PARLEZ À VOTRE ORDINATEUR... ET FAITES-VOUS COMPRENDRE !

Tristan COLOMBO

Que ce soit Hal ou Jarvis, les exemples d'ordinateurs intelligents et doués de parole ne manquent pas au cinéma ou dans la littérature. Nous laisserons de côté l'aspect « intelligent » pour nous focaliser sur la reconnaissance vocale de manière à pouvoir demander à l'ordinateur d'exécuter certaines tâches en « dialoguant » avec lui... et ce n'est déjà pas si simple !

Mots-clés : Reconnaissance vocale, Python, Jasper, Pocketsphinx, Raspbian

Résumé

Les solutions de reconnaissance vocale sont assez complexes à mettre en place, car utilisant de nombreux outils et nécessitant donc un long travail d'installation et de configuration, sans parler de l'aspect matériel ou le micro devra être reconnu. Le projet Jasper propose une solution modulaire puisque l'utilisateur peut installer différents moteurs de reconnaissance vocale et que le développement des commandes se fait à l'aide de modules Python. Dans cet article, nous installons Jasper avec le moteur STT Pocketsphinx, décrivons ses limites et créons un petit module de commande personnalisé avant de tester une utilisation pure de PocketSphinx avec Python.

Après avoir testé **Rivescript** avec Frédéric Le Roy dans l'article précédent, vous avez peut-être envie d'aller plus loin et de tester la reconnaissance vocale. C'est ce que je vous propose en utilisant le projet **Jasper** qui a été conçu spécialement pour élaborer des *chatbots* sur Raspberry Pi avec le système d'exploitation Raspbian. Cette solution peut également être mise en place sur un ordinateur fixe et c'est ce que je ferai dans cet article en utilisant une distribution Debian : toutes les indications données sont donc compatibles avec Raspbian et peuvent être utilisées pour une Raspberry Pi. L'installation est un peu longue et il vaut mieux s'assurer de ne pas faire d'erreur...

1 Installation

Il n'existe pas encore de paquetage Debian ou Raspbian, ce qui implique de passer obligatoirement par une installation manuelle.

1.1 Étape préparatoire

Il faut tout d'abord s'assurer de disposer des bons outils :

```
$ sudo aptitude install git-core python-dev python-pip bison
libasound2-dev python-pyaudio libyaml-dev
```

Sur Raspbian, ajoutez la librairie **libportaudio-dev**.

Assurez-vous ensuite que votre micro est correctement reconnu et configuré. Cela va dépendre de nombreux facteurs et je ne peux donc pas donner de solution générique si ce n'est de penser à l'activer dans l'outil « Préférences du son ». Pour tester si tout fonctionne correctement, vous pourrez employer **arecord** pour vous enregistrer (<Ctrl> + <c> pour interrompre) et **aplay** pour jouer le son contenu dans le fichier précédemment enregistré :

```
$ arecord test.wav
Capture WAVE 'test.wav' : Unsigned 8 bit, Fréquence 8000 Hz, Mono
^CInterrompu par le signal Interrompre...
$ aplay test.wav
Lecture WAVE 'test.wav' : Unsigned 8 bit, Fréquence 8000 Hz, Mono
```

L'étape la plus périlleuse étant passée, il faut définir des variables d'environnement. En se basant sur un shell bash, ajoutez les lignes suivantes à la fin de votre **~/.bashrc** :

```
export LD_LIBRARY_PATH=/usr/local/lib
export PATH=$PATH:$LD_LIBRARY_PATH
```

Bien entendu, n'oubliez pas de relire le fichier :

```
$ source ~/.bashrc
```

1.2 Installation de Jasper

Nous pouvons maintenant récupérer le code source de Jasper :

```
$ git clone https://github.com/jasperproject/jasper-client.git
jasper
```

Comme il s'agit d'un projet Python et qu'il est bien structuré, un fichier **requirements.txt** est fourni pour que **pip** puisse installer toutes les dépendances automatiquement :

```
$ sudo pip install -U -r jasper/client/requirements.txt
```

Vous pensiez que c'était fini ? Que nenni ! Il reste à installer un moteur STT (*Speech-To-Text* soit « de la parole au texte ») et un moteur TTS (*Text-To-Speech* soit « du texte à la parole »). C'est la partie la plus importante, car ce sont ces moteurs qui vont faire tout le travail et tout particulièrement le moteur STT. Le choix du moteur sera dicté par l'utilisation que vous allez avoir de votre application. En voici quelques-uns :

- **Pocketsphinx [1]** : un moteur rapide fonctionnant bien sur les systèmes embarqués (comme le Raspberry Pi), mais qui n'est pas le meilleur en terme de taux de reconnaissance vocale. Avantage de ce moteur : ne nécessite pas une utilisation du cloud ;
- **Google STT** : le système de reconnaissance de Google (celui des smartphones Android). Ne fonctionne qu'à travers le cloud avec une limite du nombre de requêtes par jour. Il faut l'activer dans la console développeurs Google **[2]** ;
- **AT&T STT [3]** : système de reconnaissance vocale d'AT&T fonctionnant comme Google STT dans le cloud ;
- **Wit.ai STT [4]** : encore un moteur fonctionnant dans le cloud...
- **Julius [5]** : et pour finir un moteur ne nécessitant pas de connexion internet. Par contre ce dernier, très performant, demande une configuration poussée.

