日本音響学会2012年春期研究発表会（#asj2012s）

ビギナーズセミナー　～若手による新人のための講演・講習会～

音声認識エンジンJuliusの利用方法おさらいとカスタマイズ

@naoh16

奈良先端科学技術大学院大学　情報科学研究科
音情報処理研究室（鹿野研究室）助教

原　直（HARA, Sunao）

hara [at] is.naist.jp

2012.03.13

良くみかけるQ&A

Q: 初心者です！
Juliusの使い方がわかりません！

1. JuliusBookを読みましょう。
2. ソースを読みましょう。
3. JuliusのForumに“英語”で質問してみましょう。

NAIST

Juliusとは？

- フリーの大語彙音声認識ソフトウェア
  - 無償で利用可能
- 商用にひけをとらない高性能な認識
  - 高速・高精度な音声認識処理
  - フリーでは世界一・・・？
- 学術・応用の両側面で貢献
  - 国内外の多くの機関で利用
    - 研究：モデル、アルゴリズムの評価
    - 開発：応用システムの開発
  - 国内の音声認識研究のデファクトスタンダード

* 奈良先端大講義資料より引用

Juliusの特徴

- 高精度・高効率な認識処理
  - 数万語を実時間認識（PC）
  - 省メモリ：マイコン上でも動作
- スケーラビリティ
  - 国会議事録作成（数万語）から
    数語のコマンド認識まで
- インタフェースの汎用性
  - 任意のシステムを構築可能
  - 多言語でも動作：英語・フランス語・タイ語・中国語・・・
  - オープンソース
    - ソースの自由な入手・解析・改変

