汎用大語彙連続音声認識エンジン
Julius / Julian
rev. 3.3
(2002/09/11)

[→ホームページ] [→開発サイト]

Juliusについて

- Juliusは、単語N-gramに基づく高性能連続音声認識ソフトウェアです。数万語の語彙を対象とした文章発声の認識が行えます。
- Juliusは高速な音声認識を一般的なスペックのPC上で実現します。ほぼ実時間で動作でき、認識率は 20,000語彙の読み上げ音声で90% 以上です。
- 最大の特徴はその可搬性にあります。発音辞書や言語モデル・音響モデルなどの各モジュールを組み替えることで，様々なタスク向けの音声認識システムを容易に構築することができます。またプログラムのソースを公開しているので、他プラットフォームへの移植や改造も容易です。
- 動作プラットフォームは Linux，Solarisなどの Unix，およびWindowsです．Windows版は Microsoft SAPI 5.1 対応です． またDLLバージョンも入手可能です．

- 本ドキュメントは Unix 版について解説します．Windows版については CSRC CD-ROM 内ドキュメント，あるいはSAPI版ホームページをご覧下さい．

Julianについて

- Julian は，有限状態文法(DFA)に基づく連続音声認識パーザです。入力音声に対して与えられた文法規則の元で最も尤もらしい単語系列を探しだし，文字列として出力します．
- 2パス構成のA*探索による解探索を行います．第1パスでは，DFAを縮退させた単語対文法に従ってビーム探索を行います．第2パスでは，その結果をヒューリスティックとして，best-firstなA*探索によりさらに高精度な認識を行います．
- Julian は言語制約以外のほとんどの部分を Julius と共有しています．使用方法や使用可能な音響モデル，音声入力デバイスの設定などは，すべて（同バージョンの）Julius と同様です．
- 最大語彙数は 65,535 語です．

連絡先・リンク

連絡先・リンク

- ホームページ: http://julius.sourceforge.jp/
- 開発サイト: http://sourceforge.jp/projects/julius/: 最新CVSスナップショットはこちらで公開中
- E-mail: julius@kuis.kyoto-u.ac.jp or julius@is.aist-nara.ac.jp

開発者

- メイン, Unix版：李 晃伸 (ri@is.aist-nara.ac.jp)
- Windows Microsoft SAPI版：住吉
- Windows (DLL版)：坂野

Last modified: 2002/09/11 22:10:03

Julius / Julian のシステム構成と仕様

Julius パッケージには以下の２種類の認識プログラムが含まれています.

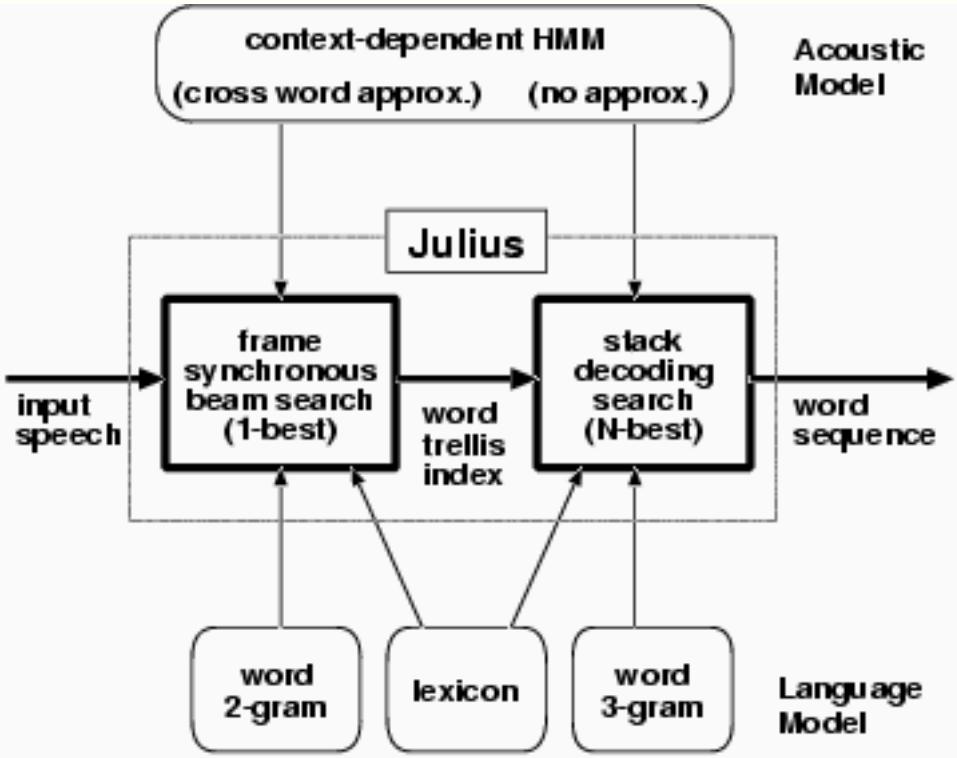
- Julius - 単語3-gramを用いた大語彙連続音声認識エンジン
- Julian - 有限状態文法を用いた連続音声認識パーザ

それぞれのシステム構成と仕様は以下のようになります.

Julius

システム構成

Julius を用いた認識システムの構成は以下の図のようになります.



- 言語モデルとして単語N-gram を，音響モデルとして HMM を使用する.
- 入力を２回に分けて処理する２パス探索を行う。
    1. 第1パス：単語2-gramを用いたフレーム同期ビーム探索（近似による高速化）
    2. 第2パス：単語3-gramを用いた N-best スタックデコーディング（精密計算）

    o

主な仕様

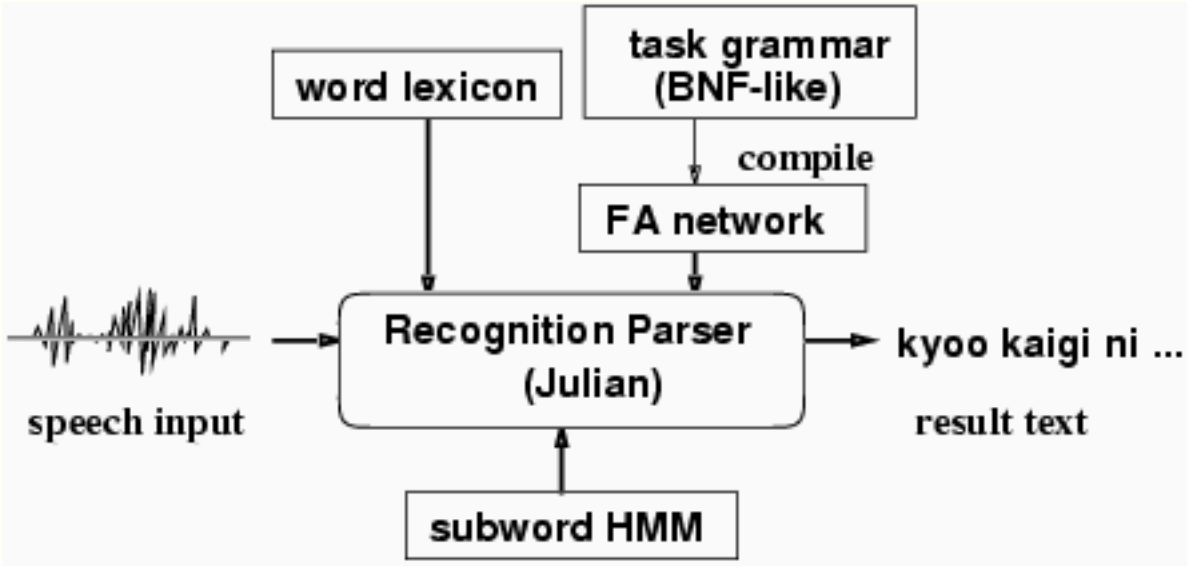
- 最大 65,535 語までの大語彙の認識に対応。
- 20kの読み上げ音声に対する単語認識性能は、精度優先の設定で 95% 以上(実時間の５倍)、速

度優先の設定では実時間で90%以上です。
- 単語N-gramは、単語2-gramと(逆向き)単語3-gramを用います。ARPA標準形式および独自のバイナリフォーマットに対応しています（変換ツール付属）。
- 音響モデルは音素(monophone)、音素環境依存(triphone)、tied-mixture モデルに対応しています。HTK 形式の音響HMM定義ファイルを読み込むことができます。tied-mixture については phonetic tied-mixture を含む任意単位のコードブック共有に対応しています。これらモデルのタイプは読み込み時に自動判別されます。
- 音声は、PCのマイクロフォン端子やDatLink(NetAudio)，ネットワーク経由の入力に対してオンラインで認識を行うことができます。音声検出と同時にオンラインで解析を始め、途中経過を漸時的に出力することもできます。その他、音声波形ファイルや、HTK形式の特徴パラメータファイルを入力とすることができます．ファイルの場合は、無限長のファイル入力に対応しています。

Julian

システム構成

Julian を用いた認識システムの構成は以下のようになります．基本アルゴリ ズムは Julius と同様の 2 パスA*探索アルゴリズムです．



- 言語モデルとして有限状態文法を，音響モデルとして HMM を使用する．
- 入力を 2 回に分けて処理する 2 パス探索を行う。
    1. 第1パス：カテゴリ対制約を用いたフレーム同期ビーム探索（近似による高速化）
    2. 第2パス：文法に基づく N-best スタックデコーディング（精密計算）

主な仕様

- 最大 65,535 語までの認識に対応。
- 言語モデルとして有限状態文法を用います．BNF風の専用形式で，カテゴ リ単位の構文制約(`.grammar`)とカテゴリごとの語彙 (`.voca`)を別ファイルに記述し，コンパイラ "`mkdfa.pl`" を用いてオートマトンネットワーク(`.dfa`) と 認識辞書(`.dict`)に変換します．
- 音響モデル，音声入力は Julius と同じのものが利用可能です．

Rev.3.3の新機能および変更点

Rev.3.3 の 3.2 からの新機能および変更点は 以下のようになっています.

- 新たな使い方を提供する機能の追加
  - サーバーモジュールモード
  - 複数文法同時認識
- 雑音対応
  - スペクトルサブトラクション
- 使用可能モデルの拡張
  - マルチパス音響モデル対応版Julius/Julian
- 性能の改善（バグフィックスによる）
- その他機能の追加

以下に詳細を述べます.

機能拡張

マルチパス音響モデル対応版 Julius/Julianの提供

従来のJulius/Julianは，高速な処理を重視して音響モデルの状態遷移に制約 がありました．すなわち，音響HMMにおいて初期状態から，あるいは最終状態 への遷移が１つであるモデルしか扱えませんでした.

Rev.3.3 からは，従来と同様の制約を持つ『通常版』と別に，新たに『マルチ パス音響モデル対応版』の Julius/Julian を提供します．このバージョンは， 初期状態からの複数遷移および最終状態への複数遷移を含むモデルを扱うこと ができます．これによって，マルチパス音響モデルなど，複雑なモデル内遷移 を持つ音響モデルを使用することができます.

この拡張された『マルチパス音響モデル対応版』は，ループ・スキップを含む ほぼ任意のモデル内遷移に対応しています．（ただし，初期状態から最終状態 への直接遷移のみ未対応です）

『マルチパス音響モデル対応版』は，通常版とは別のソースアーカイブとして提 供されます．コンパイルおよび使用方法は通常版と同一です． なおアルゴリズムの拡張により，同じモデルを用いた際の認識速度は通常版に 比べて若干(1% 前後)低下しますのでご注意ください.

マルチパス版のためのアルゴリズム変更については，参考文献[6]をご覧ください.

サーバーモジュールモードの追加

実行中の Julius/Julian をサーバー・クライアント形式で外部プロセスから 制御する機能が追加されました.

起動時にオプション "`-module`" をつけることで『サーバーモ ジュールモード』で起動します．クライアントと接続後認識を開始し，クライ アントへは認識結果や音声トリガ情報などのメッセージが送信され，クライア ントからは認識開始や中断の指示が行えます． またJulianでは，クライアント からの認

識文法切り替え・複数文法同時認識が行えます.

クライアントは tcp/ip ソケットを通じて Julius/ Julian と通信します．サ ンプルのクライアントプログラム "`jcontrol`" が収められてい ます．使用方法と使用例については，サーバーモジュー ルモードについてあるいはjcontrolマニュアルをご覧ください.

(Julian) 複数文法同時認識のサポート

文法の動的切り替え

Julianのサーバーモジュールモードにおいて，認識用文法を認識プロセス実行 中に外部から（再起動なしに）切り替えることができるようになりました．文 法はクライアントからソケットを介して Julian へ流し込みます．音声認識が 行われている最中に文法の切り替えが指示された場合は，その音声入力が終了 して入力待ち状態になったときにはじめて更新されます.

動作の詳細はサーバーモジュールモードについて，jcontrolマニュアル，および参考文献[7]をご覧ください.

複数文法の同時認識

さらに上記の拡張として，Julianで複数の文法を同時に用いて認識することが 可能になりました.

文法の追加および削除は，サーバーモジュールモードでクライアントから行い ます．音声認識実行中に指示があった場合，その音声入力が終了して入力待ち 状態になったときに更新されます．また，Julian に読み込まれた個々の文法 の一時的な ON/OFF (activate/deactivate) も可能です.

実現の仕組みとしては，全文法をJulian内部で結合し，単一のグローバル文法 を生成してその上で認識が行われる単一デコーディング方式を採っています． 全文法中の候補の中でもっとも尤度の高かった候補が認識結果として出力され ます.

詳細はサーバーモジュールモードについて，jcontrolマニュアルおよび参考文献[7]をご覧ください.

スペクトルサブトラクションのサポート

入力音声に対するスペクトルサブトラクション(SS)を実装しました． 「入力の先頭(の無音部)でSSを行う」「あらかじめ計算したノイズスペクトル を用いる」の2通りの方法が使用可能です.

1. ファイル入力の場合：入力の先頭(の無音部)でSSを行う

```
-sscalc             SS を行なう
-sscalclen msec     -sscalc 時，各ファイルの先頭の msec 分を無音部
                    とみなし，その範囲の平均スペクトルで SS を行な
                    う．単位はミリ秒（default: 300ms）
```

2. マイク入力の場合：あらかじめ計算した SS 値をファイルから読み込む

```
    -ssload filename    filename から SS 用のノイズスペクトルデータを
                        読み込む．ノイズスペクトルデータは，あらかじめ
                        作成ツール "mkss" で作成しておく．
```

また，以下のパラメータ設定オプションが追加されました．

```
    -ssalpha            SS の減算係数（default: 2.0）
    -ssfloor            SS のフロアリング値（default: 0.5）
```

SS はパワースペクトルで行なっています．

```
        |S(f)|^2 = |Y(f)|^2 - ¥alpha |N(f)|^2
```

この減算計数 alpha の値を `-ssalpha` で指定します．デフォルトは 2.0 です．

また，上記のスペクトル減算の結果マイナス値となるスペクトル成分に対して はフロアリングを行います．このフロアリング値としては「減算前の元スペク トルパワーの定数倍」を用います．この定数値を `-ssfloor` で 指定します．デフォルトは 0.5 です．

その他の機能拡張

音声入力の拡張

ネットワーク経由の入力("`-input adinnet`")が正式にサポート されました．他のクライアントホストで録音した音声をネットワーク経由で Julius/Julian に送り込んで認識することができます．音声入力クライアント としては，付属ツール "`adintool`" を用いることができま す．

音声の切り出しについては，

- 入力波形データを全て送って Julius/Julian 側で音声区間切り出しを行う
- クライアント(`adintool`)側で音声区間切り出しを行い，個々 の区間を逐次 Julius/ Julian へ送信して認識する

のいずれのスタイルも可能です． 詳しくは `adintool` のマニュアルの実行例を参照してください．

また，標準入力からの音声入力 ("`-input stdin`") が新たにサポートされました． 音声波形フォーマットは RAW(16bit, BE)あるいは WAV(無圧縮) のどちらかの みです．

CMNパラメータのファイル保存・読み込みのサポート

起動時オプション "`-cmnsave filename`" で CMN のパラメータ をファイルに書き出すことができます．Julius/Julian では CMN パラメータ は最新の5秒分の入力を用いて認識終了ごとに逐次更新しているため，このファ イルも 1 入力ごとに上書きで更新されます．

また，"`-cmnload filename`" で上記で保存した CMN パラメータを読 み込んで初期値とすることができます．ファイルが無い場合は無視されます．