Pour moi le choix ne peut être que Pocketsphinx ou Julius, le principe de la connexion internet étant rédhibitoire. Pour une première, nous n'allons pas tenter le diable avec Julius donc nous utiliserons Pocketsphinx.

1.3 Installation de Pocketsphinx

Pocketsphinx est disponible dans les dépôts Debian, ce qui simplifie nettement l'installation :

```
$ sudo aptitude install pocketsphinx
```

Il nous faut ensuite installer CMUCLMTK, le *wrapper* permettant d'accéder aux outils du CMU Sphinx (la boîte à outils contenant Pocketsphinx). Pour commencer, nous vérifions une fois encore que nous disposons de tous les outils :

```
$ sudo aptitude install subversion autoconf libtool automake
gfortran g++
```

Nous pouvons maintenant installer CMUCLMTK dans le répertoire de Jasper :

```
$ cd jasper
$ svn co https://svn.code.sf.net/p/cmusphinx/code/trunk/cmuclmtk/
$ cd cmuclmtk/
$ sudo autogen.sh && sudo make && sudo make install
$ cd ..
```

1.4 Jasper écoute... mais il parle également

Nous avons installé un moteur STT pour que Jasper puisse écouter, il lui faut maintenant un moteur TTS pour qu'il puisse parler. Là encore plusieurs moteurs sont disponibles : **eSpeak**, **Flite**, **Google TTS**, etc. Le plus simple est **eSpeak** :

```
$ sudo aptitude install espeak
```

Si vous souhaitez pouvoir changer de voix vous pouvez tester Flite :

```
$ sudo aptitude install flite
```

`flite -lv` vous permettra d'afficher la liste des voix disponibles (nous nous en servirons plus tard pour la configuration).

1.5 Et pour finir...

Le moteur Pocketsphinx a besoin de certaines dépendances. En l'occurrence, il s'agit du **MIT Language Modeling Toolkit**, **m2m-aligner** et **Phonetisaurus**. Des paquets sont disponibles pour Debian dans les dépôts expérimentaux. On va donc commencer par créer un fichier de sources expérimentales **`/etc/apt/sources.list.d/experimental.list`** contenant la ligne :

```
deb http://ftp.debian.org/debian experimental main contrib non-free
```

Après une mise à jour d'apt, il suffit d'indiquer que l'on souhaite aller chercher des paquets dans ce dépôt :

```
$ sudo aptitude update
$ sudo aptitude -t experimental install phonetisaurus m2m-aligner
mitlm
```

Il faut également ajouter l'installation d'**OpenFst** à la main. Un fst, pour *weighted Finite-State Transducer* (transducteur à états finis pondérés), est un automate **[6]** où chaque transition est dotée d'une valeur d'entrée, de sortie et d'un poids. OpenFst fournit les outils permettant de manipuler de tels automates qui sont utilisés pour déterminer la conversion des graphèmes en phonèmes (et réciproquement). Pour revenir à l'installation :

```
$ wget http://distfiles.macports.org/openfst/openfst-1.4.1.tar.gz
$ tar -xvf openfst-1.4.1.tar.gz
$ cd openfst-1.4.1
$ sudo ./configure --enable-compact-fsts --enable-const-fsts
--enable-far --enable-lookahead-fsts --enable-pdt
$ sudo make install
$ cd ..
$ rm -R openfst-1.4.1
```

Mais nous ne sommes pas pour autant tirés d'affaire... il faut maintenant configurer Jasper.

2 Configuration

Pour commencer, il faut générer un profil d'utilisation grâce au script **`populate.py`** se trouvant dans le répertoire **`client`** de notre installation (soit **`jasper/client`**) :

```
$ cd client
$ python populate.py
```

Des questions vous seront alors posées. Rassurez-vous, vous n'êtes pas obligé de répondre à tout ! L'adresse et le mot de passe gmail sont par exemple demandés pour utiliser le module permettant de vérifier si vous avez reçu des mails. Vous pouvez donc laisser ces champs vides...

```
...
First name: Tristan
Last name: Colombo
...
Gmail address:
Password:

Phone number (no country code). Any dashes or spaces will be
removed for you:

Phone carrier (for sending text notifications).
...
Carrier:

...
Location: Marseille
Location saved as Marseille,  FR

Please enter a timezone from the list located in the TZ* column at
http://en.wikipedia.org/wiki/List_of_tz_database_time_zones, or
none at all.
Timezone: Europe/Paris

Would you prefer to have notifications sent by email (E) or text
message (T)? E

If you would like to choose a specific STT engine, please specify
which.
Available implementations: %s. (Press Enter to default to
PocketSphinx):
Unrecognized STT engine. Available implementations: ['google',
'sphinx']
Writing to profile...
Done.
```