TP d'introduction à Python
Blah-blah Bot

Pierre-Marie de Rodat (LSE)

20 novembre 2012

Table des matières

1 Introduction **1**

2 Première étape : jouer les structures de données… **2**
2.1 Phase d'apprentissage 2
2.2 Mais parle donc ! 3

3 Sauvegarder la base de données **5**

4 Bonus : dessiner la base de données **5**

5 Bonus : améliorer l'algorithme **7**

6 Master Bonus Corporate : et les salons de discussion, alors ? **7**

1 Introduction

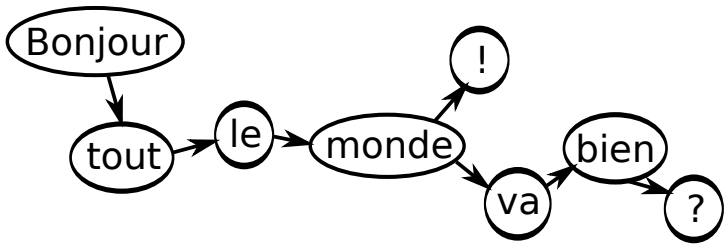
S'il vous arrive de passer du temps dans des salons de discussion (IRC, XMPP/Jabber, …), vous avez peut-être déjà croisé des robots : ceux-ci ne sont en général pas bien causants. Je parie que vous avez eu un jour envie de créer un robot qui parle tout seul. Ah ? Vous aussi ? Allez, c'est parti !

La difficulté de cet exercice réside dans le choix de la méthode utilisée pour générer du texte : « mais comment c'est possible, les ordinateurs ne savent quand même pas encore penser ?» Bonne question… Dans notre cas, nous postulerons que comme dans la rue, il suffira simplement de *paraître* intelligent, et tout ira pour le mieux. :-)

Tout comme de nombreuses méthodes de génération de texte, nous allons diviser le travail en deux étapes. En premier lieu notre robot se contentera de lire du texte venant d'humains, puis après avoir rempli une sorte de base de données (c'est la phase d'apprentissage), il utilisera cette dernière pour synthétiser des « phrases» (et là on va rire !).

Le principe sera ici fort simple : on va tenter de mémoriser les successions de mots. Par exemple, si le robot reçoit « Bonjour tout le monde !», il découpera le message en 4 morceaux (« Bonjour», « tout», « le», « monde», et « !»), puis il mémorisera que « Bonjour» peut être suivi de « tout» qui lui-même peut-être suivi de « le», etc. .

Par exemple, après avoir lu sucessivement « Bonjour tout le monde !»et « tout le monde va bien ?», la base de données de notre robot devrait ressembler à ça :



2 Première étape : jouer les structures de données...

On n'échappe pas à la règle : dans la majorité des cas, en informatique on ne fait que manipuler des données. Il faut donc commencer par décrire le format de nos données.

Dans notre cas, un seul type d'information nous intéresse : à partir d'un mot, on veut avoir accès à la l'ensemble de ses « successeurs». Cette utilisation fait penser à celle d'un dictionnaire :

```
database['monde'] = set('va', '!')
```

Les clefs du dictionnaire sont les mots, et les valeurs associées sont les ensembles des successeurs. Allez, on prend !

2.1 Phase d'apprentissage

L'algorithme d'apprentissage est celui décrit dans la section d'introduction : on découpe une phrase en mots, puis on mémorise les succes-

seurs de chaque mot.

```
database = {}

def learn(sentence):
    # Tip: 'hello world'.split()
    # returns ['hello', 'world']

    # Insert some code here...

    # ... then for each word:
    # we get the set of the successors
    # of words[i] (or an empty set)
    next_words = database.get(words[i], set())

    # we append to it one new successor
    next_words.add(words[i + 1])

    # and we put the set of successors
    # in the database
    database[words[i]] = next_words

learn(raw_input())
```

1

2.2 Mais parle donc !

Constituer une base de données, c'est bien beau, mais l'utiliser, c'est mieux ! La base de données mémorise à partir d'un mot quels sont ses successeurs possibles. Une méthode possible pour générer du texte est donc naturellement de prendre un mot (mais lequel ?), et d'ajouter à la suite les mots que l'on trouve en suivant les liens (mais jusqu'où ?).

Pour savoir à quel mot commencer et à quel mot s'arrêter, on pourrait améliorer notre structure de données pour mémoriser par exemple par quels mots les phrases utilisées pour l'apprentissage commençaient et finissaient. Bien que ce ne soit pas complexe, cet exercice est laissé au lecteur : on n'a pas toute la nuit, mais le lecteur a, lui, plusieurs années devant lui. :-)

1. Voir le fichier source blah-blah-bot-learn.py

Nous allons donc utiliser le hasard pour répondre à ces deux questions. Python nous fournit le module `random` à cet effet. Voyez plutôt :

```
import random

# Prints a number between 0 and 1
print random.random()
# Prints 'hello', 'world' or '!'
print random.choice(['hello', 'world', '!'])
```