想定する使用法としては，"`-cmnsave foo.cmn -cmnload foo.cmn`" とすることで，

1. 起動時にファイル `foo.cmn` があれば読み込む
2. 認識中の最新の CMNパラメータが随時 `foo.cmn` に保存される

という動作を行い，Julius/Julian を停止・再起動させても常に最新の CMN パラメータを最初の発話から使用することができるようになります．

状態単位のセグメンテーションのサポート

従来のセグメンテーションのオプション "`-walign`"（単語単位），"`-palign`"（音素単位）に加えて，HMM状態単位のセグメンテー ション "`-salign`" が可能になりました．モデル内の状態系列の 番号もあわせて出力されます．

RCLASSを含む hmmdefs のサポート

MLLR適応のための regression tree を含む音響モデルファイルを直接読み込 めるようになりました．

ツールの整備

付属ツールが増強・整備されました． これらは全て Julius / Julian と同時 にインストールされます．

- Juliusのライブラリを使用した音声入力ツール "`adintool`" を整備しました．音声ファイルからの音声区間連続切り出し，Julius/Julian のネットワーク経由入力 ("`-input adinnet`") への音声送信な どの使い方ができます．詳細は各マニュアルをご覧ください．
  - "`adinrec`" - マイク入力録音ツール（１発話を切り出してファイルに記録）
  - "`adintool`" - 多目的音声区間検出・記録・送信ツール
- これまで別途配布していた，Julian 用の文法作成ツールを収録しました．
  - "`mkdfa.pl`, `mkfa`" - 文法コンパイラ
  - "`yomi2voca.pl`" - ひらがな→音素表記変換スクリプト(IPA音素体系用)
  - "`generate`" - 受理文ランダム生成ツール
  - "`accept_check`" - 入力文の受理/非受理検証ツール
  - "`nextword`" - 次単語予測検証ツール

実装の改善

基本性能の改良

認識アルゴリズム実装の改善

Julius / Julian 共通：

- モデル内のループ遷移が正しく扱えていなかったバグを修正

Julius のみ：

- 第２パスのビーム幅設定 ("-b2") のアルゴリズムが正しく実装されていな かったバグを修正
- ある単語の最終音素が１状態のモデルで，かつその同音語が辞書内にある 場合に，言語確率が正しく与えられないことがあるバグを修正
- 第1パスにおいて 1-gram factoring 使用時の単語間遷移の扱いを改善し 処理量を削減

以上の実装の修正により，Julius で認識速度と認識精度が若干改善されました．

使用メモリ量の削減

以下の改善が施されました．

- マイク入力時にHMM状態の出力確率キャッシュを入力長にあわせて動的に 確保するよう改良され，使用メモリサイズが削減されました．
- (Julian) 文法のカテゴリ対情報をビット単位で保持することでメモリサ イズを押えるよう改良されました．

ドキュメンテーションの整理

オンラインマニュアルの整備

英語(.man) および日本語(.man.ja)があります． デフォルトでは英語マニュアルがインストールされます．日本語マニュアルを 上書きインストールしたい場合は，Julius/Julian インストール時に

```
% make install.man.ja
```

としてください．

ソース内のコメントを大幅に修正

ソース内のコメントを大幅に修正しました． libsent 以下は英語コメントのみ， julius 以下主要部分を日本語と英語で併記しています．

その他

- "-filelist listfile" を特徴量ファイル入力 "-input mfcfile" 入力でも可能にした．
- "-delwin" でデルタウィンドウ幅を指定可能にした．
- hmmdefs内で" <NUMMIXES> 1"という定義があるとき に読み込みに失敗する問題を修正．
- 空白を含むモデル名を扱えなかったバグを Fix．
- その他，細かいバグフィックス

動作環境

OS

Julius/Julian Rev.3.3 Unix版は以下のOSでの動作を確認しています。

- PC Linux 2.4.18 with ALSA driver
- PC Linux 2.2.18 with OSS driver
- Sun SPACC/Solaris 2.6

以下は Rev.3.1 で動作確認済みのOSです。3.3でも動作すると思われます。

- FreeBSD 3.2-RELEASE
- Sun Solaris 2.5.1
- Sun Solaris 2.6
- SGIIRIX 6.3
- DEC Digital UNIX 3.2
- Digital UNIX 4.0
- Sun SunOS 4.1.3

Windows 版の動作環境については，以下のドキュメントをお読み下さい.

- Julius for SAPI README (CSRC CD-ROM 内)
- SAPI版ホームページ(京大)

マシンスペック

Julius

最小スペック：PentiumIII 300MHz 相当, 96MB メモリ
推奨スペック：PentiumIII 700MHz 相当, 192MB メモリ

Julian

最小スペック：PentiumIII 200MHz 相当, 32MB メモリ
推奨スペック：PentiumIII 600MHz 相当, 128MB メモリ

Julian は 数百語のタスクでモノフォン使用であれば，最小スペックで10MB 以下のプロセスサイズで実時間動作します．5,000語のタスク PTM 使用であれ ば，推奨スペックの 30MB 程度のプロセスサイズで実時間動作します． なおマイク入力を行うには 16bit 録音可能なサウンドカードおよびドライバ が必要です。
詳しくは「マイク入力について」をお読みください。

Last modified: 2002/09/11 21:00:39

入手方法

Julius はオープンソースソフトウェアです． 以下の場所から入手可能です．

- http://julius.sourceforge.jp/: Julius ホームページ：最新パッチ，および Julian の機能を省いたフリー版 Julius が置いてあります．
- http://sourceforge.jp/projects/julius/: Julius 開発サイト：最新CVSはここで公開しています．
- 連続音声認識コンソーシ アム：Julian を含む Julius はここから入手できます．なお コンソーシアム終了後(2003年10月)に全てがフリーとなります．

Last modified: 2002/09/11 21:01:05

インストール

Unix版のインストール方法を説明します．Windows版については Julius for SAPI README (CSRC CD-ROM 内)あるいは SAPI版ホームページ(京大)をご覧下さい

クイックスタート

とりあえずは，以下の手順で Julius と Julian の通常版をインストールできます．

```
% tar xzvf julius-3.3.tar.gz
% cd julius-3.3
% ./configure
% make
% make install
% (日本語manをインストールする場合) make install.man.ja
% make distclean
% ./configure --enable-julian
% make
% make install
% (日本語manをインストールする場合) make install.man.ja
```

以下で各手順について具体的に説明します．

1. 準備

Julius/Julian はデフォルトでは RAW ファイルと WAV ファイル(無圧縮)しか読み込めませんが，libsndfile を用いることで AIFF, AU, NIST, SND, WAV(ADPCM)などの様々なフォーマットの音声ファイルを読み込めるようになります． libsndfile は configure 時に自動検出されますので，使用する場合はあらかじめインストールしておいてください．

```
（必要であれば）
libsndfile を入手しインストールする
```

次にJuliusのソースパッケージを展開し，展開先のディレクトリへ移動します．

```
% gunzip -c -d julius-3.3.tar.gz | tar xvf -
% cd julius-3.3
```

なお通常版ではなく『マルチパス音響モデル対応版』をインストールする場合 は，`julius-3.3.tar.gz`の代わりに `julius-3.3-multipath.tar.gz` から展開・インストールします．

2. configureの実行

configureスクリプトは，OSの種別やマイク入力サポートの有無，Cコンパイラや使用ライブラリ，CPUのエンディアンなどの動作環境を自動検出し，適切なMakefile を生成します．まずはこのconfigureを実行してください．

```
% ./configure
```

オプション指定でさまざまな設定を行えます。デフォルトでは、Julius が速度重視設定でコンパイルされます。以下は代表的なオプションです。

- Julian をコンパイルする場合

  ```
  % ./configure --enable-julian
  ```

- Julius を精度重視でコンパイルする場合

  ```
  % ./configure --enable-setup=standard
  ```

- インストール先を `/home/foobar/julius/` 以下に変えたい場合

  ```
  % ./configure --prefix=/home/foobar/julius
  ```

  実行バイナリは `${prefix}/bin`, マニュアルは `${prefix}/man/man1`にインストールされます． `prefix` のデフォルトは `/usr/local` です．

全オプションの詳細については 「configureオプション」 をご覧ください．

コンパイラは "gcc", デバッグオプションは "`-O2 -g`" がデフォルトです． Linux では "`-O6 -fomit-frame-pointer`" がデフォルトになります。 これら以外を用いるには，configure 前に使用するコンパイラとデバッグオプションをそれぞれ環境変数 "CC" および "CFLAGS" に設定します。ヘッダファイルのパスは環境変数 "CPPFLAGS" で指定します。

3. コンパイル

```
% make
```

を実行すると，各ディレクトリ下にそれぞれの実行バイナリが生成されます．

4. インストール

```
% make install
```

を実行すると，実行バイナリが${prefix}/bin, マニュアルが ${prefix}/man/man1にインストールされます.

以上でインストール作業は完了です.

インストールされる実行バイナリ一覧

julius (julian)	: Julius / Julian 本体
mkbingram	: バイナリN-gramファイル変換
adinrec, adintool	：音声入力の録音
mkdfa.pl, mkfa, yomi2voca.pl	：文法コンパイラ
generate, nextword, accept_check	：文法作成支援
mkss	：マイク入力のノイズスペクトルを計算・保存する
jcontrol	：サーバーモジュールモード用サンプルクライアント
mkgshmm	: Gaussian Selection 用音響モデ ル変換ツール

Last modified: 2002/09/11 21:00:39

マニュアル

オンラインマニュアルをHTMLに変換したものです.

- julius：大語彙連続音声認識エンジン
- julian：記述文法に基づく連続音声認識パーザ
- adinrec：マイク入力録音ツール
- adintool：汎用音声切り出し・録音・転送ツール
- jcontrol：簡単なサーバーモジュールクライアント
- mkbingram：バイナリN-gram作成ツール
- mkgshmm：GMS用モノフォンモデル変換ツール
- mkss：マイク入力のノイズスペクトル測定ツール

Last modified: 2002/09/11 21:00:39

NAME

Julius - open source multi-purpose LVCSR engine

SYNOPSIS

julius [-C jconffile] [options ...]

DESCRIPTION

Julius は数万語を対象とした大語彙連続音声認識を行うことのできるフリーの認識エンジンです．単語3-gramを用いた2パス 構成の段階的探索により高精度な認識を行うことができます．

認識対象はマイク入力，録音済みの音声波形ファイルおよび特徴抽出したパラメータファイルに対応しています．また標準的な形式の音響モデルや言語モデルを読み込んで使用することができるので，様々な音響モデルや言語モデルを切り替えて様々な条件で認識を行うことができます．

語彙数の上限は 65,535 語です．

使用モデル

Julius では以下のモデルを用います．

音素モデル

音素HMM(Hidden Markov Model)を用います．音素モデル(monophone)，音素環境 依 存 モ デ ル(triphone)，tied-mixtureモデル，phonetic tied-mixture モデルを扱えます．音素環境依存モデルの場合は単語間の依存関係も考慮されます． HTK のHMM定義言語で書かれたHMM定義ファイルを読み込むことができます．

言語モデル

言語モデルとして2-gramおよび逆向きの3-gramを用います． ARPA standard format およびそれらを付属の mkbingram で変換したバイナリ形式のN-gramを読み込むことができます．

音声入力

マイクロフォン端子やDatLink(NetAudio)からのLive入力が可能です．サンプリングした音声ファイル(16bit WAV(無圧縮)， RAW等)や特徴パラメータファイル(HTK形式)で与えることもできます．また専用クライアント adintool を用いたネットワーク経由での音声入力も可能です．

注意：Julius内部で計算できる特徴量はMFCC_E_D_N_Zのみです．これ以外の特徴抽出を必要とするHMMを使う場合は，マイク入 力や音声波形ファイル入力は使えません． wav2mfcc などの別ツールで抽出した特徴パラメータファイル(.mfc)を与えるようにして下さい．

探索アルゴリズム

Julius の認識処理は2パス構成です．まず第1パスで入力全体を完全に処理し，中間結果を出力します．モデルは単語2-gramと単語HMMの木構造ネットワークを用います．解探索はleft-to-rightにフレーム同期ビーム探索を行います．

第2パスでは3-gramを用いて逆向きに探索を行い，より精度の 高い 認識結果を求めます．第1パスの中間結果を絞り込み＋先読み情報として用い，単語単位のスタックデコーディングを行います．

音 素環境依存モデル(triphone)を用いたときは，第1パスおよび第2パスで単語間の音 素 環 境 依 存 を 考 慮 し ま す ． またtied-mixture やphonetic tied-mixture モデルではGaussian pruningによる高速な音響尤度計算を行います．

OPTIONS

以下のオプションで使用モデルやパラメータなどを指定します．コマンドライン上で指定するほか， jconf設定ファイルとして１つのテキストファイル内にまとめて記述しておき， 起 動 時 に "-C" で指定することができます．

以下は全てのオプションの説明です．

音声入力ソース

-input {rawfile|mfcfile|mic|adinnet|netaudio|stdin}
音 声 デ ータの入力ソースを選択する．それぞれ 'raw-file' は波形ファイル，フォン 入 力 ，'adinnet' は adinnet クライアントからのネットワーク経由入力，

サ ポートする音声波形ファイル形式については，コンパイル時にlibsndfileがあるかどうかで変わる．実際に その 実行バイナリでどの形式がサポートされているかはオプション "-help" で確認できる．なお標準入力について

はWAV(無圧縮) およびRAW(16bit,BE)のみをサポートする
.
(default: mfcfile)

-filelist file
(-input rawfile|mfcfile 時) 認識対象のファイルが 複数ある場合に, そのリストを与えてバッチ処理させる.

-adport portnum
(-input adinnet 時) adinnet で使用するポート番号.
(default: 5530)

-NA server:unit
(-input netaudio) 接続するDatLinkサーバ名とユニットID. netaudio 使用時必須.

音声区間検出

-cutsilence

-nocutsilence
入力に対する音声区間検出をするかしない か を 指 定 . (default: mic または adinnet は ON, ファイル入力は OFF)

-lv threslevel
波形の振幅レベルのしきい値(0 - 32767). 振幅がこ の値 を越えたときに音声区間の開始とみなし, 次にこの値を下回ったときに音声区間終了とする(default: 3000).

-zc zerocrossnum
１秒あたりの零交差数のしきい値 (default: 60)

-headmargin msec
音声区間開始部のマージン. 単位はミリ秒
(default: 300)

-tailmargin msec
音声区間終了部のマージン. 単位はミリ秒
(default: 400)

-nostrip
録 音デバイスによって生じることのある, 録音開始時あるいは終了時の無効な 0 値サンプルの自動除去を行わないようにする. デフォルトは自動除去を行う.

音響分析

-smpFreq frequency
音声のサンプリング周波数を Hz で指定
(default: 16kHz = 625ns).

-smpPeriod period
音声のサンプリング周期をナノ秒で指定
(default: 625ns = 16kHz).

-fsize sample
窓サイズをサンプル数で指定 (default: 400).

-fshift sample
フレームシフト幅をサンプル数で指定 (default: 160).

-delwin frame
デルタウィンドウ幅をフレーム数で指定 (default: 2).

-hipass frequency
高域カットオフする周波数を Hz で指定.
(default: -1 = disable)

-lopass frequency
低域カットオフする周波数を Hz で指定.
(default: -1 = disable)

-sscalc
フ ァイル先頭の無音部を用いて，入力全体に対してスペクトルサブトラクションを行う．ファイル入力に対し てのみ有効.

-sscalclen
フ ァイル先頭の無音部の長さをミリ秒で指定 (default: 300)

-ssload filename
ノイズスペクトルをファイルから読み込み，それを用 いて 入力に対してスペクトルサブトラクションを行う．ファイルはあらかじめ mkss で作成する． マ イ ク 入 力，adinnet入力では"-sscalc" ではなくこちらを使う必要がある.

-ssalpha value
スペクトルサブトラクションのアルファ係数．大きい ほど強く減じるが，歪みも大きくなる． (default: 2.0).

-ssfloor value
スペクトルサブトラクションのフロアリング係数．減 じた結果パワーが 0 以下になった部分スペクトルに対して，原信号の係数倍の信号を割り当てる (default: 0.5).

言語モデル(N-gram)

-nlr 2gram_filename
単語2-gramのファイル名(ARPA形式).

-nrl rev_3gram_filename
逆向き単語3-gramファイル名．第2パス実行時必須．指定しない場合は探索を第1パスのみ実行する.

-d bingram_filename
mkbingram で作成したバイナリ形式N-gramファイルを 指定 する． "-nlr", "-nrl" の代わりに使用することで起動を高速化できる.

-lmp lm_weight lm_penalty

-lmp2 lm_weight2 lm_penalty2
第1パスと第2パスの言語スコアの重みと単語挿入ペナ ルティ.

