大語彙連続音声認識プログラムの開発

目 次

第 1 章　はじめに *1*

第 2 章　認識プログラムの外部仕様 *3*

2.1　入出力 *3*

2.2　音響モデル (HTK フォーマット) *3*

2.3　単語辞書 (HTK フォーマット) *4*

2.4　言語モデル (ARPA 標準フォーマット) *4*

2.5　処理系の概要 *5*

2.6　認識プログラムの動作環境 *6*

第 3 章　認識プログラムの内部仕様 (アルゴリズム) *7*

3.1　第 1 パス *7*

3.1.1　木構造化辞書 *7*

3.1.2　1-best 近似 vs. 単語対近似 *8*

3.1.3　unigram の factoring vs. bigram の factoring *9*

3.1.4　単語間の音素環境依存性の扱い *10*

3.2　第 2 パス *11*

3.2.1　単語グラフ vs. 単語トレリス *11*

3.2.2　best-first 探索 vs. ビーム探索 *14*

3.2.3　N-best 探索 *15*

3.3　その他 *15*

3.3.1　言語モデル確率の重み *15*

3.3.2　単語挿入ペナルティ *16*

3.3.3　言語モデルのデータ構造 *16*

3.3.4　音響モデルの確率計算の高速化 *17*

第 4 章　Julius 2.1(98 年度版) からの変更点 *18*

第 5 章　日本語大語彙連続音声認識エンジン Julius *22*
5.1　旧バージョンをご利用の方へ *22*
5.2　Julius について *23*
5.3　パッケージの内容 *24*
5.4　インストール *24*
5.5　実行方法 *25*
5.6　問い合わせ先 *26*
5.7　作成者 *26*

第 6 章　Julius インストールの手引き *28*
6.1　Quick Start *28*
6.2　準備 *29*
6.3　configure の実行 *29*
6.4　コンパイル *30*
6.5　インストール (optional) *30*

第 7 章　Julius チュートリアル *32*
7.1　ファイルの準備 *32*
7.1.1　モデル定義ファイル *32*
7.1.2　設定ファイル (jconf ファイル) の編集 *33*
7.2　Julius の起動 *33*
7.3　音声データ入力 *34*
7.4　認識の実行と結果の出力 *34*
7.5　音声入力ソースの選択 *35*
7.6　バイナリ N-gram ファイルの利用 *37*
7.7　起動時のエラー *37*

第 8 章　マイク入力について *38*
8.1　音響モデルの制限 *38*
8.2　サポート OS *39*
8.3　コンパイル時の設定 *40*
8.4　マイク入力の認識の流れ *41*
8.5　音量・トリガレベルの調節 *41*
8.6　adinrec によるマイクデバイスの動作チェック *42*
8.7　どうしてもうまくいかないときは (Linux) *42*

第 9 章　configure オプション解説 *44*
9.1　標準オプション *44*
9.2　オプション詳細 *45*

第 10 章 サンプル設定ファイル Sample-j.jconf *48*

第 11 章 Julius の扱うファイル形式の仕様 *54*
11.1　HMM 定義ファイル *54*
11.2　HMMlist ファイル *56*
11.3　単語 N-gram 言語モデルファイル *56*
11.3.1　ARPA 標準形式 *56*
11.3.2　バイナリ N-gram 形式 *57*
11.4　単語辞書ファイル *57*
11.5　マイク入力 *58*
11.6　音声波形ファイル *59*
11.7　特徴パラメータファイル *60*
11.8　1 入力あたりの仮説単語長制限 *60*

第 12 章 音素環境依存性の扱いについて *62*
12.1　Julius での音素環境依存性の扱い方 *62*
12.2　HMMList ファイル *63*
12.3　チェック方法 *64*
12.4　注意点 *64*

第 13 章 Julius man ページ *66*

第 14 章 1999 年度 大語彙連続音声認識　評価実験結果 *77*
14.1　評価用サンプル IPA-98-TestSet *77*
14.2　評価基準 *77*
14.3　音響モデルの評価 (男性) *78*
14.4　音響モデルの評価 (女性) *78*
14.5　言語モデルの評価 *78*
14.6　デコーディングアルゴリズムの評価 *79*
14.7　20K システムの構成 *79*
14.8　60K システムの構成 *80*

第1章　はじめに

大語彙連続音声認識 (ディクテーション) 技術 [1, 2, 3] は、音声を利用した様々なアプリケーションの基盤になる技術であり、音声入力ワープロ、放送やオーディオテープの書き起こしなどの応用が考えられる一方、そこで培われる要素技術は音声対話システムや種々の音声インタフェースに利用できるであろう。

近年の音声認識技術の進展と計算機能力の向上から、IBM, Dragon Systems, Lernout&Hauspie (L&H) をはじめとする欧米の有力な研究開発機関では、ディクテーションソフトウェアの商用化を進めている。我が国においても、日本 IBM が発売している「ViaVoice 日本語版」等が反響をよんでいる。

しかしながら、音声認識技術を利用した種々のアプリケーションを開発するためには、音響モデルや単語辞書・言語モデルなどがモジュール化されており、開発者の手で、目的に応じて作成・編集・置換できたり、認識プログラムの設定を自由に変更できることが必要である。また、音声認識技術の一層の研究を促進するためには、これらがオープンになっていることも重要である。

大語彙連続音声認識システムには、高精度の音響モデル、高精度の言語モデル、効率のよい認識エンジン (デコーダ) が必要であるが、このように高度なシステム開発と要素技術の研究をバランスよく推進していくためには、共通のプラットフォームが望まれる。特に英語のディクテーション技術の開発は、DARPA のプロジェクトによる共通の研究・評価基盤の整備によって飛躍的に発展したことから、我が国においても研究機関の枠を越えて共通基盤を整備することが急務であると考えられる。

そこで我々は、毎日新聞の記事データを共通の言語・音声コーパス [4] に採用し、共有のソフトウエアプラットフォームを開発する 3ヶ年 (1997 ~ 1999 年度) のプロジェクトを推進している [5, 6]。この成果である「日本語ディクテーション基本ソフトウエア」は、標準的な音響モデル、言語モデル、及び認識エンジンから構成され、一般に無償で公開されている。これらのコーパスとソフトウエアの関連を図 1.1 に示す。

本稿では、この中で認識プログラムに関して、その外部仕様と基本的なアルゴリズム (内部仕様) を説明する。

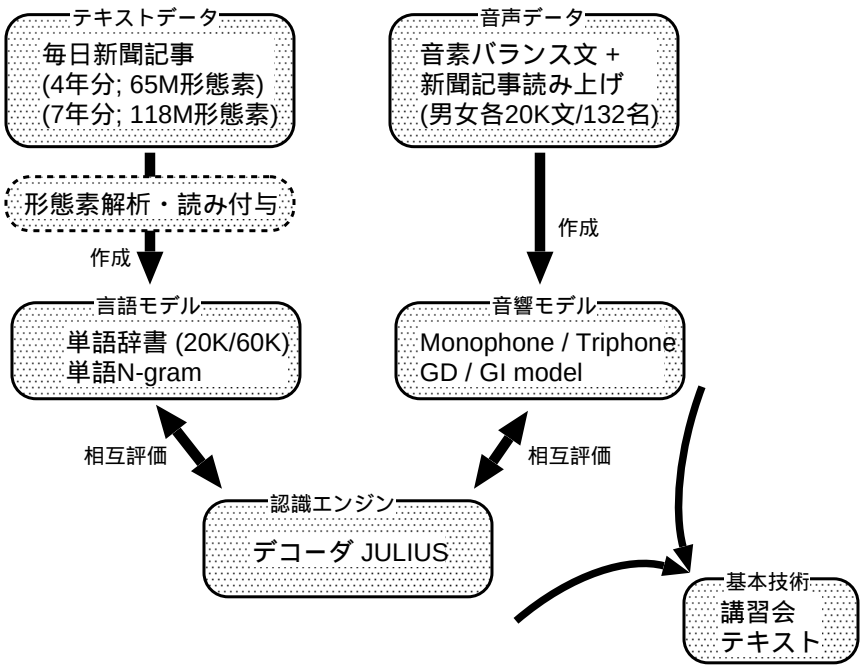

図 1.1. 大語彙連続音声認識のための研究基盤の整備

第2章　認識プログラムの外部仕様

大語彙連続音声認識プログラムは、京都大学で主要部分の開発を行っている [7]。コード名は Julius であり、今年のバージョンは 3.x である。

2.1 入出力

入力は連続音声あるいは音響特徴量のファイルである。

音声入力に関しては、音声波形ファイル以外に、netaudio を介した DAT-LINK からの入力、あるいは Sun ワークステーション (Solaris, SunOS)、Silicon Graphics ワークステーション (IRIX)、パソコン (Linux) のマイク端子からの入力が可能である。ただし、すべての機種・OS に対応していることは保証しない。

音響特徴量のファイルは HTK で用いるフォーマットと互換性を持たせており、抽出する音響特徴量は音響モデルで指定する。MFCC (Mel-Frequency Cepstral Coefficients) ベクトルとパワー及びそれらの時間差分を基本とする。CMN (Cepstral Mean Normalization) を実行するか否かも音響モデルで指定する。ただしマイク入力に対して CMN を行う場合は、前の発話のケプストラム平均を利用する。

単語間にポーズが入ってもよいが、ポーズの出現に関しては言語モデルで指定する。つまり、句読点や記号などをポーズに書き換えるようにして、それらの出現確率は N-gram の枠組みでモデル化する。ただし句点相当の長いポーズで、認識プログラムは入力を打ち切って処理する。

出力は最尤単語列である。N-best 候補を求めることもできる。

2.2 音響モデル (HTK フォーマット)

HMM を基本とし、HTK フォーマット [8] に互換性を持たせる。

混合連続分布 HMM を扱える。共分散行列は対角成分のみからなるものとする。状態間で分布を共有する tied-mixture モデルも扱える。ただし、tied-mixture(tmix) であることを明示して、分布の集合 (コードブック) を定義する必要がある。音素毎にコードブックを定

義すれば、phonetic tied-mixture モデルとなる。

状態遷移に関しては、初期状態からの遷移及び最終状態への遷移は 1 つのみに制限する。これは木構造化辞書の実装を容易にするためである。継続時間制御は実装していない。

認識プログラムでは、音素環境独立 (CI) モデルだけでなく、音素環境依存 (CD) モデルを扱える。ただし、triphone までである。これも、a-t+a のように先行・後続音素を参照して、音素環境依存モデルであることを明示する必要がある。第 1 パスでは単語間の依存性は厳密に扱わず、第 2 パスで処理する。

triphone モデルを用いる場合には、辞書中の単語に出現しうるすべての組合せについて、そのエントリ (実体のモデル) もしくは置換 (対応させるモデル) を定義しておく必要がある。この対応関係はファイルで与える。

2.3 単語辞書 (HTK フォーマット)

語彙のエントリの表記と音素記号列からなる単語辞書は、HTK フォーマットに準拠する。

語彙エントリ数 (発音表記の異なり数) の上限は、最大 65535 まで可能である。

音素表記は、日本音響学会の音声データベース委員会で策定されたものを標準とする。そうでない場合は、単語のかな表記から音素表記への変換規則 (プログラム) を、音響モデル開発者の責任で用意する。

2.4 言語モデル (ARPA 標準フォーマット)

単語 N-gram を基本とし、ARPA 標準フォーマットに互換性を持たせる。これは、CMU-TK[9] でも用いられており、また HTK のフォーマットもほぼ等価である。読み込みの高速化のため、独自型式のバイナリファイルにコンパイルした形式も実装する。

第 1 パスでは bigram のみを用い、第 2 パスで trigram を適用する。ただし第 2 パスは (第 1 パスと逆向きの) right-to-left に処理するため、通常と逆向きの trigram を用意する必要がある。また unigram の最初のエントリを未知語カテゴリとして扱い、辞書の単語で unigram 確率が定義されていないものに対しては、未知語カテゴリの確率をその総数で補正した確率が用いられる。

言語モデルと単語辞書の一貫性を保証するため、単語辞書の作成も、原則として言語モデル開発者の責任である。これには、格助詞「は」「へ」の適正な処理などのかな表記への変換も含まれる。ポーズの出現に関しても、句読点や記号などを利用して言語モデルで表現する。

2.5 処理系の概要

認識プログラムの概要を表 2.1 にまとめる。

表 2.1. 認識プログラム Julius の概要

	音響モデル	言語モデル	探索
第 1 パス	単語内 CD-HMM	単語 bigram	1-best
第 2 パス	単語間 CD-HMM	単語 trigram	N-best

