie sur le serveur vocal est le serveur HTTP.

Force est de constater que bien que la norme VoiceXML soit très en vogue, elle reste très spécifique au domaine de la téléphonie et des réseaux, ce qui est complémentaire de l'API JavaSpeech, et ne la remplace pas.

Réalisation du projet

Spécifications fonctionnelles

Le projet à pour but la conception et la réalisation d'une application Java capable de traiter la parole en plusieurs langues et d'aider l'utilisateur à améliorer sa prononciation des mots. Les caractéristiques requises sont les suivantes :

Langage de programmation	Java
Langues de la parole	Français et anglais
Plateforme cible	Windows
Interfaces utilisateur	Graphique et vocale

Le logiciel doit proposer à l'utilisateurs plusieurs types de leçon d'apprentissage de la prononciation. Ces leçons doivent proposer des mots ou des phrases que l'utilisateur doit prononcer correctement. L'ordinateur doit aider l'utilisateur dans cette tache en proposant la prononciation correcte et détecter si oui ou non le mot ou la phrase proposée est bien prononcée. Dans le cadre d'une interface multi langues, l'application doit également proposer une traduction français - anglais / anglais - français des mots choisis. Ceci dans le but d'apprentissage de langues interactif. L'interface de l'application doit supporter les commandes vocales en plus des commandes classiques (écran, clavier, souris) et doit fournir les indications verbales à l'utilisateur en deux langues.

Exemple :

Ordinateur écrit et prononce un mot : « UNE POMME »

L'utilisateur doit répéter correctement le mot dans le micro, jusqu'à ce que la prononciation soit correctement détectée par l'ordinateur. Celui-ci propose alors le mot suivant.

L'application peut être vue comme un jeu interactif à l'image du célèbre jeu « *Speak & Spell* » (« *la dictée magique* »), mais fonctionnant dans le sens contraire, d'où son nom légèrement semblable : *Voxpel*. Les leçons doivent s'inspirer de la méthode « *Assimil* » ou des logiciels éducatifs tels que : *« Talk to me »* ou *« Tell me more »*

Etude de faisabilité et choix technologiques

L'utilisation du langage Java impose les alternatives suivantes :
Utilisation de l'API JavaSpeech ;
Utilisation de la technologie VoiceXML ;
Ecriture d'une interface JNI vers une bibliothèque native de traitement de la parole (ex : SAPI de Microsoft)

La dernière alternative équivaut pratiquement à l'écriture d'une re-implémentation entière de JSAPI, tache dont la complexité sort du contexte d'un projet de valeur C. Elle a donc été écartée au profit des deux premières.

JSAPI vs VoiceXML

Après étude des outils disponibles, la technologie JavaSpeech s'est révélée plus accessible est appropriée à la réalisation du projet. En effet VoiceXML exige des solutions logicielles lourdes et une infrastructure à mettre en place qui n'est pas justifiée par rapport aux spécifications du prototype à réaliser. Au moment du démarrage du projet (printemps 2005) l'implémentation de JSAPI d'IBM était encore disponible librement et celle-ci fut utilisée pour les premiers essais de faisabilité.

Tests de IBMJS

Un premier programme fut réalisé avec IBMJS et les bibliothèques de reconnaissance et de synthèse de ViaVoice, version française. Ces essais étaient fructueux et un autre essai a été fait avec une version différente (plus ancienne) de ViaVoice en anglais. Bien que ces deux essais furent réussis indépendamment, les deux versions des bibliothèques d'IBM n'ont pas pu être utilisées en même temps à cause de l'incompatibilité entre leurs versions.

Autres implémentations de JSAPI

L'étape suivante furent les tests d'autres implémentations de JSAPI : FreeTTS et TalkingJava. FreeTTS a fonctionné correctement, mais n'offrait que la synthèse en anglais. TalkingJava fut essayé durant la période d'essai de 30 jours avec les bibliothèques du logiciel Dragon Naturally Speaking, version US. Ces essais ont montré une certaine instabilité de TalkingJava, mais surtout il fut impossible de faire fonctionner la reconnaissance vocale de Dragon. De plus le moteur de ViaVoice fut substitué par TalkingJava par celui de Microsoft, SAPI. La période d'essai de 30 jours n'a pas suffi à résoudre ces problèmes et étant donné l'absence de support ou de sources de TalkingJava, cette implémentation fut abandonnée au profit de FreeTTS.