実 際の仮説の言語スコアは，N-gramの対数尤度を以下の式によってスケーリングしたものが用いられる.

lm_score1 = lm_weight * 2-gramスコア + lm_penalty
lm_score2 = lm_weight2 * 3-gramスコア + lm_penalty2

default値：モデルによって変化する

第1パス	第２パス
5.0	-1.0
8.0	-2.0
9.0	8.0

第２パス
6.0 0.0 (monophone)
8.0 -2.0 (triphone,PTM)
11.0 -2.0 (triphone,PTM, setup=v2.1)

-transp float
透過単語用に対して追加する単語 挿 入 ペ ナ ル テ ィ
(default: 0.0)

単語辞書

-v dictionary_file
単語辞書ファイル(必須).

-silhead {WORD|WORD[OUTSYM]|#num}

-siltail {WORD|WORD[OUTSYM]|#num}
文頭／文末の無音に対応する辞書単語を指定する.
(default: "<s>" / "</s>")

こ れらは認識時に仮説の始終端として特別に扱われる.
以下のいずれかの形式で指定する.

	例
単語名	<s>
単語名[出力シンボル]	<s>[silB]
#単語ID	#14

(単語番号は辞書ファイルの並び順に0番から)

-forcedict
辞書中の誤った単語エントリをスキップして起動を続 行
する.

音響モデル(HMM)

-h hmmfilename
使用するHMM定義ファイル名(必須).

-hlist HMMlistfilename
HMMlist ファイル名. triphone体系のHMM使用時に必須で
ある.

このファイルは, 辞書の音素表記 か ら 生 成 し た 論
理triphone名からHMM定義名への写像を与える. 詳細は付
属ドキュメントを参照のこと.

-iwcd1 {max|avg}
triphone使用時, 第1パスの単語間triphoneの音響尤度計
算方法を指定する.

max: 同コンテキストtriphoneの最大値 (default)
avg: 同コンテキストtriphoneの平均値

-force_ccd / -no_ccd
単 語間の音素環境依存を考慮するかしないかを明示的に指定する．指定がない場合はモデルの名前定義から推 察す る．なおtriphone以外で -force_ccd を指定したときの動作は保証されない．

-notypecheck
入力特徴パラメータの型チェックを無効にする．
(default: チェック有効)

音響尤度計算
Gaussian pruning は tied-mixture ベースの音響モデルにお いて 自 動 的 に 有 効 となる．Gaussian Selection の使用にはmkgshmm で変換されたモノフォンモデルが必要である．

-tmix K
Gaussian pruning でコードブックごとに上位K個のガ ウス分布を計算する． (default: 2)

-gprune {safe|heuristic|beam|none}
Gaussian pruning の手法を指定する．
(default: safe (標準版) beam (高速版))

-gshmm hmmdefs
Gaussian Mixture Selection 用のモノフォン音響モデルを指定する． GMS用モノフォンは通常のモノフォンか らmkgshmm(1) によって生成できる．
デフォルトは指定無し(GMSを使用しない)．

-gsnum N
GMS 使用時，全モノフォンの状態の中から上位 N 個の状態のみトライフォンを計算する (default: 24)

ショートポーズセグメンテーション
-spdur (--enable-sp-segment 時) 第1パスの sp 継続時間長 のし きい値(単位：フレーム)．ショートポーズ単語が最尤であるフレームがこの時間以上継続したら，第１パス を中断して第２パスを実行する．(default: 10)

探索パラメータ（第1パス）
-b beamwidth
ビーム幅．HMMのノード数で指定する．値が大きいほど安定した結果が得られるが，処理時間とメモリ量を消費 する．

default値：モデルによって変化する
400 (monophone 使用時)
800 (triphone,PTM 使用時)
1000 (triphone,PTM,engine=v2.1)

-sepnum N
(./configrue --enable-lowmem2 指定時) 辞書木から分離する高頻度語の数
(default: 150)

-1pass 第1パスのみ実行する．単語3-gramの指定が無い場合自動的にこのモードになる．

-realtime

-norealtime
第1パスを実時間処理するかを明示的に指定する．デフォルトは，ファイル入力について OFF (-norealtime)， マイク・NetAudio・ネットワーク入力について ON (-realtime)．このオプションは CMN と密接な関係にある：OFF の際は CMN は１入力ごとにそれ自身から計算されるが，ON の場合は直前の５秒分の入力の値を常に用いる．-progout も参照のこと．

-cmnsave filename
認識中に計算したCMNパラメータをファイルへ保存する．保存は一入力認識のたびに行われる．すでにファイル があ る場合は上書きされる．

-cmnload filename
初期CMNパラメータをファイルから読み込む．ファイルは "-cmnsave" で保存したファイル．これによってマイク入力やネットワーク入力においても起動後の入力第1発話からCMNを適用できる．

探索パラメータ（第2パス）

-b2 hyponum
仮説エンベロープの幅．仮説の各長さごとに，この数 を越 える仮説が展開されたらそれより短い仮説を展開しないようにする．探索失敗を防ぐ効果がある．
(default: 30)

-n candidatenum
この数の文仮説が得られるまで探索を続ける．得られ た仮 説 は スコアで再ソートして結果を出力する． (参考：-outputオプション)．

Juliusでは第2パスの探索の最適性は厳密には保証されない ため，最尤候補が常に最初に得られるとは限らない．この値が大きいほど真の最尤仮説が得られる可能性が 高くなるが， 長く探索するため処理時間は大きくなる．(default: 1)

default値：エンジン設定(--enable-setup=)に依存
10 (standard)
1 (fast,v2.1)

-output N
"-n"オプションで指定した仮説数のうち，上位N個を出力する
(default: 1).

-sb score
ス コアエンベロープの幅．各フレームごとに，それまでの最大スコアからこの幅以上離れた部分について はscanしない．第2パスの音響尤度計算の高速化に効果がある．
(default: 80.0)

-s stack_size
解探索中にスタックに保持する仮説の最大数．値が大 きい ほど安定した結果が得られるが必要メモリ量が増える．(default: 500)

-m overflow_pop_times
解探索打ち切りと判断する展開仮説数のしきい値．展 開さ れた仮説数がこの数を越えたとき，そこで探索を打ち切る．値が大きいほどあきらめずに探索を続けるが， 探索失敗時の処理時間は長くなる．(default: 2000)

-lookuprange nframe
単 語展開時に前後何フレームまでみて展開単語を決めるかを指定する．短い単語の脱落防止に効果があるが， 値が大きいと展開仮説が増えるため遅くなる．
(default: 5)

Forced alignment

-walign
認識結果に対して，単語単位のViterbiアラインメントを行う．単語ごとにマッチした区間，およびフレームご との平均音響尤度が出力される．

-palign
認識結果に対して，音素単位のViterbiアラインメントを行う．音素ごとにマッチした区間，およびフレームご との平均音響尤度が出力される．

-salign
認識結果に対して，状態単位のViterbiアラインメントを行う．状態ごとにマッチした区間，およびフレームご との平均音響尤度が出力される．

サーバーモジュールモード

-module [port]
サ ーバーモジュールモードで起動する．起動後はクライアントからのtcpip接続を待ち，クライアントからのコマン ドの処理およびクライアントへの認識結果や入力トリガ情報を送信する．複数文法認識はこのサーバーモジ ュー ルモードでのみ使用することができる．詳細は関連ドキュメント参照のこと．ポート番号のデ フ ォ ル ト は 10500 である．

-outcode [W][L][P][S][w][l][p][s]
サ ーバーモジュールモード時に，クライアントへ送信する認識結果の内容を指定する．それぞれ 'W' は単語の通常 の 出 力文字列，'L' は文法エントリ，'P' は音素列，'S' はスコアを表す．大文字は第2パス，小文字は 第1パ スに対応する．例えば第２パスの単語と音素列のみを送信したい場合は，"-outcode WP" のように指定する．

メッセージ出力

-separatescore
言語スコアと音響スコアをわけて出力する．

-quiet 音素列やスコアを省略して，ベストの仮説の単語列だ け出力する．

-progout
第1パスの途中結果を一定時間おきに漸次出力する．

-proginterval msec
-progout 時の出力インターバルを指定(単位：ミリ秒)

-demo "-progout -quiet" と等価.

その他

-debug 大量のデバッグ用内部メッセージを出力する.

-C jconffile
jconf設定ファイルの読み込み. これらの実行時オプションをあらかじめ記述して読み込ませることができる. また, あるjconf設定ファイル内でこのオプションにより他のjconf設定ファイルを include することができる.

-check wchmm
(デバッグ用)木構造化辞書の構造を対話的にチェック する.

-check triphone
(デバッグ用)音響モデル(とHMMList)による辞書上の単語の実際のマッピングを対話的にチェックする.

-version
プログラム名・コンパイル時刻・コンパイル時オプシ ョンを表示して終了する.

-help 簡単なオプション一覧を表示後終了する.

EXAMPLES

使用例については付属のチュートリアルをご覧下さい.

NOTICE

jconf 設定ファイル内でファイルを相対パスで指定する場合, それは実行時のカレントディレクトリは無関係に, その jconf ファイルが置いてある場所からの相対パスとして解釈される点に注意してください.

SEE ALSO

julian(1), mkbingram(1), mkss(1), jcontrol(1), adinrec(1), adintool(1), mkdfa(1), mkgsmm(1), wav2mfcc(1)

http://julius.sourceforge.jp/ (main)
http://sourceforge.jp/projects/julius/ (development site)

DIAGNOSTICS

正常終了した場合, Julius は exit status として 0 を返します. エラーが見付かった場合は異常終了し, exist status として 1 を返します.

入力ファイルが見つからない場合やうまく読み込めなかった場合は, そのファイルに対する処理をスキップします.

BUGS

Julius で使用できるモデルにはサイズやタイプに若干の制限 があります. 詳しくはパッケージに付属のドキュメントを参照してください.

バグ報告・問い合わせ・コメントな ど は julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

Rev.1.0 (1998/02/20)
河原 達也 と 李 晃伸 (京都大学)
が設計を行いました.

李晃伸 (京都大学)
が実装しました.

Rev.1.1 (1998/04/14)

Rev.1.2 (1998/10/31)

Rev.2.0 (1999/02/20)

Rev.2.1 (1999/04/20)

Rev.2.2 (1999/10/04)

Rev.3.0 (2000/02/14)

Rev.3.1 (2000/05/11)
李 晃伸 (京都大学)
が実装しました.

Rev.3.2 (2001/08/15)

Rev.3.3 (2002/09/11)
李 晃伸 (奈良先端大)
が主に実装しました.

THANKS TO

このプログラムは Rev.3.1 まで，情報処理振興事業協会(IPA)独創的情報技術育成事業「日本語ディクテーションの基本ソフトウェアの開発」(代表者：鹿野清宏 奈良先端科学技術大学院大学教授)の援助を受けて行われました.

Rev.3.2以降は「情報処理学会 連続音声認識コンソーシアム」において公開されています.

Windows DLL版 は板野秀樹氏(名古屋大学)の手によって作成・公開されています.

Windows Microsoft Speech API対応版は住吉貴志氏(京都大学)の手によるものです.

上記の協力・貢献してくださった方々，およびさまざまな助言・コメントをいただく関係者各位に深く感謝いたします.

また，開発に際して言語モデルを提供して頂いた伊藤克亘氏(電子技術総合研究所)，音素モデルを提供して頂いた武田一哉氏(名古屋大学)をはじめとする関係各位に感謝します．また伊藤克 亘氏をはじめ多くの方に動作確認とデバッグを行って頂きましたことを感謝します.

最後に，バグ報告や提案をしていただいている Julius users ML の メンバーの方々をはじめとするLinuxコミュニティの方々に感謝します.

Last modified: 2002/09/11 21:00:39

NAME

Julian - open source grammar-based continuous speech recognition parser

SYNOPSIS

julian [-C jconffile] [options ...]

DESCRIPTION

Julian は有限状態文法に基づく連続音声認識を行う認識パー ザで す. 記述文法を用いた2パス構成の段階的探索により高精度な認識を行うことができます.

認識対象はマイク入力, 録音済みの音声波形ファイルおよび特徴抽出したパラメータファイルに対応しています. また標準的な形式の音響モデルや言語モデルを読み込んで使用することができるので, 様々な音響モデルや言語モデルを切り替えて様々な条件で認識を行うことができます.

語彙数の上限は 65,535 語です.

使用モデル

Julian では以下のモデルを用います.

音素モデル

音素HMM(Hidden Markov Model)を用います. 音素モデル(monophone) , 音 素 環 境依存モデル(triphone) , tied-mixtureモデル, phonetic tied-mixture モデルを扱えます. 音素環境依存モデルの場合は単語間の依存関係も考慮されます. HTK のHMM定義言語で書かれたHMM定義ファイルを読み込むことができます.

言語モデル

タスク文法は, 構文制約を単語のカテゴリを終端規則としてBNF風に記述した grammar ファイルと, カテゴリ ご と の 単語の表記と読み（音素列）を登録する voca ファイルに分けて記述します. これをコンパ イラ mkdfa.pl (1) を用いて決定性有限状態オートマトンファイル(.dfa)と辞書ファイル(.dict)に変換し て Julian に与えます.

音声入力

マ イクロフォン端子やDatLink(NetAudio)からのLive入力が可能です．サンプリングした音声ファイル(16bit WAV(無圧縮)， RAW等)や特徴パラメータファイル(HTK形式)で与えることもできます．また専用クライアント adintool を用いたネットワーク経由での音声入力も可能です．

注意：Julian内部で計算できる特徴量はMFCC_E_D_N_Zのみです．これ以外の特徴抽出を必要とするHMMを使う場合は，マイク入 力や 音声波形ファイル入力は使えません．wav2mfcc(1)などの別ツールで抽出した特徴パラメータファイル(.mfc)を与えるようにして下さい．

探索アルゴリズム

Julian の認識処理は2パス構成となっています．まず第1パスでは，与えられた文法よりも弱い制約規則を用いて高速かつ荒い認識を行います．ここでは文法規則からカテゴリ間の制約のみを抜きだしたカテゴリ対制約を用いて，LRビーム探索を行いま す ．第2 パスでは第１パスの結果を元に本来の文法による再探索行い，高精度な解を高速に得ます．第２パスの探索は最適解が保証されるA*探索となります．

音 素環境依存モデル(triphone)を用いたときは，第1パスおよび第2パスで単語間の音 素 環 境 依 存 を 考 慮 し ま す ． またtied-mixture やphonetic tied-mixture モデルではGaussian pruningによる高速な音響尤度計算を行います．

OPTIONS

以下のオプションで使用モデルやパラメータなどを指定します．コマンドライン上で指定することもできますが， jconf設定ファイルとして１つのテキストファイル内にまとめて記述しておき，起動時に "-C" で指定することができます．

ほとんどのオプションは Julius と共通です．
Julian のみ：-dfa, -penalty1, -penalty2, -sp, -looktrellis
Julius のみ：-nlr, -nrl, -d, -lmp, -lmp2, -transp, -sil-head, -siltail, -spdur, -sepnum, -separatescore

以下は全てのオプションの説明です．

音声入力ソース
-input {rawfile|mfcfile|mic|adinnet|netaudio|stdin}

音 声 デ ータの入力ソースを選択する．それぞれ 'raw-file' は波形ファイル，フォン 入 力 ，'adinnet' は adinnet クライアントからのネットワーク経由入力，

サ ポートする音声波形ファイル形式については，コンパイル時にlibsndfileがあるかどうかで変わる．実際に その 実行バイナリでどの形式がサポートされているかはオプション "-help" で確認できる．なお標準入力についてはWAV(無圧縮) およびRAW(16bit,BE)のみをサポートする．
(default: mfcfile)

-filelist file
(-input rawfile|mfcfile 時) 認識対象のファイルが 複数ある場合に，そのリストを与えてバッチ処理させる．

-adport portnum
(-input adinnet 時) adinnet で使用するポート番号．
(default: 5530)

-NA server:unit
(-input netaudio時) 接続するDatLinkサーバ名とユニットID．netaudio 使用時必須．

音声区間検出

-cutsilence

-nocutsilence
入力に対する音声区間検出をするかしな い か を 指 定．(default: mic または adinnet は ON，ファイル入力は OFF)

-lv threslevel
波形の振幅レベルのしきい値(0 - 32767)．振幅がこ の値 を越えると音声区間の開始とみなし，次にこの値を下回ったときに音声区間終了とする(default: 3000)．

-zc zerocrossnum
１秒あたりの零交差数のしきい値 (default: 60)

-headmargin msec
音声区間開始部のマージン．単位はミリ秒
(default: 300)

-tailmargin msec
音声区間終了部のマージン．単位はミリ秒
(default: 400)

-nostrip
録音デバイスによって生じることのある，録音開始時 あるいは終了時の無効な 0 サンプルの自動除去を行わないようにする．デフォルトは自動除去を行う．

音響分析

-smpFreq frequency
音声のサンプリング周波数を Hz で指定
(default: 16kHz = 625ns)．

-smpPeriod period
音声のサンプリング周期をナノ秒で指定
(default: 625ns = 16kHz).

