Activité : Synthèse vovale

Thierry Vaira <tvaira@free.fr>

01/02/2016 (rev. 1)

Table des matières

Synthèse vovale

Expression du besoin

Certaines applications ont besoin de communiquer de manière sonore avec l'utilisateur. Ces applications doivent utiliser un système Text To Speech permettant de transformer un message écrit en un message parlé.

Notions de base

La synthèse vocale est une technique informatique de synthèse sonore qui permet de créer de la parole artificielle à partir de n'importe quel texte. Pour obtenir ce résultat, elle s'appuie à la fois sur des techniques de traitement linguistique, notamment pour transformer le texte orthographique en une version phonétique prononçable sans ambiguïté, et sur des techniques de traitement du signal pour transformer cette version phonétique en son numérisé écoutable sur un haut parleur.

Terminologie

Text To Speech Text To Speech est un système informatique permettant de transformer un texte écrit en un texte parlé.

Phonème Un phonème est la plus petite unité discrète ou distinctive (c'est-à-dire permettant de distinguer des mots les uns des autres) que l'on puisse isoler par segmentation dans la chaîne parlée. Un phonème est en réalité une entité abstraite, qui peut correspondre à plusieurs sons.

Transcription phonétique La première opération à réaliser par un logiciel de synthèse vocale est de transcrire le texte, généralement écrit sous une forme orthographique, en une séquence de phonèmes, qui représentent exactement les sons qui doivent être prononcés. La technique classique pour réaliser la transcription phonétique est d'appliquer à la suite de lettres composant le texte une série de règles de réécritures dépendant du contexte.

Formants La synthèse par formants repose typiquement sur la description des trois premiers formants du spectre de la parole. Chaque formant (maximum du spectre de parole) est classiquement décrit par trois paramètres, sa fréquence (en Hz), son amplitude (en dB) et sa bande passante (en Hz). L'amplitude représente l'intensité du signal à la fréquence du formant tandis que la bande passante représente la largeur du spectre autour du maximum formantique.

Prosodie La prosodie est l'inflexion, le ton, la tonalité, l'intonation, l'accent, la modulation que nous donnons à notre langage oral en fonction de nos émotions et de l'influence que nous désirons avoir sur nos interlocuteurs. En outre, c'est l'étude des traits phoniques, c'est-à-dire l'étude du rythme (vitesse d'élocution), l'accent et l'intonation.

Intonation L'intonation constitue l'un des trois éléments de la prosodie, les deux autres paramètres prosodiques étant le rythme et l'intensité. En fait, pour être perçue comme naturelle, la synthèse vocale nécessite d'imiter une prosodie naturelle dans son ensemble. Ceci nécessite de reproduire aussi un rythme naturel, c'est-à-dire une durée naturelle des sons élémentaires (phonèmes). Quant à l'intensité, qui correspond aussi au volume sonore, elle est beaucoup moins critique que l'intonation ou le rythme pour obtenir un rendu naturel. L'intonation se mesure par la fréquence fondamentale de la voix. C'est une fréquence variable au cours du temps correspondant à la fréquence de vibration des cordes vocales pendant l'énonciation de la phrase, et qui s'observe aisément comme la périodicité du signal vocal. Typiquement la fréquence fondamentale d'une voix masculine possède une plage de variation dans la zone des 80 Hz à 150 Hz tandis que celle d'une voix féminine se situera plutôt dans la zone des 140 Hz à 200 Hz.

MBROLA et eSpeak

MBROLA est un algorithme de synthèse vocale. Il ne fait pas directement la transformation de texte en voix. Il traite des fichiers de phonèmes. Une autre application, capable de décomposer du texte en phonème est donc nécessaire. Les voix de mbrola donnent un rendu plus naturel que les voix de espeak (lesquelles sont très "robotiques"), d'où l'intérêt de combiner les deux. La page web du projet MBROLA fournit des bases de données de diphones pour un grand nombre de langues parlées.

