TR-IT-0207

1日でできる音声認識システム

シンガー ハラルド
Harald Singer

岩澤 亮祐
Ryousuke Iwasawa

1997.03

第1研究室で開発中のATRSPREC音声認識ツールキットを使用した簡単な音声認識システムの作成手順を紹介する。本稿では、システムに受け付けられるメッセージを規定する文法の作成、SPRECのモジュールをアプリケーションに実装する方法について述べられる。本レポートを参考に、誰にでもATRSPRECを用いて音声認識研究に関する実験用ツールや、アプリケーション作成をごく単期間に作成することができる。

目次

1 簡単な音声認識アプリケーション：atrCard

本稿では、ATRSPREC[1] を用いた音声認識システムの作り方について解説する。解説をわかりやすくするために、実際にアプリケーションを作成し、具体的な作成手順を説明する。

本稿で作成されたアプリケーションは、トランプのカードを指定すると、そのカードの絵を表示させるものである。カード指定の方法に、ATRSPREC を用いた音声認識機能を実現する。また、ラジオボタンを使用したカード指定の GUI も提供している。これは、GUI による入力方法と音声認識による指定方法とを比較するためである。

本稿で作成されたアプリケーションは、atrCard という名称で ATRSPREC のサンプルディレクトリ (`$ATRSPREC/sample/ATRcard`) に収められている。アプリケーション作成の際の参考にしていただきたい。

1.1節では atrCard の概要を説明する。atrCard の使用方法、機能について説明する。

1.2節では音声メッセージを atrCard に認識させるために必要なファイルの作成手順について説明する。作成されたファイルは ATRSPREC のモジュールに読み込まれ、音声認識処理に用いられるが、本稿ではファイル自体の詳細な説明までは触れない。

音声認識処理において、発話と非発話とをわける波形切り出しは [2] によって行なわれる。使用される音響モデルは ML-SSS[3] によって学習された文脈依存 HMM である。探索はマルチパスで行なわれ、第一パスで単語グラフが生成され、次に、単語グラフの再評価を行なう [4]。本稿において音声認識処理自体の流れや詳細についてはこれ以上触れない。

1.3節では、atrCard の実装について概説する。atrCard のメインルーチンはオブジェクト指向プログラミング言語 Python/Tk （以下 Python）を使用して作成されている。この節では主に ATRSPREC の音声認識処理を Python 上で実行させる方法について述べられている。Python 言語自体の説明についてこの節で詳しく触れることはない。詳しくは [5][6] を参考にしていただきたい。

1.1 atrCard の概要

まず atrCard を起動してみよう。atrCard は SPREC のサンプルディレクトリに収められている。サンプルディレクトリ上から、シェルスクリプト`atrCard.csh` を実行する。

```
% cd $ATRSPREC/sample/ATRcard
% atrCard.csh
```

`atrCard.csh` は、atrCard の実行に必要な環境設定を行ない、メインの Python スクリプト`atrCard.py` を実行する。`atrCard.csh` が行なう環境設定などについては、付録 A を参照。また、atrCard.py が起動時に読み込むコンフィグレーションファイルが付録 B に収録してある。

なお、atrCard は HP-UX、OSF1 で動作確認されている。

atrCard を起動すると、図 1のような画面が立ち上がる。

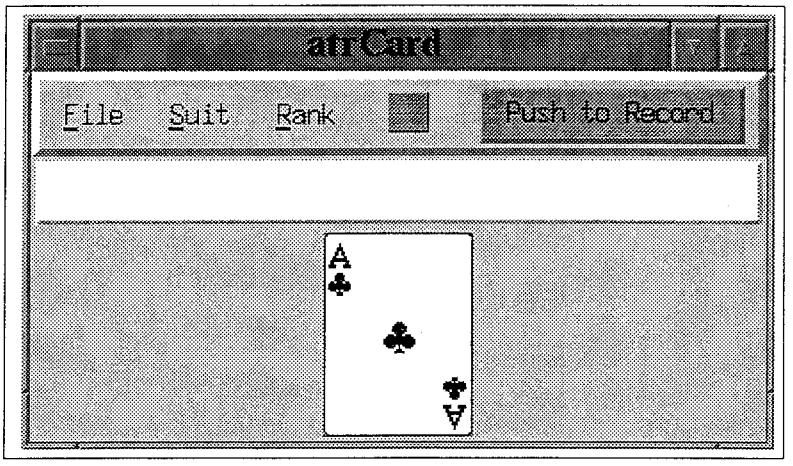

図 1: atrCard

トランプの、組と番号を指定するとそのカードが表示される、単純なアプリケーションである。ウインドウの一番下にトランプのカードが表示される。メニューバーとカード表示ディスプレイの間のウインドウには、音声入力の認識結果が表示される。メニューバーコマンドについて以下に概説する。

File メニュー [Quit]：アプリケーションを終了する。

Suit ラジオボタン カードの組を選択するラジオボタン。

Rank ラジオボタン カードの番号を選択するラジオボタン。

Push to Record ボタン このボタンをクリックすると、ボタンの色が変わる。赤色のときは、音声入力ができる状態である。

緑色のときは、音声入力はできない。

カードの組と種類を入力する方法は次のとおりである。

チェックボックスで指定 [Color] メニューから組を、続けて [Rank] メニューから番号を選択。

音声入力 [Push to Record] ボタンをクリックし、ボタンが赤色の時にマイクロフォンから入力。「[組] の [番号]」の順番で発声する（例：「ええと、ダイヤのキング」「クローバーの 7」など）。音声入力により結果を表示している例を図 2に示す。