-fsize sample
窓サイズをサンプル数で指定 (default: 400).

-fshift sample
フレームシフト幅をサンプル数で指定 (default: 160).

-delwin frame
デルタウィンドウ幅をフレーム数で指定 (default: 2).

-hipass frequency
高域カットオフする周波数を Hz で指定.
(default: -1 = disable)

-lopass frequency
低域カットオフする周波数を Hz で指定.
(default: -1 = disable)

-sscalc
ファイル先頭の無音部を用いて，入力全体に対してス ペク トルサブトラクションを行う．ファイル入力に対してのみ有効.

-sscalclen
ファイル先頭の無音部の長さをミリ秒で指定 (default: 300)

-ssload filename
ノ イズスペクトルをファイルから読み込み，それを用いて入力に対してスペクトルサブトラクションを行う． ファ イ ル は あ らかじめ mkss で作成する．マイク入力，adinnet入力では"-sscalc" ではなくこちらを使う必要がある.

-ssalpha value
ス ペクトルサブトラクションのアルファ係数．大きいほど強く減じるが，歪みも大きくなる． (default: 2.0).

-ssfloor value
ス ペクトルサブトラクションのフロアリング係数．減じた結果パワーが 0 以下になった部分スペクトルに対して，原信号の係数倍の信号を割り当てる (default: 0.5).

言語モデル(記述文法)
-dfa dfa_filename
文法の有限状態オートマトンファイル(.dfa)を指定する(必須).

-penalty1 float
第 1 パ スの単語挿入ペナルティを指定する (default: 0.0)

-penalty2 float
第２パスの単語挿入ペナルティを指定 す る (default:

0.0)

単語辞書

-v dictionary_file
単語辞書ファイル(.dict) (必須).

-sp {WORD|WORD[OUTSYM]|#num}
「文中の短いポーズ」に対応する音響HMM名を指定する.
(default: "sp")

以下のいずれかの形式で指定する.

	例
単語名	<s>
単語名[出力シンボル]	<s>[silB]
#単語ID	#14

(単語番号は辞書ファイルの並び順に0番から)

-forcedict
辞書中の誤った単語エントリをスキップして起動を続 行する.

音響モデル(HMM)

-h hmmfilename
使用するHMM定義ファイル名(必須).

-hlist HMMlistfilename
HMMlist ファイル名. triphone体系のHMM使用時に必須である.

このファイルは, 辞書の音素表記 か ら 生 成 し た 論理triphone名からHMM定義名への写像を与える. 詳細は付属ドキュメントを参照のこと.

-iwcd1 {max|avg}
triphone使用時, 第1パスの単語間triphoneの音響尤度計算方法を指定する.

max: 同コンテキストtriphoneの最大値 (default)
avg: 同コンテキストtriphoneの平均値

-force_ccd / -no_ccd
単 語間の音素環境依存を考慮するかしないかを明示的に指定する. 指定がない場合はモデルの名前定義から推 察 す る. なおtriphone以外で -force_ccd を指定したときの動作は保証されない.

-notypecheck
入力特徴パラメータの型チェックを無効にする.
(default: チェック有効)

音響尤度計算

Gaussian pruning は tied-mixture ベースの音響モデルにお い て 自 動 的 に 有 効 となる. Gaussian Selection の使用にはmkgshmm で変換されたモノフォンモデルが必要である.

-tmix K
Gaussian pruning でコードブックごとに上位K個のガ ウ

ス分布を計算する．(default: 2)

-gprune {safe|heuristic|beam|none}
Gaussian pruning の手法を指定する．
(default: safe (標準版) beam (高速版))

-gshmm hmmdefs
Gaussian Mixture Selection 用のモノフォン音響モデルを指定する． GMS用モノフォンは通常のモノフォンか ら mkgshmm(1) によって生成できる．
デフォルトは指定無し(GMSを使用しない)．

-gsnum N
GMS 使用時，全モノフォンの状態の中から上位 N 個の状態のみトライフォンを計算する (default: 24)

探索パラメータ（第1パス）

-b beamwidth
ビーム幅．HMMのノード数で指定する．値が大きいほど安定 した結果が得られるが，処理時間とメモリ量を消費する．

default値：モデルによって変化する
400 (monophone 使用時)
800 (triphone,PTM 使用時)
1000 (triphone,PTM,engine=v2.1)

-1pass 第1パスのみ実行する．単語3-gramの指定が無い場合自動的にこのモードになる．

-realtime

-norealtime
第1パスを実時間処理するかを明示的に指定する．デフォルトは，ファイル入力について OFF (-norealtime)， マイク・NetAudio・ネットワーク入力について ON (-realtime)．このオプションは CMN と密接な関係に ある：OFF の際は CMN は１入力ごとにそれ自身から計算されるが，ON の場合は直前の５秒分の入力の値を常に用いる．-progout も参照のこと．

-cmnsave filename
認識中に計算したCMNパラメータをファイルへ保存する．保存は一入力認識のたびに行われる．すでにファイル があ る場合は上書きされる．

-cmnload filename
初期CMNパラメータをファイルから読み込む．ファイルは "-cmnsave" で保存したファイル．これによってマイク入力やネットワーク入力においても起動後の入力第1発話からCMNを適用できる．

探索パラメータ（第2パス）

-b2 hyponum
仮説エンベロープの幅．仮説の各長さごとに，この数 を越 える仮説が展開されたらそれより短い仮説を展開しないようにする．探索失敗を防ぐ効果がある．
(default: 30)

-n candidate_num
この数の文仮説が得られるまで探索を続ける．得られ た仮 説はスコアで再ソートされ、上位順に出力される． (参考：-outputオプション)．

Juliusでは第2パスの探索の最適性は厳密には保証されない ため，最尤候補が常に最初に得られるとは限らない．この値が大きいほど真の最尤仮説が得られる可能性が 高く な る が ， 長く探索するため処理時間は大きくなる．(default: 1)

default値：エンジン設定(--enable-setup=)に依存
10 (standard)
1 (fast,v2.1)

-output N
"-n"オプションで指定した仮説数のうち，上位N個を出力する
(default: 1).

-sb score
ス コアエンベロープの幅．各フレームごとに，それまでの最大スコアからこの幅以上離れた部分について はscanしない．第2パスの音響尤度計算の高速化に効果がある．
(default: 80.0)

-s stack_size
解探索中にスタックに保持する仮説の最大数．値が大 きい ほど安定した結果が得られるが必要メモリ量が増える．(default: 500)

-m overflow_pop_times
解探索打ち切りと判断する展開仮説数のしきい値．展 開さ れた仮説数がこの数を越えたとき，そこで探索を打ち切る．値が大きいほどあきらめずに探索を続けるが， 探索失敗時の処理時間は長くなる．(default: 2000)

-lookuprange nframe
単 語展開時に前後何フレームまでみて展開単語を決めるかを指定する．短い単語の脱落防止に効果があるが， 値が大きいと展開仮説が増えるため遅くなる．
(default: 5)

-looktrellis
単 語仮説を完全にトレリス内の単語だけに絞り，最尤フレーム検索も第１パスのトレリス単語の前後に絞り込 む． 特に大語彙では高速化の効果が高いが，誤りが増大することもある．

Forced alignment
-walign
認識結果に対して，単語単位のViterbiアラインメントを行 う．単語ごとにマッチした区間，およびフレームごとの平均音響尤度が出力される．

-palign
認識結果に対して，音素単位のViterbiアラインメントを行 う．音素ごとにマッチした区間，およびフレームごと

の平均音響尤度が出力される.

-salign
認識結果に対して，状態単位のViterbiアラインメントを行 う．状態ごとにマッチした区間，およびフレームごとの平均音響尤度が出力される.

サーバーモジュールモード
-module [port]
サーバーモジュールモードで起動する．起動後はクラ イアントからのtcpip接続を待ち，クライアントからのコマンドの処理およびクライアントへの認識結果や入力ト リガ 情報を送信する．複数文法認識はこのサーバーモジュールモードでのみ使用することができる．詳細は関連 ドキ ュ メ ン ト 参照のこと．ポート番号のデフォルトは 10500 である.

-outcode [W][L][P][S][w][l][p][s]
サーバーモジュールモード時に，クライアントへ送信 する認識結果の内容を指定する．それぞれ 'W' は単語の通常の出力文字列，'L' は文法エントリ，'P' は 音 素 列 ，'S' はスコアを表す．大文字は第2パス，小文字は第1パスに対応する．例えば第２パスの単語と音素列のみ を送信したい場合は，"-outcode WP" のように指定する.

メッセージ出力
-quiet 音素列やスコアを省略して，ベストの仮説の単語列だけ出力する.

-progout
第1パスの途中結果を一定時間おきに漸次出力する.

-proginterval msec
-progout 時の出力インターバルを指定(単位：msec)

-demo "-progout -quiet" に同じ.

その他
-debug デバッグ用内部メッセージを出力する.

-C jconffile
jconf設定ファイルの読み込み．これらの実行時オプション をあらかじめ記述して読み込ませることができる．また，あるjconf設定ファイル内でこのオプションにより他のjconf設定ファイルを include することができる.

-version
プ ログラム名・コンパイル時刻・コンパイル時オプションを表示して終了する.

-help 簡単なオプション一覧を表示後終了する.

EXAMPLES

使用例については付属のチュートリアルをご覧下さい.

NOTICE

jconf設定ファイル内でファイルを相対パスで指定する場合，それ は実行時のカレントディレクトリは無関係に，その jconf ファイルが置いてある場所からの相対パスとして解釈される点に注意してください.

SEE ALSO

julius(1), mkbingram(1), mkss(1), jcontrol(1), adinrec(1), adintool(1), mkdfa(1), mkgsmm(1), wav2mfcc(1)

http://julius.sourceforge.jp/ (main)
http://sourceforge.jp/projects/julius/ (development site)

DIAGNOSTICS

正常終了した場合， Julian は exit status として 0 を返します. エラーが見付かった場合は異常終了し， exist status として 1 を返します.

入力ファイルが見つからない場合やうまく読み込めなかった場合は，そのファイルに対する処理をスキップします.

BUGS

Julian で使用できるモデルのサイズやタイプには若干の制限があります. 詳しくはパッケージに付属のドキュメントを参照してください.

バ グ 報 告・問い合わせ・コメントなどは julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

Rev.1.0 (1998/07/20)

河原 達也　と　李 晃伸 (京都大学)
が設計を行いました.

Rev.2.0 (1999/02/20)

Rev.2.1 (1999/04/20)

Rev.2.2 (1999/10/04)

Rev.3.1 (2000/05/11)
李 晃伸 (京都大学)
が実装しました.

Rev.3.2 (2001/08/15)

Rev.3.3 (2002/09/11)
李 晃伸 (奈良先端大)
が主に実装しました.

THANKS TO

このプログラムは Rev.3.1 まで京都大学音声メディア研究室(旧堂 下 研)において開発されました. Rev.3.2 より以降は Julius と統合され, 「情報処理学会 連続音声認識コンソーシアム」 において公開されています.

Windows Microsoft Speech API対応版は住吉貴志氏(京都大学)の手によるものです.

さまざまな助言・コメント・御指導いただく関係者各位に深く感謝いたします.

Last modified: 2002/09/11 21:00:39

NAME

adinrec - record one sentence utterance to a file

SYNOPSIS

adinrec [options..] filename

DESCRIPTION

adinrec はマイク入力の音声区間を切り出してファイルに記録するツールです.

サンプリング周波数は任意に設定可能です. 形式は 16bit monoral です. 書き出されるデータ形式は RAW, 16bit, big endian です. 既に同じ名前のファイルが存在する場合は上書きします.

また, ファイル名に "-" を指定すると標準出力へ出力します.

音声区間の切り出しは, 一定時間内の零交差数とパワー（振幅レベル）のしきい値を用います.

OPTIONS

-freq threshold
: サンプリング周波数. 単位は Hz (default: 16000)

-lv threslevel
: 波形の振幅レベルのしきい値 (0 - 32767) . (default: 3000).

-zc zerocrossnum
: １秒あたりの零交差数のしきい値 (default: 60)

-margin msec
: 音 声区間開始部および終了部の前後のマージン. 単位はミリ秒 (default: 300)

-nostrip
: 無効な 0 サンプルの自動除去を行わないようにする. デフォルトは自動除去を行う.

SEE ALSO

adintool(1), julius(1), sox(1), wavplay(1), wavrec(1), aplay(1), arecord(1)

BUGS

バグ報告・問い合わせ・コメントな ど は julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

李 晃伸 (奈良先端大) が実装しました.

LICENSE

Last modified: 2002/09/11 21:00:39

NAME

adintool - multi-purpose tool to record/split/send/receive speech data

SYNOPSIS

adintool -in inputdev -out outputdev [options...]

DESCRIPTION

adintool は，音声波形データ中の音声区間の検出および記録 を連続的に行うツールです．入力音声に対して零交差数と振幅レベルに基づく音声区間検出を逐次行い，音声区間部分を連続出力します．

adintool は adinrec の高機能版です．音声データの入力元として，マイク入力・音声波形ファイル・標準入力・ネットワーク入力(adinnet サーバーモード)が選択できます．また，出力先として，音声波形ファイル・標準出力・ネットワーク 出 力(adinnet ク ラ イ アントモード)が選択できます．特にネットワーク出力（adinnet クライアントモード）では， julius へネットワーク経由で音声を送信して音声認識させることができます．

入力音声は音声区間ごとに自動分割され，逐次出力されます．音声区間の切り出しには adinrec と同じ，一定時間内の零交差 数とパワー（振幅レベル）のしきい値を用います．音声区間開始と同時に音声出力が開始されます．出力としてファイル出力を選んだ場合は，連番ファイル名で検出された区間ごとに保存します．

サンプリング周波数は任意に設定可能です．形式は 16bit monoral です．書き出されるデータ形式は RAW，16bit，big endian です．既に同じ名前のファイルが存在する場合は上書きします．

INPUT

音声を読み込む入力デバイスは以下のうちどれかを指定します．

-in mic
: マイク入力（デフォルト）．

-in file
: 音声波形ファイル．形式は RAW (16bit big endian)，WAV(無圧縮)など（コンパイル時の設定による）．

な お，入力ファイル名は起動後に，プロンプトに対して入力する．

-in adinnet
adinnet サーバーとなってネットワーク経由で adinnet クライアントから音声データを受け取る． adinnet クライアントからのTCP/IP接続を待ち，接続が確立した後 は adinnet クライアントから音声データを受け取る．
ポ ート番号のデフォルトは 5530 である．これはオプション "-port" で変更可能．

-in stdin
標準入力．音声データ形式は RAW，WAV のみ．

OUTPUT

検出した音声区間の音声データを書き出す出力デバイスとして，以下のうちどれかを指定します．

-out file
フ ァ イ ルへ出力する．出力ファイル名は別のオプション"-filename foobar" の形で与える．実際に は "foobar.0000" ，"foobar.0001" ... のように区間ごとに，指定した名前の末尾に4桁のIDをつけた名前で記録されなる． ID は 0 を初期値として，音声区間検出ごとに１増加する．初期値はオプション "-startid" で変更可能 であ る ． また，出力ファイル形式は RAW，16bit signed (big endian) である．

-out adinnet
adinnet クライアントとなって，ネットワーク 経 由 で adinnet サーバへ音声データを送る．入力の時とは逆に， adintool は adinnet クライアントとなり ，adinnet サ ーバーへ接続後，音声データを送信する．adinnet サーバーとしては， adintool および Julius のadinnet入力が挙げられる．
"-server" で送信先のadinnetサーバのホスト名を指定する．またポート番号のデフォルトは 5530 である．こ れはオプション "-port" で変更可能．

-out stdout
標 準 出力へ出力する．形式は RAW，16bit signed (big endian) である．

OPTIONS

-nosegment
入力音声に対して音声区間の検出を行わず，そのまま 出力 へリダイレクトする．ファイル出力の場合，ファイル

名の末尾に4桁のIDは付与されなくなる.

-oneshot
入力開始後，一番最初の１音声区間のみを送信後，終 了する.