Lien MBROLA : tcts.fpms.ac.be

eSpeak est un logiciel open source compact de synthèse vocale fonctionnant sous les systèmes d'exploitation Linux, Windows, Mac OS et d'autres plateformes.

eSpeak utilise les formants comme méthode de synthèse, permettant de fournir un large choix de langues pour une taille réduite. La plupart des développements sur les langues gérées par eSpeak sont basés sur les informations trouvées sur Wikipédia et sur les retours des utilisateurs de langue maternelle. eSpeak est utilisé par des projets comme NVDA (NonVisual Desktop Access), Ubuntu et OLPC, et est également utilisé par Google Traduction.

Lien eSpeak : espeak.sourceforge.net

Ressources

- Synthèse vocale Ubuntu
- Faire parler un robot
- Activer le son sur Raspberry Pi
- Synthèse vocale avec espeak et mbrola sur Raspberry Pi

Objectifs

Être capable de faire "parler" une application.

Préparation

Il vous faudra tout d'abord installer `espeak` :

```
$ sudo apt-get install espeak libespeak-dev
```

Il vous faudra ensuite installer `mbrola` :

```
$ sudo vim /etc/apt/sources.list
deb-src http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free rpi

$ sudo apt-get update

$ sudo apt-get build-dep unrar-nonfree

$ sudo apt-get source -b unrar-nonfree

$ sudo dpkg -i unrar*.deb

$ wget http://tcts.fpms.ac.be/synthesis/mbrola/bin/armlinux/mbrola.rar
```



```
$ unrar e mbrola.rar

$ sudo mv mbrola /usr/bin

$ sudo chmod 777 /usr/bin/mbrola

$ sudo mkdir /usr/share/mbrola

$ sudo mkdir /usr/share/mbrola/voices

$ wget http://tcts.fpms.ac.be/synthesis/mbrola/dba/fr1/fr1-990204.zip

$ unzip fr1*.zip

$ sudo mv fr1/fr1 /usr/share/mbrola/voices/
```

Séquence 1 : test de eSpeak et MBROLA

```
$ espeak -v mb/mb-fr1 "Toto"

$ espeak -X -q -v mb/mb-fr1 "Toto"
Translate 'toto'
  1 t        [t]
  5 _) t     [t]

  1 o        [O]

  1 t        [t]

 22 o (_     [o]
  1 o        [O]

 tOt'o

$ espeak -s 110 -v mb/mb-fr1 "Toto"

$ espeak -v mb/mb-fr1 "Toto mange des bananes" | mbrola /usr/share/mbrola/fr1/fr1 - - | aplay
```

Séquence 2 : mise en oeuvre d'une classe SyntheseVocale

La déclaration de la classe `SyntheseVocale` :

```
#ifndef SyntheseVocale_H
#define SyntheseVocale_H

#include <iostream>
#include <string>
#include <cstring>

using namespace std;

class SyntheseVocale
{
   public:
      // Constructeurs et Destructeur
      SyntheseVocale();
      SyntheseVocale(int speed, int volume, int pitch, int range);
      ~SyntheseVocale();

      // Accesseurs
      string  getVoice() const;
      int  getSpeed() const;
      int  getVolume() const;
```

```
      int  getPitch() const;
      int  getRange() const;
      int  getPunctuation() const;
      int  getGender() const;
      void setVoice(string voice);
      void setSpeed(int speed);
      void setVolume(int volume);
      void setPitch(int pitch);
      void setRange(int range);
      void setGender(int gender);
      void setPunctuation(int punctuation);

      // Services
      void parle(const string texte);
      void muet();
      void liste() const;
      void version();

   private:
      string texte;
      string voice;
      int speed; // espeakRATE
      int volume; // espeakVOLUME
      int pitch; // espakPITCH
      int range; // espeakRANGE
      int punctuation; // espeakPUNCTUATION
      int gender;
      const char *eSpeakVersionInfo;
      int samplerate;

      void reglages();
};

#endif // SyntheseVocale_H
```