Choix de configuration et tests supplémentaires

La configuration finale retenue : IBMJS + ViaVoice français et FreeTTS anglais a été testée avec succès. Cependant la qualité de la synthèse de FreeTTS étant très médiocre, des essais supplémentaires ont été réalisés afin d'utiliser le synthétiseur vocal MBROLA avec le moteur TTS de FreeTTS. Bien que documentée, cette extension a posé quelques problèmes de mise en œuvre. MBROLA étant un logiciel UNIX à la base, son moteur est l'exécutable **mbrola** sans extension particulière. Or la distribution Windows de MBROLA n'inclut pas ce moteur batch en tant que tel, mais un outil ayant une interface graphique *mbroli.exe*. Cet outil n'est pas utilisable avec FreeTTS et il a fallu rechercher une version batch de mbrola spécialement compilée pour Win32. De plus pour que FreeTTS trouve l'exécutable il faut impérativement enlever l'extension .exe ! C'est ainsi qu'à la fin les essais se sont révélés fructueux et FreeTTS a pu bénéficier de 3 voix américaines de meilleure qualité.

Ainsi la reconnaissance vocale ne pouvait être réalisée qu'en français en l'absence d'outils librement disponibles, la synthèse, quant à elle, fonctionnant en deux langues.

Conception

Architecture logicielle

Les objets de l'application Voxpel se divisent en trois packages principaux :

- Traitement de la parole : **`fr.cnam.voxpel.speech`** ;
- Interface graphique : **`fr.cnam.voxpel.ui`** ;
- Gestion du contenu des leçons : **`fr.cnam.voxpel.lesson`** ;

Les packages supplémentaires : **speech.ibm** et **speech.freetts** contiennent des programmes de test particulières pour chacune des implémentations et ne sont pas utilisés par l'application.

Les classes principales sont :

- **ApplicationControl** – interface graphique principale, contient la fonction main. Il s'agit d'un singleton ;
- **VoiceManager** – facilite l'utilisation de JavaSpeech en encapsulant les objets Synthesizer et Recognizer appropriés à l'application. Gère également les voix et les grammaires ;
- **LessonManager** – classe abstraite définissant une interface unique aux leçons. Chaque leçon doit étendre cette classe et implémenter les méthodes abstraites selon son contenu ;
- **LessonPanel** – classe de base des interfaces graphiques et vocales des leçons. Contient une instance de LessonManager appropriée afin de gérer les actions de l'utilisateur ;
- **SpeakingTutor** – à l'origine une classe utilitaire qui donne des instructions vocales à l'utilisateur ;

Cette architecture offre une bonne séparation entre les parties JavaSpeech, Interfaces et paramètres de l'application. De plus le niveau d'abstraction qu'offrent les classes LessonManager et LessonPanel permet d'intégrer facilement d'autres types de leçons avec des interfaces différentes. L'application actuelle ne compte que 3 leçons.

La figure de la page suivante présente le diagramme des classes complet de l'application Voxpel.