CD: Context-Dependent model

認識プログラムを中心とする認識システムの構成と処理の流れを図 2.1 に示す。

認識処理は、基本的に 2 パスから構成される。第 1 パスでは、単語 bigram を用いてフレーム同期ビームサーチを行う。音素環境依存モデルについては、単語内は適用されるが、単語間については近似値を与える。この探索においては、最尤 (1-best) 近似を用いる。これは、実時間の認識処理が可能である。その結果、各フレーム毎にビーム内に終端ノードが残った単語候補についてそのスコアと始端フレームのリストを、単語トレリスのインデックス形式で保存する。第 2 パスでは、この中間結果として得られた仮説に対して、高精度なモデルを適用して再評価及び再探索を行う。すなわち、単語 trigram と単語間の音素環境依存モデルを厳密に適用しながら、スタックデコーディングサーチを行う。正しい N-best 候補を求める。

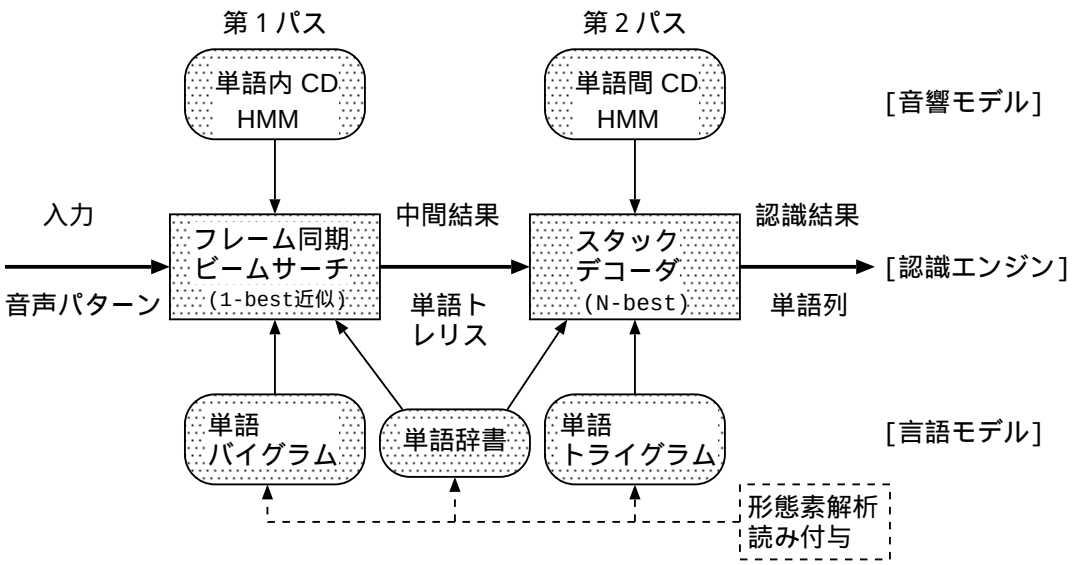

図 2.1. 認識システムの構成と処理の流れ

2.6 認識プログラムの動作環境

標準的な Unix の計算機で動作することをめざす。少なくとも Solaris, SunOS, IRIX, Linux では動作確認を行う。

処理に要する時間は、ハイエンドの計算機で実時間の 10 倍以下とする。ただし、簡略なモデルを用いる場合には、実時間に近い応答ができるようにめざす。

処理に要するメモリ量は、ハイエンドの計算機で標準的なメモリサイズ (64 ~ 256MB) におさまるようにする。

ただし、処理時間とメモリ量に関してはトレードオフの関係があるので、メモリ量の制約を優先するよう指定できるようにする。

第3章　認識プログラムの内部仕様 (アルゴリズム)

認識プログラム (Julius) で採用するアルゴリズムについて説明する。

3.1 第1パス

時間フレームに同期して探索を進める。HMM のトレリス上の探索と等価である。アルゴリズムの設計や実装が比較的容易である。仮説間の評価値もそのまま比較できる。ただし、1 フレーム毎の情報に基づいて枝刈りを行うので、安定な照合を行えず、局所的な変動の影響を受けやすい。このため、マルチパスで総合的な認識を行う。

3.1.1 木構造化辞書

N-gram 言語モデルの最も単純な実装は、確率有限状態オートマトンとみなして、静的な単語ネットワークを構成する方法である。この場合、単語間の遷移に言語モデルの確率を付与する (図 **3.1** 参照)。bigram を使用する場合、ネットワークのノード (HMM の状態数) は、語彙サイズに対してほぼ線形に増加する。また、単語間のアーク (遷移数) は語彙サイズの 2 乗になる。trigram を使用する場合の事態はもっと深刻である。

単語辞書に関しては、プレフィックスを共有する木構造化を行うことにより、状態数を削減できる (図 **3.2** 参照)。この効果は、語彙サイズが大きくなるにつれて顕著になり、大語彙連続音声認識では不可欠となる。しかし、木構造化辞書においては、単語の終端 (=木の葉ノード) の方に達しないと単語を同定できないので、N-gram の確率を静的に埋め込むことはできない。そこで、単語の終端に達する毎に N-gram の確率を動的に加える。ただし、直前の単語 (履歴) をいちいちトレースバックすることなく同定できるように、最尤経路を与える直前の単語 (履歴) の情報を伝播 (コピー) しておく必要がある。なお、始終端を特定したい際には、単語境界のフレームもコピーする。

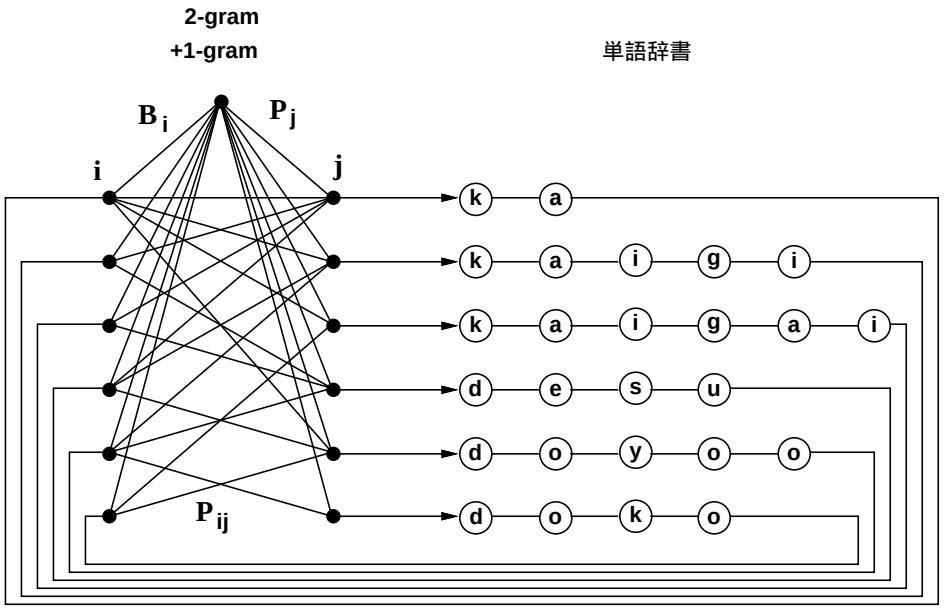

図 3.1. 単語辞書と N-gram の有限オートマトン

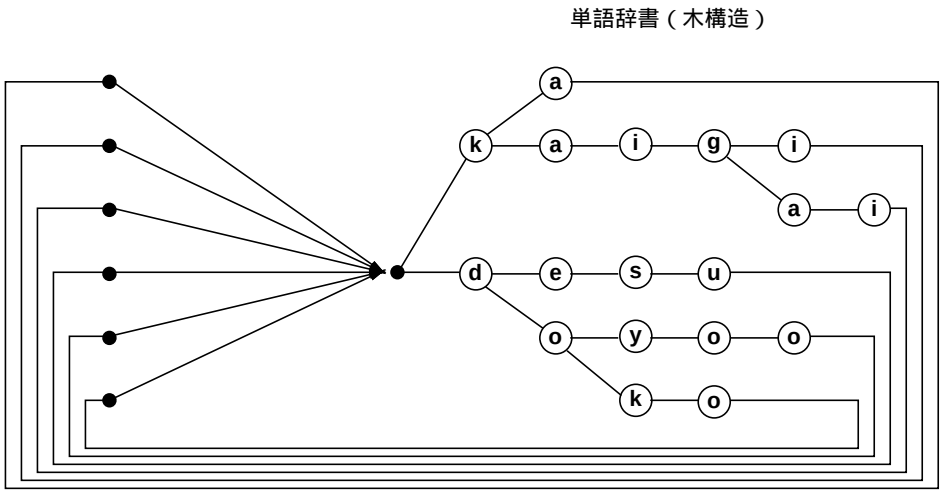

図 3.2. 木構造化単語辞書

3.1.2 1-best 近似 vs. 単語対近似

木構造化辞書と N-gram の実装に関しては、あくまでも静的な木を 1 つ用意しておいて、すべてのノードにバックポインタ (単語履歴) をコピーする方法と、単語履歴毎に木を動的に生成する方法とがある [10]。

前者の場合は最尤の直前単語からのビタビ経路のみを計算することになるのに対して、後者の場合は直前の単語に応じたビタビ計算を行なえる。一般に、単語のスコアや境界はそれ以前の単語の影響を受けるので、(N-best 候補を求める場合) 単語履歴毎に異なった候補を求めるべきであるが、そうすると仮説が膨大になるので、何らかの近似を仮定して仮

説のマージを行なう。前者は履歴に依存しない 0 次近似 (=1-best 近似)、後者は直前の単語のみに依存する 1 次近似 (=単語対近似)[11] といえる。

ここでは前者を基本的に採用するが、後者も比較のために実装する。

なお木構造化辞書で 1-best 近似を採用すると、次節で述べるように不完全 (楽観的) な言語モデル確率を用いたまま最尤経路選択 (マージ) を行なうことになるので、N-best だけでなく最尤の候補の探索にも影響を与える (図 **3.3** 参照)。

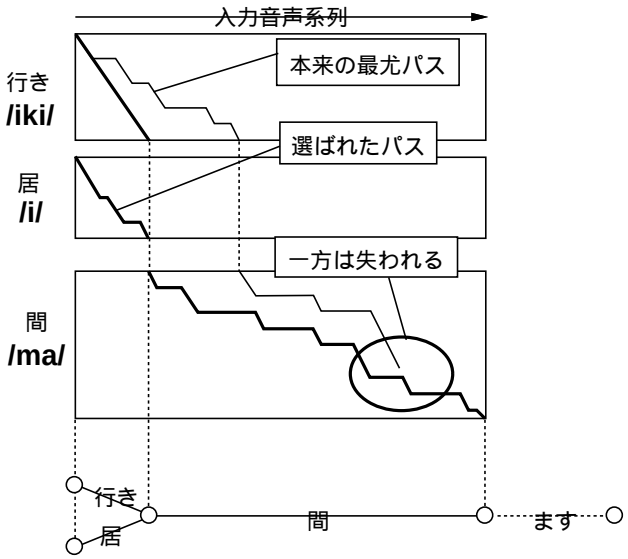

図 **3.3.** 1-best 近似による仮説のマージ

3.1.3 unigram の factoring vs. bigram の factoring

木構造化辞書と N-gram を単純に組み合わせると、単語の終端に到達するまで、言語モデルの確率が与えられないことになる。言語的制約が適用されるタイミングが遅れると、枝刈りの際の評価値に反映されないために、最適な解が単語の途中でビーム幅からあふれる可能性がある。語彙サイズが大きくなるにつれて、この問題は顕在化する。

そこで、言語モデルの確率を分解して、木の途中のノードにも割り振るようにする。この factoring は、プレフィックスを共有する単語に対する N-gram 確率の最大値を割り振り、累積値を順に精算していくのが基本である (図 **3.4** 参照)[10, 12]。

この際に、bigram 確率を factoring する方法と unigram 確率を factoring する方法が考えられる。unigram 確率の方は言語モデルの制約がかなり弱くなるが、単語辞書を読み込んで木構造化辞書を構築した際に、事前に静的に確率を求めておくことができる。これに対して、bigram 確率の factoring には多くの計算量あるいは記憶量を要するが、特に日本語の形態素のように短い語彙エントリが多い場合には、bigram の制約を早期に適用すること

