プロジェクト A　Julius を用いた音声対話システム超入門 (1)　2016/05/29

フリーの音声認識システムである Julius を用いて、簡単に音声対話システムを構築してみる。

1　Julius のインストール

まず、Julius 記述文法音声認識実行キットの Windows 版を `http://julius.osdn.jp/index.php?q=grammar-kit.html` からダウンロードする。Windows 版は zip アーカイブなので、適当なツールを用いて解凍する。

練習 1 展開したフォルダにある `HOWTO-ja.txt` を読んで、「果物購入タスク」を試してみよ。安価なものでよいので、マイクを用いて試すことが望ましい。

練習 2 Julius のシステム構成と認識のしくみを調べよ。

2　音声認識用文法の理解

一般に、音声認識システムは、言語モデルで生成できる文だけしか認識できない。今回ダウンロードした「記述文法音声認識実行キット」では、「文法」と呼ばれる規則で言語モデルが定義される。

Julius では「文法」は、文法規則と辞書から構成される。文法フォルダに含まれる 拡張子 `.grammar` のファイルが文法規則 (grammar) を記述したファイルである。拡張子 `.voca` のファイルが辞書ファイルである。この 2 つのファイルを理解すれば、文法から生成される文が理解できる。

「果物購入タスク」を例に、文法から生成される文を生成する方法を説明する。

「果物購入タスク」の辞書 `fruit.voca` を以下に示す。

```
% FRUIT
蜜柑            m i k a N
リンゴ          r i N g o
ぶどう          b u d o:
% NUM
0         z e r o
1         i ch i
2         n i:
3         s a N
4         y o N
5         g o:
6         r o k u
7         n a n a
8         h a ch i
9         ky u:
% KO
個              k o
% WO
を              o
% KUDASAI
ください        k u d a s a i
% NISHITE
にして          n i sh i t e
```

```
% DESU
です            d e s u
% NS_B
<s>             silB
% NS_E
</s>            silE
```

この辞書では、単語クラスごとに単語とその語がどのように発声されるかが定義されている。例えば、最初の部分では、FRUIT という単語クラスには、「蜜柑」「リンゴ」「ぶどう」という 3 語が属していることが定義されている。さらに、「蜜柑」という語は、m i k a N という 5 つの音素から構成されることが定義されている。(Julius における読みと音素記号の対応は、SampleGrammar フォルダの下にある type というフォルダにある type.voca を見よ。) ファイルの末尾の silB と silE は、それぞれ発話の開始前・終了後の無音部分に対応する音素記号である。

このファイルで定義される単語がどのようにつながって文になるのかを定義するのが .grammar ファイルである。「果物購入タスク」の .grammar ファイルを以下に示す。

```
S       :  NS_B FRUIT_N PLEASE NS_E
FRUIT_N :  FRUIT
FRUIT_N :  FRUIT NUM KO
PLEASE  :  WO KUDASAI
PLEASE  :  NISHITE KUDASAI
PLEASE  :  DESU
```

Julius の文法は文脈自由文法で書く。上記の式は、通常は以下のように書かれる。

```
S       →  NS_B FRUIT_N PLEASE NS_E
FRUIT_N →  FRUIT
FRUIT_N →  FRUIT NUM KO
PLEASE  →  WO KUDASAI
PLEASE  →  NISHITE KUDASAI
PLEASE  →  DESU
```

この文法を使って実際の文を生成する例を示す。

1. Julius の文法では、S は特殊な記号であり、文全体を意味する。1 番目の規則は、文全体の S という記号が、NS_B FRUIT_N PLEASE NS_El という記号列に置き換えられることをあらわす。
2. NS_B と NS_E は、文法規則ではこれ以上置き換えられない記号である。これらの記号は、上記の辞書を参照して単語に置き換える。NS_B は <s> に、NS_E は</s> に置き換えられる。これらは、それぞれ、Julius で文の最初、文の最後を示す特別な記号である。
3. FRUIT_N は、2 番目、3 番目の規則で置き換えることができる。ここでは、2 番目の規則で置き換えてみる。すると、FRUIT に置き換えられる。FRUIT は、文法規則ではこれ以上置き換えられない記号なので、単語に置き換える。辞書には、FRUIT に属する単語は 3 つある。このいずれに置き換えても構わない。
4. 引き続き、残った PLEASE を置き換える。
5. このような手順を繰り返し、全ての記号が単語に置き換えられるまで続ける。すると、例えば、「<s> 蜜柑 を下さい </s>」というような文が生成される。

練習 3「果物購入タスク」の文法を用いて、異なる文を 10 文生成せよ。また、その文を発話して認識させて、認識できるか確認せよ。

3 音声認識用文法の作成

Julius では、自作の文法で認識させることができる。作成しなければならないのは、拡張子 .grammar のファイルに書く文法規則と、拡張子 .voca のファイルに書く辞書項目である。