* 奈良先端大講義資料より引用

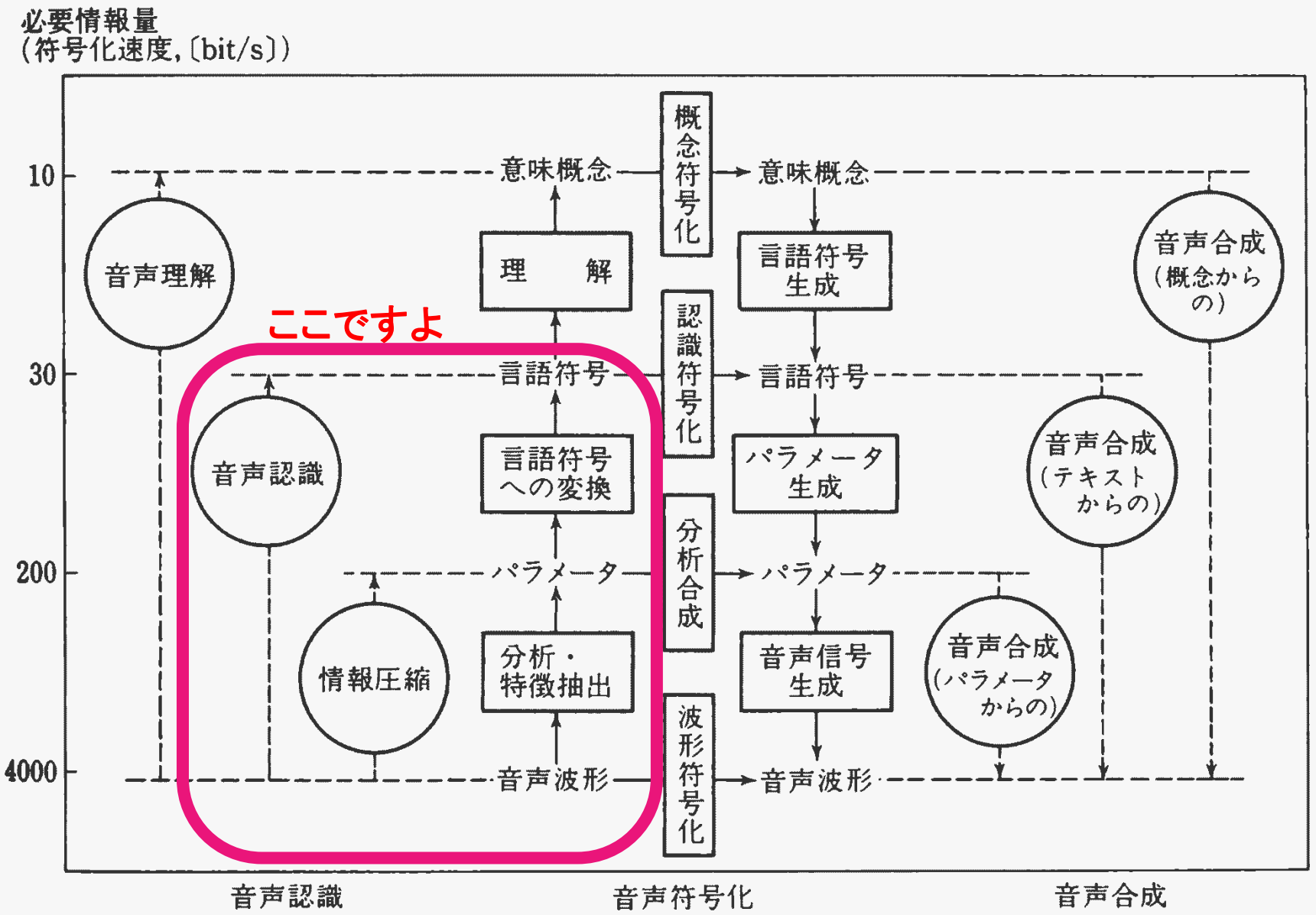


図 10.15　音声情報処理の階層構造

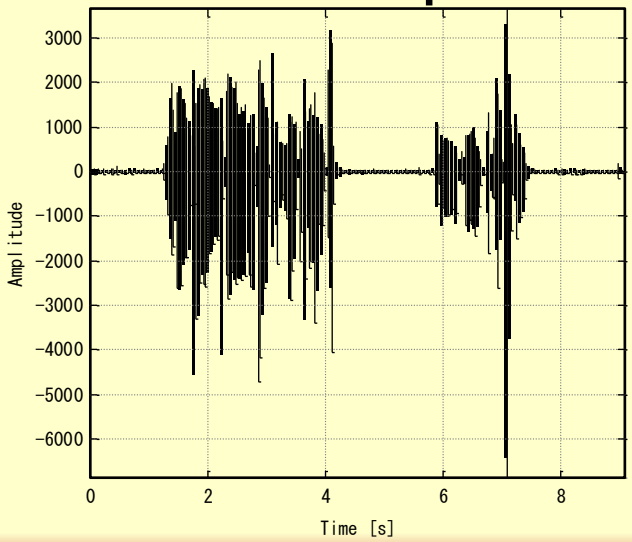
* 古井, “10章 音声の基本的性質”, 音響・音声工学, p.111, 近代科学社, 1992.

まずは使ってみる

- Sourceforgeからダウンロード
  - 一通り入っているJulius dictation kitが良いと思います（現在 v4.2が最新; juliusはv4.2.1）

```
bin/julius -C fast.jconf -input rawfile
```

  - 音声の例 sample2.wav



（発話内容）
衆議院議員はどう考えているのか？
神の声を聞くがよい！

入力音声

- -input で色々変更可能
  - マイクロホンからの入力　　mic, alsa, osd
  - ネットワーク経由　　adinnet, esd
  - PCM音声ファイル入力　　rawfile, wavfile, stdin
  - 特徴量ファイル入力　　mfccfile

- エンディアン（バイトオーダー）に注意
  - rawfile, mfccfileはHTK準拠＝ビッグエンディアン
- 後段の処理に注意
  - ケプストラム正規化の処理が変化する
- 20秒ルール
  - 内部バッファの都合上20秒で強制的に分割される

音声区間の検出

- VAD (Voice activity detection)
  - 音声入力の入り口でリアルタイム処理には必須

1. レベルとゼロ交差による入力検知
   - 「一定レベル(-lv)を越える振幅について
     ゼロ交差数が一定数(-zc)を超えた時」
     - 振幅はそのまま音の大きさに対応し、ゼロクロス回数は低域通過フィルタの遮断周波数に“ほぼ”対応する
   - 明示的にon/offするなら `-cutsilense / -nocutsilence`

音声区間の検出

2. GMMによる音声区間検出（GMM-VAD）

- コンパイル前のconfigureで指定する

```
./configure –enable-gmm-vad
```

- 起動時オプション（GMMによる棄却には使用不可）

```
-gmm adult.gmmdefs –gmmreject laugh,caugh,noise
```

3. デコーダベースの音声区間検出

- コンパイル前のconfigureで指定する

```
./configure –enable-decoder-vad
```

- 起動時オプション `-spsegment`
  - 複数インスタンスでは最初のインスタンスでの結果が用いられる

特徴量抽出

- ともかく音響モデルに合わせること！

- 陥りやすい罠パラメータ
  - サンプリングレート（-smpFreq）
    - 48kHzの入力に関しては-48to16を使うのはアリ
  - 周波数カットオフの下限・上限（-lo/-hifreq）
    - 低域（高域）通過フィルタのカットオフじゃなくて、フィルタバンク処理におけるカットオフ