-freq threshold
サンプリング周波数. 単位は Hz (default: 16000)

-lv threslevel
波 形 の振幅レベルのしきい値 (0 - 32767). (default: 3000).

-zc zerocrossnum
１秒あたりの零交差数のしきい値 (default: 60)

-headmargin msec
音声区間開始部の直前のマージン . 単 位 は ミ リ 秒 (default: 400)

-tailmargin msec
音 声 区 間 終 了 部 の直後のマージン. 単位はミリ秒 (default: 400)

-nostrip
無効な 0 サンプルの自動除去を行わないようにする. デフォルトは自動除去を行う.

EXAMPLE

マ イクからの音声入力を，発話ごとに "data.0000" から順に記録する：

```
% adintool -in mic -out file -filename data
```

巨大な収録音声ファイル "foobar.raw"を音声区間ごとに "foobar.1500" "foobar.1501" ... に分割する：

```
% adintool -in file -out file -filename foobar
  -startid 1500
  (起動後プロンプトに対してファイル名を入力)
  enter filename->foobar.raw
```

ネットワーク経由で音声ファイルを転送する(区間検出なし)：

```
[受信側]
  % adintool -in adinnet -out file -nosegment
[送信側]
  % adintool -in file -out adinnet -server hostname
    -nosegment
```

マイクからの入力音声を別サーバーの Julius に送る：

(1) 入力データを全て送信し，Julius側で区間検出・認識：

```
[Julius]
  % julius -C xxx.jconf ... -input adinnet
[adintool]
  % adintool -in mic -out adinnet -server hostname
    -nosegment
```

(2) 入力データはクライアント(adintool)側で区間検出し，検出した区間だけを順に Julius へ送信・認識：

```
[Julius]
  % julius -C xxx.jconf ... -input adinnet
[adintool]
  % adintool -in mic -out adinnet -server hostname
```

SEE ALSO

julius(1), adinrec(1)

BUGS

バグ報告・問い合わせ・コメントな ど は julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

李 晃伸 (奈良先端大) が実装しました.

LICENSE

Last modified: 2002/09/11 21:01:05

NAME

jcontrol - simple program to control Julius / Julian module via API

SYNOPSIS

jcontrol hostname [portnum]

DESCRIPTION

jcontrol は，他のホストで動作中の julius を，APIを介してコントロールする簡単なコンソールプログラムです． Julius へのコマンド送信，およびJuliusからのメッセージ受信を行うことができます．

起 動後，jcontrol は，指定ホスト上において「モジュールモード」で動作中の Julius または Julian に対し，接続を試みます． 接続確立後，jcontrol はユーザーからのコマンド入力待ち状態となります．

jcontrol はユーザーが入力したコマンドを解釈し，対応するAPI コ マンドを Julius へ送信します．また，Julius から認識結果や入力トリガ情報などのメッセージが送信されてきたときは，その内容を標準出力へ書き出します．

APIの詳細については関連文書をご覧下さい．

OPTIONS

hostname
: 接 続先のホスト名（Julius / Julian がモジュールモードで動作中）

portnum
: (optional) ポート番号 (default=10500)

COMMANDS (COMMON)

起動後，jcontrol に対して以下のコマンド文字列を標準入力 か

ら入力できます.

pause　認識を中断する．認識途中の場合，そこで入力を中断して第2パスまで認識が終わってから中断する.

terminate
　　認識を中断する．認識途中の場合，入力を破棄して即 時中断する.

resume 認識を再開.

inputparam arg
　　文 法切り替え時に音声入力であった場合の入力中音声の扱いを指定． "TERMINATE", "PAUSE", "WAIT"のうちいずれかを指定.

version
　　バージョン文字列を返す

status システムの状態(active/sleep)を返す.

GRAMMAR COMMANDS (Julian)

Julian 用のコマンドです：

changegram prefix
　　認識文法を "prefix.dfa" と "prefix.dict" に切り替える． Julian 内の文法は全て消去され，指定された文 法に置き換わる.

addgram prefix
　　認識文法として "prefix.dfa" と "prefix.dict" を追加する.

deletegram ID
　　指定されたIDの認識文法を削除する．指定文法は Julian か ら削除される．ID は Julian から送られる GRAMINFO 内に記述されている.

deactivategram ID
　　指定されたIDの認識文法を，一時的にOFFにする．OFF にされた文法は認識処理から一時的に除外される．このOFFにされた文法は Julian 内に保持され， "activategram" コマンドで再び ON にできる.

activategram ID
　　一時的に OFF になっていた文法を再び ON にする.

EXAMPLE

Julius および Julian からのメッセージは "> " を行の先頭につけてそのまま標準出力に出力されます．出力内容の詳細については，関連文書を参照してください．

(1) Julius / Julian をモジュールモードでホスト host で起動する．

```
% julian -C xxx.jconf ... -input mic -module
```

(2) (他の端末で) jcontrol を起動し，通信を開始する．

```
% jcontrol host
connecting to host:10500...done
> <GRAMINFO>
>  # 0: [active] 99words, 42categories, 135nodes (new)
> </GRAMINFO>
> <GRAMINFO>
>  # 0: [active] 99words, 42categories, 135 nodes
>   Grobal:      99words, 42categories, 135nodes
> </GRAMINFO>
> <INPUT STATUS="LISTEN" TIME="1031583083"/>
-> pause
-> resume
> <INPUT STATUS="LISTEN" TIME="1031583386"/>
-> addgram test
....
```

SEE ALSO

julius(1), julian(1)

BUGS

バ グ 報 告・問い合わせ・コメントなどは julius@kuis.kyoto-u.ac.jp までお願いします．

AUTHORS

李 晃伸 (奈良先端大) が実装しました．

LICENSE

Julius の使用許諾に準じます.

Last modified: 2002/09/11 21:01:05

NAME

mkbingram - make binary N-gram from two arpa LMs

SYNOPSIS

mkbingram 2gram.arpa rev3gram.arpa bingram

DESCRIPTION

mkbingram は, Julius で使用する言語モデルであるARPA形式の2-gram と 逆向き3-gram を 1つのバイナリファイル に結合 ・変 換するツールです. これを使用することで, Juliusの起動を大幅に高速化することができます.

なお2gramと逆無き 3-gram は, 同一のコーパスから同一の条 件（カットオフ値, バックオフ計算方法等）で学習されてあり, 同一の語彙を持っている必要があります.

mkbingram は gzip 圧縮された ARPA ファイルをそのまま読み込めます.

OPTIONS

2gram.arpa
: ARPA標準形式の単語 2-gram ファイル.

rev3gram.arpa
: ARPA標準形式の逆向き単語 3-gram ファイル.

bingram
: 出力ファイル（Julius用バイナリ形式）

USAGE

Julius で 言 語 モデル指定時に, 元の ARPA 形式ファイルを "-nlr 2gramfile -nrl rev3gramfile" とする代わり に mkbin-gram で変換したバイナリ形式ファイルを "-d bingramfile" と指定します.

SEE ALSO

julius(1)

BUGS

バグ報告・問い合わせ・コメントな ど は　julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

李 晃伸（京都大学）が実装しました.

LICENSE

Julius の使用許諾に準じます.

Last modified: 2002/09/11 21:00:39

NAME

mkgshmm - convert monophone HMM to GS HMM for Julius

SYNOPSIS

mkgshmm monophone_hmmdefs > outputfile

DESCRIPTION

mkgshmm はHTK形式のmonophone HMMを Julius の Gaussian Mixture Selection (GMS) 用に変換するperlスクリプトです.

GMSはJulius-3.2からサポートされている音響尤度計算の高速 化手法です． フレームごとに monophone の状態尤度に基づいてtriphoneやPTMの状態を予備選択することで，音響尤度計算 がおよそ30%高速化されます.

EXAMPLE

まずターゲットとするtriphoneやPTMに対して，同じコーパスで学習した monophone モデルを用意します.

次にそのmonophoneモデルを mkgshmm を用いて GMS 用に変換 します（実際には状態定義をマクロ化しているだけです）.

```
% mkgshmm monophone > gshmmfile
```

これを Julius で "-gshmm" で指定します.

```
% julius -C foo.jconf -gshmm gshmmfile
```

注意：GMS用モデルはtriphoneやPTMと同一のコーパスから作成する必要がある点に注意してください. gshmm がミスマッチだと選択誤りが生じ，性能が劣化します.

SEE ALSO

julius(1)

BUGS

バ グ 報 告・問い合わせ・コメントなどは julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

李 晃伸（奈良先端大）が実装しました.

LICENSE

Julius の使用許諾に準じます.

Last modified: 2002/09/11 21:00:39

NAME

mkss - compute average spectrum from mic input for SS

SYNOPSIS

mkss [options..] filename

DESCRIPTION

mkss は指定時間分の音をマイク入力から録音し，その平均スペクトラムをファイルに出力するツールです．出力されたファイルは Julius でスペクトルサブトラクションのためのノイズスペクトルファイル（オプション "-ssload"）として利用できます．

音声の切り出しは行わず，起動と同時に録音を始めます．

サンプリング条件は16bit signed short (big endian), monoral で 固定です．データ形式はRAW(ヘッダ無し), big endian形式です．既に同じ名前のファイルが存在する場合は上書きします．

なおファイル名に "-" を指定することで標準出力へ出力する こともできます．

OPTIONS

-freq threshold
: サンプリング周波数をHzで指定する．(default: 16000)

-len msec
: 録音時間長をミリ秒単位で指定する (default: 3000).

-fsize samplenum
: 分析のフレームサイズをサンプル数で指定する (default: 400).

-fshift samplenum
: 分析のフレームシフトをサンプル数で指定する (default: 160).

SEE ALSO

BUGS

バ グ 報 告・問い合わせ・コメントなどは julius@kuis.kyoto-u.ac.jp までお願いします.

AUTHORS

李 晃伸（京都大学）が実装しました.

LICENSE

Julius の使用許諾に準じます.

ドキュメント

- チュートリアル
  Julius の基本的な使用方法の解説です．

- 文法の記述方法
  Julian 用の文法の記述方法の解説です．

- マイク入力について
  マイク入力の認識に関する解説です．OS ごとの留意点やセットアップ方法，録音レベルの調節などについて解説します．

- configureオプション
  コンパイル時の configure オプションの詳細です．

- ファイル仕様と制限 Julius でサポートするモデルや音声ファイルなどの形式の詳細な仕様です．

- トライフォンとHMMList トライフォンの扱い，HMMListの形式と注意点，論理triphoneと物理triphoneについて解説します．

- サーバーモジュールモードについて
  Julius/Julian Rev. 3.3 の機能であるサーバーモジュールモード，およびそ の中での Julian による複数文法動的切り替えの機構について解説します．

Last modified: 2002/09/11 21:00:39

チュートリアル ～起動から認識実行まで～

Unix版 Julius の基本的な使い方について説明します．モデルの準備、システムの起動、そして実際に認識を行ってみるまでの流れを説明します。

1. 準備するもの
2. 音声ファイルの認識
3. マイク入力の認識

1. 準備するもの

Julius はそれ単体では動作しません。言語モデルと音響モデルを与えることで、認識システムとして音声認識を行うことができるようになります。

最低限必要なのは，以下の3種類のファイルです．

音響HMM定義ファイル	HTKのHMM定義ファイル形式
単語辞書ファイル	HTKの辞書形式とほぼ同じ
単語N-gramファイル	2-gram＋逆向き3-gram (ARPA標準形式)、 またはバイナリN-gram形式(独自形式)

音響HMMとして音素環境依存モデル(triphone, PTMなど)を用いる場合、さらに以下のファイルが必要です。

HMMListファイル	独自形式

各ファイルの詳細については「ファイル仕様と制限」を参照してください．

基本的な音響モデル・言語モデルがこれまでに配布されています：

- 連続音声認識コンソーシアム配布CD-ROM
- IPA「日本語ディクテーション基本ソフトウェア」配布CD-ROM
- オーム社教科書「音声認識システム」[6] 付属CD-ROM

以下では、これらの CD-ROM を使用することを想定して説明を進めます。

設定ファイル(jconfファイル)の準備

モデルファイルの指定や認識のパラメータ、入力ソースの選択などの設定は、すべて「jconf ファイル」と呼ばれるファイルに記述します．サンプルのjconfファイルがソースアーカイブ内に "Sample-j.jconf" としてありますので，これを適当な場所にコピーして，自分の環境に合わせて編集します。

前節で述べた Julius のCD-ROMをお持ちの方は、各CD-ROMに含まれている音響モデル・言語モデル

を使うよう設定されたjconfファイルがCD-ROMに含まれていますので、それを使って下さい。

2. 音声ファイルの認識

最初に録音済みの音声波形ファイルの認識について説明します。

音声ファイルの形式は、16bit の .WAV ファイル(無圧縮)あるいは RAW ファイル(big endian) です。サンプリング周波数は音響モデルの分析条件に依存しますが，通常は 16kHz です.

Juliusの起動はコマンドラインから行います．"-C" オプションで jconf ファイルを指定します.

```
% julius -C jconfファイル名
```

jconf内の各設定は、ここでコマンドラインオプションとして与えることもできます。特に"-C jconfファイル名" の後ろで指定することで，jconfファイル内の設定を上書きできます．たとえば，ここでは音声ファイル入力を行うので、以下のように指定することで jconf 内の指定によらず音声ファイル入力を指定することができます。

```
% julius -C jconfファイル名 -input rawfile
```

起動が終了すると，以下のようなプロンプトが出てキー入力待ち状態になります。

```
enter filename->
```

ファイル名を与えると、Juliusはその入力に対して音声認識を行います.

認識処理全体は2パスで行われます。まず入力全体に対して2-gramを用いたフレーム同期の認識を行います．以下は第１パスの出力例です.

```
input speechfile: sample/EF043002.hs
58000 samples (3.62 sec.)
### speech analysis (waveform -> MFCC)
length: 361 frames (3.61 sec.)
attach MFCC_E_D_Z->MFCC_E_N_D_Z
### Recognition: 1st pass (LR beam with 2-gram)
.......
pass1_best:  師匠 の 指導 力 が 問わ れる ところ だ 。
pass1_best_wordseq: <s> 師匠+シショー+2 の+ノ+67 指導+シドー+17 力+
リョク+28 が+ガ+58 問わ+トワ+問う+44/21/3 れる+レル+46/6/2 ところ+ト
コロ+22 だ+ダ+70/48/2 。+。+74 </s>
pass1_best_phonemeseq: silB | sh i sh o: | n o | sh i d o: | ry o k
u | g a | t o w a | r e r u | t o k o r o | d a | sp | silE
pass1_best_score: -8944.117188
```

pass1_best: が第１パスのベスト仮説の文字列(中間結果)です.
pass1_best_wordseq: は同じくN-gram表記の文字列,
pass1_best_phonemeseq: は同じく音素表記列（"|" は単語区切り）
pass1_best_score: は仮説のスコア（対数尤度）です.

第1パス終了後、第2パスが実行されて最終的な認識結果がでます。 第2パスでは第1パスで得られた中間結果を元に，3-gramを用いて単語単位の スタックデコーディングを行います.

```
### Recognition: 2nd pass (RL heuristic best-first with 3-gram)
samplenum=361
sentence1:  首相 の 指導 力 が 問わ れる ところ だ 。
wseq1: <s> 首相+シュショー+2 の+ノ+67 指導+シドー+17 力+リョク+28 が
+ガ+58 問わ+トワ+問う+44/21/3 れる+レル+46/6/2 ところ+トコロ+22 だ+
ダ+70/48/2 。+。+74 </s>
phseq1: silB | sh u sh o: | n o | sh i d o: | ry o k u | g a | t o w
a | r e r u | t o k o r o | d a | sp | silE
score1: -8948.578125
478 generated, 478 pushed, 16 nodes popped in 361
```

sentence1: が最終的な認識結果となります． 認識終了後はファイル名入力プロンプトへ戻ります.

結果の出力を変更できます。起動時に "-progout" をつけると第１パスの途中結果を漸次的に出力します。また "-quiet" で以下のようにシンプルな出力にできます。

```
58000 samples (3.62 sec.)
pass1_best:  師匠 の 指導 力 が 問わ れる ところ だ 。
sentence1:  首相 の 指導 力 が 問わ れる ところ だ 。
```

以上でファイルの認識は終わりです.

3. マイク入力の認識

マイク入力を直接認識することができます。マイク入力を行うにはオプション "-input mic" を指定します。

起動後、以下のようなプロンプトが出て入力開始待ちになります。

```
<<< please speak >>>
```

マイクに向かって発声を開始すると、同時に第1パスの認識処理が開始されます． 発声を止めると，そこで第1パスを打ち切って第2パスを実行し、最終的な認識結果を出力してまた上記プロンプトへもどります.