が必要であると考えられる。この実装においては、探索時に動的に前のノードで与えた値を差し引く方法と、単語履歴毎に木を生成する際に木の全ノードの値を計算する方法とがある。

ただし、後述する単語トレリス形式を採用した 2 パス探索では、unigram 確率の factoring を用いても、第 1 パスの誤りを第 2 パスで回復できることがわかった。

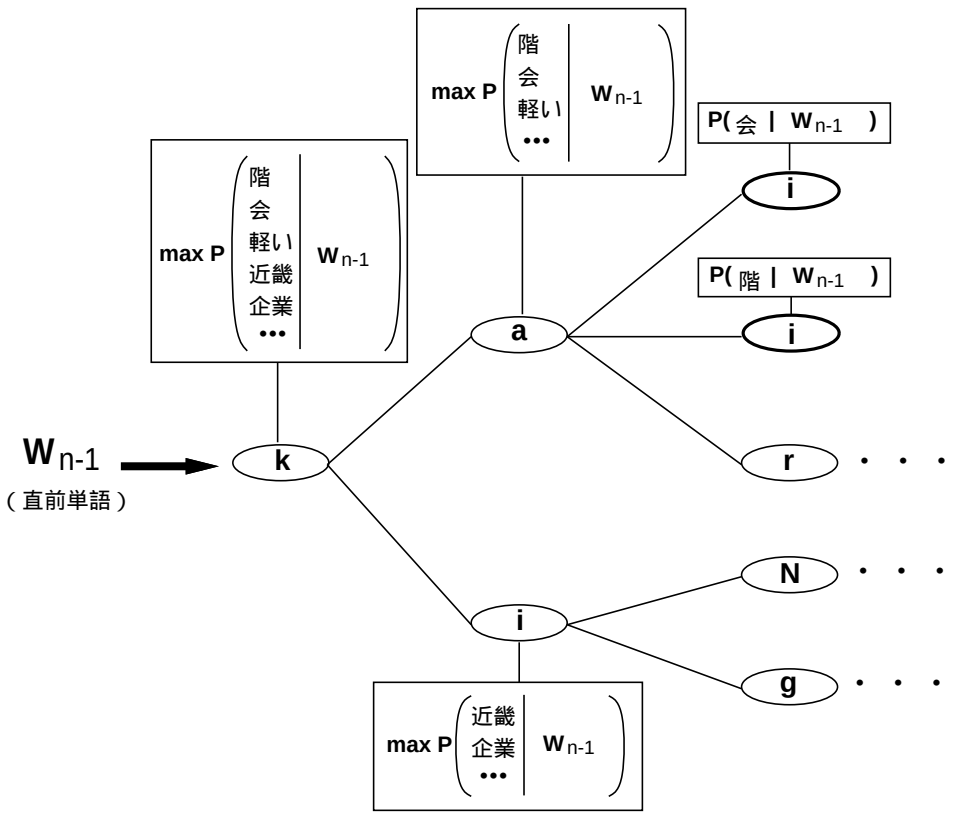

図 3.4. bigram 確率の factoring

3.1.4 単語間の音素環境依存性の扱い

一般に単語終端では、次単語が未知であるために次の音素を同定できず、音素環境依存の音響モデルを直接適用することができない。

厳密な方法は、すべての可能な音素環境に関して仮説を分離して、個別に確率を計算し、次単語の時点で適切な仮説と接続することであるが、仮説管理が複雑になりオーバーヘッドが大きい。そのためとりあえず、可能な音素環境依存モデルの確率の最大値や平均値で近似したり、biphone や monophone で代用しておいて、次単語がある程度絞れた後に厳密な再評価をすることが多い。

特にマルチパス探索においては、最初のパスで単語間の音素環境依存性を考慮せずに、

階直前単語)

リスコアリングの段階で適用することが多い。ただしこれは後の段階で高い音響スコアを与えることになるので、A*探索の適格性を著しく損ない、探索を不安定にする要因となる。これに対して、可能な音素環境依存モデルの最尤値で近似しておくと、計算量は増大するが、A*適格性を満たす [13]。

単語始端の音素環境依存性の扱いは、単語履歴の管理方法に依存する。単語対近似を採用していれば正しい音素環境依存モデルで計算できるが、1-best 近似の場合は、最尤履歴以外からの経路に対して誤ったモデルを適用することになる。

ここでは、単語終端では可能な音素環境依存モデルの最大値で近似し、単語始端では最尤履歴から与えることにする。

3.2 第 2 パス

1 パスの処理において、高精度のモデルを適用すると多くの仮説を扱うことになり、全体としてかえって処理量が大きくなりかねない。特に trigram を適用するには、2 単語履歴を考慮する必要があるので、実装が複雑になる。

そこで、効率よく高精度のモデルを適用するために、マルチパス探索 [14, 15] を採用する。第 1 パスで、ある程度の精度の音響モデル・言語モデルを用いて入力 (ポーズまで) を完全に処理して、この中間結果を基に、第 2 パスにおける探索空間を限定すると共に、先読み情報 (ヒューリスティック) として利用する。ここでは認識精度を優先して、最初から音素環境依存モデルを使用する [1]。ただし、単語間の結合に関しては処理が煩雑になるため、最初は近似値で与える。言語モデルは、やはり処理の簡便性から bigram を最初に用いて、第 2 パスで trigram を適用する。

第 2 パスでは、候補もかなり絞られている上に、完全な先読みもできるので、音素・単語単位で探索を行う。これは単語の木の上の探索と等価であり、安定した照合に基づいて枝刈りを行うことができる。仮説の評価は、第 1 パスで計算された入力全体のスコアを反映 (先読み) して行う [16]。

3.2.1 単語グラフ vs. 単語トレリス

第 1 パスと第 2 パスの間の中間表現の形式 (インタフェース) としては、以下が考えられる。

- N-best 単語列 [11, 17]

  単語列 (文) の複数 (N-best) 候補を受け渡す。単語間の音素環境依存モデルや trigram

による再評価の実装が容易である。第二パスで探索機構が不要であり、一般的な自然言語処理技術との親和性が高い。入力 (文長) が長い場合、かなり多数の候補を求めても、一単語のみ異なる類似候補しか得られないので、結果として効率が悪くなる。

- 単語グラフ (単語ラティス) [18, 14, 19][20]

  単語のスコアと始端・終端の集合を求める。そのグラフをたどることにより、結果として多数の N-best 候補が求められるので、効率のよい表現といえる。単語のスコアや境界は、それ以前の単語の影響を受けるので、単語履歴毎に異なった候補を求めるべきであるが、そうすると候補が膨大となるので、直前の単語のみに依存させる単語対近似 [11] を仮定することが多い。

- 単語トレリス [16, 21, 7]

  単語毎のスコアや区間を決定的に求めるのではなく、単語の終端の状態のトレリス (=スコアと対応する始端) を保存する。次のパスにおいてもトレリス接続の計算が必要になるが、より高精度なモデルを用いた仮説の正確な再評価ができる。精度が高い表現である。そのままでは単語の絞り込みが直接的に行えないので、ビームに残った単語ノードを逆引きできるようにインデックスを作成する。

これらを、図 3.5・図 3.6・図 3.7 に図示する。

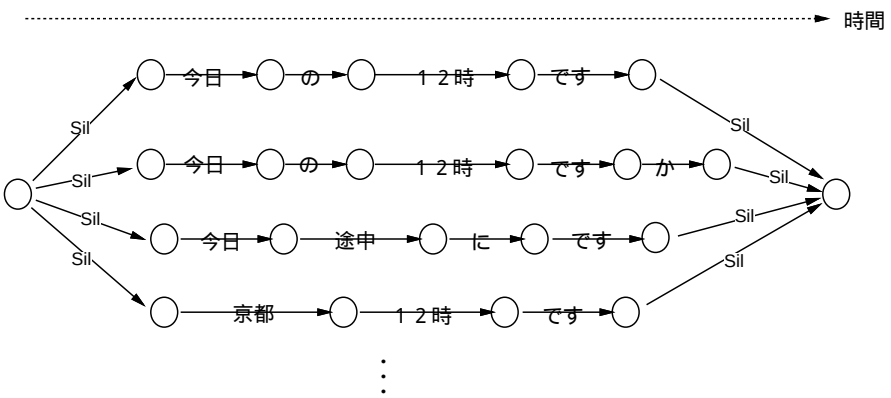

図 3.5. N-best 単語列インタフェース

単語トレリスと単語グラフのいずれを用いるかは、第 1 パスの計算において、1-best 近似を仮定するか、単語対近似を用いるかとも関係する。単語対近似を利用して、直前の単語毎に異なったビタビ経路を求めていれば、その単語境界のみを残す単語グラフで十分であるし、1-best 近似によるビタビ計算であれば、直前の最尤単語との境界のみを残すのは危険であり、単語終端状態がビームに残った範囲すべて (単語トレリスのインデックス) を

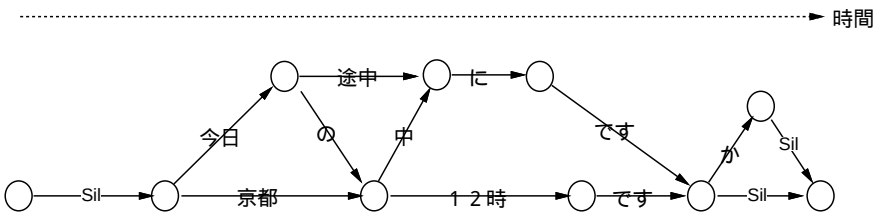

図 3.6. 単語グラフインタフェース

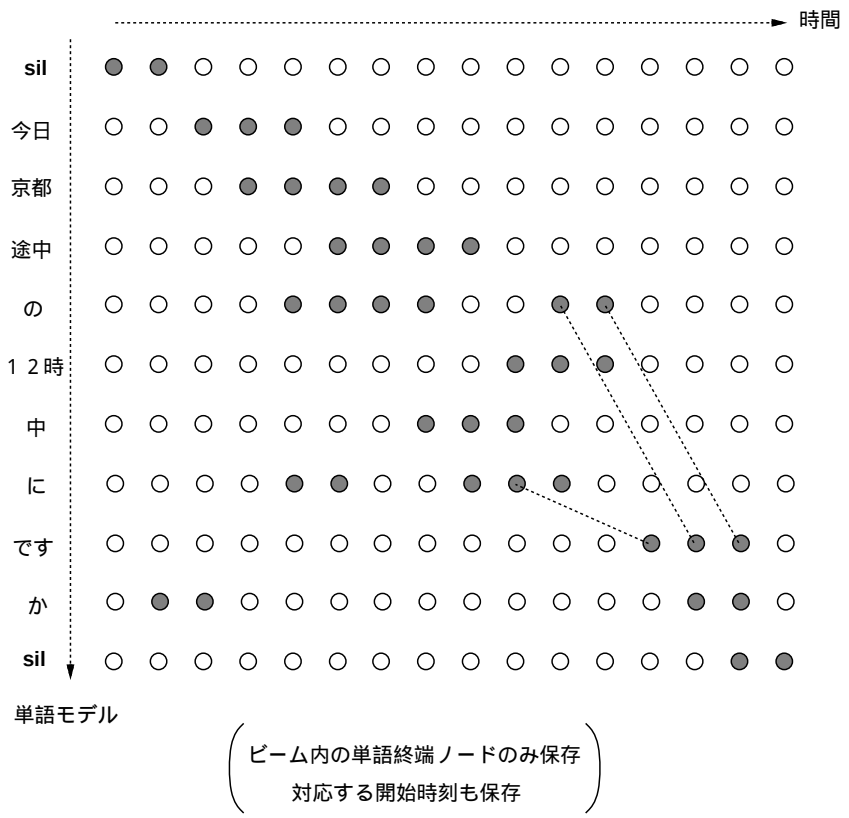

図 3.7. 単語トレリスインタフェース

残すのが適当である。もちろん、単語対近似とトレリス表現を併用すれば、より精密な計算ができる。

ここでは、単語対近似による計算の方が大きいと考え、1-best 近似とトレリスインデックスを基本とする。1-best 近似による第 1 パスでの精度の低下は、トレリス表現を介した第 2 パスにより回復する。

なお実際には、第 2 パスで単語間の音素環境依存処理を行うことにより、第 1 パスと単語境界がずれる可能性があるので、数 (=5) フレーム程度のずれを許容して次単語の接続を行う。