- 最近のJuliusではHTKの設定ファイルを読めるので、そちらを使った方が安全

ケプストラム正規化
Cepstral Mean Normalization (CMN)

- 長時間のケプストラム平均を差し引く
  - Cepstal mean subtraction (CMS) とも
  - HTKのパラメータ形式では、“_Z”
  - マイクロホンなどの伝達系による乗法性歪みの補正
- 利用パターン
  - ファイル毎のCMN【-norealtimeのデフォルト】
    - ファイル毎に算出されるため初期値不要
  - MAP-CMN【-realtimeのデフォルト】
    - 初期値無し・更新有り
    - 初期値有り・更新有り

      ```
      -cmnload adult.dat –cmnupdate –cmnsave adult.dat
      ```

    - 初期値有り・更新無し

      ```
      -cmnload adult.dat -cmnnoupdate
      ```

  - CMNを使わない
    - 音響モデルを”_Z”なしで学習しておく

認識パラメータの調節

1. 第一パスのビーム幅を設定する！

2. 言語スコア重みと挿入ペナルティを設定する！

3. 文仮説数や出力形式を設定する！

ビーム幅の指定

- Juliusのチューニングとは、
  枝刈りマシンとの生き残りをかけた仮説達の戦いである

- Juliusは2パス探索
  - 第一パスで荒く探索（2-gramなど）
  - 第二パスで高精度な探索（3-gramなど）

- そもそも、第一パスで仮説が残ってなければ第二パスで回復することもあり得ない

- 第一パスのビーム幅は計算時間が許す限り大きく取る
  - デフォルトはモノフォン `-b 400`　トライフォン `-b 1000`
  - 究極は `-b 0` 枝刈りなしの全探索）

* 数字はJuliusBookより抜粋

ビーム幅の指定（補足）

- 無指定のデフォルト値は自動推定している
  - libjulius/src/m_chkparam.c#437 set_beam_width()
  - 基準値　辞書語彙数の平方根の15倍
  - 上限（m_chkparam.c#369 default_width()）

	高速版（fast）	標準版（Standard）
Monophone	400	700
Triphone	800	1500
Triphone (PTM)	600	800

  - 下限
    - 200 (`#define MINIMAL_BEAM_WIDTH 200`)
- 実験の時は自分で指定したほうがよい

言語スコア重みと挿入ペナルティ

- 言語スコア重み
  - 音響モデルの尤度と言語モデルの尤度のダイナミックレンジを補正する
  - 数字を大きくすると言語制約が強くなる
- 挿入ペナルティ
  - 単語数の多い文章の方が言語モデルの尤度は高くなる
  - マイナスになるほど短めの文章が残りやすくなる

- 第一パス(-lmp)と第二パス(-lmp2)は同じように変更する
- 最適値は Grid Search（網羅的に値を振る）

```
(triphone)
-lmp 8.0 -2.0
-lmp2 9.0 -2.0
```

```
(monophone)
-lmp 5.0 -1
-lmp2 6.0 0
```

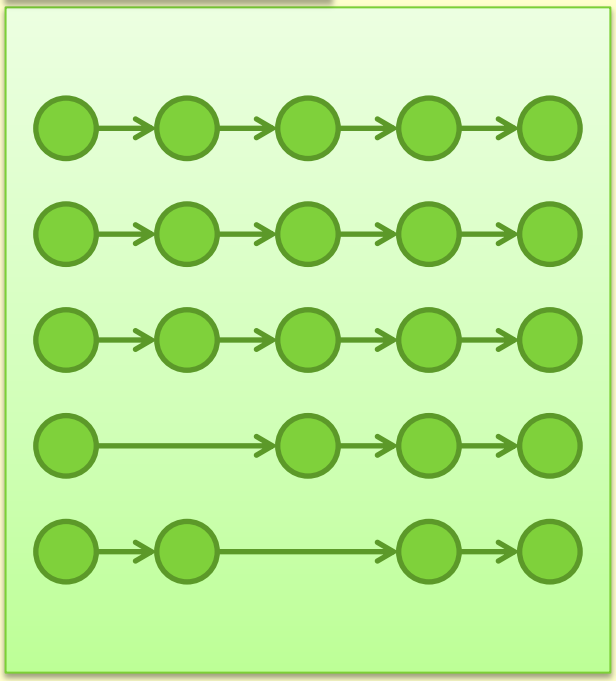
文仮説数と出力結果

- 通常はN-Best探索（N=1）→出力
- 第二パスのN-Best候補の数（-n）
  - その中で実際に出力する数（-output）
  - ヒューリスティクスに基づいた探索なので、-outputより-nを大きくする