起動して最初の１入力は正しく認識できませんので注意してください。また周囲の雑音を検出して認識が始まってしまう場合や，逆に喋っても認識が始まらない場合は， マイクの録音レベルを調節してください.

動作条件などマイク入力に関する詳細は「マイク入力について」をご覧下さい.

Last modified: 2002/09/11 21:00:39

文法の記述方法

Julian における文法の記述方法について概説します．詳しくは別途配布のツールキット「Julian 文法キット」をご覧下さい．

タスク文法

Julian では Julius と異なり，認識対象とする音声の構文構造や語彙を人の手で明示的に記述します．Julian は与えられた文法で許される範囲のみで最尤解探索を行い，入力を最も良く説明できる文字列を結果として出力します．

この構文制約や語彙は，通常システムが対象とするタスクごとに記述します．これをここでは「タスク文法」と呼びます．

文法の記述

タスク文法は，構文制約を単語のカテゴリを終端規則としてBNF風に記述する grammar ファイルと，カテゴリごとの単語の表記と読み（音素列）を登録する voca ファイルに分けて記述します．

構文制約は，形式上はCFGのクラスまで記述可能ですが，Julian は正規文法のクラスまでしか扱えません．この制限はコンパイル時に自動チェックされます．また再帰性は左再帰性のみ扱えます． 以下，例として，以下のような文章を受理する文法を考えます．

- 「白にしてください」
- 「赤にします」
- 「赤色にしてください」
- 「終了します」

この文を受理する grammar ファイルは以下のように書けます．シンボル "S" が固定の開始記号です．`NS_B`, `NS_E`, `NOISE` はそれぞれ始端，終端，文中の無音区間（ポーズ）に対応します．

```
S          : NS_B CHGCOLOR_S NS_E
S          : NS_B QUIT_S NS_E
CHGCOLOR_S : COLOR_NP NI NOISE SURU_VP
COLOR_NP   : COLOR_N
COLOR_NP   : COLOR_N IRO_N
SURU_VP    : SURU_V
SURU_VP    : KUDASAI_V
QUIT_S     : QUIT_V SURU_V
```

左辺に出てこなかったシンボル(上図中赤文字)が終端記号，すなわち単語カテゴリとなります．voca ファイルではその各単語カテゴリごとに単語を登録します．

```
% COLOR_N
赤              a k a
```

```
白              sh i r o
黄緑            k i m i d o r i
青              a o
% IRO_N
色              i r o
% SURU_V
します          sh i m a s u
% KUDASAI_V
してください    sh i t e k u d a s a i
% QUIT_V
終了            sh u: ry o:
% NI
に              n i
% NS_B
silB            silB
% NS_E
silE            silE
% NOISE
sp              sp
```

Julian 形式への変換

grammar ファイルと voca ファイルは，専用コンパイラ "mkdfa.pl" を用いて決定性有限状態オートマトンファイル(.dfa)と辞書ファイル(.dict)に変換します.

```
(grammarファイル=sample.grammar, vocaファイル=sample.voca の場合)
% mkdfa.pl sample
sample.grammar has 8 rules
sample.voca    has 9 categories and 12 words
---
Now parsing grammar file
Now modifying grammar to minimize states[0]
Now parsing vocabulary file
Now making nondeterministic finite automaton[10/10]
Now making deterministic finite automaton[10/10]
Now making triplet list[10/10]
---
-rw-r--r--    1 foo      users         134 Aug 17 17:50 sample.dfa
-rw-r--r--    1 foo      users         212 Aug 17 17:50 sample.dict
-rw-r--r--    1 foo      users          75 Aug 17 17:50 sample.term
```

Julian の起動

変換したファイルを使って以下の要領で Julian を起動できます.

```
% julian -dfa sample.dfa -v sample.dict...
```

文法以外に音響モデルも指定する必要があります．また探索パラメータも設定できます．すべての設定を jconf ファイルに書いておくこともできます．
このあたりのやり方は Julius と全く同じです．

以上です．

Last modified: 2002/09/11 21:00:39

マイク入力について

Julius/Julian-3.3 でマイクロフォンから直接入力する際の，留意点や環境設定，制限事項等について説明します．

１．使用可能な音響モデル

Julius/Julian 内部で可能な特徴抽出量は，現在のところIPAの音響モデルと同じ特徴量(MFCC_E_D_N_Z)のみです．よってLPC等の異なるタイプの音響モデルを使用する場合，マイク入力は使用できず，特徴パラメータファイル(HTK形式)での入力のみとなりますので注意してください．

２．サポートするOS

現在マイク入力がサポートされているOSは，以下の通りです．

1. Linux (kernel driver, OSS/Linux, ALSA)
2. Sun Solaris 2.x
3. Sun SunOS 4.x
4. SGI IRIX
5. FreeBSD

各OSに関する詳細と注意事項は以下のとおりです：

1. Linux

Linux では，以下のサウンドドライバに対応しています．

- カーネル標準のドライバ
- OSS/Linux 4Front Technologies社の商用ドライバ
- ALSA: Advanced Linux Sound Architecture

これらは configure 実行時に自動判別されます．

使用ドライバを明示的に指定したい場合は，configure で"--with-mictype=TYPE" を指定してください(TYPEには oss もしくは alsa を指定)．

16bit,16kHz,monoral で録音できることが必須です．

正しく録音できるどうかは，現状ではドライバやチップとの相性に大きく左右されます．よくあ

る "Sound-Blaster Pro 互換" の設定ではおそらく正しく動作しません．特に，NotePC についてはドライバの相性が問題となることが多く，16bit,16kHz録音がサポートされているとされるチップであっても，実際には録音音質が非常に悪く使用に耐えないものもあります．ご留意下さい．

Juliusはミキサーデバイスの設定を行いません．録音デバイスの選択や各デバイスのボリューム調節は xmixer などで別途行ってください．

関連リンク:

- Linux Sound-HOWTO
- ALSA
- OSS/Linux

2. Sun Solaris 2.x

Solaris 2.5.1 および 2.6 で動作確認をしています．

デフォルトのデバイス名は "/dev/audio" です．環境変数 AUDIODEV で指定できます．

起動後オーディオ入力がマイクに自動的に切り替わります．また音量は 14 に自動設定されます．

3. Sun SunOS 4.x

SunOS 4.1.3 で動作確認をしています．コンパイルにはヘッダ が必要です．

デフォルトのデバイス名は "/dev/audio" です．環境変数 AUDIODEV で指定できます．

起動後オーディオ入力がマイクに自動的に切り替わります．音量は 20 に自動設定されます．

4. SGI IRIX

IRIX 6.3 で動作確認をしています．（5.x でも動作する可能性は大）

起動後オーディオ入力はマイクに自動的に切り替わりますが，ボリュームは自動調節されません．apanelコマンドで別途調節してください．

5. FreeBSD

FreeBSD 3.2-RELEASEで、ドライバはsndで動作確認をしました．"--with-mictype=oss" で動作します．

ミキサーの設定は行われません．入力デバイスの選択(MIC/LINE)やボリューム調節を他のツールで行うようにしてください．

サウンドカードなどの注意点は Linux の項をご覧下さい．

３．コンパイル時の設定

基本的に特別な設定は必要ありません．"configure" がOSを自動判別し，必要なライブラリを組み込みます．うまく検出されたかどうかは，configure の最後に出力される以下のメッセージをチェックしてください．

```
mic API type    : oss (Open Sound System compatibles)
```

自動判別に失敗する場合は，configure にオプション"--with-mictype=TYPE"を指定してください．TYPE は oss, alsa, freebsd, sol2,sun4, irix のどれかを指定します．

なお 3.1p1 よりドライバが ALSA であっても OSS API使用がデフォルトとなりました。ALSA ドライバーをお使いの方も OSS emulation での使用をお奨めします．ALSA native API 使用版(β版)を使用したい方は "--with-mictype=alsa" を指定してください．

４．マイク入力の認識手順

ここで Julius の起動からマイク入力までの流れを説明します．途中で不具合が起きた場合は ５，６ へ進んでください．

マイク入力を認識させるには，起動時に "-input mic" を指定します．すると起動後，以下のようなプロンプトが出て音声のトリガ待ちになります．(プロンプトが出る前に発声した内容は無視されます)

```
<<< please speak >>>
```

プロンプトを確認後，マイクに向かって発声します．口の位置はマイクから15cm程度離し，できるだけはっきり発声して下さい．

一定以上のレベルの入力があるとJuliusは第1パスの処理をはじめます．解析は入力と平行して進みます．次に長い無音区間が現れたらそこで第1パスの解析をやめて第2パスへ移行し，最終結果を出力します．その後また入力待ちになる，を繰り返します．

！ 注意 ！
マイク入力の場合，基本的に最初の第1発話は正しく認識できません． これは，実時間処理では直前の入力で計算したCMNパラメータを次の入力で使用する仕組みになっているため，最初の入力では CMN を行えないためです． 最初の入力は「マイクテスト」などと適当な入力を行い，2回目以降から本 入力を開始するようにしてください． ただし "-cmnload filename" で初期 CMN パラメータをファイル から読み込み，第1発話からCMNを適用することができます．また Julius/Julian 内のCMNパラメータは "-cmnsave filename" で保存できます．

なお起動時に "-demo" を指定することで第1パスの解析途中の候補をリアルタイムに出力することができます．

５．音量・トリガレベルの調節

うまく認識できないときは，マイクの音量や，音声の開始を検出するためのトリガレベルを動作環境に応じて設定してやる必要があります．

調節の流れですが，まずマイクのボリュームを調節してから，トリガレベルを決定します．ボリュームは入力音声が割れない程度に大きくします（他の録音ツール等で正しく録音できているかチェックすると良いでしょう）．

感度が鈍くて音声を検出できない場合や，逆に周囲の雑音でトリガしてしまう場合は，トリガレベルを調節します．トリガレベルはオプション "-lv" で指定できます．値の範囲はunsigned short の振幅 (0-32767) で，デフォルトは 3000 です．トリガレベルが大きいと感度が鈍り，小さいと感度が鋭くなります．

また，発話開始や発話終了部分の音声が切れてしまう場合は，トリガレベルを下げると 途切れずに収録できることがあります．

６．録音音声の検証

Rev.2.2 以降の Julius には，１発話をマイクから録音するプログラム adinrec が付属しています．これを用いて，Julius が取り込む音声をチェックできます．

```
% ./adinrec/adinrec myfile
```

上記のように実行すると，adinrec はマイクから1回分の発声をファイル myfile に記録します．この adinrec は Julius 本体と同じ取り込みルーチンを使用しているので，この録音ファイルの音質がすなわちJuliusが認識しようとしている音声の音質になります．

オプションを指定しない場合，録音ファイルはヘッダなしの 16kHz, monoral, signed 16bit big endian です．以下のコマンドで再生できます（要sox）

```
Linux/OSS: sox -t .raw -r 16000 -s -w -x -c 1 myfile -t ossdsp -s -w /dev/dsp
Linux/ALSA: aplay -f s16b -r -s 16000 myfile
Solaris2: sox -t .raw -r 16000 -s -w -c 1 aaa -t sunau -w -s /dev/audio
```

記録した音声波形を見るには，例えば以下のツールを使用してみてください．

- spwave
- wavesurfer
- snd

Last modified: 2002/09/11 21:00:39

configure オプションの詳解

Julius-3.3 のコンパイル時に "configure" に与えられる全オプションを解説します.

Julius / Julian 切り替え

"--enable-julian" で, Julius の代わりに Julian をコンパイルします.

エンジン設定変更オプション

"--enable-setup=..." で認識アルゴリズムを以下の３種類のプリセットから選択できます.

```
1) standard: 標準版            最高精度だが低速
2)   fast   : 高速版            高速, 高精度            （デフォルト）
3)   v2.1   : rev.2.1互換   全オプション OFF
```

実行バイナリのアルゴリズム設定は下記のようにオプション "-version" をつけて起動することで確認できます.

```
% julius -version
###### booting up system
Julius rev.3.2 (fast)
built on jale at Sun Aug 12 20:58:44 JST 2001
 - compiler: gcc -O6 -fomit-frame-pointer
 - options: UNIGRAM_FACTORING LOWMEM2 PASS1_IWCD SCAN_BEAM GPRUNE_DEFAULT_BEAM
```

以下に詳しく説明します.

1) 標準版：--enable-setup=standard

精度を特に重視した設定です.
triphoneモデル使用時に第2パスで厳密な単語間triphoneを計算するようになります. 精度は 1% 弱改善されますが,
第2パスの処理速度は大幅に増加します.
また TM, PTM モデルにおける Gaussian Pruning [3] は
エラーを生じない safe pruning がデフォルトになります.
(実行時に -gprune で他のpruning法に切り替えられます)

2) 高速版：--enable-setup=fast (default)

速度と精度のバランスを取った設定です.
Gaussian pruning 法のデフォルトは最も足切り性能の高い
beam pruning となります. 無指定時のデフォルトです.

3) rev.2.1互換：--enable-setup=v2.1

Rev.2.1 と同じアルゴリズムに戻します.

1-gram factoring と第1パスでの単語間triphoneの扱いを行わなくなります.

上記を表にまとめると以下のようになります．表中の ○ のオプションが実際に configure 内で enable されます.

	1-gram factoring	1st pass IWCD	2nd pass strict IWCD	tree separation	Gauss. pruning default method
--enable-	factor1	iwcd1	strict-iwcd2	lowmem2	
standard	○	○	○	×	safe
fast	○	○	×	○	beam
v2.1	×	×	×	×	safe

Julius 第１パス重視の設定

Julius において，デフォルト設定では、第１パスにおいて 1-gram factoring や 1-best 近似などの速度を重視した近似計算を行うため、
第１パスの時点での認識精度は近似の誤差を多く含み、高くはありません。
(この誤差は第２パスで解消する仕組みです)

一方、2-gram のみを用いる場合など、第１パスの精度を重視したい場合もあります。この場合、以下の設定が有効です。

```
% ./configure --enable-setup=v2.1 --enable-iwcd1 --enable-wpair
```

上記の設定によって、第１パスで 2-gram factoring を行い、単語間triphoneを計算し、かつ1-best近似の代わりに単語対近似を用います。これによって計算コストはかなり高くなりますが、第１パスの精度を上げることができます。

オプション詳細

個々の configure オプションの詳細です.

○コンパイル・インストール環境

--prefix=dir　　インストール先のディレクトリを変更します.
実行ファイルは ${dir}/bin,
マニュアルは　${dir}/man/{man1,cat1}
にインストールされます.
デフォルトは /usr/local/bin です.

--with-netaudio-dir=dir　DatLink を使用する際に，DatLink付属のNetAudioライブラリの include と lib があるディレクトリを指定します.

○マイク関連

--disable-pthread　音声入力の取り込みに POSIX thread を使用しないようにします.

--with-mictype=TYPE ドライバタイプを指定します.
alsa, oss, sol2, sun4, irix のどれかを指定します.
デフォルトは自動判別です.

○アルゴリズム関連

--enable-sp-segment (Julius) ショートポーズセグメンテーションによる逐次デコーディングを
ON にします．第1パスを短いポーズでより細かく区分化しながら、
逐次第2パスを実行・確定していきます.
長文の入力に対して人手による発話切出が不要となります。
また、無限長のファイルを扱うことができます。

※現時点ではファイル入力のみ対応です.
マイク入力には対応していないのでご注意下さい

--enable-words-int 単語IDの変数型を unsigned short から int に変更します.
使用メモリ量が増えますが，65535 語以上の語彙数が扱えるようになります.
なお現在のところ動作は保証しません.

--enable-factor1 (Julius) 第1パスで1-gram factoringを行います．デフォル
ト(2-gram factoring)に比べ処理速度が大幅に
向上しますが，第1パスの精度は下がります.

--enable-lowmem2 (Julius) 高頻度語のみ木から分離することで，1-gram factoring
時に少ないビーム幅でも認識精度を落ちにくくします.

--enable-iwcd1 第1パスで単語間triphoneを扱います．処理量は増えま
すが，精度は良くなります.

--enable-strict-iwcd2 第2パスで単語間triphoneを厳密に扱います.
処理量が劇的に増える分，精度は僅かに良くな
ることがあります.

--disable-score-beam 第2パスでスコアに基づくビームの設定をOFFにします.
通常は ON のままにしておいてください.