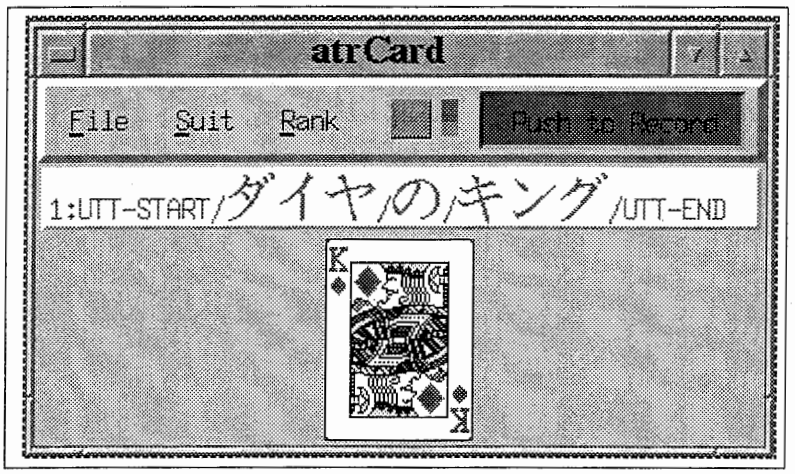

図 2: atrCard

1.2　認識辞書、言語モデルファイルの作成

この節では atrCard がどんな音声入力を認識するかを決定し、ATRSPREC で認識処理を行うためのファイルを作成する。前節で少しふれたように、atrCard でカード表示をさせるためには次のように話せばよかった。

「ハートの六」
「ええと、スペードのジャック」

これは、‘[ええと、あの etc.](カードの組) の (カードの番号)’ という構文を ATRSPREC が認識できる唯一の文法と規定しているからである。

したがって、atrCard では、音声入力が‘[ええと、あの etc.](カードの組) の (番号)’ という形をとっていれば正しく認識される (言い換えれば、どんな風に話したとしても atrCard は‘[ええと、あの etc.](カードの組) の (カードの番号)’ という形でしか音声入力を認識できない)。

ATRSPREC の音声認識処理で認識可能な単語や構文を規定するファイルを認識辞書ファイル、言語モデルファイルと呼ぶ。以下では、認識辞書ファイルと言語モデルファイルの作成について説明する。

atrCard で音声認識に使用するために以下の認識辞書ファイルと言語モデルファイルを作成する。

```
card.lex card.fsa.bin
```

言語モデルファイルを作成するために次のようなファイルを作成する。

```
card.class card.class.def card.htkin card.htkout card.ANS
```

これらのファイルが作成される手順を図3に示す。以下のセクションではこの順番にしたがってファイルの作成方法を説明する。実際に作成されたファイルは`$ATRSPREC/sample/ATRcard/GRAMMAR` にある。なお、ここで作られるファイル (`card.htkin card.htkout` を除く) の仕様を、ATRSPREC のマニュアル [1] の Index—file format から調べることができる。

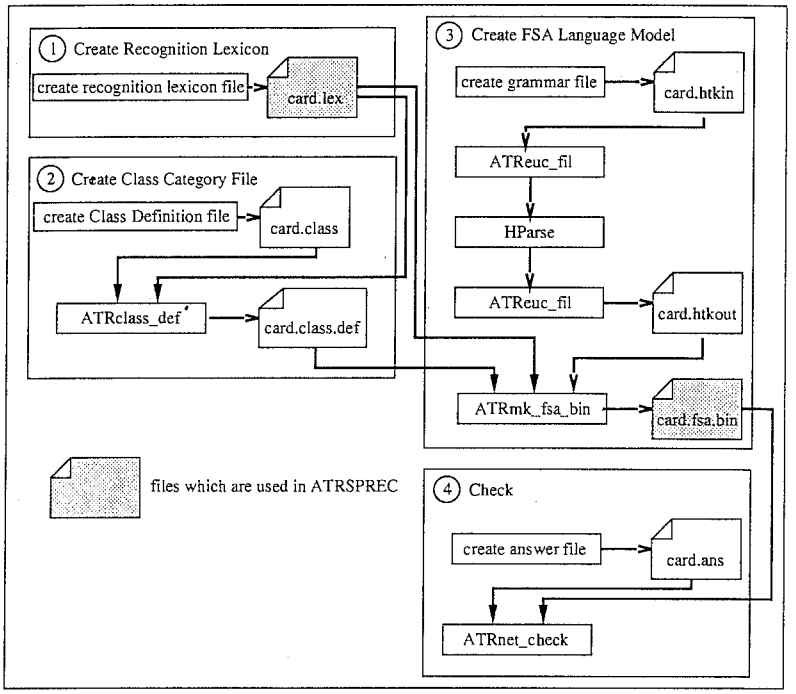

図 3: 文法作成手順

(1.2.1) 認識辞書ファイルを作る

`card.lex` というファイル。音声入力で ATRSPREC が認識できる単語を定義する。作り方としては、

1. 自分で作る
2. ATRSPREC の辞書メンテナンスツールを利用する

音声入力の使用する単語数が限られているため、ここでは自作でファイルを作成する。ATRSPREC の辞書メンテナンスツールの詳細については、ATRSPREC のマニュアル [1] の Reference Manual—Lexicon/Dictionary Tools セクションを参考にされたい。

単語の種類は、カードの組、カードの番号、'の'、'あの'、'ええ'、'ええと' を受け付けることにする。ファイルの内容は次のようになる。

```
card.lex
        5 [UTT-START] -         # UTT-START
        6 [UTT-END]   -         # UTT-END
        40000 [ええと]  e e t o {|o} {|-}  # filler|
        40001 [あの]  a n o {|o} {|-}      # filler|
        40002 [ええ]  e e  {|-}  # filler|
```

```
10000 [の]    n o {|-}      # part
# 10001 [一]    i ch i    # 数詞 |
10002 [二]    n i {|i}  # 数詞 |
10003 [三]    s a ng    # 数詞 |
10004 [四]    j o ng    # 数詞 |
10005 [五]    g o       # 数詞 |
10006 [六]    r o k u   # 数詞 |
10007 [七]    n a n a   # 数詞 |
10008 [八]    h a ch i  # 数詞 |
10009 [九]    k j u u   # 数詞 |
10010 [十]    zh j u u  # 数詞 |
10011 [エース] e e s u    # 絵札 |
10012 [ジャック] zh j a q k u    # 絵札 |
10013 [クイーン] k u i i ng    # 絵札 |
10014 [キング] k i ng g u    # 絵札 |
10015 [スペード] s u p e e d o {|-} # 色 |
10016 [ダイヤ] d a i j a {|-}      # 色 |
10017 [ハート] h a a t o {|-}      # 色 |
10018 [クラブ] k u r a b u {|-}    # 色 |
10019 [クローバー] k u r o o b a a {|-} # 色 |
```