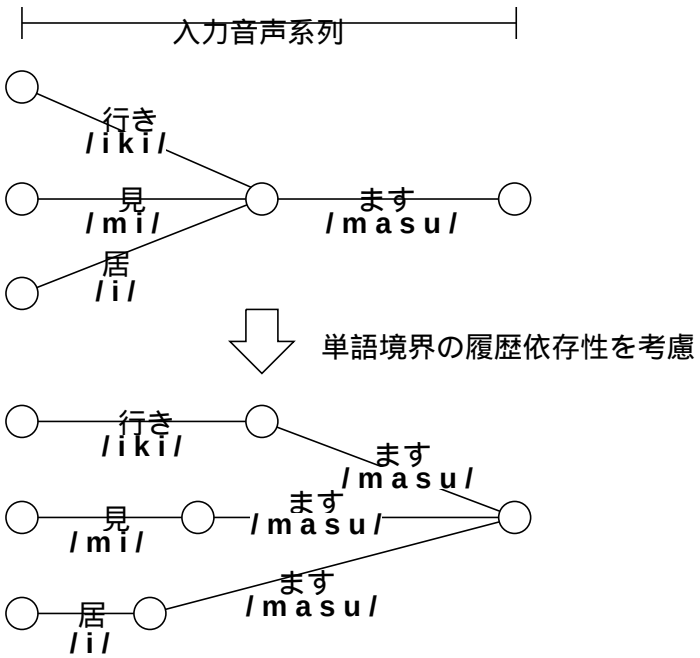

図 3.8. 単語対近似による単語境界の保持

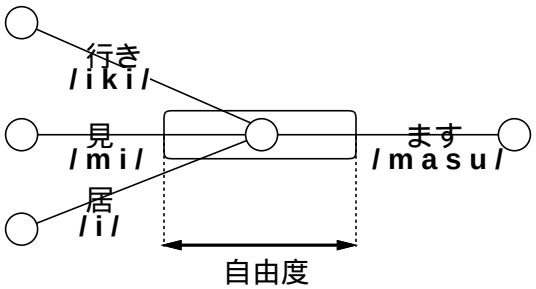

図 3.9. トレリス表現による単語境界の保持

3.2.2 best-first 探索 vs. ビーム探索

第 2 パスの探索アルゴリズムとしては以下が考えられる。

- best-first 探索

  最尤の仮説から順に展開していく。第 1 パスの認識精度 (ヒューリスティック) がよい場合は効率よく探索が進行するが、そうでない場合は仮説展開が幅優先的になり、探索に失敗する (解が得られない) 場合もある。

- ビームサーチ

  仮説の長さ (=単語数) に同期して、一定の数 (=ビーム幅) だけ仮説を展開していく。一定時間において必ず解が得られるが、多くの無駄な仮説展開も必ず生じる。

- エンベロープサーチ

単語境界の履歴依存性を考慮

フレームを基準として (時間長の短い順に) 仮説を選択して、単語単位で展開する。先に展開された仮説のスコアに基づいて、枝刈りのしきい値 (エンベロープ) を設定し、仮説が展開される毎に順次更新していく。

- ビーム幅つき best-first 探索

  上記の best-first 探索とビームサーチの中間的な探索手法である。基本的には best-first に探索を進めていくが、各単語長の仮説数に上限 (=ビーム幅) を設定し、それに達したら、その長さ以下の仮説を破棄する。これにより、幅優先な展開に陥る事態を回避し、必ず解が得られるようにする。さらにエンベロープサーチと同様に、既に展開した仮説のスコアに基づいて、各フレーム毎に枝刈りのしきい値を設定することにより効率化できる。

ここでは、ビーム幅つき best-first 探索を実装する。

3.2.3 N-best 探索

このスタックデコーダを利用した探索により、sentence-dependent な仮説の評価が行えるが、第 1 パスと第 2 パスで言語モデル・音響モデルともに異なり、第 2 パスのモデルの方が高いスコアを与える可能性があるために、A*適格性は満たしていない。そのため最初に得られる解が最尤解である保証はない。したがって、N 個の候補を出力してから、それらをスコアでソートして最尤解を求めるのが望ましい。通常は 10 個程度を求めれば十分である。

3.3 その他

3.3.1 言語モデル確率の重み

情報理論に基づく定式化であるベイズ則に従うと、各仮説の評価値の計算は、言語モデルの確率と音響モデルの確率を単純にかけ合わせることになるが、実際には、言語モデルの確率値の分布が、音響モデル (特に連続分布 HMM) の確率値の分布に比べてかなり狭いため、言語モデルの確率に 1 より大きい重みを乗じることが効果的であることが知られている。

また bigram より trigram の方が言語的な予測精度が高いことから、bigram を用いるときより trigram を用いるときの方が、重みを大きくするのが効果的である。また 2 パスサー

チにおいては、第 1 パスで候補が絞れていることから、第 2 パスの方が大きい重みを用いるのが一般的である。

3.3.2 単語挿入ペナルティ

言語モデルの確率や音響モデルの継続時間制御などにもかかわらず、局所的なマッチングの連続により、多数の短い単語からなる系列の方がスコアが高くなり、短い単語の挿入誤りが生じる場合がある。こうした挿入誤りを防ぐ (correctness より accuracy を優先する) ために、単語が遷移する毎に単語挿入ペナルティ(insertion penalty) を加えることが効果的であることが知られている。

このペナルティの値は、前節の言語モデルの重みと関連がある。すなわち、言語モデル重みを大きくするほど、単語が挿入されにくくなるので、ペナルティの値は小さくする必要がある [22]。

3.3.3 言語モデルのデータ構造

大語彙の N-gram は膨大な記憶量を必要とする。参照する回数が多いので、記憶量を削減すると共に、参照の局所性を反映したコンパクトなデータ構造を採用する必要がある。具体的には、同一の単語履歴に対する trigram (及び bigram) は連続してシーケンシャルにアロケートする、またこれらは単語履歴に対応する 2 単語の bigram (及び 1 単語の unigram) からリンクすることにより、効率よく表現する (図 **3.10** 参照)。

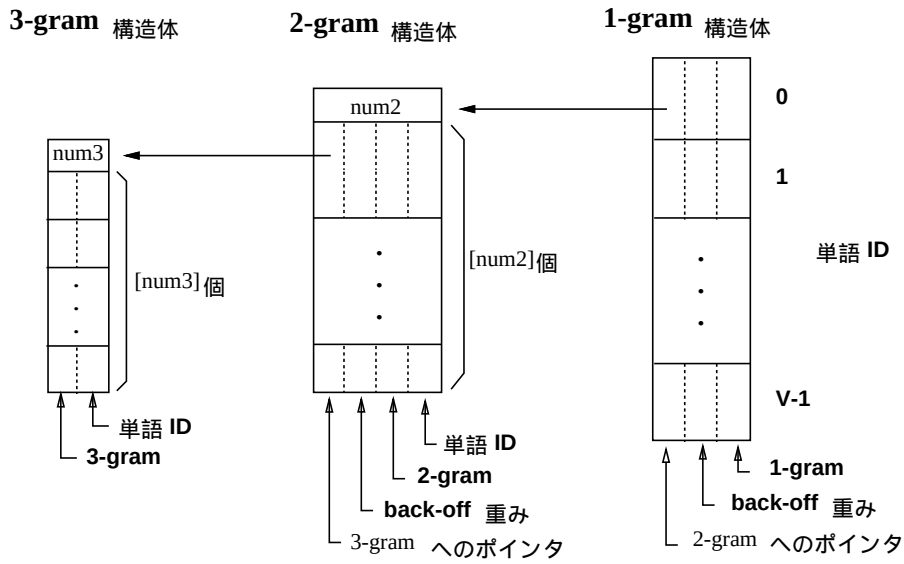

図 **3.10.** N-gram のデータ構造

基礎研究インタ

3.3.4 音響モデルの確率計算の高速化

認識処理において、音響モデルの確率計算が占める割合は非常に大きいので、この効率化が重要である。特に、音素環境依存モデルを使用する際には、ガウス分布の数が膨大(数千~数万)であるので、すべてを計算するのではなく、必要なもののみを計算するのが望ましい。

ガウス分布の出力確率計算の高速化の方法は、Gaussian Pruning と Gaussian Selection (shortlists) の 2 つに大別される [23]。

Gaussian Pruning は、多次元ベクトルの距離 (=確率密度関数の指数部分) の計算を行う際に、途中の次元で枝刈りするものである。この際のしきい値として、既に計算された k-best の値を用いるのが安全であるが、効率はよくない。途中の次元でもビーム幅を設定したり、未計算の次元をヒューリスティックに見積ったりすることにより、適格性は失われるが、さらに効率化される [24]。

Gaussian Selection (shortlists) は、あらかじめ計算の対象とする分布を絞り込むものである。そのために分布をクラスタリングしたコードブックを用意しておく。入力ベクトルに対して、マッチングしたコードに対応づけられた分布に絞り込む [25][26]。

今回は、tied-mixture モデル用に Gaussian Pruning のみ実装した。なお Julius では、探索途上でビームに残っている仮説から必要な音素のモデルのみを計算するようにしている。また、最初に計算した確率はキャッシュしておき、他の仮説や第 2 パスでは新たに計算することのないようにしている。

第4章　Julius 2.1(98年度版)からの変更点

1. 第1パスでの単語間triphoneの近似処理の導入
2. 第2パスでの単語間triphoneの早期厳密処理の導入
3. tied-mixtureモデルへの対応
4. tied-mixtureモデルのためのGaussian Pruningの導入
5. 語彙サイズの65Kまでの拡張
6. その他、バグフィックス多数

参考文献

[1] S.J.Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing magazine*, Vol. 13, No. 5, pp. 45–57, 1996.

[2] 松岡達雄, 大附克年, 森岳至, 古井貞熙, 白井克彦. 新聞記事データベースを用いた大語い連続音声認識. 電子情報通信学会論文誌, Vol. J79-DII, No. 12, pp. 2125–2131, 1996.

[3] 西村雅史, 伊東伸泰. 単語を認識単位とした日本語ディクテーションシステム. 電子情報通信学会論文誌, Vol. J81-DII, No. 1, pp. 10–17, 1998.

[4] 伊藤克亘, 伊藤彰則, 宇津呂武仁, 河原達也, 小林哲則, 清水徹, 田本真詞, 荒井和博, 峯松信明, 山本幹雄, 竹沢寿幸, 武田一哉, 松岡達雄, 鹿野清宏. 大語彙日本語連続音声認識研究基盤の整備–学習・評価用テキストコーパスの作成–. 情報処理学会研究報告, 97-SLP-18-2, 1997.

[5] 河原達也, 李晃伸, 小林哲則, 武田一哉, 峯松信明, 伊藤克亘, 伊藤彰則, 山本幹雄, 山田篤, 宇津呂武仁, 鹿野清宏. 日本語ディクテーション基本ソフトウェア (97 年度版). 日本音響学会誌, Vol. 55, No. 3, pp. 175–180, 1999.

[6] 河原達也, 李晃伸, 小林哲則, 武田一哉, 峯松信明, 伊藤克亘, 山本幹雄, 山田篤, 宇津呂武仁, 鹿野清宏. 日本語ディクテーション基本ソフトウェア (98 年度版) の性能評価. 情報処理学会研究報告, 99-SLP-26-6, 1999.

[7] 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.

[8] S.Young, J.Jansen, and J.Odell D.Ollason P.Woodland. *The HTK BOOK*, 1995.

[9] *The CMU-Cambridge Statistical Language Modeling Toolkit v2*, 1997.

[10] J.J.Odel, V.Valtchev, P.C.Woodland, and S.J.Young. A one pass decoder design for large vocabulary recognition. In *Proc. ARPA Human Language Technology Workshop*, pp. 405–410, 1994.

[11] R.Schwartz and S.Austin. A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses. In *Proc. IEEE-ICASSP*, pp. 701–704, 1991.

[12] G.Antoniol, F.Brugnara, M.Cettolo, and M.Federico. Language model representations for beam-search decoding. In *Proc. IEEE-ICASSP*, pp. 588–591, 1995.

[13] 李晃伸, 河原達也. 大語彙連続音声認識エンジン Julius における A*探索法の改善. 情報処理学会研究報告, 99-SLP-27-5, 1999.

[14] H.Murveit, J.Butzberger, V.Digalakis, and M.Weintraub. Large-vocabulary dictation using SRI's DECIPHER speech recognition system : Progressive search techniques. In *Proc. IEEE-ICASSP*, Vol. 2, pp. 319–322, 1993.