--enable-wpair (Julius) 第１パスで単語対近似を用います。デフォルトの
1-best 近似に比べて第１パスの精度は高くなりま
すが、計算コストが増大します。

以上

Last modified: 2002/09/11 21:00:39

ファイル形式の仕様と制限

Julius が扱うことのできるデータ形式は以下のようになっている。

なお，ファイルは全て .gz のままで読み込み可能である。

各ファイルの用途／形式や仕様上の制限等を、以下に述べる

---

1. HMM定義ファイル (起動時指定オプション：-h)

HTKのHMM定義言語で記述されたHMM定義ファイルを読み込むことができる。
音素(monophone)，音素環境依存(triphone), tied-mixture ベースモデルに対応している．これらは読み込み時に自動判別される．

triphoneモデルの場合は，辞書の音素表記からtriphoneへの変換を行うために，辞書上で登場しうる全てのtriphoneに対して実際のモデル上のtriphoneの割り付けを指定する HMMlist ファイルが必須である（後述）．

JuliusではHTKの仕様を全て実装してはおらず，必要性が低いと思われる部分は省いている。また実装の都合上，状態間遷移に制限が加わる。

また 3.3 からはMLLR適応のための regression tree を含む音響モデルファイルを直接読み込める．ただし Julius/Julian自身が MLLR 適応を行えるわけではない．

○形式

(各項目の意味の詳細は "The HTK Book for HTK V2.0" 7.9 節を参考のこと)

・出力分布形式

連続HMMのみ（離散HMMは不可）
対応。Phonetic Tied-Mixture モデルを含む任意の
mixture tyingに対応

混合数・ベクトル次元数は任意
共分散行列の型は，デフォルトの対角成分(diagonal)のみ。
InvCover, LLTCover, XForm, FULL は不可。
継続時間制御(duration)パラメータは不可。

・状態間遷移

初期状態と最終状態には以下の制限がある。
・出力分布を持たないこと
・初期状態から遷移は1つのみであること
・最終状態への遷移は1つのみであること
なおこの2状態を除く内部では任意のskipやloopを許す.

・共有マクロ

~t(遷移), ~s(状態), ~m(分布), ~v(共分散) を扱える.
これ以外の共有マクロ(~w ~u ~i ~x)は不可.

・multi input stream

入力ストリームは1つのみ.

以上の条件に合わないHMM定義ファイルを読み込もうとした場合，Julius/Julian はエラーメッセージを出力して異常終了する.

○サイズの制限

HMMの定義数や状態数，マクロ数に上限は無い.
メモリの許す限りのどんな大きさのものでも読み込むことができる.

○Tied-Mixture モデルの判別

バージョン 3.0 以降では，tied-mixture ベースのモデルがサポートされている.
単一のコードブックからなる通常のtied-mixtureモデルのほかに，音素単位でコードブックを持つ phonetic tied-mixture モデルも扱える.
HTKと同じくコードブックは任意数定義できる.
ただし，通常の混合分布定義との混在は許されない.

Julius は音響尤度の計算手順として以下の2種類を実装している.
使用するモデルのタイプを自動判別してどちらかを自動的に選択する.

1. state-driven

   monophoneや状態共有triphone用の計算方式.
   出力確率の計算は状態単位で行い，キャッシュも状態単位で行う.

2. mixture-driven

   tied-mixtureモデルの計算方式.
   コードブック（ガウス分布集合）単位で計算を行う.
   各フレームごとに，まず全コードブックの各ガウス分布の尤度を計算し，コードブックごとに各分布尤度をキャッシュする.
   HMM状態の出力確率は，対応するコードブックの上記のキャッシュを参照しながら，重みをかけて算出する.

tied-mixtureモデルかどうかの判別は，hmmdefs中に "" 定義が用いられているかどうかで行う．hmmdefs読み込み中に "" 定義が見つかると，Julius はそのモデルを tied-mixture ベースと判断してmixture-drivenの計算方式を選択する.

このように，音響モデルをtied-mixtureモデルとしてJuliusに計算させるには音響モデルを ディレクティブを用いて定義する必要がある.

ディレクティブの扱いについては HTK に準ずる．詳細は HTK Book を参照のこと.

2. HMMlistファイル (-hlist)

HMMListファイルは，"e-i+q" といった辞書の音素表記から生成される論理的な音素表記から，hmmdefs で定義される音響モデル名への任意のマッピングを記述する．triphone HMM を用いる場合は必須．

フォーマットや制限についてはトライフォンとHMMListについてを参考にされたい．

3. 単語N-gram言語モデルファイル

■３．１．ARPA標準形式 (-nlr, -nrl)

2-gramと逆向きの3-gramのARPA標準形式のN-gramデータを読み込むことができる．

○形式

ARPA標準形式を読み込める．未知語カテゴリ(等)のエントリが1-gramの最初のエントリとして必ず含まれていること．(CMU SLM ToolKitで作成した場合，仕様上常にこの条件は満たされる)

単語辞書の単語のうちN-gramにない単語のN-gram確率は，この未知語カテゴリの確率をその総数で補正した値が用いられる．

2-gramと（逆向きの）3-gramの両方を読み込む場合，逆向き3-gramにおいて出現するコンテキストが 2-gramに無い場合は，警告メッセージを出力しつつそのtupleを無視して処理を続行する．

○サイズの制限

語彙数の上限は 65,535 語である．

configure 時に"--enable-word-int" を指定することで上限を(2^31)語まで拡張することができる．ただし現在のところ動作は保証の限りではない．

また，上記 configure で作成した Julius でバイナリN-gramを用いる場合は，そのバイナリN-gramも，上記 configure でコンパイルした mkbingram を使って生成されたものである必要がある点に注意されたい

■３．２．バイナリN-gram形式 (-d)

2-gramと逆向き3-gramファイル(ARPA形式)から生成できる，バイナリ形式のN-gram ファイルである．添付のツール "mkbingram" で生成する．

あらかじめ内部でインデックス化されているので起動が非常に高速になる．またサイズも小さくなるメリットがある．

なお，CMU-TKのbinary n-gram (.binlm) とは非互換である．

4. 単語辞書ファイル　(-v)

HTKのdictionary formatとほぼ同等のフォーマットを用いる．
違いは第2フィールドが必須であることだけである．

○形式

・第1フィールド（言語制約エントリ）

Julius においては，その単語が参照する N-gram エントリの文字列．
Julian においては，その単語が属するカテゴリ番号．

N-gramエントリを検索するため，辞書ファイルとN-gramファイルの日本語コードは一致させる必要がある．

N-gramエントリにない単語はとして参照される．そのN-gram確率は
N-gramエントリにない単語総数で補正した値が用いられる．

・第2フィールド（出力シンボル）

認識結果として出力する文字列を指定する．値は `[' `]' で囲まれていなくてはならない．"[]"と指定することで出力無しにできる．

・第3フィールド以降（音素HMMの並び）

辞書の音素記述は monophone で行う．
（triphone HMM使用時は，辞書読み込み時に単語内コンテキストを考慮して
　自動変換される．）

[例](ソート済みである必要はない)
```
課税+1  [カゼイ]        k a z e i
課題+1  [カダイ]        k a d a i
課長+1  [カチョウ]      k a ch o:
課長+1  [カチョウ]      k a ch o u
過ぎ+過ぎる+102 [スギ]  s u g i
過ぎ+過ぎる+114 [スギ]  s u g i
過去+11 [カコ]  k a k o
過激+14 [カゲキ]        k a g e k i
過程+1  [カテイ]        k a t e:
```

○サイズの上限

認識単語数の上限は 65,535 語である．

ただし，configure 時に"--enable-word-int" を指定することで上限を(2^31)語まで拡張することができる．ただし現在のところ動作は保証の限りではない．

5. DFAファイル (-dfa)

オートマトン文法制約を表現する，カテゴリ単位の有限状態オートマトンネッ

トワークを記述する．

○形式

1行で1遷移を定義する．状態番号は 0 から．

・第1フィールド：状態番号
・第2フィールド：入力カテゴリ番号
・第3フィールド：遷移先状態番号
・第4フィールド：最下位ビットが１なら，第1フィールドで示された状態が
　終了状態（受理状態）であることを表す．最下位ビット以外は未使用．
・第5フィールド：未使用

初期状態は状態番号 0 に固定（１つのみ）．
終了状態は「状態番号 -1 -1 1」の形で指定（複数指定可能）
入力カテゴリ番号は，単語辞書中のカテゴリ番号（第1フィールド）に
対応する．

6. マイク入力（-input mic)

マイクデバイスは16kHz,16bitでサンプリングが行える必要がある．

最大長はデフォルトで20秒(320kサンプル)．
増やしたいときは include/sent/speech.h の MAXSPEECHLEN を上げ
て再コンパイルすればよい．

なおJuliusは現在のところMFCC_E_D_N_Zしか特徴抽出できないため，
MFCC_E_D_N_Z以外の音響モデルではマイク入力を使用できない点に注意．

7. 音声波形ファイル (-input rawfile)

音声データは16kHz,16bitで与える必要がある．

ファイル形式は以下のものを読み込める（自動判別）．

1. RAW(別名no header)ファイル
   ただし16kHz,16bit(signed short),mono,big-endian
2. Microsoft Windows WAV ファイル
   ただし16kHz,16bit,無圧縮

libsndfile付きでコンパイルした場合はさらに以下の形式のファイルを読み込
める(libsndfileドキュメントより)．ただしどれも16bit,16kHzである必要が
ある（内部でサンプリングレート変換などは行わない）

3. Microsoft WAV 16 bit integer PCM.
4. Apple/SGI AIFF and AIFC uncompressed 16bit interger PCM
5. Sun/NeXT AU/SND format (big endian 16bit PCM)
6. Dec AU format (little endian 16 bit PCM)
7. Microsoft IMA/DVI ADPCM WAV format (16 bits per sample
   compressed to 4 bits per sample).
8. Microsoft ADPCM WAV format (16 bits per sample compressed

```
   to 4 bits per sample)
9. Microsoft 8 bit A-law an u-law formats (16 bits per sample
   compressed to 8 bits per sample)
10. Ensoniq PARIS big and little endian, 16 bit PCM files (.PAF).
```

なお[libsndfile](http://www.zip.com.au/~erikd/libsndfile/)は以下のURLから取得できる：

http://www.zip.com.au/~erikd/libsndfile/

Juliusの探索アルゴリズムの性質上，長い入力は第2パスの探索が不安定になるため，無音などで短く区切って入力するのが望ましい．

なおJuliusは現在のところMFCC_E_D_N_Zしか特徴抽出できないため，MFCC_E_D_N_Z以外の音響モデルでは音声ファイルの入力はできない．外部ツールで特徴抽出した特徴パラメータファイルを与えることになる．

8. 特徴パラメータファイル (-input mfcfile)

HTKの用いる特徴パラメータファイルを認識対象として与えることができる．

○形式

特徴パラメータの型(base kind,qualifier)およびベクトル長は，使用するHMMの学習パラメータと一致するか，もしくは学習パラメータを含んでいる必要がある．認識に必要なパラメータが全て含まれていない場合はエラーとなる．

なんらかの理由でチェックがうまく機能しない場合は -notypecheck オプションでチェックを回避できる．

○与えるパラメータの型について

特徴パラメータの型は本来使用するHMMの学習パラメータと一致している必要がある．しかし，与える特徴パラメータが，HMMが必要とするパラメータの構成要素を内部に含んでいる場合は，Juliusが自動的にその中から必要な要素を抜き出して認識に使用する．

例えば，

・HMM学習パラメータ
```
MFCC_E_D_N_Z = MFCC(12)+ΔMFCC(12)+ΔPow(1) (CMN)          計25次元
```

のときは，MFCC_E_D_N_Z以外に

・特徴パラメータファイル
```
MFCC_E_D_Z   = MFCC(12)+Pow(1)+ΔMFCC(12)+ΔPow(1) (CMN) 計26次元
```
または
```
MFCC_E_D_A_Z = MFCC(12)+Pow(1)+ΔMFCC(12)+ΔPow(1)
               + ΔΔMFCC(12) + ΔΔPow(1)      (CMN)          計39次元
```

を与えても認識を実行可能である．

9. 1入力あたりの仮説単語長制限

文仮説中の単語数には制限がある（デフォルト：150単語）.

非常に長い入力を与えた時に以下のエラーがでた場合は，Juliusのソースプログラム内のMAXSEQNUMの定義を変更して再コンパイルすれば良い.

```
    sentence length exceeded ( > 150)
```

具体的には，ソースパッケージの include/sent/speech.h 内の

```
#define MAXSEQNUM     150
```

の値を必要に応じて大きい値に書き換えた後,

```
    % make distclean; make
```

を実行する.

以上

Last modified: 2002/09/11 21:00:39

トライフォンとHMMListについて

この文書では，Julius における音素環境依存モデル(トライフォン)の扱いと， HMMList ファイルについて解説します．

1. 音素環境依存モデル

Julius は音素環境依存モデル(トライフォン)が与えられた時，辞書上の音素表記からコンテキストを考慮したトライフォン表記を生成し，その表記に対応するHMMを適用します．

モノフォンからトライフォン表記の生成は，ある音素 "X" に対して直前の音素が "L"，直後の音素が "R" である場合に "L-X+R" の形で行われます．以下は単語「英訳」のトライフォン表記への変換例です．

```
英訳+エイヤク+17        [英訳]  e     i     y     a     k     u
        ↓
英訳+エイヤク+17        [英訳]  e+i e-i+y i-y+a y-a+k a-k+u k-u
```

Julius ではこのような単語辞書の音素表記，およびそこから生成されるトライフォン表記を「論理トライフォン(logical triphone)」と呼びます．またこれに対して，実際に hmmdefs で定義されているHMM名を「物理トライフォン(physical triphone)」と呼びます．

2. HMMListファイル

論理トライフォンと物理トライフォンとの対応は，HMMList ファイルで指定します．HMMListファイルは，登場しうるすべてのトライフォン表記について，hmmdefs で定義されている HMM を対応付けます．以下はそのフォーマットです．

1. 一行に1つの論理トライフォンの対応を定義します．
2. 第1カラムに論理トライフォン，第2カラムに対応する hmmdefs 内のHMM名を指定します．
3. hmmdefs 内で表記と同じ名前で定義されているトライフォンについては，第 2 カラムを空白にします．
4. すべての登場しうる論理トライフォンについて定義する必要があります．
5. 二重登録はエラーとなります．

以下は例です．第 2 カラムが空白のエントリは，その HMM 名が直接 hmmdefs 内で定義されていることを表しています．

```
a-k
a-k+a
a-k+a: a-k+a
a-k+e
a-k+e: a-k+e
a-k+i
a-k+i: a-k+i
a-k+o
```

```
a-k+o: a-k+o
a-k+u
a-k+u: a-k+u
...
```

システムにおいて実際にどのようにマッピングされているかは，julius を "-check wchmm" を付けて起動することでチェックできます．初期化終了後チェックモードに入りプロンプトがでてくるので，"h 論理トライフォン名" と打つとそのトライフォンに関する情報が出力されます．

3. 単語間トライフォンの扱い

Julius において，音素環境依存性の扱いはパスごとに異なります．第１パスでは単語間トライフォンについて，同じコンテキストを持つ全トライフォンHMMの最大値を用います．例を挙げると，前述の単語「英訳」の場合，単語の末端において， "k-u+a", "k-u+s", "k-u+e:" などのHMMの音響尤度を計算し，その最大値を割り当てます．

第2パスでは正確な単語間の依存性を計算します．ある仮説から次の単語を展開する際，接続部のモデルを依存性を考慮して切り替えながら探索を行います．

```
今日 + は       ky+o: ky-o:   w+a    w-a
          ↓
今日 は         ky+o: ky-o:+h o:-w+a w-a
```

4. HMMListファイル作成にあたっての注意点

HMMListファイルの作成に当たっては，以下の点に注意して下さい．

- HMMList ファイルでの指定は hmmdefs内の名前定義を上書きします．つま り，実際に hmmdefs 内で定義されている名前とマッピングの名前が重なっ た場合，マッピングのほうが優先されます．例を挙げると， hmmdefs 内に

  ```
  ~h "j-u+q"
  ```

  という定義があるとき，HMMList ファイルで

  ```
  j-u+q y-u+q
  ```

  などど指定すると，j-u+q のHMM定義は実際には使用されずに y-u+q が使われ ることになります．

Last modified: 2001/08/17 09:40:09

サーバーモジュールモードについて

しくみ

外部クライアントと Julius/Julian は，ソケットを介したサーバークライアントの形式で接続する．クライアントは認識開始・一時停止などの命令や，文法の流し込み(Julian)などの動作命令を送り，Julius/Julian からは，認識結果や音声のトリガ情報などを受け取る．