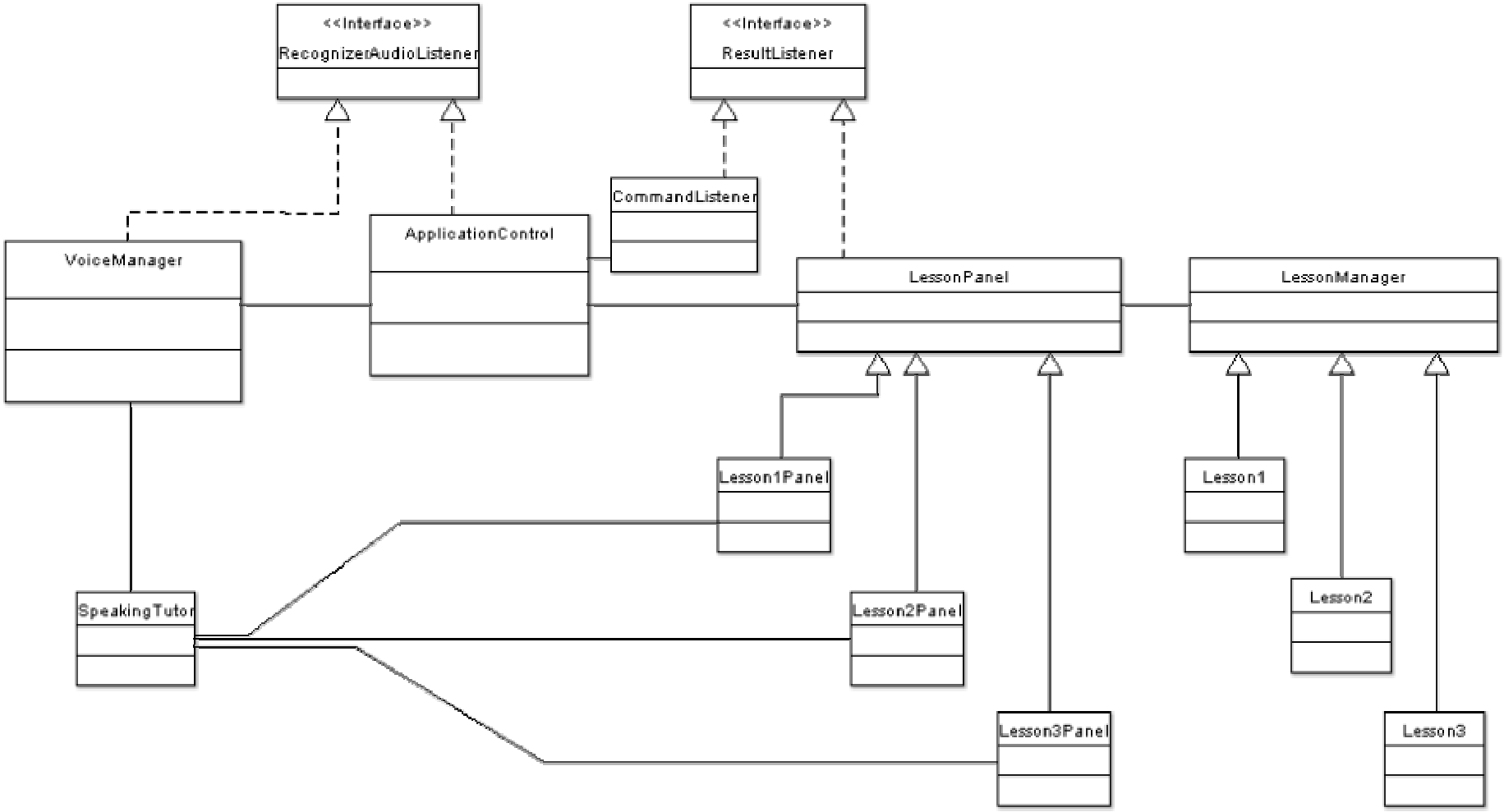

Figure 4 – diagramme UML des classes de l'application Voxpel

Interface graphique

L'interface graphique devait être simple et fonctionnelle. Toutes les leçons devaient être rapidement accessibles et leur contenu clair. Pour cela les mots et les phrases à prononcer devaient être représentés avec une police de grande taille et en gras. Des indications visuelles devaient aider l'utilisateur à commencer la leçon et à suivre son déroulement.

Seuls les composants standards Java Swing ont été utilisés afin de créer l'interface qui constitue une fenêtre avec une zone principale, un menu et une barre d'informations en bas.