[15] L.Nguyen, R.Schwartz, Y.Zhao, and G.Zavaliagkos. Is N-best dead ? In *Proc. ARPA Human Language Technology Workshop*, pp. 411–414, 1994.

[16] F.K.Soong and E.F.Huang. A tree-trellis based fast search for finding the N best sentence hypotheses in continuous speech recognition. In *Proc. IEEE-ICASSP*, pp. 705–708, 1991.

[17] 吉田航太郎, 松岡達雄, 大附克年, 古井貞熙. 単語 trigram を用いた大語彙連続音声認識. 情報処理学会研究報告, 96-SLP-14-14, 1996.

[18] H.Ney and X.Aubert. A word graph algorithm for large vocabulary continuous speech recognition. In *Proc. ICSLP*, pp. 1355–1358, 1994.

[19] L.Nguyen, R.Schwartz, F.Kubala, and P.Placeway. Search algorithms for software-only real-time recognition with very large vocabularies. In *Proc. ARPA Human Language Technology Workshop*, pp. 91–95, 1993.

[20] 清水徹, 山本博史, 松永昭一, 匂坂芳典. 単語グラフを用いた自由発話音声認識. 情報処理学会研究報告, 95-SLP-9-8, 1995.

[21] 河原達也, 松本真治, 堂下修司. 単語対制約をヒューリスティックとする A*探索に基づく会話音声認識. 電子情報通信学会論文誌, Vol. J77-DII, No. 1, pp. 1–8, 1994.

[22] 小川厚徳, 武田一哉, 板倉文忠. 文長を考慮した言語モデルの検討. 情報処理学会研究報告, 97-SLP-16-5, 1997.

[23] A.Sankar. A new look at HMM parameter tying for large vocabulary speech recognition. In *Proc. ICSLP*, pp. 2219–2222, 1998.

[24] 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. 情報処理学会研究報告, 99-SLP-29-8, 1999.

[25] Enrico Bocchieri. Vector quantization for the efficient computation of continuous density likelihoods. In *Proc. IEEE-ICASSP*, Vol. 2, pp. 692–695, 1993.

[26] K.M.Knill, M.J.F.Gales, and S.J.Young. Use of Gaussian selection in large vocabulary continuous speech recognition. In *Proc. ICSLP*, pp. 470–473, 1996.

第5章　日本語大語彙連続音声認識エンジン Julius

Japanese
Large Vocabulary Continuous Speech
Recognition Engine

Julius

(Rev 1.0 1998/02/20)
(Rev 1.1 1998/04/14)
(Rev 1.2 1998/10/24)
(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)

Copyright (c) 1998-2000 Information-technology Promotion Agency, Japan (IPA)
Copyright (c) 1991-2000 Kyoto University
All rights reserved

5.1 旧バージョンをご利用の方へ

2.1 からの変更点を 00Changes-2.1-3.1 にまとめてありますのでご覧下さい．
各バージョンごとの変更点は以下のファイルををご覧下さい．

3.0 -> 3.1	doc/00Changes-3.0-3.1
2.2 -> 3.0	doc/00Changes-2.2-3.0
2.1 -> 2.2	doc/00Changes-2.1-2.2

5.2 Julius について

- Julius は，単語 N-gram を用いて大語彙連続音声認識を行なう認識エンジンです．
- 入力 (ポーズを切れ目とする連続発声) に対し，あらゆる単語の組合わせからモデル上で最も尤もらしい単語系列を探し出し，出力します．
- 2 パス構成の段階的探索で，第 1 パスでの 1-best 近似計算と単語トレリスインデックスを用いたパス間インタフェースにより，高効率かつ高精度な認識を行います．
- 語彙数は最大 65,535 語まで対応しています．
- 認識性能は実験室環境で 単語認識精度 85 ~ 95%程度です．30MB ~ 128MB 程度のメモリ量で動作し，実行時間は実時間 ~ 15 倍程度です．(使用する音素モデルや言語モデルによります)
- 言語モデルは単語 2-gram と (逆向き) 単語 3-gram，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．これらは読み込み時に自動判別されます．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．
- モデルのファイル形式は，他のツールで用いられている標準的形式がそのまま使えます (若干の制限あり)．音素モデル (HMM) は HTK 形式ファイル，言語モデル (単語 N-gram) は ARPA 標準形式ファイルで与えます．また言語モデルは Julius 独自のバイナリフォーマットでも与えることができます．
- 入力は，マイクロフォン端子や DatLink(NetAudio) からの直接入力が可能です．またサンプリングした音声波形ファイルや，HTK 形式の特徴パラメータファイルでも与えることができます．
- Julius は Solaris 2.5.1 with gcc の環境で開発されました．以下の環境での動作を確認しています．

  Sun Solaris 2.5.1 / gcc, egcs
  Sun Solaris 2.6 / egcs
  Sun SunOS 4.1.3 / gcc
  PC Linux 2.0.36 libc5 / gcc, egcs, pgcc
  PC Linux 2.2.12 glibc2 / gcc, egcs

SGI IRIX 6.3 / cc, gcc
DEC Digital UNIX 3.2 / gcc
DEC Digital UNIX 4.0 / gcc

5.3 パッケージの内容

本パッケージには以下のファイルが含まれています．

00readme	最初に読む文書 (この文書)
LICENCE	使用許諾書
INSTALL	インストールの手引き
README.configure	configure オプションの詳細
README.mic	マイクロフォンを使った認識の手引き
Sample-j.jconf	jconf 設定ファイルのサンプル (日本語コメント)
Sample.jconf	jconf file example (English comments)
00Changes-2.1-3.1	Rev.2.1 → 3.1 の変更履歴
doc/00Changes-*	各リリースの変更履歴
doc/Tutorial.txt	初めて使う人のためのチュートリアル
doc/Format.txt	Julius が使用するファイルの形式に関する詳細
doc/Triphone.txt	音響モデルの音素環境依存性の扱いについて
configure	コンパイル環境自動設定スクリプト
julius.jman	日本語オンラインマニュアル (troff 形式)
julius.jcatman	日本語オンラインマニュアル (catman 形式)
julius/	Julius-3.1 本体のソース
mkbingram/	mkbingram — バイナリ N-gram 作成ツール
adinrec/	adinrec — マイク録音テストツール
libsent/	汎用ライブラリ ヘッダ・ソース
Makefile.in	以下，コンパイル用のソース類
support/	

5.4 インストール

基本的には

```
% ./configure
% make
% su                （必要であれば）
% make install
```

で終了です．以下のファイルがインストールされます

- 実行バイナリ

```
/usr/local/bin/julius
/usr/local/bin/mkbingram
/usr/local/bin/adinrec
```

- オンラインマニュアル

```
/usr/local/man/man1/julius.1
/usr/local/man/cat1/julius.1
/usr/local/man/cat1/mkbingram.1
/usr/local/man/man1/adinrec.1
/usr/local/man/cat1/adinrec.1
```

以上の方法では高速版がインストールされます．Julius ではこの他に精度重視・速度重視などの細かいコンパイル時設定が可能です．詳しくは付属の INSTALL および README.configure をご覧下さい．

5.5 実行方法

まず動作に必要なモデル定義ファイルや入力ソースの指定，認識パラメータ等の各種設定を行います．設定ファイル（以下 jconf ファイル）のサンプルが Sample-j.jconf にあるので，これをコピーして必要に応じて各項目を編集します．

Julius は，コマンドラインから以下の要領で起動します．

```
% julius -C jconf ファイル名
```

起動後は入力待ちとなり，音声データが与えられるごとに認識を行い，結果を標準出力に出力します．最後のほうに出てくる“sentence1: ...”の一行が最終的な認識結果となります．

初めての方はチュートリアルが `doc/Tutorial.txt` を参考にしてください．

なお現状では一文の認識にはある程度時間がかかり，メモリ量も数十 MB を必要とします．できるかぎり高スペックの計算機上で運用することをお勧めします．

5.6 問い合わせ先

問い合わせ・バグ報告・コメントなどは

julius@kuis.kyoto-u.ac.jp

までお願いします．

5.7 作成者

Rev.1.0 (1998)

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 (1999/10/03)
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．

謝辞

このプログラムは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．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)，及び峯松信明氏 (豊橋技術科学大学) をはじめとする関係各位に感謝します．

また伊藤克亘氏 (電子技術総合研究所) 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．

また，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．

第6章　Julius インストールの手引き

Julius インストールの手引き

(1999/02/20 新訂)
(1999/04/20 改訂)
(1999/10/04 改訂)
(2000/02/14 改訂)
(2000/05/11 改訂)

6.1 Quick Start

```
% ./configure
% make
```

builds “julius”, “mkbingram”, “adinrec” unser their directories. Optionally,

```
% make install
```

installs the executables under /usr/local/bin.

When you are using Linux and have trouble with mic input, first doubt the mixer setting, and then try these configuration each by one or both at once. Input threading support:

Force OSS API even on ALSA:

```
% ./configure --with-mictype=oss
```

Input threading support:

```
% ./configure --enable-pthread
```

6.2 準備

以下のものは必須ではありませんが，あれば Julius の機能が拡張されます．

- GNU readline/history ライブラリ
  ファイル名入力での補完と履歴参照が使用可能になります．

- libsndfile (音声ファイル入出力ライブラリ)
  様々な形式の音声ファイルが読み込み可能になります．
  `http://www.zip.com.au/~erikd/libsndfile/`
  使用不可の場合 raw 形式と wav 形式 (無圧縮) のみ読み込めます．

使用する場合は Julius のインストールに先だってこれらのインストールを済ませておいてください．

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-3.1.tar.gz | tar xvf -
% cd julius-3.1
```

6.3 configure の実行

configure スクリプトは，OS の種別やマイク入力の有無，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
```

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．明示的に指定したい場合は，configure を実行する前に，環境変数CC およびCFLAGS を設定してください．なおコンパイラは ANSI C 準拠のコンパイラをご使用下さい．以下は IRIX6.3 でcc を使用する場合の例です．

```
csh,tcsh 系:
% setenv CC cc
% setenv CFLAGS '-n32 -O'
% ./configure
sh,zsh 系:
```

```
% CC=cc; export CC
% CFLAGS='-n32 -O'; export CFLAGS
% ./configure
```

標準でないパスに必要なヘッダがある場合は，環境変数 CPPFLAGS を指定してください．

```
% setenv CPPFLAGS '-I/home/somewhere/in/my/dir/include'
```

configure 実行時のオプションで，Julius のアルゴリズムの設定を変えることができます．基本的には以下の 3 種類のプリセット設定から選択してください．

1) --enable-setup=standard	標準版 (低速，最高精度)
2) --enable-setup=fast	高速版 (高速，高精度) （デフォルト）
3) --enalbe-setup=v2.1	rev.2.1 互換

なお何も指定しなければ fast(高速版) になります．./configure のオプションの詳細については，付属の README.configure をご覧ください．

6.4 コンパイル

```
% make
```

を実行すると実行バイナリ julius, mkbingram, adinrec の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．

6.5 インストール (optional)

```
% make install
```

を実行すると，実行バイナリとオンラインマニュアルがインストールされます．

・実行バイナリ

```
/usr/local/bin/julius
/usr/local/bin/mkbingram
/usr/local/bin/adinrec
```

・オンラインマニュアル

```
/usr/local/man/man1/julius.1
```

```
/usr/local/man/cat1/julius.1
/usr/local/man/cat1/mkbingram.1
/usr/local/man/man1/adinrec.1
/usr/local/man/cat1/adinrec.1
```

なお julius は実行バイナリのみで動作しますので，上記のインストールは特に必要ではありません．

以上でインストール作業は完了です．以下の文書もあわせてご覧下さい．

- doc/Tutorial ... チュートリアル
- README.mic ... マイク入力の手引き
- README.configure ... configure オプションの詳細

以上

第7章　Julius チュートリアル

Julius チュートリアル

(1998/02/20)
(1998/04/15 改)
(1999/02/20 改)
(1999/04/20 改)
(2000/02/14 改)

ここでは，Julius の基本的な使い方について説明します．準備から起動，音声データの認識を流れを追って説明したあと，選択可能な音声入力ソースやバイナリ N-gram ファイルの利用，エラーについて解説します．

7.1 ファイルの準備

7.1.1 モデル定義ファイル

Julius が動作するために必要なモデル定義ファイルを用意します．最低限必要なのは，

- HMM 定義ファイル ... HTK の HMM 定義ファイル形式
- 単語 2-gram ファイル ... ARPA 標準形式
- 単語辞書ファイル ... HTK の辞書形式とほぼ同じ