単語 id、単語表記、音素記号列、コメントの順に記述する。単語 id から一意に単語が定まるようにする。単語 id は 0 番から 99 番までをシステムの予約語に使用する。UTT-START、UTT-END は発話の始まり、終りを示す予約語である。残りの単語には 5 桁の番号をふった。

音素記号列中のので囲まれている音素記号は、発音時に省略されることが許される。また、音素記号のハイフンは無発音状態 (pause) を表す。たとえば、40000 番「ええと」は「ええとぉ」または「ええと、」と発音しても認識される。日本語音声認識に使用される音素記号の一覧と音素列の記述例を付録 C に示している。

以降はコメントとなる。コメント部分は次項で説明する単語のクラスを作成するのに必要となる。| で区切られたコメントがクラス分けのキーとなる。ここで、(カードの組) は色、(カードの番号) は数詞と絵札という分類になっている。card.lex は単語の種類も数も少ないので 1 つのキーで十分であろう。

(1.2.2)　単語のクラス分類ファイルをつくる

名詞、動詞とか、地名、人名、といったカテゴリー別に単語を分類する作業である。この単語の分類をクラスと呼ぶ。音声認識に用いられる認識辞書ファイルにある単語は必ずどれか 1 つのクラスに属する。

まず、クラス分けのルールを記述するファイルをつくる。これをクラス定義ファイルと呼ぶ。card.class というファイルである。

card.class

```
filler=[filler|*]
number=[数詞 |*]
picture=[絵札 |*]
suit=[色 |*]
の=10000
UTT-START=5
UTT-END=6
```

'クラス名 = 分類キー'、というフォーマットになる。分類キーは、認識辞書の通し番号か、コメント部分の記述に沿ったものである。分類キーと認識辞書のコメントでマッチングを行なうため、分類キーのフォーマットが、認識辞書のコメント部分の縦棒の数、コメントの数に一致していないとエラーになる。ファイルができたら、次のようなコマンドを実行し、クラス分類ファイルを作成する。

```
% $ATRSPREC/bin/ATRclass_def -lexicon=card.lex \
    -rule=card.class -class_def=card.class.def
```

-lexicon が認識辞書ファイル、-rule が今作ったクラス定義ファイル、-class_def はクラス分類ファイルである。もしエラーがあった場合、以下のようなメッセージが出力される。

```
Class of word 5 is not defined.
Class of word 6 is not defined.
```

この場合、クラス定義ファイルに5番と6番のための記述が抜けていないか、間違っていないか、確認したら、もうコマンドを一度実行する。

正しくファイルがつくられると、クラス分類ファイルcard.class.def の内容は次のようになる。

```
card.class.def
        $filler={ 40000 , 40001 , 40002 };
        $number={ 10002 , 10003 , 10004 , 10005 , 10006 , 10007 , 10008 , 10009 , 10010 };
        $picture={ 10011 , 10012 , 10013 , 10014 };
        $suit={ 10015 , 10016 , 10017 , 10018 , 10019 };
        $の={ 10000 };
        $UTT-START={ 5 };
        $UTT-END={ 6 };
```

(1.2.3) 文法を定義し、SPREC で使用できるようにする

音声認識処理のための文法を定義し、言語モデルファイルを作成する。ATRcard は言語モデルとして FSA モデルを使用する。モデル作成の過程で HTKtools を使用する。HTKtools を使用するのに必要な環境設定については、付録 A を参照していただきたい。

まず、文法を定義するファイルcard.htkin を作成する。文法定義ファイルは HTK フォーマットにしたがって書かれたものである。'[ええと、あの etc.](カードの組) の (カードの番号)' という文法だと次のようになる。

```
card.htkin
        (
                UTT-START
                [filler] suit の (number|picture)
                UTT-END
        )
```

先ほどつくったクラスが記号にはさまれて並んでいる。[] に囲まれたクラスに属する単語は省略可で、| はどちらかのクラスに属する単語が許されるという意味である。記号の意味、ルールの記述方法についての詳細は HTK-tool のマニュアルを参照していただきたい。

次に、HTK 文法定義ファイルを SLF 文法定義ファイルに変換するという作業が必要となる。HTK-SLF 変換ツールを実行する前に、クラス名が日本語の場合、フィルターにかけなければならない。

ただしこのフィルターは EUC 日本語コードのみ受け付けるため、card.htkin のコードを EUC にしなければならない。ファイルを mule で編集しているなら、C-x-k f というコマンドで、文字コードを変換できる。コードの種類は*euc-japan* と指定すればよい。

card.htkin の準備ができたら、次のようなコマンドを実行し、フィルターにかける。

```
% cat card.htkin | $ATRSPREC/bin/ATReuc_fil -i > ! card.htkin_fil
```

次に、HTK-SLF 変換ツールを実行する。

```
% HParse card.htkin_fil  card.htkout_fil ; HFree
```

できたファイルを再度フィルタにかける。

```
% cat card.htkout_fil | $ATRSPREC/bin/ATReuc_fil -o > ! card.htkout
```