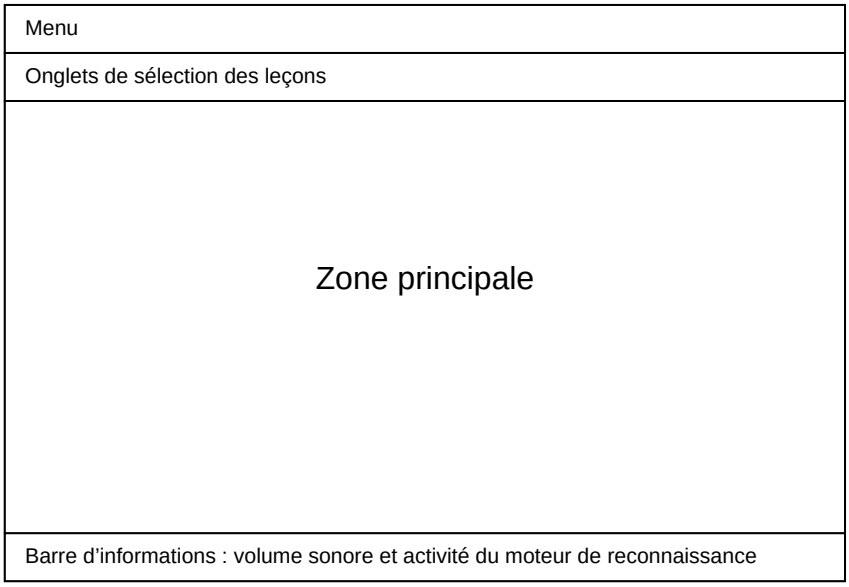

Figure 5 - disposition des éléments d'interface graphique

Interface vocale

On peut distinguer 3 catégories d'interactions pour l'interface vocale :

- Les commandes de l'utilisateur ;
- Les messages d'information et d'aide de l'ordinateur ;
- Les entrées/sorties des leçons de prononciation ;

La particularité de Voxpel est d'être toujours à l'écoute de l'utilisateur. Celui-ci peut donner une commande à tout instant, même pendant une leçon. Afin que le logiciel puisse ignorer une commande issue par inadvertance, un système trie les entrées voix en plus de la grammaire de règles. L'utilisateur dispose de deux possibilités d'attirer l'attention du programme : soit d'abord de l'appeler par son nom, soit d'adjoindre une formule de politesse à sa demande. Ainsi le fait juste de dire :

- Quitte le programme

n'a aucune chance d'aboutir. En revanche :

- Quitte le programme, s'il te plait

ou bien

- S'il te plait, leçon suivante

fonctionnera.

Si l'on veut donner plusieurs commandes (éventuellement possible, si l'application est plus complexe) l'utilisateur peut d'abord appeler l'application par son nom. Dans notre cas, le nom n'est guère original - « Ordinateur ». Ce dernier, s'il a bien entendu l'appel, répond. Ainsi l'utilisateur sait qu'il a obtenu l'attention du programme. Afin de quitter ce mode, l'utilisateur donne une commande tout à fait naturelle - « Merci ». Le programme retourne alors dans le mode normal.

En ce qui concerne les indications vocales données par l'application, le soin a été apporté sur le fait de ne pas les rendre répétitifs et agaçants. Les voix ne doivent pas bloquer l'exécution du programme (sauf cas particuliers, où une temporisation est nécessaire). Aussi l'ordinateur doit parler avec les voix différentes afin de palier au manque d'expression de la prosodie. Enfin, les voix françaises et anglaises ne doivent pas parler en même temps !

Afin d'obtenir un résultat acceptable un système de temporisations, de verrous et de choix aléatoires des voix a été mis en place. L'ordinateur parle d'une voix unique lorsqu'il s'adresse à l'utilisateur, mais change de voix, lorsqu'il prononce les mots des leçons.

Réalisation

En dehors des difficultés éprouvées lors des tests de faisabilité, la réalisation s'est déroulée sans souci majeur. Les problèmes touchaient essentiellement la partie de la parole, et notamment la reconnaissance : le bruit sonore, même très faible, parvient à sensibiliser le moteur de reconnaissance qui génère des lors une suite d'échecs d'écoute, ce qui est gênant dans le contexte d'une application d'apprentissage. La définition du paramètre de sensibilité : `RecognizerProperties.setSensitivity(float s)` n'étant pas supportée par IBMJS (`PropertyVetoException` obtenu), un mécanisme de surveillance du niveau sonore a du être mis en place. En utilisant l'interface `RecognizerAudioListener` il est possible de surveiller l'intensité de la voix et définir un seuil en dessous duquel les résultats rejetés sont ignorés.