On va donc choisir un mot au hasard pour commencer la phrase. Quant à la fin de la phrase : à chaque mot ajouté, on s'arrêtera avec une probabilité, par exemple, de $0,05$ (une chance sur 20). La suite devrait venir d'elle-même :

```
import random

database = {}
# ... Fill the database...

def speak():
    # Start with a random word
    words = [random.choice(database.keys())]
    # We'll stop with a probability of 0.05
    while random.random() > 0.05 :
        # Get the successors of the last word.
        # successors = TODO

        # If there is no successor, stop.
        if len(successors) == 0 :
            break

        # Otherwise, add one random successor
        # to the list of words
        # words.append( TODO )

    # Finally concatenate the list of words
    # and return it.
    return ' '.join(words)

for i in range(5):
    print speak()
```

2

3 Sauvegarder la base de données

Tout comme en Java (mais en bien plus simple et élégant), il est possible de « sauvegarder»des objets Python dans des fichiers. C'est le module `pickle` qui permet cette prouesse. Dans notre cas, ceci va vous permettre de ne pas perdre votre base de données à chaque redémarrage !

```
import pickle

database = {}
# ... Fill the database

# Open the 'database.bin' file in write mode
my_file = open('database.bin', 'w')
pickle.dump(database, my_file)
my_file.close()
# The 'database'ăobject is saved in the
# 'database.bin' file!

# Open the 'database.bin' file in read mode
my_file = open('database.bin', 'w')
my_file = open('database.bin', 'r')
restored_database = pickle.load(my_file)
my_file.close()
# 'database' and 'restored_database' have the
# same content!
```

3

4 Bonus : dessiner la base de données

Il est possible de représenter graphiquement la base de données : le graphe de l'introduction en est une illustration. Ce graphe a été fait à la main, mais il est possible de le dessiner automatiquement grâce à la commande `dot(1)`.

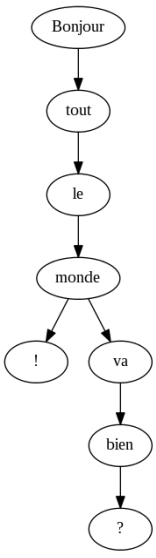
2. Voir le fichier source blah-blah-bot-speak.py
3. Voir le fichier source blah-blah-bot-pickle.py

Cette commande lit un fichier texte dans un format spécifique décrivant un graphe et produit une représentation visuelle de ce graphe. Par exemple, pour le schéma de l'introduction, on aurait pu enregistrer le contenu suivant dans le fichier `database.dot` :

```
digraph Database
{
    0 [label="Bonjour"];
    1 [label="tout"];
    2 [label="le"];
    3 [label="monde"];
    4 [label="!"];
    5 [label="va"];
    6 [label="bien"];
    7 [label="?"];
    0 -> 1;
    1 -> 2;
    2 -> 3;
    3 -> 4;
    3 -> 5;
    5 -> 6;
    6 -> 7;
}
```

4

Puis exécuter la commande `dot -Tpng -o database.png database.dot`, ce qui aurait produit l'image :



4. Voir le fichier source blah-blah-bot-dot.dot

Magie ! Essayez vous-mêmes de dessiner vos bases de données. Vous verrez : la plus grande difficulté est d'associer un numéro unique aux mots. Vous pouvez pour cela (par exemple) créer un autre dictionnaire `numbers` de sorte qu'à chaque ajout d'un nouveau mot dans `database`, le mot soit aussi ajouté dans `numbers` :

```
if word not in numbers :
    numbers[word] = len(numbers)
```

5 Bonus : améliorer l'algorithme

Eh oui, vous avez dû vous en rendre compteă : beaucoup de choix dans cet algorithme sont assez maladroits. Pensez à la gestion de la ponctuation, au choix du premier et du dernier mot, Les possibilités damélioration sont très diverses, et cest sans parler des autres types dalgorithmes. Si vous voulez explorer dautres pistes, allez jetter un coup dil sur le Web :

- *Dissociated Press* : `http://en.wikipedia.org/wiki/Dissociated_press`
- Chaînes de Markov : `http://en.wikipedia.org/wiki/Markov_chain`

6 Master Bonus Corporate : et les salons de discussion, alors ?

Blah blah bot est fantastique mais le but à la base, c'était de l'introduire dans les salons de discussion, non ? Je dis ça, parce que la console, vous n'irez pas lui parler très souvent

À l'instar de Perl, Python rassemble aujourd'hui *énormément* de modules venant compléter la bibliothèque stantard (fournie par défaut). Vous pouvez ainsi utiliser les modules irclib [5] pour communiquer avec les réseaux IRC et xmppy [6] [7] pour les réseaux XMPP/Jabber et ainsi donner à votre robot une portée internationale !

5. `http://python-irclib.sourceforge.net/`
6. `http://xmpppy.sourceforge.net/`
7. Mais il existe encore bien d'autres bibliothèques si celles-ci ne vous plaisent pas : faites une simple recherche sur le Web…