の 3 つです．第 2 パス探索により高精度な認識を行うためには，さらに

- 逆向き単語 3-gram ファイル ... ARPA 標準形式

が必要です．また triphone HMM を用いる場合は

- HMMlist ファイル ... 独自形式

が必要です．triphone 使用時は第 2 パスで単語間コンテキストが考慮され，より精度の高い認識結果が得られます．

※ 単語 2-gram ファイルと逆向き単語 3-gram ファイルの代わりに，それらをあらかじめコンパイルした『バイナリ N-gram ファイル』を使うことで，読み込みを高速にできます．これについては第 6 節を参照してください．

各ファイルの形式に関する詳細，Julius での制限事項については付属の `Format.txt` を参照してください．

7.1.2 設定ファイル (jconf ファイル) の編集

Julius の動作設定は基本的に「jconf ファイル」と呼ばれるファイルに記述します．ソースアーカイブ内にサンプル `"Sample-j.jconf"` があるので，これを適当な場所にコピーしてから自分の環境に合わせて編集します．

各設定については，`Sample-j.jconf` 内のコメントおよびオンラインマニュアルを参考にしてください．

7.2 Julius の起動

Julius の起動はコマンドラインから行います．起動時に `"-C jconf ファイル"` 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，この時点でコマンドラインオプションとして与えることも可能です．`-C jconf` ファイルの後ろ側で指定することで，jconf ファイル内の設定を上書きできます．又複数の jconf ファイルを指定したときは後の設定が優先されます．

たとえば，jconf ファイル内ではマイク入力 `-input mic` が指定してあるが音声ファイル入力を行いたい場合，以下のように指定できます．

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

ここでエラーが発生した場合は 7.7 を参照してください．

7.3 音声データ入力

起動が終了すると，音声データの入力待ちになります．

音声入力ソースは起動時に-input {rawfile|mfcfile|mic|netaudio} の形で指定します．例えば -input rawfile としたときは，次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->
```

与えることのできる音声データは16kHz,16bitのみです．詳しくは付属の文書 “Format.txt” をご覧下さい．

7.4 認識の実行と結果の出力

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(-1pass で1パスのみの実行も可)．

まず入力全体に対し第1パスの2-gramを用いたフレーム同期のビーム探索でを行います．途中結果としてベスト仮説を出力します．以下は出力例です．

```
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
```

第1パス終了後，さらに高精度な第2パスが実行されます．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
```

認識終了後は 第 7.3 節のプロンプトへ戻ります . 続けてファイル名を入力することで何度も音声認識を行うことができます .

なお起動時に -quiet オプションを付けた場合 , 以下のように結果だけの出力になります .

```
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 。
```

以上で基本的な使い方は終りです .

7.5 音声入力ソースの選択

入力音声のソースは , 以下の 2 ~ 4 種類から選択できます .

-input rawfile 音声波形データファイル

一文連続発声を記録した音声波形ファイルを読み込んで認識します . ファイル形式は RAW(16kHz,16bit signed short,mono,big-endian) ,もしくは WAV(無圧縮,16kHz,16bit) です . コンパイル時に libsndfile を組み込んでいれば , さらに多様な形式の音声ファイルを入力とすることができます .

なお一回の入力で認識できる時間長にはシステム上の制限があります .詳しくは `Format.txt` をご覧下さい .

-input mfcfile HTK 形式のパラメータファイル (MFCC)

音声波形データから特徴抽出済みの特徴パラメータファイルを読み込んで認識します . ファイル形式は HTK のパラメータファイル形式です .

音響 HMM の学習パラメータと同じ特徴パラメータ型であることが必要です . 特徴パラメータファイルはこの学習パラメータと同じ型であるか , もしくは含んでいる必要があります .

特徴パラメータファイルの作成は別のツールを使います．wav2mfcc は MFCC タイプの特徴パラメータファイルを HTK フォーマットで生成できます．HTK がある場合は，もちろん`HCopy` コマンド で作ることができます．

-input mic マイクからの直接入力（β 版）

マイク端子からの音声 Live 入力を認識します．
詳しくはパッケージに付属の `README.mic` をご覧下さい．サポート OS は Linux (OSS or ALSA), Solaris, IRIX です．

入力開始と同時に解析が始まります．このとき，起動時に `-demo` を指定しておけば，解析途中の候補が漸次的に出力されます．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．

なお第 1 発話目は正しい結果が出てきません (CMN パラメータ取得のため) のでご注意下さい．

-input netaudio -NA server:0 DatLink からの直接入力

DatLink サーバから入力音声をパイプラインで受け取り，認識します．`-input mic` と同じく，入力と同時に解析が始まります．

使用するには，DatLink に付属の NetAudio ライブラリが必要です．configure がライブラリのチェックに失敗する場合は，NetAudio の lib, include があるディレクトリを`--with-netaudio-dir=directory` の形で指定してください．

`-NA` で，音声を読み込んでくる NetAudio サーバのホスト名とユニット番号を指定します．多くの場合 “DatLink の接続されたホスト名:0” となります．

`-input mfcfile` 以外は Julius が内部で音響特徴抽出を行いますが，使用できるのは 25 次元の MFCC_E_D_N_Z のみです．これ以外の特徴パラメータを必要とする HMM 音響モデルを使用する場合は，抽出済みの特徴パラメータファイル (.mfc) を `-input mfcfile` で与えるようにしてください．

7.6 バイナリ N-gram ファイルの利用

ARPA 形式の単語 2-gram ファイルおよび逆向き 3-gram ファイルは，あらかじめコンパイルして 1 つのバイナリ N-gram ファイルに変換しておくことができます．

バイナリ N-gram を使うことで，Julius 起動時の N-gram の読み込み時間が大幅に短縮できます (数十秒 → 数秒) ．

変換には `mkbingram` を使います．以下は `2-gram.arpa` と`rev-3-gram.arpa` からバイナリ N-gram ファイル `new.bingram` に変換する場合のコマンドです．

```
% mkbingram 2-gram.arpa rev-3-gram.arpa new.bingram
```

jconf ファイルでは，`"-nlr 2-gram.arpa -nrl rev-3-gram.arpa"` の代わりに `"-d new.bingram"` と指定します．

(付記) このバイナリ N-gram は CMU-TK のバイナリ形式とは非互換です．

7.7 起動時のエラー

Julius が扱うことの出来る HMM や単語 N-gram のタイプやサイズには，若干の制限があります (`Format.txt` を参照してください) ．

Julius 起動時に，モデルファイル名を正しく指定したにも関わらずエラーが生じる場合は，そのモデルがこの制限を満たしていない可能性があります．

特に，以下の制限には注意してください．

1) HMM の遷移に関する制限
2) N-gram の語彙数制限 (65,535 語)
3) 単語語彙のサイズ制限 (65,535 語)

以上

第8章　マイク入力について

マイク入力について

(1999/04/20 新訂)
(1999/10/04 改訂)
(2000/02/14 改訂)
(2000/05/06 改訂)

Julius-3.1 でのマイクロフォンからの音声入力について，環境の設定や制限事項等について説明します．

目次

1. 音響モデルの制限
2. サポート OS
3. コンパイル時の設定
4. マイク入力の認識の流れ
5. 音量・トリガレベルの調節
6. マイクデバイスの動作チェック

8.1 音響モデルの制限

Julius 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．

8.2 サポート 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．

a) Linux (kernel driver, OSS/Linux, ALSA)
b) Sun Solaris 2.x
c) Sun SunOS 4.x
d) SGI IRIX

各 OS に関する詳細と注意事項です：

a) Linux

Linux では，以下のサウンドドライバに対応しています．

1. OSS compatibles
   (a) カーネル 2.0.x および 2.2.x に含まれるドライバ (2.3.x は未テスト)
   (b) OSS/Linux: 4Front Technologies 社の商用ドライバ
2. 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`alsa` もしくは `oss` を指定)．

16bit,16kHz,monoral で録音できることが必須です．

うまく入るかどうかは，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．また特に NotePC に関しては，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が多数ありますのでご留意下さい．

うまくいかなければ OSS, ALSA などいろいろなドライバを試してみるのがよいでしょう．

Julius はミキサー設定を行いません．入力デバイスの選択 (MIC/LINE) やボリューム調節は `xmixer` などで別途行ってください．

関連 `URL`:

Linux Sound-HOWTO
`http://www.linux.or.jp/JF/JFdocs/Sound-HOWTO.txt`

ALSA
http://www.alsa-project.org/
OSS/Linux
http://www.opensound.com/

b) 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．

c) 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 <multimedia/*> 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．また音量は 20 に自動設定されます．

d) 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．

8.3 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．うまく検出出来たかどうかを調べるには，configure の最後に出力される以下のようなメッセージをチェックしてください．

```
mic API type    : alsa (Advanced Linux Sound Architecture)
```

何らかの理由で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, sol2, sun4, irix のどれかを指定します．

8.4 マイク入力の認識の流れ

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には，起動時に -input mic を指定します．すると起動後，プロンプトが出て音声のトリガ待ちになります．なお同時に-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．

プロンプトがでたらマイクに向かって発声します．口の位置はマイクから 15cm 程度で，ゆっくりはっきり発声して下さい．Julius は一定以上のレベルの入力があるとマイク入力開始とみなし，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，最初の第 1 発話は必ずおかしな認識結果が出ます．

実時間処理では正確な CMN が与えるのが難しいため，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ます．このため，起動して最初の発話では正しい特徴抽出ができません．

最初の入力は「マイクテスト」などとして，2 回目以降から本入力を開始するようにしてください．

8.5 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）．周囲の環境に合わせて増減して調節してください．

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．

8.6 adinrec によるマイクデバイスの動作チェック

2.2 以降の Julius には adinrec というマイク録音テストプログラムが一緒に収められています．これを使って Julius がちゃんとマイク入力を取り込めているのかどうかを確かめることができます．

```
% ./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ということになります．

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
```

また波形を目で見てみることで，ノイズの乗り方やクリッピングの様子などより詳しく分かります．snd という音波形編集ツールが知られています．

```
http://www-ccrma.stanford.edu/CCRMA/Software/snd/snd.html
```

8.7 どうしてもうまくいかないときは (Linux)

Linux ではドライバもサウンドチップも多様で，動く/動かないがあります．adinrec での録音音質が悪い，あるいは動作しないという場合は，以下の順で試してみてください．

1. (ALSA の方は) OSS 互換モードで使用する
   Julius の configure 時に --with-mictype=oss をつける．

   ```
   % ./configure --with-mictype=oss
   ```

2. 入力部をスレッド化する
   Julius の configure 時に --enable-pthread をつける．

   ```
   % ./configure --enable-pthread
   ```

3. ドライバを換えてみる．(ALSA ←→ OSS ←→カーネル付属)

以上

第9章　configure オプション解説

---

configure オプション解説

(2000/02/14 新訂)
(2000/05/11 改訂)

Rev.3.1 以降の Julius は，デフォルトで速度と精度のバランスを取った設定になっています．モデルの評価時など，特に精度を重視した認識を行いたい場合は，精度を優先した認識アルゴリズムを用いるよう configure を変更する必要があります[1]

そのほか，マイク入力に関する指定も configure オプションで行います．

ここでは，これら configure に与えられる設定オプションについて解説します．

9.1 標準オプション

オプション `--enable-setup=...` で認識アルゴリズムを以下のプリセットの３種類から選択できます．

1) `standard`: 標準版（低速，最高精度）

2) `fast`：高速版（高速，高精度）（デフォルト）

3) `v2.1`：rev.2.1 互換（最高速，低精度）

なお，現在の設定は `-version` をつけて起動することで確認できます．以下は例です (rev. 横の括弧内)．

```
% julius -version
Julius rev.3.1 (fast)
built on louvre at Fri May  5 23:54:14 JST 2000
```