起動方法

Julius/Julian を "-module ポート番号" をつけて起動する．
起動直後，指定されたポートからの接続待ちとなる．
（マイク入力を行う場合 "-input mic" も必須）

起動
→初期化（ポート初期化，音響モデル読み込み，パラメータ設定）
→接続待ち・接続（fork）
→ループに入る
・文法の処理
・認識の実行
・認識suspend, resumeなど

Julius/Julian は「音声認識サーバー」となり，接続はクライアントから Julius/Julian 側へ行われる．
エンジンの初期状態は sleep である．

Julian からクライアントへ送信される情報

下記の情報が送信される．

・音声検出開始時
<INPUT STATUS="LISTEN" TIME="システム時間(秒)"/>
．

・音声トリガ→録音開始時
<INPUT STATUS="STARTREC" TIME="システム時間(秒)"/>
．

・音声録音終了時
<INPUT STATUS="ENDREC" TIME="システム時間(秒)"/>
．

・認識結果

```
<RECOGOUT>
  <SHYPO RANK="1" SCORE="-7375.335449">
    <WHYPO WORD="silB" CLASSID="23" PHONE="silB"/>
    <WHYPO WORD="嵯峨山" CLASSID="2" PHONE="s a g a y a m a"/>
    <WHYPO WORD="先生" CLASSID="3" PHONE="s e N s e:"/>
    <WHYPO WORD="の" CLASSID="4" PHONE="n o"/>
    <WHYPO WORD="電話番号" CLASSID="6" PHONE="d e N w a b a N g o:"/>
    <WHYPO WORD="を" CLASSID="8" PHONE="o"/>
    <WHYPO WORD="教えて" CLASSID="9" PHONE="o sh i e t e"/>
    <WHYPO WORD="下さい" CLASSID="10" PHONE="k u d a s a i"/>
    <WHYPO WORD="silE" CLASSID="24" PHONE="silE"/>
  </SHYPO>
</RECOGOUT>
.
```

RANK属性は順位を表す．例えば "-n 5 -output 5" とするとRANK=1..5が出力される．またCLASSID属性はその単語の言語モデルエントリ（Juliusなら N-gramエントリ名，Julian ならカテゴリID）を表す．

どの情報を出力するか，または第1パスを出力するかなどは，起動時オプション "-outcode string" で指定する．
string は "WLPSwlps" のうち出力したい情報に対応するものを続けて指定する．
各文字の対応は以下の通り：

- W: 単語文字列
- L: カテゴリID
- P: 音素列
- S: スコア
- w: 単語文字列（第1パス）
- l: カテゴリID（第1パス）
- p: 音素列（第1パス）
- s: スコア（第1パス）

なおデフォルトは "-outcode WLPS" である．

コマンドAPI

Julius/Julian への命令は，クライアントからソケットを介して送り込む．
コマンド文字列に続けて，改行コード '¥n' を送る．

コマンド一覧・引数を持たないもの：

DIE	Julian を強制終了させる．
VERSION	エンジンのバージョンを返す
STATUS	システムの状態(active/sleep)を返す
PAUSE	認識を中断し sleep にする． （認識途中の場合，そこで入力を中断し， 第2パスまで認識が終わってから中断する）
TERMINATE	認識を中断し sleep にする． （認識途中の場合，入力を破棄して即時中断）
RESUME	認識を再開，sleep から active状態へ移行

コマンド一覧・'¥n' に続けてデータを送信（Julianのみ）：

INPUTONCHANGE	音声入力中の切り替え方法を指定
CHANGEGRAM	文法を切り替える
ADDGRAM	文法を加える

```
        DELGRAM          文法を削除する
        DEACTIVATEGRAM   文法を一時的に無効化する
        ACTIVATEGRAM     一時的に無効化された文法を有効化する
```

Julian においては，文法情報が適宜，下記のタグ形式で送られてくる．

```
------------------------------------------------------------------------
<GRAMINFO>
  #01: [ACTIVE     ]   45 words,  22 categories,  547 nodes
  #02: [INACTIVE   ]  247 words,  14 categories,   30 nodes (new)
  Total:               292 words,  36 categories,  577 nodes
</GRAMINFO>
------------------------------------------------------------------------
```

Julius/Julian から送られてくる認識結果を含む任意のメッセージは，
最後にピリオドのみの行 ".¥n" が付与される．よって Julius/Julian からの
メッセージを受け取るルーチンでは，ピリオドのみの行 ".¥n" をメッセージ
の区切りとして処理する．

Julius/Julian の状態

以下の2状態を持つ．

```
active:          音声認識を実行中
sleep:           音声認識を中断中．ただしコマンドは受け付ける．
```

状態の切り替えは PAUSE/TERMINATE/RESUME コマンドで行う（後述）．

文法の流し込みと制御(Julian)

CHANGEGRAM，ADDGRAMでは "コマンド¥n" に続けて
使用したい文法を以下の形式で Julian に転送する．

```
------------------------------------------------------------------------
CHANGEGRAM                      <- コマンド文字列
[.dfa ファイルの中身]
DFAEND                          <- .dfaファイルの終端を示す文字列
[.dict ファイルの中身]
DICEND                          <- .dictファイルの終端を示す文字列
------------------------------------------------------------------------
```

Julianは文法の送信要求を受けると割り込みを発生し，文法を受け取る．
たとえ active (認識中)であっても，認識を中断して文法を受け取る．
よってサイズの大きい文法を送る際は認識の遅延を考慮して，
sleep 状態のきに送るなどの配慮が望ましい．

Julian に流し込まれた文法は，単語対文法の抽出などを経て，global
lexicon tree に組み込まれる．エンジンが sleep 状態であればすぐに処理さ
れるが，受け取り時点で active の場合は次の認識待ち状態(sleep)で初めて
処理される．このため認識中の一文の前後で文法が切り替わることはない．

文法の削除の場合は，該当文法を削除したあと global lexicon tree 全体が
再構築される．このためサイズの大きい文法では削除が遅くなる可能性がある．

・ADDGRAM は使用中の文法群に新たな文法を加える．

・CHANGEGRAM は，使用中の文法群を全て破棄し，新たな文法を使用する．
・DELGRAM は指定した文法を削除する．文法は ID 番号で指定する．
  ID番号は，コマンドに続けて削除したい文法のID暗号をスペースで区切って
  与え，最後に改行で閉じる．例： "コマンド¥n1 3 4¥n"
・DEACTIVATEGRAM, ACTIVATEGRAM は指定した文法を一時的に OFF/ON する．
  DEACTIVATE された文法は認識に使用されなくなるが global lexicon には
  残っており，再び ACTIVATE することができる．このため高速な切り替えが
  可能である．

・INPUTONCHANGE では文法を送ったときやactivate/deactivate時に音声入力中
  だったときのエンジンの動作を決める．引数の意味は以下の通り：

- TERMINATE	現在の入力を破棄する．探索を即時中断後， 文法を変更し，認識を再開．
- PAUSE	現在の入力をそこで強制的に区切って認識を終わらせる． 認識終了後文法を変更し，認識を再開．
- WAIT	現在の入力が（ユーザによって）区切れるまで待つ． 入力終了後，文法を変更して認識を再開

  デフォルトは PAUSE である．

以上

Last modified: 2002/09/11 21:01:05

参考文献

国内中心

1. 李晃伸, 河原達也, 堂下修司. "単語トレリスインデックスを用いた段階的探索による大語彙連続音声認識." 電子情報通信学会論文誌, Vol. J82-DII, No. 1, pp. 1--9, 1999.
2. 李晃伸, 河原達也, 堂下修司. "文法カテゴリ対制約を用いた{A*}探索に基づく大語彙連続音声認識パーザ." 情報処理学会論文誌, Vol. 40, No. 4, pp. 1374--1382, 1999.
3. 李 晃伸, 河原達也, 武田一哉, 鹿野清宏. "Phonetic Tied-Mixture モデルを用いた大語彙連続音声認識." 電子情報通信学会論文誌, Vol. J83-DII, No. 12, pp. 2517--2525, 2000.
4. A.Lee, T.Kawahara, and K.Shikano. "Gaussian mixture selection using context-independent HMM." In Proc. IEEE Int'l Conf. Acoust., Speech & Signal Process., 2001.
5. 李 晃伸，河原達也，鹿野清宏．"透過語処理を用いた単語N-gramにおける不要語のモデル化と評価." 日本音響学会研究発表会講演論文集，1-5-15，春季 2001.
6. 李晃伸, 河原達也, 鹿野清宏. " 認識エンジンjuliusにおけるマルチパス音韻モデルの実装." 日本音響学会研究発表会講演論文集, 2-5-8, 春季 2002.
7. 李 晃伸，鹿野清宏．"複数文法の同時認識および動的切り替えを行う認識エンジン Julius/Julian-3.3." 日本音響学会研究発表会講演論文集，3-9-12,秋季, 2002.

8. オーム社 IT Text 「音声認識システム」
   鹿野清宏，伊藤克亘，河原達也，武田一哉，山本幹雄 編著
   ISBN4-274-13228-5
   定価 3,500円

国際会議

1. A. Lee, T. Kawahara, and K. Shikano. "Julius --- an open source real-time large vocabulary recognition engine." In Proc. European Conf. on Speech Communication and Technology, pp. 1691--1694, 2001.
2. A. Lee, T. Kawahara, K. Takeda, M. Mimura, A. Yamada, A. Ito, K. Itou, and K. Shikano. "Continuous Speech Recognition Consortium --- an open repository for CSR tools and models ---." In Proc. IEEE Int'l Conf. on Language Resources and Evaluation, 2002.

Last modified: 2002/09/11 21:00:39

ライセンス

Julius はオープンソースソフトウェアであり，ソースを含めて無償公開され ています.

- 改変は自由
- バイナリのみの配布を認める
- 商用ソフトへの組み込みも可能
- 変更部分のソースは公開が望ましいが，非公開でも構わない

以下が使用許諾文書です.

```
          「大語彙連続音声認識エンジン Julius」
                        使用許諾書

    Copyright (c)   1991-2002 京都大学
    Copyright (c)   1997-2000 情報処理振興事業協会(IPA)
    Copyright (c)   2000-2002 奈良先端科学技術大学院大学

-------------------------------------------------------------------------

「大語彙連続音声認識エンジン Julius」（Julianを含む）は、京都大学音声
メディア研究室、奈良先端科学技術大学院大学音情報処理学講座で開発されて
います。1997年度から3年間、情報処理振興事業協会(IPA)が実施した「独創的
情報技術育成事業」の援助を受けました。

京都大学、IPA、及び奈良先端科学技術大学院大学は、著作者であり著作権を
留保していますが、本使用条件の全てを受諾し遵守する限り、ソースコードを
含む本プログラム及びドキュメンテーション(以下あわせて「本ソフトウェア」
と言う)を無償であなたに提供します。あなたが本ソフトウェアを使用したと
きは、本使用条件の全てを受諾したものと看做されます。

                        【使用条件】

1. 京都大学、IPA、及び奈良先端科学技術大学院大学は、あなたが本使用条件
の全てを受諾し遵守する限り、本ソフトウェアの全部又は一部について使用、
複製、翻案、変更、組込み、結合することおよびそれらの複製物、翻案物、変
更物等を配布、頒布、送信することに関し、著作権及び著作者人格権を行使し
ません。ただし、あなたを含め本ソフトウェアの使用者は、本ソフトウェアの
全部又はその一部を変更してその複製物を配布、頒布、送信などして第三者に
提供するときは第2項の表示記載に加え本ソフトウェアを変更した旨、変更者
及びその変更日を明確に表示するものとします。

2. あなたは、使用、複製、翻案、変更、組込み、結合その他本ソフトウェア
の使用態様の如何にかかわらず、その複製物、翻案物、変更物等の全部又は一
部を第三者に提供するときは、本ソフトウェアに下記の著作権表示及び公開の
趣旨を含む本使用条件の全て(この文書ファイル)をいささかも変更することな
くそのまま表示し添付しなければなりません。

                        記
  Copyright (c) 1991-2002 京都大学
  Copyright (c) 1997-2000 情報処理振興事業協会(IPA)
  Copyright (c) 2000-2002 奈良先端科学技術大学院大学

3. 京都大学、IPA、及び奈良先端科学技術大学院大学は、本ソフトウェアを研
究開発の試作物をあるがままの状態で無償公開提供するものであり、本ソフト
ウェアに関し、明示、黙示を問わず、いかなる国における利用であるかを問わ
ず、また法令により生じるものであるか否かを問わず、一切の保証を行いませ
ん。ここで言う保証には、プログラムの品質、性能、商品性、特定目的適合性、
```

欠陥のないことおよび他の第三者の有する著作権、特許権、商標権等の無体財産権や営業秘密その他の権利利益を侵害しないことの保証を含みますがそれに限定されるものではありません。あなたを含め本ソフトウェアの使用者は、本ソフトウェアが無保証であることを承諾し、本ソフトウェアが無保証であることのリスクを使用者自身で負うものとします。裁判所の判決その他何らかの理由によりあなたに課せられた義務と本使用条件が相容れないときは、あなたは、本ソフトウェアを使用してはなりません。本ソフトウェアの使用又は使用できないことに関してあなた及び第三者に生じる通常損害、特別損害、直接的、間接的、付随的、派生的な損害(逸失利益を含む)一切につきそれが契約、不法行為、ネグリジェンス、製造物責任等いかなる国のいかなる法律原因によるかを問わず、賠償しません。

4．あなたは、本ソフトウェアを原子力関連、航空管制その他の交通関連、医療、救急関連、警備関連その他人の生命、身体、財産等に重大な損害が発生する危険を有するシステムに使用してはいけません。

5．本ソフトウェアの使用に関しては、日本国の法律を準拠法とし、東京地方裁判所を第一審の専属管轄裁判所とします。

6．本ソフトウェアを利用して得られた知見に関して発表を行なう際には、「大語彙連続音声認識エンジン Julius」を使用したことを明記して下さい。

7．本ソフトウェアのメンテナンスやサポート、上記条件以外の使用等に関しては、奈良先端科学技術大学院大学音情報処理学講座、または京都大学音声メディア研究室に照会下さい。

Last modified: 2002/09/11 22:10:03

Julius/Julian 開発履歴

```
1998
     02/20    Julius Rev.1.0 release
     04/14    Julius Rev.1.1 release
     07/30    Julian Rev.1.0 release
     10/24    Julius Rev.1.2 release
1999
     02/20    Julius Rev.2.0 release
     03/01    Julian Rev.2.0 release
     04/20    Julius/Julian Rev.2.1 release
     10/04    Julius/Julian Rev.2.2 release
2000
     02/14    Julius/Julian Rev.3.0 release
     05/11    Julius/Julian Rev.3.1 release
     06/23    Julius/Julian Rev.3.1p1 release
2001
     02/27    Julius/Julian Rev.3.1p2 release
     06/22    Julius/Julian Rev 3.2(beta) release
     08/18    Julius/Julian Rev 3.2 release
2002
     07/01    Julius/Julian Rev 3.3(beta) release
     09/09    http://sourceforge.jp/projects/julius launched
     09/11    Julius/Julian Rev 3.3 release
```

Last modified: 2002/09/11 21:00:39

Julius users ML

Julius users ML とは

Julius users ML (julius-users@luky.org) はJulius全般に関して議論するメーリングリストです．柴田(ひ)@福岡さんのご厚意により運営していただいております．

Juliusのバグ報告やインストールに関する質問などどんな話題でもOKです． Juliusの開発関係者も参加しております．お気軽にご参加下さい．

参加方法

参加（登録）希望の方は

subscribe あなたのお名前

というコマンドをメール本文の1行目の行頭に書いて，アドレス julius-users-ctl@luky.org へ送って下さい。たとえば、

subscribe Akinobu Lee

などと書いて頂ければOKです。

なお、MLに投稿された内容はWeb上で閲覧可能な形て公開され、DOC-CD等へ収録される場合もあります。投稿に際してはこの点を御理解いただきますようお願いします。

過去の投稿記事アーカイブ

過去にJulius users MLに投稿された記事は，以下のWebページから閲覧できます．

- 一覧リスト
- 全文検索

Last modified: 2001/08/17 09:40:23

連絡先・リンク

リンク

- Julius ホームページ
- Julius 開発サイト：最新CVSスナップショットはこちらで公開中
- SAPI版Julius/Julianホームページ(京大)
- Julius for Windows (DLL版)(坂野@名大)
- 連続音声認識コンソーシアム

連絡先

- 問い合わせ先: julius@kuis.kyoto-u.ac.jp
- 製作者：李 晃伸 (ri@is.aist-nara.ac.jp)

Last modified: 2002/09/11 21:00:39