音声認識ソフトウェア Julius

河原 達也（京都大学）
李 晃伸（名古屋工業大学）

JSAI2011 AIレクチャー1：ツールボックス
2011年6月2日（水）9:00〜10:30

概要

- 音声認識の概要
- Juliusの概要
- 言語モデルの構築
  - 単語認識
  - 記述文法
  - N-gram
  - クラスN-gram
  - 複数モデル
- アプリケーションとの連携
  - パイプ
  - サーバモード
  - JuliusLib
- 音声インタラクション構築ツール
  - 対話スクリプト

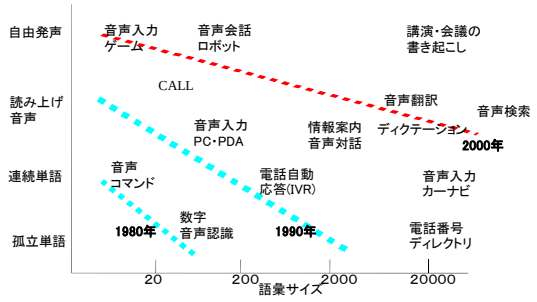
音声認識の現在のレベル

- かなりできた？
  - MS Windows 標準装備
  - Google Voice Search
- まだまだ使えない？
  - 携帯端末やタブレット端末で音声入力している人はほとんどいない
  - ロボットでまともに音声会話できるものもない
  - 使った?人の評価?

音声認識の現在のレベル

- 外国語話者？ 母国語話者？
  - かなり高いレベルではあるが、
  - コンピュータにとってはしょせん外国語
    → 話し言葉や騒音のバリエーションに頑健でない
  - ただしユーザは母国語話者を期待
- 万能でない
  - OCRのようにブラックボックス的に使えない
  - アプリ（タスクドメインや入力環境）に応じて、パーツや設定を作成／適応する必要
  - チューンすればよくなるが、ミスマッチがあると大きく性能低下

語彙サイズと発声スタイルによるアプリケーションの分類



解決されていない課題

- 自由発声（発声スタイル）
  - 話し言葉、感情的な音声
  - なまり、ささやき声
- 実環境（使用環境）
  - 雑音（特に非定常な雑音）
  - 遠隔発話・集音マイクによる入力

…利用者が発音の怠けや背景音の存在に気づきにくい

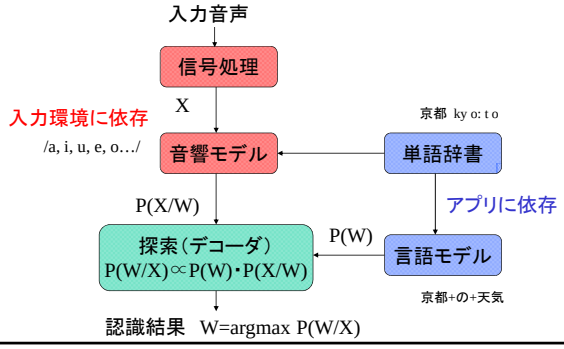
音声認識の原理

- 入力音声Xに対して、事後確率 p(W|X) が最大となる単語列Wを見つける問題として定式化

$$p(W \mid X) = \frac{p(W) * p(X \mid W)}{p(X)}$$

- 分母 $p(X) = \sum_{W} p(W) * p(X \mid W)$ はWの決定に影響しないので認識の際には無視
  - ただし、信頼度計算には利用する

音声認識の原理



音声認識の基本要素

- 音響モデル：音素の周波数パターンを保持し、入力音声（を分析した量）と照合
- 単語辞書：認識対象の語彙（=単語の集合）とその発音を規定
  → 規定されているもののみが照合の対象
  …音素を認識してから単語を照合するのでなく、単語辞書を照合しながら音素を認識
- 言語モデル：単語の連鎖を規定
  → 記述文法の場合は規定されている連鎖のみが照合の対象
- デコーダ：最尤の単語列を探索（音声Xをテキストwにデコード）

音声認識における言語モデル

単語列の先見確率 p(W) を規定

- 言語的な知見・統計量
  - 「私」→「は」「の」…
- 対話の一貫性からの制約
  - 「どこですか？」「値段は？」
- タスクドメインの制約
  - 天気案内、旅行会話
  - 最も強力 → タスクドメインに応じて単語辞書と言語モデルを用意する必要

音声認識における言語モデル

通常、先頭の単語から逐次的に適用

$$p(W) = \prod_{i} p(w_i \mid w_1 \ldots w_{i-1})$$

- 記述文法
  - （部分）文として受理できるか(1/0)
- N-gramモデル
  - $p(w_i|w_1 \cdots w_{i-1})$を直近のN単語連鎖$p(w_i|w_{i-N+1},..w_{i-1})$で近似
  - N=3（3単語連鎖）の場合がトライグラムモデル

音声認識における音響モデル

パターン認識処理により、p(X|W)を計算

- HMM (Hidden Markov Model)
  - 基本的には混合ガウス分布モデル
  - 話者層（性別・年齢層）に依存
    → 個別のモデルを構築するか、混合分布で吸収
  - 入力環境（パソコン、電話、自動車内）に大きく影響される
    → 個別のマッチしたモデルを用いるか、適応を行う