De manière générale la qualité de reconnaissance vocale a été très médiocre. Afin que les mots aient plus de chance d'être reconnus, une grammaire spécifique a été définie est activée pour chaque leçon. Ainsi la fonctionnalité de dictée n'a pas été utilisée, mais uniquement celle de reconnaissance des règles. La reconnaissance est très sensible à la qualité de la prise du son : la qualité du micro et de la carte son joue énormément sur le taux de succès. Un micro et une carte bas de gamme n'ont que 10% de réussite contre un micro haut de gamme (50% à 80%)

Certaines lettre posent plus de difficultés que d'autres. Ainsi les mots commençant par la lettre F : fête, fenêtre, foudre, sont détectés lorsqu'on parle très fort et loin du micro, à cause de l'effet de projection de l'air. Dans ces cas le fait de parler à coté du micro, ou bien de mettre un anti-pop improvisé entre le micro et la bouche, améliore la reconnaissance. Malgré tout entre deux mot semblables : fête et tête, le second est reconnu au moins deux fois plus souvent. Il y a également ce qu'on pourrait presque appeler *« le syndrome du chat »*. En effet, le mot « le chat » est parfois détecté sans qu'il soit prononcé ! Le contrôle du seuil du volume permet parfois d'y remédier.

A noter également certains plantages inopinés de IBMJS, qui se sont produit sans cause apparente.

Prototype fonctionnel : Voxpel

Utilisation du logiciel

L'application dans son état actuel est un prototype pleinement fonctionnel, mais nécessitant sans doute quelques ajustements au niveau de l'interaction vocale. Afin de décrire son fonctionnement, voici un bref manuel d'utilisation.

Lorsque le programme démarre, la leçon numéro 1 est activée. L'interface qui se présente à l'utilisateur est la suivante :

Figure 6 - L'interface utilisateur de la leçon 1 de Voxpel

Ici elle est présentée pendant le déroulement de la leçon. Au début, l'ordinateur explique brièvement, et en deux langues, ce que l'utilisateur doit faire :

- Prononcez distinctement les mots suivants…
- Please speak carefully the following french words…

L'utilisateur a le choix de prononcer l'un des mots de la liste à gauche, ou bien de sélectionner l'un des mots afin que l'ordinateur le prononce d'abord. En actionnant la touche « *Hint* » il peut l'entendre avec des voix différentes, car toutes les voix synthétiques ne se valent pas. Supposons que l'utilisateur prononce :

- La tragédie

En cas de succès l'ordinateur répète en anglais :

- A tragedy

Et le mot passe dans la liste à gauche. En cas d'échec, « *Try again* » s'affiche et le nombre de tentatives s'incrémente avec le pourcentage de succès diminuant. Il n'y a pas de réaction vocale à l'échec afin de ne pas importuner l'utilisateur. Cependant si au début un mot est mal prononcé 8 fois de suite, un message vocal incite une fois l'utilisateur à parler fort et faire des pauses entre les mots, ceci toujours en deux langues.

En cas de difficulté, l'utilisateur peut passer un mot ou une phrase en cliquant sur le bouton « *Skip* » ou remettre à zéro la leçon avec le bouton « Reset ».

Lorsque tous les mots sont correctement prononcés, l'ordinateur le signale par une phrase de félicitation, puis bascule automatiquement vers la leçon suivante, pour laquelle il donne les instructions.

Les interfaces des leçons, bien que différentes, sont construites de manière similaire. Les mêmes contrôles : *Repeat (Hint) ; Skip et Reset* y sont présentes, ainsi que la zone des statistiques qui résume la proportion et le pourcentage des succès et le résultat instantané de la dernière tentative.