La définition de la classe `SyntheseVocale` :

```
#include "SyntheseVocale.h"
#include "espeak/speak_lib.h"

// Constructeurs et Destructeur
SyntheseVocale::SyntheseVocale() : texte(""), voice("mb-fr1"), speed(130), volume(80), pitch(40), range(60),
    punctuation(0), gender(2), eSpeakVersionInfo(NULL), samplerate(22050)
{
   version();

   const char *data_path = NULL; // use default path for espeak-data
   samplerate = espeak_Initialize(AUDIO_OUTPUT_PLAYBACK, 0, data_path, 0);
   //samplerate = espeak_Initialize(AUDIO_OUTPUT_SYNCH_PLAYBACK, 0, data_path, 0);
   cout << "Samplerate : " << samplerate << endl;

   //liste(); // liste toutes les langues disponibles
   // Attention : les langues MBROLA ne sont pas affichées mais potentiellement disponibles aussi

   setVoice(voice);

   // 0=no punctuation
   // 2=female
   reglages();
}

SyntheseVocale::SyntheseVocale(int speed, int volume, int pitch, int range) : texte(""), voice("fr"), speed(
    speed), volume(volume), pitch(pitch), range(range), punctuation(0), gender(2), eSpeakVersionInfo(NULL),
    samplerate(22050)
{
   version();
```



```
   const char *data_path = NULL; // use default path for espeak-data
   samplerate = espeak_Initialize(AUDIO_OUTPUT_PLAYBACK, 0, data_path, 0);
   //samplerate = espeak_Initialize(AUDIO_OUTPUT_SYNCH_PLAYBACK, 0, data_path, 0);
   //cout << "Samplerate : " << samplerate << endl;

   //liste(); // liste toutes les langues disponibles
   // Attention : les langues MBROLA ne sont pas affichées mais potentiellement disponibles aussi

   setVoice(voice);

   // 0=no punctuation
   // 2=female
   reglages();
}

SyntheseVocale::~SyntheseVocale()
{
   espeak_Terminate();
}

// Accesseurs
string SyntheseVocale::getVoice() const
{
   return voice;
}

int  SyntheseVocale::getSpeed() const
{
   return speed;
}

int  SyntheseVocale::getVolume() const
{
   return volume;
}

int  SyntheseVocale::getPitch() const
{
   return pitch;
}

int  SyntheseVocale::getRange() const
{
   return range;
}

int  SyntheseVocale::getPunctuation() const
{
   return punctuation;
}

int  SyntheseVocale::getGender() const
{
   return gender;
}

void SyntheseVocale::setVoice(string voice)
{
   if(espeak_SetVoiceByName(voice.c_str()) != EE_OK)
         cerr << "espeak_SetVoiceByName error !" << endl;
   else
   {
     cout << "Selected language : " << voice << endl;
     this->voice = voice;
   }
```



```
}

void SyntheseVocale::setSpeed(int speed)
{
   // speed in words per minute Values 80 to 450.
   if(speed >= 80 && speed <= 450)
   {
      this->speed = speed;
      espeak_SetParameter(espeakRATE, speed, 0);
   }
}

void SyntheseVocale::setVolume(int volume)
{
   // volume in range 0-100 (0=silence)
   if(volume >= 0 && volume <= 100)
   {
      this->volume = volume;
      espeak_SetParameter(espeakVOLUME, volume, 0);
   }
}

void SyntheseVocale::setPitch(int pitch)
{
   // base pitch in range 0-100 (50=normal)
   if(pitch >= 0 && pitch <= 100)
   {
      this->pitch = pitch;
      espeak_SetParameter(espeakPITCH, pitch, 0);
   }
}

void SyntheseVocale::setRange(int range)
{
   // pitch range in range 0-100 (0=monotone, 50=normal)
   if(range >= 0 && range <= 100)
   {
      this->range = range;
      espeak_SetParameter(espeakRANGE, range, 0);
   }
}

void SyntheseVocale::setPunctuation(int punctuation)
{
   // 0 no punctuation, 1 say punctuation
   if(range >= 0 && range <= 1)
   {
      this->punctuation = punctuation;
      espeak_SetParameter(espeakPUNCTUATION, punctuation, 0);
   }
}

void SyntheseVocale::setGender(int gender)
{
   // 0=none, 1=male, 2=female
   if(gender >= 0 && gender <= 2)
   {
      this->gender = gender;
      espeak_VOICE *voice_spec = espeak_GetCurrentVoice();
      //memset(voice_spec, 0, sizeof(espeak_VOICE)); // Zero out the voice first
      //voice_spec->languages = "fr";
      //voice_spec->identifier = "mb-fr1";
      voice_spec->gender = gender;
      //voice_spec->age = 25;
      //voice_spec->variant = 1;
      int status = espeak_SetVoiceByProperties(voice_spec);
```



```
      //cout << "Status : " << status << endl;
      voice_spec = espeak_GetCurrentVoice();
      //cout << "Language shortsign : " << voice_spec->languages << endl;
      //cout << "Identifier : " << voice_spec->identifier << endl;
      //cout << "Age : " << (int)voice_spec->age << endl;
      //cout << "Gender : " << (int)voice_spec->gender << endl;
      //cout << "Name : " << voice_spec->name << endl;
      //cout << "Variant : " << (int)voice_spec->variant << endl;
      //cout << endl;
   }
}

// Services
void SyntheseVocale::parle(const string texte)
{
   cout << "-- " << texte << endl;
   espeak_Synth((char *)texte.c_str(), texte.length()+1, 0, POS_CHARACTER, 0, espeakCHARS_AUTO, NULL, NULL);
   espeak_Synchronize(); // appel bloquant
}

void SyntheseVocale::muet()
{
   setVolume(0);
}

void SyntheseVocale::liste() const
{
   const espeak_VOICE **voices = NULL;
   espeak_VOICE voice_select;
   const espeak_VOICE *v;

   voices = espeak_ListVoices(NULL);

   for(int ix=0; (v = voices[ix]) != NULL; ix++)
   {
      cout << "Language shortsign : " << v->languages << endl;
      cout << "Identifier : " << v->identifier << endl;
      cout << "Age : " << v->age << endl;
      cout << "Gender : " << v->gender << endl;
      cout << "Name : " << v->name << endl;
      cout << "Variant : " << v->variant << endl;
      cout << endl;
   }
}

void SyntheseVocale::version()
{
   eSpeakVersionInfo = espeak_Info(NULL);
   cout << "SyntheseVocale version : " << eSpeakVersionInfo << endl;
}

void SyntheseVocale::reglages()
{
   cout << "Gender : " << gender << endl;
   setGender(gender);
   cout << "Speed : " << speed << endl;
   setSpeed(speed);
   cout << "Volume : " << volume << endl;
   setVolume(volume);
   cout << "Pitch : " << pitch << endl;
   setPitch(pitch);
   cout << "Range : " << range << endl;
   setRange(range);
   cout << "Punctuation : " << punctuation << endl;
   setPunctuation(punctuation);
}
```