[1] もちろん，実行時の探索パラメータもビーム幅を大きくするなどする必要があります．

```
 - compiler: gcc -g -O2
 - options: UNIGRAM_FACTORING LOWMEM2 PASS1_IWCD CCD_ON_2PASS
SCAN_BEAM_1 WORST_BEAM LOOKAROUND LOGTABLE 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 法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

9.2 オプション詳細

その他の configure オプションについての詳細です．なお configure スクリプトの本体は，実際には

- 汎用ライブラリ libsent： `./libsent/configure`
- エンジン本体 julius： `./julius/configure`

に分かれていますが，オプションの指定はトップの `./configure` 実行時に行ってください．

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．

`--disable-readline`

GNU readline/history ライブラリの使用を禁止します．コンパイル時に`-lreadline -lhistory` の部分でエラーが出る場合は試してみてください．

`--with-netaudio-dir=dir`

DatLink に付属の NetAudio ライブラリの検出に失敗するときは，これで NetAudio の include と lib があるディレクトリを直接指定します．

マイク関連

`--enable-pthread`

音声入力の取り込みに POSIX thread を使用します．ドライバによっては入力が改善されることがありますが，動かなくなることもあります．

`--with-mictype=TYPE`

ドライバタイプを明示的に指定します．alsa, oss, sol2, sun4, irix のどれかを指定します．

アルゴリズム関連

`--enable-factor1`

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`

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-monotree`

triphone であっても第 1 パスで monophone lexicon を使います．通常は指定しないでください．

`--enable-words-int`

単語 ID の型を unsigned short ではなく int にする (実験的) . 語彙数の上限は 2^31 語になるが，現在のところ動作は保証しない．

以上

第10章　サンプル設定ファイル Sample-j.jconf

```
# Copyright (c) 1991-2000 Kyoto University
# All rights reserved
#
# Julius 設定ファイル for rev.3.1
#
# 1) 重要: 相対パスは，*このファイルを起点に*指定してください．
# 2) 空行および # から行末まではコメントとみなします．
#    '#'をコマンドに含めたい場合は '\#' としてください．
# 3) 1行の長さは 512 バイト以内に収めてください．
# 4) 各オプションの書式は，コマンドラインで与える場合と同一です．
# 5) このファイル内の各値は Julius のデフォルト値です．
#


#
# 高速版システム for IPA99 CDROM
#   語彙数 20k
#   言語モデル: 75ヶ月, 圧縮3-gram
#   音響モデル: PTM, 3000状態, 129コードブック, 64混合, 性別非依存
#   デコーダ:  Julius 高速版
#   入力デフォルト：音声ファイル
#

#############################################################################
#### ファイル指定
#############################################################################
```

```
##
## 単語 2-gram,3-gram ファイル
##
# ARPA 標準形式
#-nlr /cdrom/lang_m/4.wit.arpa          # 2-gram
#-nrl /cdrom/lang_m/rev-4-8.wit.arpa    # 逆向き 3-gram
# もしくは
# mkbingram で作成したバイナリ形式のとき
-d ../lang_m/20k/bingram_for_julius/75.20k.1-1.10p.wit.bingram.gz

##
## 音響 HMM 定義ファイル
##
-h ../phone_m/model/PTM/gid/tri/hmmdefs,tmix.gz # HTK 形式

## triphone モデルの場合，さらに
##     論理的に出現しうる triphone -> 定義されている triphone
## の対応を指定した「HMMList ファイル」が必要です．
-hlist ../phone_m/parms/logicalTri.added

##
## 単語辞書ファイル
##
-v ../lang_m/20k/20k.htkdic                    # HTK 形式


#######################################################################
#### 言語モデル詳細設定
#######################################################################
##
## 言語重みと挿入ペナルティ
##
## 例えば，"-lmp 8.0 7.0" のとき，言語確率の対数尤度 'log p(w)' は
```

```
## (log p(w)) * 8.0 + 7.0' として適用されます．
##
##デフォルト値 ( 実験結果の最適値 )
##monophone 向け
#-lmp 5.0 -1.0          # 第 1 パス (2-gram)
#-lmp2 6.0 0.0          # 第 2 パス (3-gram)
##triphone 向け
#-lmp 8.0 -2.0
#-lmp2 8.0 -2.0
##triphone 向け ( v2.1 設定 ): 単語間 triphone を第 1 パスで扱わない場合
#-lmp 9.0 8.0
#-lmp2 11.0 -2.0

##
## 文頭単語の名前（単語辞書では無音の読みを付与する）
##
#-silhead '<s>'

##
## 文末単語の名前（同上）
##
#-silhead '</s>'


###########################################################################
#### 音響モデル詳細設定
###########################################################################
##
## Julius が triphone/monophone の自動判別に失敗する場合，
## 以下を試してみてください．
##
#-no_ccd                # 音素環境依存性を（強制的に）考慮しない
#-force_ccd             #        〃        （強制的に）考慮する
```

```
##
## 特徴パラメータの型チェックをスキップしたい場合は，
## 以下を試してみてください．
##
#-notypecheck

#############################################################################
#### 探索パラメータ
#############################################################################
-b 600                  # 第1パスのビーム幅( ノード数 )
#-b2 30                 # 第2パスの仮説数ビームの幅( 仮説数 )
#-s 500                 # 第2パスの最大スタック数（仮説数）
#-m 2000                # 第2パスの仮説オーバフローのしきい値
#-lookuprange 5         # 第2パスで単語展開時のトレリス制約緩和幅( フレーム数 )

#-n 1                   # 第2パスで見つける文の数( 文数 )
#-n 10                  #   ( 'standard' 設定時のデフォルト)
#-output 1              # 第2パスで見つかった文のうち出力する数 ( 文数 )

### "./configure --enable-lowmem" 時に有効
#-iwcache 10            # 単語間言語確率キャッシュの大きさ比率
                        # (100でオプションなしと同等)

### "./configure --enable-lowmem2" 時に有効
#-sepnum 150            # 第1パスで単語木から独立させる高頻度語の数( 単語数 )

#############################################################################
#### Gaussian Pruning パラメータ( tied-mixture, PTMでのみ有効 )
#############################################################################
## コードブックあたり計算するガウス分布計算数( 上位N個 )
## 以下のデフォルト値は IPA99 の PTM モデル( 1コードブックあたり64混合 )
## に合わせた値
```

```
#-tmix 2

## Gaussian pruning 法の選択
## 高速版では beam , それ以外では safe がデフォルトです
#-gprune safe           # safe pruning 上位 N 個が確実に求まる．正確．
#-gprune heuristic      # heuristic pruning
#-gprune beam           # beam pruning 次元ごとに足切り．高速．

######################################################################
#### 音声入力ソース
######################################################################
## どれかを選んでください ( デフォルト：mfcfile )
#-input mfcfile         # HTK 形式のパラメータファイル (MFCC)
#-input rawfile         # 音声波形データファイル (フォーマット自動判別)
                        # 形式：WAV(16kHz,16bit) または
                        #       RAW(16kHz,16bit(signed short),mono,big-endian)
                        #       libsndfile 付きでコンパイルした場合は
                        #       AIFF,AU も OK
#-input mic             # マイクから直接入力
#-input netaudio -NA host:0    # host 上の DatLink(NetAudio) から入力

#-lv 3000               # 切出レベルのしきい値 (0-32767)
#-nostrip               # 無効な入力部の除去を OFF にする
                        # (default: 無効な入力部分は除去される)

######################################################################
#### Forced alignment
######################################################################
#-walign                # 認識結果の単語ごとのアラインメント結果を出力
#-palign                # 認識結果の音素ごとのアラインメント結果を出力

######################################################################
#### 出力スタイル
```

```
#########################################################################
#-progout                # 第 1 パスで解析途中から漸次的に結果を出力
                         # オフライン認識には不向き
#-quiet                  # 第 1 パス・第 2 パスの認識結果のみ出力
#-demo                   # "-progout -quiet" と同じ
#-debug                  # 探索中の内部状態を出力させる ( デバッグ用 )
#-separatescore          # 言語スコアと音響スコアを分けて出力する

#########################################################################
#### その他
#########################################################################
#-help                   # ヘルプを出力して終了 ( jconf 内では無効 )
#-version                # バージョン情報を出力して終了 ( jconf 内では無効 )
#-C jconffile            # 他の jconf ファイルを挿入

################################################################## end of file
```

第11章　Juliusの扱うファイル形式の仕様

Julius の扱うファイル形式の仕様

(1998/02/20)
(1998/04/14 改)
(1998/10/27 改)
(1999/02/20 改)
(1999/10/04 改)
(2000/02/14 改)

Julius に与えるデータの形式は以下のようになっている。(なおファイルは全て，gzip 圧縮済であってもそのまま読み込める)

1. HMM 定義ファイル ... HTK の HMM 定義フォーマット
2. HMMlist ファイル ... 独自
3. 単語 N-gram 言語モデルファイル ... ARPA 標準形式 もしくはバイナリ N-gram 形式
4. 単語辞書ファイル ... HTK の辞書フォーマットに類似
5. マイク入力 ... 16bit 16kHz サンプリング
6. 音声波形ファイル ... .wav, .raw, その他
7. 特徴パラメータファイル ... HTK の特徴パラメータファイル形式

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。

11.1 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．

triphone モデルの場合は，辞書の monophone 表記から生成される triphone 体系と実際のモデル定義名とのマッチングを記述した 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．

○ 形式

(各項目の意味の詳細は The HTK Book for HTK V2.0 の 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)

  <TMix>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．

Julius は，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，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，マクロ数に上限は事実上無い．

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 以降の Julius は，tied-mixture モデルのために音響尤度の計算方法として以下の 2 種類を実装しており，使用モデルのタイプを判別してどちらかを自動的に選択する仕様になっている．

1. **state-driven**
   monophone や状態共有 triphone で用いられる．状態単位で音響尤度計算を行い，出力確率のキャッシュも状態単位で行う．
2. **mixture-driven**
   tied-mixture ベースのモデルで用いられる．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，まずコードブックごとの各ガウス分布の尤度を (Gaussian pruning を行いながら) 計算し，コードブック単位でキャッシュする．HMM 状態の出力尤度は，後に対応するコードブックのキャッシュを参照しながら重みをかけて算出する．

この判別には，hmmdefs 中に <Tmix> 定義が用いられているかどうかで判断している．初期の hmmdefs 読み込み中に <Tmix> 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，ガウス分布の定義名からコードブックを生成しつつ計算方法を mixture-driven にセットする．

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 <Tmix> ディレクティブを用いて定義する必要がある点に注意されたい．

(なお<Tmix>ディレクティブについて詳細は HTK Book を参照のこと)

11.2 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。

書き方や制限など，詳しくはファイル Triphone.txt に HMMList と triphone 名の扱いの解説があるのでそちらを参考にされたい．

11.3 単語 N-gram 言語モデルファイル

11.3.1 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 (<UNK>等) のエントリが必ず含まれていなくてはならない．
1-gram の最初の単語を未知語カテゴリとして扱う．(CMU SLM ToolKit で作成した場合，仕様上常にこの条件は満たされている)

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，その tuple を無視する旨警告メッセージを出力しつつ処理を続行する．

○ サイズの制限

語彙数の上限は 65,535 語である．

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。またこの場合，同時にコンパイルした `mkbingram` でバイナリ N-gram ファイルを生成し直す必要があるので注意すること。

11.3.2 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．

11.4 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 (単語名)

単語名から 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<UNK>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しなくすることができる．

- 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変換される．

  [例](ソート済みの必要はない)

  ```
  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。

11.5 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．

11.6 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 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，長い入力では第 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 .