正常に実行できたら、SLF 文法定義ファイルcard.htkout は次のようになる。

card.htkout

```
VERSION=1.0
N=10   L=11
I=0    W=!NULL
I=1    W=!NULL
I=2    W=UTT-START
I=3    W=filler
I=4    W=suit
I=5    W=の
I=6    W=number
I=7    W=!NULL
I=8    W=picture
I=9    W=UTT-END
J=0     S=9     E=1
J=1     S=0     E=2
J=2     S=2     E=3
J=3     S=2     E=4
J=4     S=3     E=4
J=5     S=4     E=5
J=6     S=5     E=6
J=7     S=6     E=7
J=8     S=8     E=7
J=9     S=5     E=8
J=10    S=7     E=9
```

card.htkin をコンパイルし FSA 言語モデルファイルを作成する。

```
% $ATRSPREC/bin/ATRmk_fsa_bin -lexicon=card.lex -class_def=card.class.def \
    -FSA_net=card.htkout -FSA_bin=card.fsa.bin
```

(1.2.4)　言語モデルファイルをチェックする

音声認識結果の例を記述したファイルをつくる。これを正解ファイルと呼ぶ。

card.ANS

```
5
10015
10000
10002
6
5
10016
10000
10011
6
5
40001
10016
10000
10011
6
5
10003
```

```
10000
10016
6
5
10011
6
```

正しい認識結果は、'UTT-START/ えー / ハート / の / 六 /UTT-END' のような形になる。このような単語の並びを単語 id で記述する。これと先ほど作成した言語モデルファイルをチェックプログラムにかける。

```
% $ATRSPREC/bin/ATRnet_check -FSA_bin=card.fsa.bin -answer=card.ANS
```

正しく言語モデルファイルがつくられていれば、

```
Sample 1 OK.
Sample 2 OK.
Sample 3 OK.
Sample 4 Fail.
Sample 5 Fail.
```

というメッセージがでる。正解ファイル card.ANS のサンプル 1 ～ 3 は正しい認識結果を記述しているが、サンプル 4 はカードの組と番号の順番が違っており、サンプル 5 は単語が足りないため、エラーとなっている。

1.3 アプリケーションの実装

この節では、前節で作成したファイル（`card.lex`、`card.fsa.bin`）を使用する音声認識システムを atrCard に実装する。atrCard のメインルーチンはプログラミング言語 Python/tk （以下 Python) で実装される。Python は高水準のデータ型や GUI ツールキットを含む、単純だが非常に強力なプログラミング言語である。また完成度の高いオブジェクト指向言語でもある。Python はインタプリタによる言語なので、コンパイルやリンクは不要であり、さらに、C 言語で新しい Python の関数やデータ型を実装することが容易にできる。これらの特徴から、プロトタイピングや、C 言語で開発したプログラムの拡張を行なうのに適した言語だといえる。

ATRSPREC の音声認識処理は Python スクリプト上で実行される。ATRSPREC の音声認識処理を実行するために、C 言語とのインターフェイスを提供する Python の関数が用意されている。このライブラリ関数は大別すると次の 3 種類に分けられる。

音声認識処理のイベントハンドリングループ実行関数 ATRSPREC の音声認識処理を Python 上で実行する。

C コールバック関数 音声認識処理中に Python の関数を呼び出し、実行する

ATRSPREC ライブラリ関数のラッパー 音声認識結果を処理するために ATRSPREC で提供されているライブラリ関数を Python 上で実行する

これらのライブラリ関数はシェアードライブラリで提供されているため、必要なときに直接 Python のソースコードからロードし、使用することが可能である。

(1.3.1) 処理のながれ

ATRSPREC と Python/C インターフェイスモジュールの基本的なデータのながれを図 4に示す。それぞれのモジュールはイベントハンドリングループの中で呼ばれるため、正しくデータが与えられれば、モジュールのどの部分からでも音声認識処理を動作させることができる。

このイベントループを Python 上で実行する関数がシェアードライブラリとして提供されるため、Python と容易にリンクさせることができる。したがって、全ての処理 (認識、認識結果をつかった Python での処理) を 1 プロセスとして実行できる。