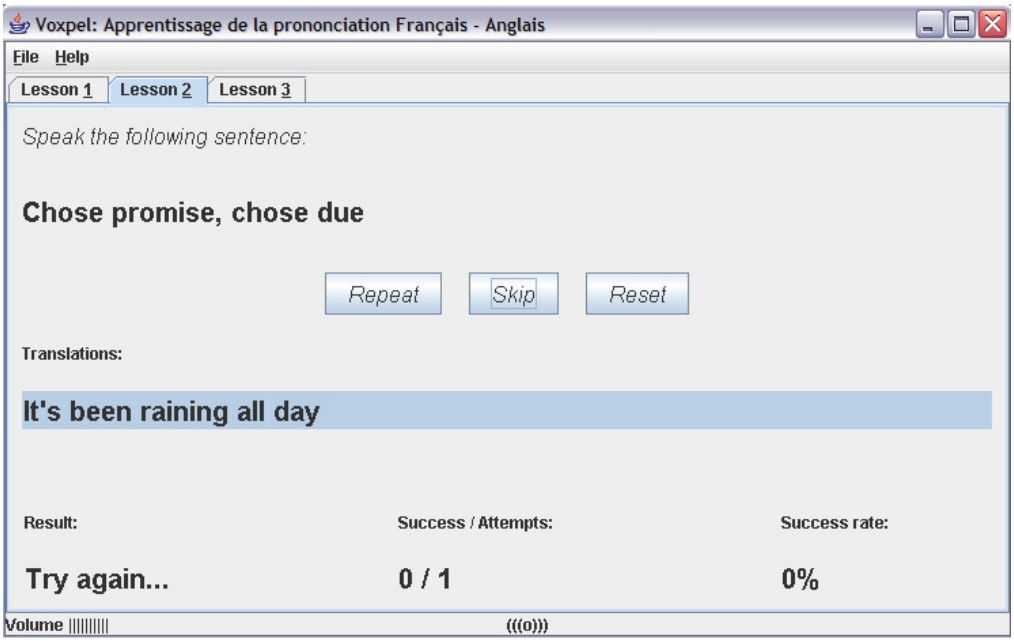

Figure 7 - interface de la leçon 2 diffère légèrement de celle de la leçon 1

Comme il fut précisé dans les spécifications, l'interface vocale permet également de commander un certain nombre de fonctionnalités. Ainsi il est possible de prononcer les ordres suivants :

- Leçon 1, 2, 3 ;
- Leçon suivante/précédente ;
- Quitte/Arrête le programme ;

Toutefois le simple fait de prononcer ces ordres ne suffit pas. Afin d'éviter des malentendus, il faut attirer l'attention de l'application, soit en adjoignant les mots « S'il te plait » au début ou à la fin de la commande, soit en appelant l'ordinateur avec le mot « Ordinateur ». Celui-ci se manifeste alors et a toute son attention pour exécuter les commandes. Lorsque c'est fait, le

mot « Merci » met fin à ce dialogue. L'ordinateur indique à chaque fois qu'il a bien compris la demande.

Possibilités d'extension

Etant donné l'aspect légèrement rustique de ce prototype, il peut être envisagé d'étendre ses fonctionnalités.

La première amélioration peu être d'étendre le nombre et le contenu des leçons. A cette fin l'architecture de l'application définit des *interfaces* standard qui peuvent être suivies et réutilisées.

Une deuxième amélioration peut concerner l'interface graphique : ajout de couleurs ou d'images représentant les mots ou les phrases.

Enfin, une fonctionnalité prévue, mais non implémentée à cause du manque d'outils logiciels, est l'ajout de reconnaissance vocale en anglais, qui permettrait de rendre l'outil totalement bilingue. Un meilleur moteur de reconnaissance vocale serait également le bienvenu, car, comme précisé précédemment, le taux de réussite de cette reconnaissance reste médiocre.

Conclusion

En conclusion, l'utilisation de l'API JavaSpeech au sein d'une application Java est efficace et relativement simple à mettre en œuvre, à condition de disposer d'une implémentation complète et d'un moteur de traitement de la parole de qualité. Ce dernier point est le maillon faible d'une interface vocale, car même si les voix synthétiques restent acceptables, le taux d'erreurs de la reconnaissance limite très sérieusement les possibilités d'un dialogue entre l'utilisateur et l'application.

Outre les problèmes de qualité, une bonne conception d'une interface vocale homme-machine doit tenir compte des redondances dans les commandes et les message vocaux afin de pas rendre ces fonctionnalités ennuyeuses, voire insupportables ! Une combinaison équilibrée entre les éléments d'interface graphique et vocales en est le meilleur moyen. L'emploi plus étendu de la prosodie permet également d'améliorer le confort de l'utilisation. A noter que le modèle de dialogue avec l'écoute continue, expérimenté dans ce prototype, a confirmé son efficacité espérée !