Un programme de test de la classe `SyntheseVocale` :

```
#include <iostream>
#include <unistd.h>

using namespace std;

#include "SyntheseVocale.h"

int main()
{
   SyntheseVocale espeak;
   //SyntheseVocale espeak(130, 80, 40, 60); // speed, volume, pitch, range

   espeak.parle("alarme température");
   sleep(1);
   espeak.parle("alarme ph");
   sleep(1);
   espeak.parle("alarme niveau d'eau");
   sleep(2);

   espeak.setVoice("fr+f3");
   espeak.parle("alarme température");
   sleep(1);
   espeak.parle("alarme ph");
   sleep(1);
   espeak.parle("alarme niveau d'eau");
   return 0;
}
```

Pour fabriquer ce programme de test (ajout de `-lespeak` à l'édition de liens), il vous faudra utiliser ce fichier `Makefile` :

```
CXX=g++

GENERIC_CFLAGS := -fpermissive
GENERIC_LIBS := -lespeak

TARGET=test-espeak

HEADERFILES := $(wildcard *.h)
SRCFILES    := $(wildcard *.cpp)
OBJFILES    := $(patsubst %.cpp, %.o, $(SRCFILES))

.PHONY: clean $(TARGET)

all: $(TARGET)

%.o: %.cpp %.h
    $(CXX) -o $@ -c $< $(GENERIC_CFLAGS)

$(TARGET): $(OBJFILES)
    $(CXX) -o $(TARGET) $^ $(GENERIC_CFLAGS) $(GENERIC_LIBS)

clean:
    rm -f $(TARGET) $(OBJFILES) *~
```

Code source : test-mo-espeak.zip

Retour au sommaire