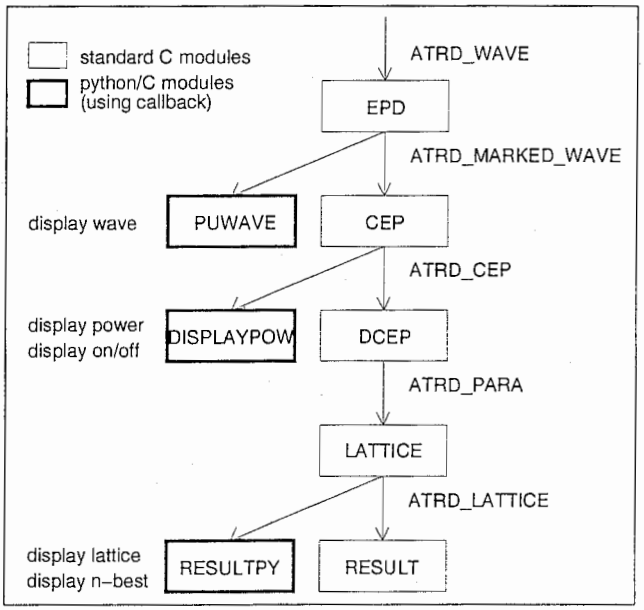

図 4: Basic modular architecture

(1.3.2) Python と C プログラムのリンク

atrCard での Python と C プログラムの関係を図 5に示す。

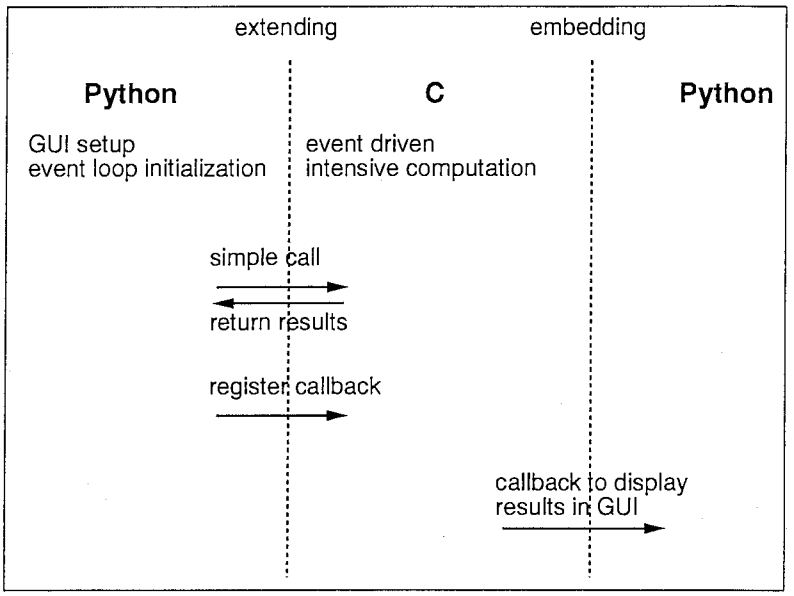

図 5: Python/C interface

ATRSPREC のモジュールは純粋に C のみのプログラムである。Python は GUI を作成し、ATRSPREC のイベントハンドリングループを初期化する。イベントが発生したらイベントの内容にしたがって ATRSPREC のモジュールが音声認識処理を実行していく。Python 側から ATRSPREC ライブラリ関数を使って認識結果を処理することができる。

(1.3.3) C のコールバック関数

イベントが発生した時に ATRSPREC の音声認識処理から Python の関数を呼ぶことができる (図 4の太枠で囲まれた処理に相当する)。音声認識の結果を利用したコードを Python で記述することができるのである。atrCard でも使用している`EventLoop` クラス (`$ATRSPREC/script/python/tk/atrEvent.py`) は、ATRSPREC のイベントループの初期化、実行、コールバック関数の定義や初期化を行なうクラスである。例えば、音声認識の最終結果が出た時点で呼び出される`NextResult` という関数が、`EventLoop` クラスに定義されている。

```
def NextResult( self, lattice_ptr = None ):
    print 'default NextResult', lattice_ptr
```

ここでは、認識結果を取り込み、認識結果からカードの組と番号を決定し、カードの表示を行なう。そのために、`EventLoop` クラスを継承する新しいクラス`MyVoiceInput` クラスで、次のようなコードを記述する。

```
001     def NextResult(self,lattice_ptr):
002         # make new n_best
003         self.CreateInputString( lattice_ptr )
004
005         # display n_best
006         self.n_best_disp.ReDraw(self.n_best)
007
008         # define suit and rank from n_best wordids
009         num = apply(GetCardIndex,self.ParseInputString())
010
011         self.x.configure(image=self.img[num-1])
```

3行めでC側から渡されたラティスのポインタから認識結果の文字列をPython側に保存する。6行めで結果の文字列をウィンドウに表示する。9行めは結果の文字列から何番目のカードかを決定し、11行めでカードの絵柄を変更している。

このように、コールバック関数を再定義することによってやりたい処理を自由に設定することができる。

`MyVoiceInput`クラスを含む、atrCardのメインファイル`atrCard.py`のソースコードは付録 D を参照していただきたい。

謝辞

本報告書作成の機会を与えて下さったATR音声翻訳通信研究所の山崎泰弘社長、並びにATR音声翻訳通信研究所第一研究室の匂坂芳典室長に深く感謝致します。また、本報告書の作成に際し有益な助言、協力をいただいた山本研究技術員、西野研究技術員をはじめとするATR音声翻訳通信研究所の研究員諸氏に感謝の意を表します。

参考文献

[1] ATR ITL. **SPREC** *User, Programmer and Reference Manual (Ver04)*, 1997.
(online via http://www.itl.atr.co.jp/~singer/software/SPRECDOC/release.html).

[2] B. Reaves. Parameters for noise robust speech detection. In *Proc. Acoust. Soc. Jap.*, pages 145–146, Fall 1993.

[3] H. Singer and M. Ostendorf. Maximum likelihood successive state splitting. In *Proc. ICASSP*, pages 601–604, Atlanta, 1996.

[4] T. Shimizu, H. Yamamoto, S. Matsunaga, and Y. Sagisaka. Spontaneous dialogue speech recognition using cross-word context constrained word graphs. In *Proc. ICASSP*, pages 145–148, 1995.

[5] M. Lutz. *Programming Python.* O'Reilly, 1996.

[6] A. Watters, G. van Rossum, and J. Ahlstrom. *Internet Programming with Python.* M&T, 1996.

付録 A　動作環境

atrCard の実行環境 atrCard は以下のような環境で動作する。

Python バージョン 1.4— ダイナミックローディングが可能であること（Python の実行ファイルを作成する際に、コンパイルオプション -Wl,+s を指定する）。コンパイルには、gcc（2.7.2 以上）または cc を用いる。

Tcl/Tk バージョン 7.4(tcl)/4.0(tk) 以上 —Python の GUI には Tk のプログラムが使用されている (Tkinter というモジュールが Tk のライブラリと Python のインタフェイスを提供している)

gmake バージョン 3.67 以上

gcc バージョン 2.7.2 以上

Python/SPREC インタフェイスを用いたアプリケーションを使用するには、次のような環境の設定が必要である。

OSF1, SunOS and Linux

```
$ setenv ATRSPREC /usr/local/SPREC/r04a10
$ setenv LD_LIBRARY_PATH $ATRSPREC/shlib
```