En ce qui concerne les alternatives, telles que VoiceXML, la démarche de définition d'interface vocale structurée est très appropriée à la tache poursuivie par ce projet. Néanmoins la spécificité des outils VXML existants, tournés vers les réseaux et la téléphonie, rend complexe la mise en œuvre des applications classiques. L'approche utilisant JSAPI fut, par conséquent, plus justifiée dans notre cas. Reste à regretter l'abandon de cette norme par les éditeurs des logiciels, parmi lesquels IBM dont l'outil fut utilisé pour ce prototype, et l'absence d'implémentations compètes, efficaces et fiables.

Bibliographie et références Internet

Publications

- Huang Xuedong : *«Spoken Language Processing: a guide to theory, algorithm and system development »*, Prentice Hall, Upper Saddle River, NJ, 2001
- Vivek Malhotra : *"Developing dynamic VoiceXML applications"*, IBM DeveloperWorks Tutorials
- Andy Aaron, Ellen Eide, John Pitrelli : *"Les ordinateurs ont la parole"*, Article, Pour la Science, août 2005
- Gene Frantz, Richard Wiggins : *"Design case history : Speak & Spell learns to talk"*, Article, IEEE Spectrum, février 1982

Liens Internet

- JavaSpeech API : http://java.sun.com/products/java-media/speech/
- VoiceXML Forum: http://www.voicexml.org
- Cloud Garden TalkingJava: http://www.cloudgarden.com/JSAPI/
- Microsoft Speech SDK: http://www.microsoft.com/speech/download/sdk51/
- FreeTTS: http://freetts.sourceforge.net/docs/index.php
- MBROLA: http://tcts.fpms.ac.be/synthesis/mbrola.html
- Carnegie Mellon University, Speech Group: http://www.speech.cs.cmu.edu/hephaestus.html
- Procédés de synthèse de la parole: http://www.tcom.ch/Tcom/Laboratoires/digivox2000/chap/chap5/synthese.htm

Glossaire

API	Application programming interface – interface ou bibliothèque de programmation destinée à la réalisation d'une application
ASR	Automatic Speech Recognition – reconnaissance automatique de la parole
Diphone	Combinaison de deux phonèmes, particulière à une langue donnée
Diphtongue	Son de voyelle, dont la sonorité change entre le début et la fin (ex : boat)
Formant	Signature particulière du spectre sonore dont la répétition forme un son de la voix (voyelle monophtongue ou diphtongue). La répétition d'un formant ou la transition entre plusieurs formants constitue un phonème
Monophtongue	Son de voyelle, dont la sonorité ne change pas au cours du temps (ex : boot). Un son monophtongue est en général composé d'un seul formant
Phonème	Unité sonore élémentaire de la parole. La parole dans une langue donnée est composée d'un nombre limité de phonèmes (34 en français, 40 en anglais etc.). Les phonèmes sont composés de formants et se composent ils mêmes en *diphones*
Prosodie	Accentuation et intonations dans la parole
SDK	Software Development Kit – similaire à la signification de l'API
TTS	Text To Speech – la synthèse vocale d'un texte écrit
VXML	Format de documents VoiceXML
IVR	Interactive Voice Response – réponse vocale interactive ou serveur vocal interactif

Annexes

A1 - Documentation du programme (JavaDoc)

Voir pages suivantes.

A2 - Récapitulatif des commandes vocales dans le fichier JSGF

```
grammar fr.cnam.command;


public <prepareCommand> = Ordinateur {prepareCommand};

<number> = un {1} | deux {2} | trois {3} | suivante {next} | précédente
{prev} ;

public <lesson> = Leçon {lesson} <number>;

public <quit> = (Quitte le programme | Arrête le programme) {quit};
public <please> = (S'il te plaît | S'il vous plaît) {please};
<command> = <lesson> | <quit>;

public <immediateCommand> = ( <command> <please> ) | ( <please> <command>
);

public <endCommand> = Merci {endCommand};
```