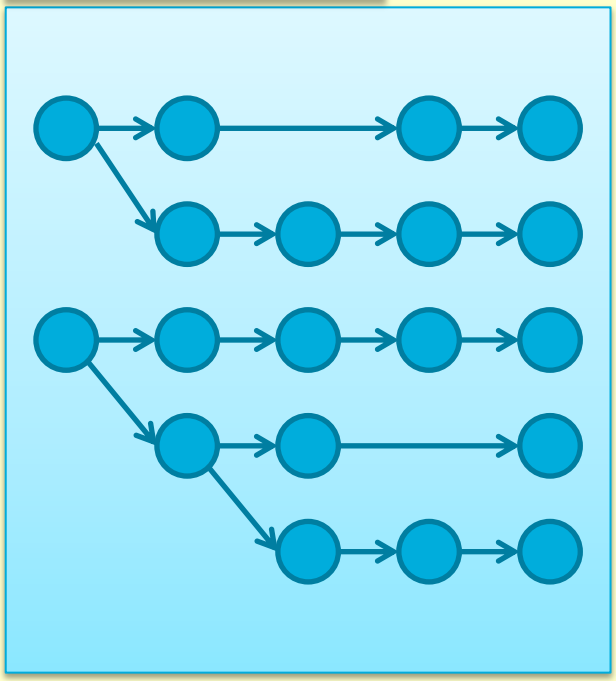
- その他の出力形式
  - Word lattice (graph) `-lattice`
  - Confusion network `-confnet`

出力形式のイメージ図

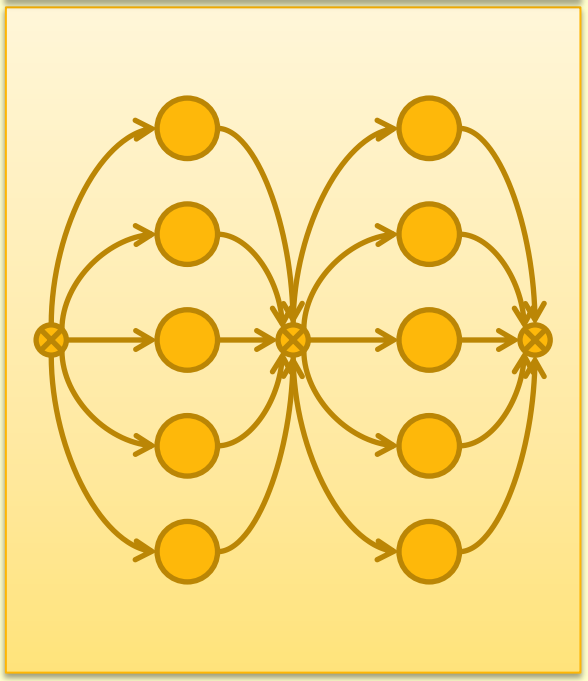
N-best list



Word lattice



Confusion network



- 認識結果をどの程度信頼して後の処理で使うかによって選ぶ

(参考)音声認識

- 音声認識:音声入力$\boldsymbol{O}$に対する単語列Wの推定
- 例:事後確率最大化基準による推定

$$\hat{W} = \arg\max_{W_n} P(\boldsymbol{W}|\boldsymbol{O})$$

$$= \arg\max_{W_n} \frac{P(\boldsymbol{W},\boldsymbol{O})}{P(\boldsymbol{O})}$$

$$\simeq \arg\max_{W_n} P(\boldsymbol{O}|\boldsymbol{W})P(\boldsymbol{W})$$

実用上の仮定

音響モデル

言語モデル

$$\hat{W} \simeq \arg\max_{W_n} P(\boldsymbol{O}|\boldsymbol{W};\mathcal{H}_{\boldsymbol{\alpha}})P(\boldsymbol{W};\mathcal{H}_{\boldsymbol{\lambda}})$$