HP-UX

```
$ setenv ATRSPREC /usr/local/SPREC/r04a10
$ setenv SHLIB_PATH $ATRSPREC/shlib
```

atrCard.csh スクリプト HP-UX/OSF1 の環境変数を設定し、スクリプトatrCard.py を実行する。ATRSPREC をホームディレクトリにインストールしていなくても、atrCard の実行が可能である。

```
#!/usr/local/bin/tcsh -f
# $Header: $ATRSPREC/sample/ATRcard/atrCard.csh,v1.3 1997/02/28 01:03:14 riwa Exp $$
# script for atrCard

# default SPREC location
# setenv ATRSPREC /usr/local/SPREC/latest_release

# finding the correct shared libraries for SPREC
if ( $OS == "HP-UX" ) then
        setenv SHLIB_PATH $ATRSPREC/shlib/bugfix:$ATRSPREC/shlib
else
        # OSF1, SunOS, Linux
        setenv LD_LIBRARY_PATH .:$ATRSPREC/shlib/bugfix:$ATRSPREC/shlib
endif

# run atrCard
cd $ATRSPREC/sample/ATRcard
/home/atrh31/riwa/local/$OS/bin/python atrCard.py -config=config
```

HTKtools の環境設定 環境設定ファイル (.cshrc など) を編集して使用する環境にあわせて以下のような記述を加える。

```
# for HTK V2.0
if ( $OS == "SunOS" ) then
        setenv HBIN /usr/local/HTK/HTK_V2.0
        setenv CPU sun4
        set path = ( $path $HBIN/bin.$CPU )
endif

if ( $OS == "HP-UX" ) then
       setenv HBIN /usr/local/HTK/HTK_V2.0
       setenv CPU hp700
       set path = ( $path $HBIN/bin.$CPU )
endif
```

付録 B　コンフィグレーションファイル

atrCard が実行時に読み込むファイルである。ATRSPREC のモジュールへのさまざまなオプションを指定する。例えば今回作った辞書や文法ファイルは 'ATRlattice' に対して指定されている。各モジュールのオプションの詳細については ATRSPREC のマニュアル [1] の Reference Manual—Executables セクションを参照していただきたい。

$ATRSPREC/sample/ATRcard/config

```
Demo:OnOffLine=online          # offline, online

I/Ocontrol:inputFormat=NULL
I/Ocontrol:inputParamSize=90
I/Ocontrol:inputParamType=short
I/Ocontrol:inputFd=stdin
I/Ocontrol:inputByteorder=BigEndian
I/Ocontrol:outputFormat=NULL
I/Ocontrol:outputFd=stdout
I/Ocontrol:outputByteorder=BigEndian
I/Ocontrol:rpcNumber=2

ATRepd:SamplingFrequency=12000
ATRepd:energyThreshold=100
ATRepd:upperDispersionThreshold=50
ATRepd:lowerDispersionThreshold=10
ATRepd:orderInMs=300
ATRepd:alpha=1.06        # 1.02 originally, 1.1 for a realistic demo
ATRepd:skewInMs=300
ATRepd:s1TimerLimitInMs=200
ATRepd:epdFramesInMs=2000
ATRepd:framePointsInMs=7.5
ATRepd:inputByteOrder=none
ATRepd:sendNonspeech=0
ATRepd:track=0

ATRwave2cep:inputParameter=waveRaw
ATRwave2cep:Preemphasis=0.98
ATRwave2cep:FrameLength=20
ATRwave2cep:FrameShift=10
ATRwave2cep:SamplingFrequency=12000
ATRwave2cep:TimeWindow=hamming
ATRwave2cep:LagWindowFactor=0.01
ATRwave2cep:LpcOrder=16
ATRwave2cep:CepstrumOrder=16
ATRwave2cep:FrequencyWarping=linear
ATRwave2cep:DebuggingLevel=0
ATRwave2cep:IntermedCepstrumOrder=32

ATRdisplaypow:position=700,0

ATRcep2para:CepstrumOrder=16
ATRcep2para:DeltaCepstrumWindow=9
ATRcep2para:deltaCepstrumPadding=repeat
ATRcep2para:DDCepstrumWindow=3
ATRcep2para:DDCepstrumPadding=zero
ATRcep2para:rho=0.0
ATRcep2para:OutputParameter=pow+cep(16)+dpow+dcep(16)

ATRlattice:lexicon=/usr/local/SPREC/latest_release/sample/ATRcard/GRAMMAR/card.lex
ATRlattice:FSA=/usr/local/SPREC/latest_release/sample/ATRcard/GRAMMAR/card.fsa.bin
ATRlattice:amname=/usr/local/SPREC/latest_release/sample/ATRlattice/model.Mi.bin,\
                  /usr/local/SPREC/latest_release/sample/ATRlattice/model.Fi.bin
ATRlattice:lmscale=4.0,8.0
ATRlattice:work_area=200,50
ATRlattice:beam=50,50
ATRlattice:word_merge=all
ATRlattice:null_trance=OFF

ATRresult:N_best=1
ATRresult:N_best_out=/dev/null
ATRresult:Lattice_out=/dev/null
```

付録 C　音素セット

表 1に、26 個の音素記号と発音の例を示す。

表 1: 日本語音声認識に用いられる音素セット

label	example
a	akai (赤い)
b	benri (便利)
ch	chikara (力)
d	deguchi (出口)
e	eng (円)
g	gizhjutsu (技術)
h	hata (旗)
i	ima (今)
j	jama (山)
k	keqtei (決定)
m	mang (万)
n	namae (名)
ng	dengwa (電話)
o	otoko (男)
p	pang (パン)
q	aqsari (あっさり)
r	rongbung (論文)
s	sa (差)
sh	shjori (処理)
t	tani (谷)
ts	tsukau (使う)
u	ushi (牛)
w	dengwa (電話)
z	angzeng (安全)
zh	azhi (味)
-	pause or silence

付録 D　atrCard.py のソースコード

今回作成したメインのソースコードである。このファイルでインポートしているその他の Python のソースコードについては、`$ATRSPREC/script/python/tk` に収録されている。図 6に、atrCard で使用されるすべての Python モジュールのリストを示す。