3.1 辞書の作成

「果物購入タスク」の辞書 fruit.voca の冒頭部分は以下のようになっている。

```
% FRUIT
蜜柑            m i k a N
リンゴ          r i N g o
ぶどう          b u d o:
% NUM
0         z e r o
1         i ch i
2         n i:
```

辞書では、1 行で 1 項目を表わす。フォーマットは、冒頭の「蜜柑」が、認識結果として表示される語の表記である。発音部分では、音素記号を用いて、このエントリの発音を定義する。ここで使える音素記号は、音素モデルが用意されているものだけである。音素記号は、半角スペースで区切る。どのような音素が使えるのか、また、音節 (おおよそ、かな一文字に対応する) がどのような音素で表現されるかを簡単に知るには、SampleGrammars フォルダにある type フォルダにある type.voca を見るのがよい。

0 のように複数の発音がある語は、発音ごとにエントリを定義する。

```
% NUM
0         z e r o
0         r e:
```

3.2 文法のコンパイル方法

辞書を修正したら、文法をコンパイルし直さなければならない。それらのツールを cygwin のシェル上で利用する方法について述べる。

1. まず、cygwin Bash Shell (もしくは cygwin terminal) を右クリックし、管理者で実行する。
2. Julius を展開したトップフォルダ (ディレクトリ) に移動する。

```
$ cd (インストールした場所)/grammar-kit-v4.3.1
$ ls
00readme-ja.txt     SampleGrammars_en   sample.wav
00readme.txt        bin         testfile.jconf
HOWTO-ja.txt        doc         testmic.jconf
LICENSE.md      hmm_mono.jconf      tools
README.md       hmm_ptm.jconf
SampleGrammars      model
```

3. ツールがおかれている bin/win32 ディレクトリに移動

```
$ cd bin/win32
```

4. (初めての場合) 作業用のディレクトリ tmp を作成

```
$ mkdir tmp
```

5. (初めての場合) bin ディレクトリのコマンドを実行可能にする。

```
$ chmod +rx *
```

6. 環境変数 PATH を設定

```
$ PATH=.:$PATH
```

途中に空白を含めてはいけない

7. (初めての場合) mkdfa.pl の 11 行目を編集

```
$usrtmpdir = "";
```

これを次のように変更する。

```
$usrtmpdir = "./tmp";
```

8. 文法のコンパイル

```
$ perl mkdfa.pl ../../SampleGrammars/fruit/fruit
```

これで、追加したエントリも認識できるようになったはずである。

練習 4「果物購入タスク」に果物名を 10 個追加して、認識できるか確認せよ。

練習 5「果物購入タスク」の辞書にエントリを追加して、「りんごを 3 つ (みっつ) 下さい」というように「つ」を使って個数を数える発話も認識できるようにし、認識できるか確認せよ。

3.3 文法の作成

「果物購入タスク」では、「リンゴ 1 個です」のように、同時には、一種類の果物しか注文できない。
そこで、「リンゴ 3 個と蜜柑です」というように二種類の果物が同時に注文できるようにしてみる。

```
S       :  NS_B FRUIT_N PLEASE NS_E
FRUIT_N :  FRUIT
FRUIT_N :  FRUIT NUM KO
PLEASE  :  WO KUDASAI
PLEASE  :  NISHITE KUDASAI
PLEASE  :  DESU
```

「りんご 3 個」の部分は、FRUIT_N なので、その部分を 2 回繰り返す文法規則を用意する。

```
S       :  NS_B FRUIT_NN PLEASE NS_E
FRUIT_NN : FRUIT_N
```

```
FRUIT_NN : FRUIT_N TO FRUIT_N
FRUIT_N :  FRUIT
FRUIT_N :  FRUIT NUM KO
PLEASE  :  WO KUDASAI
PLEASE  :  NISHITE KUDASAI
PLEASE  :  DESU
```

TO に関しては、新たに辞書エントリを追加する。
追加したら前節の手順で文法を再コンパイルする。
文法を変更した場合は、上手く変更できているか確認した方がよい。
ツールのディレクトリに、文法を使ってランダムに文を生成するツール generate がある。

```
$ generate -n 10 ../../SampleGrammars/fruit/fruit
```

とすることで、10 文生成する。

練習 6「果物購入タスク」の文法を改造して、一度に何種類でも注文できるようにせよ。(「リンゴ 3 個と蜜柑 2 個とぶどう下さい」などが認識できるようにする)

練習 7 自分で適当な文法を設計し、文法と辞書を作成せよ。作成できたら、ランダムに文を生成し、意図通りに作成できたか確認せよ。さらに、実際に発話してみて認識できるか確認せよ。