特徴量ベクトルの時系列集合

特徴量ベクトル

$$\mathbf{O} = [\mathbf{o}_1, \cdots, \mathbf{o}_t, \cdots, \mathbf{o}_T]$$

$$\mathbf{o}_t = [o_{t,1}, \cdots, o_{t,m}, \cdots, o_{t,M}]^T$$

仮説の集合

単語列の候補

$$\mathbf{W} = \{W_1, \cdots, W_n, \cdots\}$$

$$W_n = [w_{n,1}, \cdots, w_{n,\ell}, \cdots]$$

(参考)音声対話

- 音声対話:音声入力Oに対する対話行動Aの推定

$$\hat{A} = \arg\max_{A_k} P(\boldsymbol{A}|\boldsymbol{O})$$

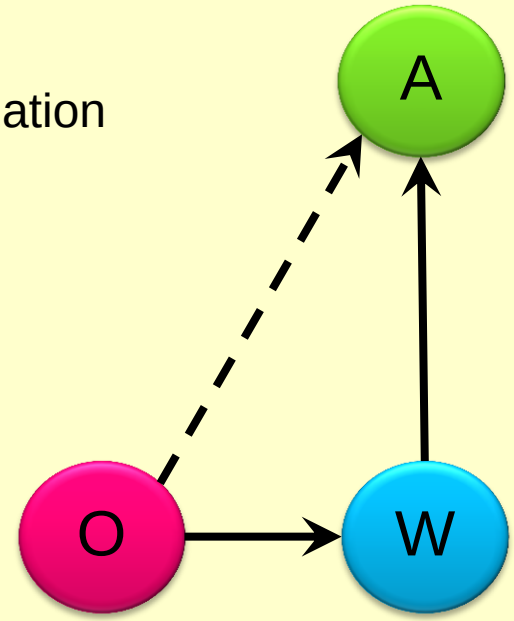
$$= \arg\max_{A_k} \sum_{\boldsymbol{W}} P(\boldsymbol{A}, \boldsymbol{W}|\boldsymbol{O})$$

$$= \arg\max_{A_k} \sum_{\boldsymbol{W}} P(\boldsymbol{A}|\boldsymbol{W}, \boldsymbol{O}) P(\boldsymbol{W}|\boldsymbol{O})$$

$$\simeq \arg\max_{A_k} \sum_{\boldsymbol{W}} P(\boldsymbol{A}|\boldsymbol{W}) P(\boldsymbol{W}|\boldsymbol{O})$$

$$\simeq \arg\max_{A_k} \sum_{\boldsymbol{W}} P(\boldsymbol{A}|\boldsymbol{W}) P(\boldsymbol{O}|\boldsymbol{W}) P(\boldsymbol{W})$$

周辺化
marginalization



対話
戦略

音響
モデル

言語
モデル

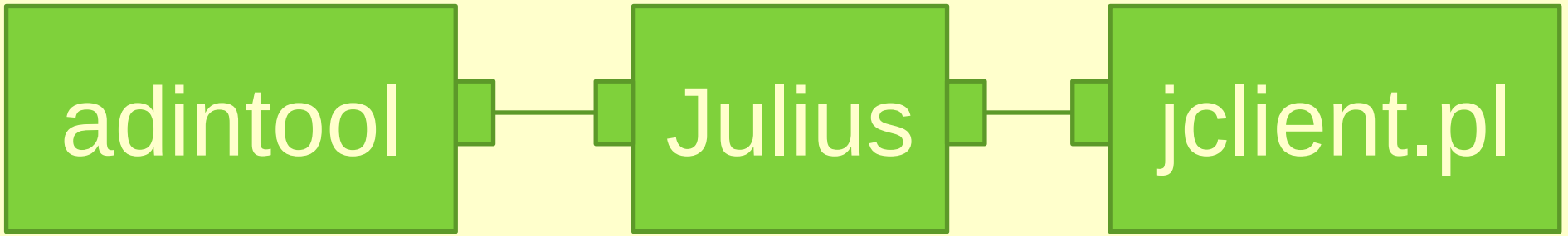
初期化の高速化

- 音響モデルのバイナリ化(mkbinhmm)
  - HTKのパラメタ設定ファイルも埋め込める
- 音素リストのバイナリ化(mkbinhmmlist)
- 言語モデルのバイナリ化(mkbingram)

- 繰り返し実験を行う場合には、一日がかりだった実験が30分で終わるぐらいに変わる

ネットワーク型

- モジュールモード(-module)とネットワーク音声入力(-input adinnet)を用いる事で、サーバ型の音声認識システムにできる
  - 接続はTCP/IP



プラグインの開発

- 音声入力プラグイン
- 音声後処理プラグイン
- 特徴量入力プラグイン
- 特徴量後処理プラグイン
- ガウス分布計算プラグイン
- 結果取得プラグイン

- コールバック関数の追加

NAIST

まとめ

- Juliusを使って楽しい研究をしましょう。