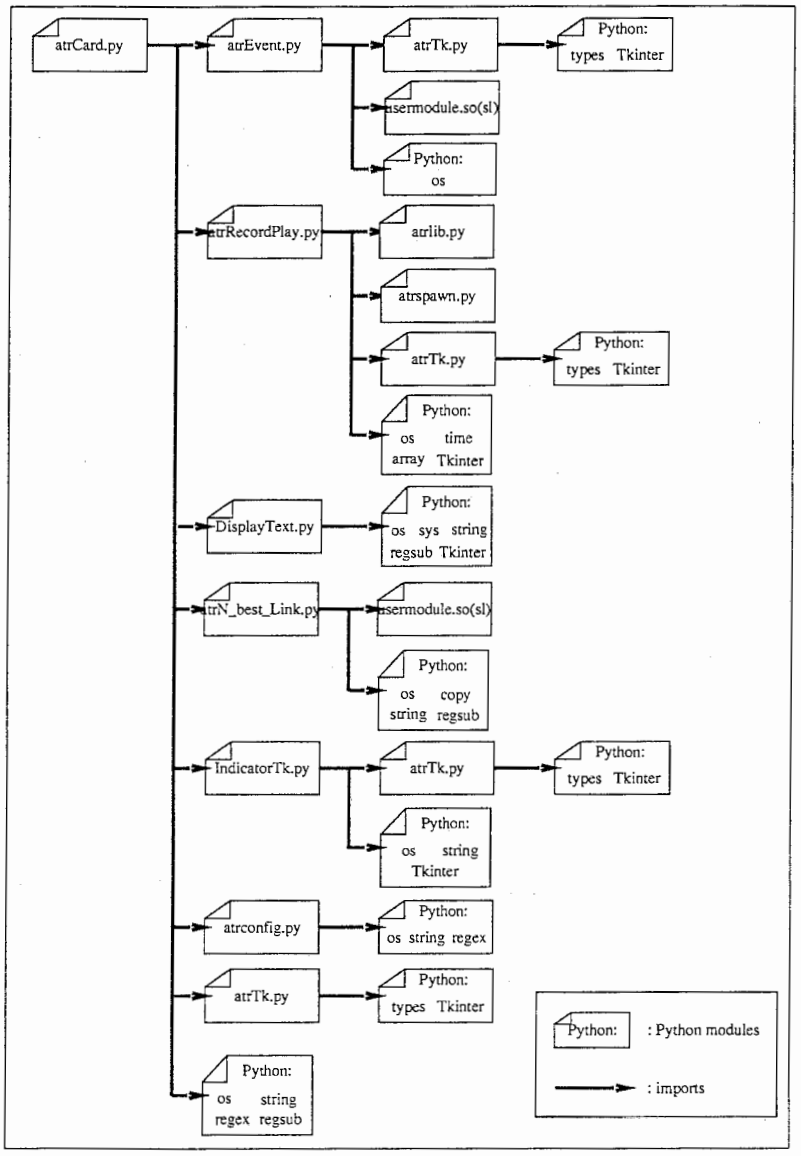

図 6: Python module listing of atrCard

$ATRSPREC/sample/ATRcard/atrCard.py

```
#!/usr/local/bin/python
""" $Header: /homes/singer/CVSROOT/SPREC/sample/ATRcard/atrCard.py,v 1.9 1997/02/28 01:03:15 riwa Exp $
H. Singer and R. Iwasawa, Dec 1996
Simple card viewer: example of Python/C integrated module.
example:
        atrCard.py -config=config
"""
import os
import string, regex, regsub
import Tkinter

os.sys.path.append(os.environ['ATRSPREC']+'/script/python/tk')
import atrEvent, atrN_best_Link, DisplayText, atrRecordPlay, IndicatorTk, atrTk, atrconfig

def GetCardIndex(c='ダイヤ',r='二'):
    """ mapping between lexical entries and file number
    """
    Suit = {'クローバー':1,'クラブ':1, 'ダイヤ':2,'ハート':3, 'スペード':4}
    Rank = {'エース':1,'二':2,'三':3,'四':4,'五':5,'六':6,
            '七':7,'八':8,'九':9,'十':10,'ジャック':11,'クイーン':12,'キング':13}
    number = (Rank[r] - 1) * 4 + Suit[c]
    return number

class MyVoiceInput(atrEvent.EventLoop):
    """all of voice input stuff.
       --get recognition result
       --display result
       --parse input string from result
    """

    def __init__(self,Mode='online',master=None,num_n_best=1):
        # initialize superclass first
        atrEvent.EventLoop.__init__(self,
                                    Master=master,
                                    Mode=Mode)

        # recognition result (to be instance of N_best class)
        self.n_best = None

        # function set of creating lattice/n_best
        self.slf = atrN_best_Link.My_SLF(num_n_best)

        # init n_best display
        self.n_best_disp = DisplayText.DisplayText (master=master,
                                                    cnf ={
                                                        'n_best_show':1,
                                                        'bg':'white',
                                                        'fg':'red',
                                                        'relief':'raised',
                                                        "height":1,
                                                        "width": 48,
                                                        "wrap":"none",
                                                        'kanjifont':'k24'
                                                        })

        # get gif data of 52 cards
        self.img = []
        for i in range(1,53):
            self.img.append(Tkinter.PhotoImage(file='GIF/%d.gif'%i).zoom('2'))

        # create card display. stratup picture is " ace of club"
        self.card_disp = Tkinter.Label(master=master,
                                       image=self.img[0])
        self.card_disp.pack()

        # create dictionaly from lexicon
        self.ParseLexFile(fname='GRAMMAR/card.lex')

    def ParseLexFile(self,fname='../card.lex'):
        """parse lexicon and create dictionary whose key is wordid
        """
        fd = open(fname)
        buf = map(lambda x: string.strip(x), fd.readlines())
        fd.close()

        # 10002 [二]    n i {|i} {|-} # 数詞 |
        #   ----> { '10002': ['二','n i {|i} {|-}','数詞'] }
        val = regex.symcomp('\(<s>[0-9]+\) \[\(<k>.*\)\] \(<p>.*\)\#\(<m>.*\)')

        dict={}
        for x in buf:
```

```
            if val.match(x)!=-1:
                dict[val.group('s')] = [val.group('k'),
                                        string.strip(val.group('p')),
                                        string.strip(val.group('m'))]
        self.lex = dict

    def DisplayPow(self,val1,val2):
        """ override the default callback from C
        """
        if val1==0:
            self.OnOffIndicator.updatedisplay(val2)
        elif val1==1:
            self.LevelIndicator.updatedisplay(val2)
        # self.n_best_disp.update_idletasks()

    def NextResult(self,lattice_ptr):
        """override default callback:EventLoop.NextResult
        """
        # make new n_best
        self.CreateInputString( lattice_ptr )

        # display n_best
        self.n_best_disp.ShowText(self.n_best)

        # define suit and rank from n_best wordids
        num = apply(GetCardIndex,self.ParseInputString())
        self.card_disp.configure(image=self.img[num-1])

    def CreateInputString(self, lattice_ptr = None ):
        """ make n_best from lattice_ptr.
            (part of callback function, called from callbackfunction "NextResult")
        """
        # read the next utterance
        self.slf.scanutteranceblock( lattice_ptr )

        # create and parse n_best
        self.slf.createN_best()
        self.slf.parseN_best()

        # "save" it
        self.n_best = self.slf.copyN_best()

    def ParseInputString(self):
        """set suit and rank from wordids
        """
        wordids = self.n_best[0].wordids