11.7 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)+Δ MFCC(12)+Δ Pow(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)+Δ MFCC(12)+Δ Pow(1) (CMN)
  計 26 次元

  または

  FCC_E_D_A_Z = MFCC(12)+Pow(1)+Δ MFCC(12)+Δ Pow(1)
  + Δ Δ MFCC(12) + Δ Δ Pow(1) (CMN) 計 39 次元

を与えても認識は実行可能である．

11.8 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 clean; make
```

を実行する．

以上

第12章　音素環境依存性の扱いについて

12.1 Juliusでの音素環境依存性の扱い方

Julius における音素環境依存性の扱いは，与えるモデルが以下のどれかの条件にマッチしたときに ON になります．

- hmmdefs 内の HMM 定義名に ”-” と ”+” を含む名前が存在する場合
- HMMList(後述) 内の音素表記 に ”-” と ”+” を含む名前が存在する場合 (rev.2.3 以降)

音素環境依存性の扱いが ON になった場合，Julius は辞書ファイル内の単語の音素名から ”-” や ”+” をつけた名前を生成して，HMM 定義へのアクセスを行います．この音素環境依存性の扱いの ON / OFF は，起動時に出力される System Info メッセージ内の HMMInfo セクションで 確認できます．

音素環境依存性を扱うタイミングは，単語内と単語間で異なります．まず単語内の依存性は第 1 パスから扱います．Julius は初期化時に辞書ファイルを読み込む際に，辞書の音素表記を以下のように変換しながら，対応する HMM を割り当てていきます．このとき単語の先頭と終端は biphone の表記でアクセスします．また 1 音素からなる単語については monophone としてアクセスします．

```
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
```

単語間の依存性は第 2 パスで扱います．第 2 パスの探索時に，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ていきます．

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

Julius では基本的に，全ての出現し得る triphone について対応する HMM の定義を用意しておく必要があります．また，単語の始終端の音素や，1 音素からなる単語を考慮する

必要があるため，片方のコンテキストのみの e+i,k-u といった biphone や，e, k などの monophone も全て同様に参照可能にしておく必要があります（詳細はこの文書の「注意」を参照のこと）

hmmdefs で定義されていないモデルについては，次節で解説する **HMMList** ファイルでマッピングを指定します．

12.2 HMMList ファイル

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指定します．

1. 一行に 1 つのマッピングを記述します．
2. 第 1 コラムに，論理的な音素表記を指定します．
3. 第 2 コラムに，対応する hmmdefs 内の音響モデルの定義名を指定します．
4. 表記の名前の HMM が hmmdefs 内で直接定義されている triphone については，第 2 コラムを空白にしておけばその定義の存在チェックだけを行います．
5. 二重登録はエラーになります．

以下は例です．名前が一つしか書いていないエントリは，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
...
```

なお，rev.2.3 以降ではそのような名前 1 つのエントリを無視して以下のように記述してもかまいません．

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

12.3 チェック方法

実際にどのように論理的な音素表記から HMM 定義へマッピングされたかは，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 音素表記” と打つとその音素表記に関する HMM 情報が出てきます．

12.4 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．

- `e+i`, `k-u` といった biphone や，`e`, `k` などの monophone も全て参照可能にしておく必要があります．これは，たとえ第 1 パスから単語間 triphone を扱うように Julius を設定した場合でも必要です (この場合 monophone, biphone のマッピングは実際には使用されません．あくまで辞書読み込み部分の実装上の都合です)．

- 一方 triphone では ,もし定義が見つからなかった場合は ,暗黙のうちに元の **biphone**(**1** 音素の単語の場合 **monophone**) モデルを流用するようになっています . たとえば「で」という単語と「も」という単語を接続するとき , 以下のようにモデルを適用します .

  「で |d+e d-e|」 + 「も |m+o m-o|」
  ->「で も |d+e d-e+m e-m+o m-o|」(もし |d-e+m| と |e-m+o| があれば)
  ->「で も |d+e d-e e-m+o m-o|」(もし |d-e+m| がなければ)

  ただし , 無用な混乱を避けるために , 出現しうる全ての triphone について HMMList で明示的にマッピングを指定しておくことを推奨します .

なお Julius の将来のバージョンでは ,最初にマッピングをチェックして存在しない triphone があると起動できないようになる予定です .

第13章　Julius man ページ

JULIUS(1)　　User Commands　　JULIUS(1)

【NAME】

Julius - Japanese LVCSR engine（大語彙連続音声認識エンジン）

【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います．

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．

なお語彙数の上限は 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 の N-gram ファイルを読み込むことができます．

【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 (.wav もしくは raw 形式：16kHz,16bit) 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．
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(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】

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】

コマンドライン上で指定することもできますが，jconf 設定ファイル内にまとめて記述しておいて起動時に "-C" で指定する方法をお奨めします．
詳しくは付属の Sample-j.jconf および doc/Tutorial.txt をご覧下さい．

[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(1) で作成したバイナリ形式 N-gram ファイルを指 定する．"-nlr", "-nrl" の代わりにこちらを使えば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
5.0 -1.0 / 6.0 0.0 (monophone 使用時)
8.0 -2.0 / 8.0 -2.0 (triphone,PTM 使用時)
9.0 8.0 / 11.0 -2.0 (triphone,PTM,engine=v2.1)

[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．
辞書の音素表記から生成した論理 triphone 名から HMM 定義名への写像を表す．詳細は doc/Triphone.txt を参照．

-force_ccd / -no_ccd
triphone 使用時，第 2 パスの単語間音素環境依存を考慮する / し ないを強制指示する．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)

tied-mixture および PTM 用オプション

-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

Gaussian pruning の手法を指定する

(default: safe (標準版) beam (高速版))

[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 下のうちど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)

[探索パラメータ (第 1 パス)]

-b beam_width

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 リ量がかかる .

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 使用時 ただし engine 設定=v2.1)

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

-force_realtime on|off
on-the-fly デコーディング (入力と同時に解 析 開 始) の ON/OFF を強制指定 .
(default: マイク・DatLink では ON, ファイルでは OFF)

-1pass
第 1 パスのみ実行する . -nrl 指定無しなら自動的にこの モードになる .

[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あり .
(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 . 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がかかるよ うになる .

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)．

-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)

-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あり．
(default: 80.0)

-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)

[音声入力]
-input rawfile|mfcfile|mic(|netaudio|adinserv)

音声波形データの入力ソースを指定 (default: mfcfile)． 各形式については doc/Format.txt を参照のこと

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

-lv threslevel
マイク入力時の切出しレベルのしきい値を 0 から 32767 の範 囲で指定する．(default: 3000)

-nostrip
0 サンプルの連続などの無効な入力区間は自動的に判別・除去されるが，このオプションを指定すると OFF になる．無効区間の自動判別が失敗するときに使う．

[メッセージ出力]

-separatescore
言語スコアと音響スコアを別々に出力する．

-quiet
ベストの候補の文字列だけを出力する．

-progout
第 1 パスで解析開始と同時に途中結果を漸次出力する．

-demo
"-progout -quiet" に同じ．

-walign
認識結果に対して単語ごとのアラインメントを出力する．

-palign

認識結果に対して音素ごとのアラインメントを出力する．

[その他]

-debug

冗長なデバッグ用メッセージを出力する．

-C jconffile

jconf 設定ファイルの読み込み．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．

-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 "doc/Tutorial.txt" をご覧下さい．

【SEE ALSO】

mkbingram(1), adinrec(1), wav2mfcc(1)

パッケージ付属のドキュメント：

Rev 1.1 から 2.1 の変更点	(doc/00Changes-1.1-2.1)
Rev 2.1 から 2.2 の変更点	(doc/00Changes-2.1-2.2)
Rev 2.2 から 3.0 の変更点	(doc/00Changes-2.2-3.0)
Rev 3.0 から 3.1 の変更点	(doc/00Changes-3.0-3.1)
jconf ファイルの例	(Sample-j.jconf)
チュートリアル	(doc/Tutorial.txt)
ファイル形式の仕様	(doc/Format.txt)
Triphone の扱いに関する詳細	(doc/Triphone.txt)

HTK manual:

The HTK Book for HTK V2.0

【DIAGNOSTICS】

正常終了した場合，Julius は exit status として 0 を返します．指定した使用モデルファイルが見つからない場合や大きさが制限を越え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 Format.txt 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)

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．

【THANKS TO】

このプログラムは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．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），及び峯松信明氏（豊橋技術科学大学）をはじめとする関係各位に感謝します．

また伊藤克亘氏（電子技術総合研究所）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．

また，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．

第14章　1999年度 大語彙連続音声認識 評価実験結果

14.1 評価用サンプル IPA-98-TestSet

ASJ-JNAS 日本音響学会 新聞記事読み上げ音声コーパス

テキスト: NORMAL:76 + LONG:24, LPP:26 + MPP:45 + HPP:29

話者 23 名, 合計 100 文 (男女とも)

男性: www.milab.is.tsukuba.ac.jp/jnas/test-set/male/male1_LARGE.txt

女性: www.milab.is.tsukuba.ac.jp/jnas/test-set/female/female1_LARGE.txt

未知語率: 0.44% (=7/1575)

正解基準: 自動集計ツール (漢字レベル；複合語処理あり)

14.2 評価基準

表の数値: 認識率 (第 1 パス認識率) / 平均認識時間

認識率は Word Acc.

CPU: UltraSPARC 300MHz, サンプルの平均時間: 5.8 sec.

音響・言語モデルの評価の際はデコーダは精度優先の設定 (=標準設定)

14.3 音響モデルの評価 (男性)

言語モデル: 75-month cutoff-1-1

デコーディング： 標準

	mix.4	mix.8	mix.16
GD monophone	75.3(65.4)/16.0	79.6(71.0)/16.1	83.9(76.2)/16.9
GI monophone	68.3(58.4)/16.9	78.0(66.6)/16.5	81.7(70.6)/17.7
GD triphone 2000	92.0(81.9)/51.9	92.6(82.3)/59.8	94.3(85.0)/76.3
GI triphone 2000	89.3(77.2)/58.4	91.8(80.3)/62.3	92.5(81.1)/80.6
GD PTM 129x64 (3000)		92.4(82.3)/29.6	
GI PTM 129x64 (3000)		89.5(79.9)/30.7	

14.4 音響モデルの評価 (女性)

言語モデル: 75-month cutoff-1-1

デコーディング： 標準

	mix.4	mix.8	mix.16
GD monophone	75.5(68.8)/16.0	80.7(71.0)/15.8	88.9(80.7)/16.5
GI monophone	76.0(64.5)/16.7	80.8(71.8)/16.4	84.7(77.2)/17.3
GD triphone 2000	92.0(80.7)/53.4	94.4(85.0)/57.7	95.2(84.6)/72.7
GI triphone 2000	92.3(80.8)/57.4	93.4(84.3)/62.9	94.8(85.6)/77.0
GD PTM 129x64 (3000)		94.6(85.5)/30.8	
GI PTM 129x64 (3000)		94.3(85.0)/30.6	

14.5 言語モデルの評価

男性, **音響モデル**: GD triphone 2000x16

デコーディング：標準

		LM size
20K 75month cutoff-1-1	94.3(85.0)/76.3	(79MB)
20K 75month compress10p	94.3(85.0)/75.9	(38MB)
60K 75month cutoff-1-1	93.7(84.0)/99.8	(100MB)
60K 75month compress10p	93.5(84.0)/105.6	(55MB)

14.6 デコーディングアルゴリズムの評価

男性

言語モデル: 20K 75month cutoff-1-1

GD triphone 2000x16	
98 年度版 (Julius 2.0)	92.0(78.9)/49.8
標準 (-iwcd2; -b 1500 -n 10)	94.3(85.0)/76.3
(-iwcd1)	93.0(85.2)/60.7
高速 (-iwcd1; -shortlist)	TBD
GD PTM 129x64	
(-iwcd2; -tmix 2 -b 800)	
Gaussian pruning = no	92.8(83.1)/73.5
Gaussian pruning = safe (標準)	92.4(82.3)/29.6
Gaussian pruning = heuristic	90.7(82.6)/29.5
Gaussian pruning = beam	91.2(80.6)/25.9
Gaussian shortlists	TBD
高速 (-iwcd1; -Gbeam -b 600 -n 1)	89.7(81.0)/13.2
GD monophone 129x16	
標準 (-b 700, -n 10)	83.8(73.8)/12.7
高速 (-b 400, -n 1)	82.5(75.7)/ 6.5

14.7 20K システムの構成

	簡易版	高速版	標準版	高精度版
音響モデル	monophone 16 (0.5MB)	PTM 129x64 (3.0MB)	triphone 2000x16 (8.6MB)	
言語モデル	75month compress10p (38.0MB)		75month cutoff-1-1 (78.5MB)	
デコーダ	高速	高速	(98 年度版)	高精度
認識時間	1.1x RT	2.3x RT	8.4x RT	12.8x RT
認識率 (男性 D)	82.6, 83.5c	89.1, 91.1c	92.0, 93.2c	94.3, 95.4c
認識率 (女性 D)	85.7, 87.1c	91.8, 93.1c	93.2, 94.1c	95.2, 96.2c
認識率 (GD 平均)	84.2, 85.3c	90.5, 92.1c	92.6, 93.7c	94.8, 95.8c
認識率 (GI 平均)	81.5, 84.0c	89.7, 91.1c	90.3, 91.7c	93.7, 94.7c

14.8 60K システムの構成

	高速版	高精度版
音響モデル	PTM 129x64 (3.0MB)	triphone 2000x16 (8.6MB)
言語モデル	75 compress10p (54.5MB)	75 cutoff-1-1 (99.7MB)
デコーダ	高速	高精度
認識時間	2.9x RT	16.9xRT
認識率 (男性 D)	89.1, 90.9c	93.7, 94.6c
認識率 (女性 D)	91.6, 92.7c	93.4, 94.9c
認識率 (GD 平均)	90.4, 91.8c	93.6, 94.8c
認識率 (GI 平均)	88.9, 90.5c	93.2, 94.2c