        Rank = None
        Suit = None
        for wordid in wordids:
            if self.lex[wordid][2] == '絵札|' or self.lex[wordid][2] == '数詞|':
                Rank = self.lex[wordid][0]
            elif self.lex[wordid][2] == '色|':
                Suit = self.lex[wordid][0]
        return Suit,Rank

    def quit(self):
        """terminate SPREC modules, and exit application
        """
        self.terminate(self.NORMAL_EXIT)

class SprecApp:
    def __init__(self,cnfopt={}):
        self.CreateWidget(cnfopt)

    def CreateWidget(self,cnfopt={}):
        """create instances, and show application displays, buttons, menubar
        """
        ##########################################
        # main initialization
        ##########################################
        # set root window
        root = Tkinter.Tk()
        root.geometry('+0+0')

        # connect mainloop() to self
        self.mainloop = root.mainloop

        # set configuration options from config file
        Mode = cnfopts['Demo:OnOffLine']
        N_best = string.atoi(cnfopts['ATRresult:N_best'])
        SamplingFrequency = string.atof(cnfopts['ATRwave2cep:SamplingFrequency'])

        # create a menubar
```

```
        MenuBar = atrTk.SimpleMenuBar(master=root)

        # create a voice input
        myvoice = MyVoiceInput(Mode=Mode,master=root,num_n_best=N_best)

        ########################################
        # set instances as new members of myvoice
        ########################################
        # create an audio input stuff.
        # function: record/play button, execution of external audio input device
        myvoice.AudioFrame = atrRecordPlay.AudioFrame(master=MenuBar.mBar,
                                                samplingfrequency=SamplingFrequency)
        myvoice.AudioFrame.pack(side='right',fill='x')

        # delete 'play' button, no need to use
        myvoice.AudioFrame.playbutton.forget()

        # override default AudioFrame method
        def PySendEvent(typeD,typeT,buf):
            """ redefine for AudioFrame """
            PySendEvent = atrN_best_Link.user.PySendEvent
            PySendEvent(typeD,typeT,buf)

        myvoice.AudioFrame.PySendEvent = PySendEvent

        # create a power level indicator
        myvoice.LevelIndicator = IndicatorTk.LevelIndicator(master=MenuBar.mBar,
                                                cnf={'width':10,
                                                     'height':20})
        myvoice.LevelIndicator.pack(side='right',fill='x')

        # create an activation indicator
        myvoice.Colors= [ 'green','yellow','red']
        myvoice.OnOffIndicator = IndicatorTk.OnOffIndicator(master=MenuBar.mBar,
                                                cnf={'width':20,
                                                     'height':20})
        myvoice.OnOffIndicator.pack(side='right',fill='x')

        ########################################
        # define menubar function
        ########################################
        # radiobutton's selected/not selected value
        Suit = Tkinter.StringVar()
        Rank  = Tkinter.StringVar()

        # define radiobutton bind function
        def doit(Suit=Suit,Rank=Rank,myvoice=myvoice):
            """parse card index from radiobutton's value,
               and change card display picture.
            """
            num = GetCardIndex(Suit.get(), Rank.get())
            myvoice.card_disp.configure(image=myvoice.img[num-1])

        ########################################
        # create menu commands
        ########################################
        # 'File' menu
        MenuBar.add(name='File',type='command',
                    label=['Quit',],
                    action=[myvoice.quit,])

        # 'Suit' radiobutton
        MenuBar.add(name='Suit',
                    type='radiobutton',
                    label=['クローバー','ダイヤ','ハート','スペード'],
                    variable=Suit)

        # 'Rank' radiobutton
        MenuBar.add(name='Rank',
                    type='radiobutton',
                    label = ['エース','二','三','四','五','六',
                             '七','八','九','十','ジャック','クイーン','キング'],
                    variable=Rank,
                    action=doit)

if __name__ == '__main__':
    # make dictionary of command line option( {'option':'value'}
    lineopts = atrconfig.ParseCommandLine(os.sys.argv[1:])

    # make dictonary of config file options
    if lineopts.has_key('config'):
        cnfopts = atrconfig.ParseConfig(lineopts['config'])
```

```
    # merge config file dictionary into command line dictionary
    atrconfig.MergeDict(cnfopts, lineopts)

    # create application instance
    atrCard = SprecApp(cnfopts)

    # run the application
    atrCard.mainloop()

### EOF
```