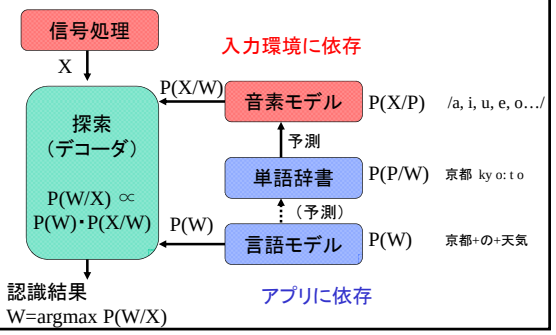
音声認識における音響モデル

- 音素単位のテンプレート → 音素モデル
- ただし、前後の音素に依存したモデル化（トライフォン）が一般的
  - → 一つのモデル集合内で別のエントリを定義
    i-k+a…前がi,後がaのk
- 単語w_iと音素列p_{ij}（発音）の対応づけ$p(p_{ij}|w_i)$は単語辞書で記述

  「京都」: ky o: t o　「は」: w a　「へ」: e

$$p(X|W)=\prod_i\prod_j p(x|p_{ij})$$

音声認識の原理



音声認識の主要技術

- 音響モデル：HMM
  - 1990年頃から約20年間
  - モデル学習（識別学習）、適応（最尤線形回帰）、特徴量正規化などで大きな進歩
  - 学習データベースの大規模化
    - 数百時間～、数千人～
- 言語モデル：N-gram
  - 同上；ただし技術的進歩は比較的小さい

音声認識におけるデコーダ（認識エンジン）

- 様々な単語列Wについて仮説を生成し、確率を計算し、最大のものを選択

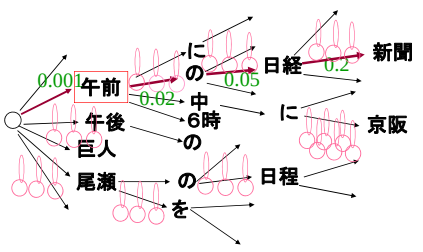
$$\hat{W}=\arg\max\ p(W|X)$$
$$=\arg\max\ p(W)*p(X|W)$$
$$=\arg\max\ \{\log p(W)+\log p(X|W)\}$$

- 実際には下記の尤度が一般的

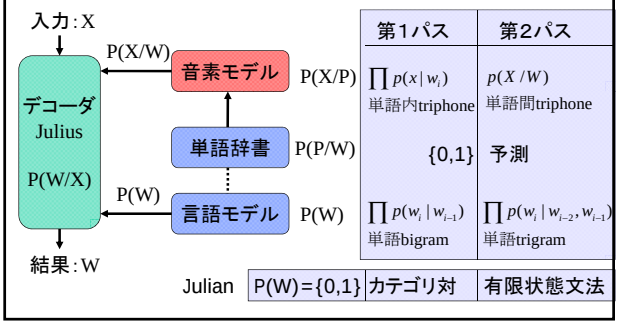
$$\log p(X|W)+\alpha\log p(W)+\beta*N$$

- 仮説数は膨大（語彙サイズ^長さN）なので、対数尤度の計算を先頭から逐次的に行い、尤度が大きい候補のみを展開（＝探索）

単語列仮説の探索



Juliusの動作原理



概要

- 音声認識の概要
- Juliusの概要
- 言語モデルの構築
  - 単語認識
  - 記述文法
  - N-gram
  - クラスN-gram
  - 複数モデル
- アプリケーションとの連携
  - パイプ
  - サーバモード
  - JuliusLib
- 音声インタラクション構築ツール
  - 対話スクリプト

27

大語彙音声認識エンジンJulius

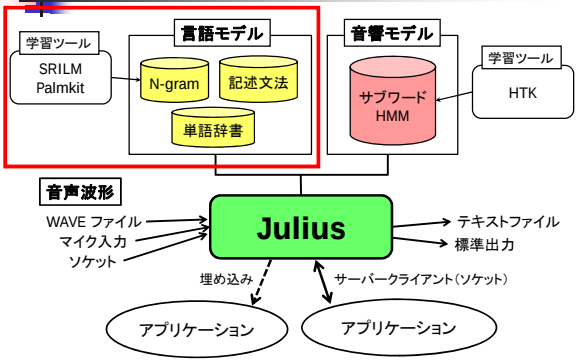
28

Julius

- 大語彙連続音声認識ソフトウェア
  - 2パス探索
  - 計算効率の良い探索アルゴリズム
- オープン性
  - HTK 等のツールで学習したモデルを使用可能
  - タスク・対象環境ごとのカスタマイズが可能
  - オープンソース
- 言語独立性
  - 対象言語の音響モデル・言語モデルで動作
- プラットフォーム独立性
  - C 言語
  - Linux で開発（Solaris, MacOSX, FreeBSD で動作）
  - Windows でも動作（MSVC, cygwin）

29

Julius を用いた音声認識システムの構成



30

アクセス

http://julius.sourceforge.jp/

- ダウンロード
  - Julius 最新版（実行バイナリ・ソースコード(CVS)）
  - ディクテーションキット（日本語標準音響・言語モデル）
  - 文法認識キット
  - カスタム実行キット
    - モデルの組合せを選択可能（@京大）
  - 応用キット
    - 単語・音素セグメンテーションキット
- 関連文書
  - 解説書（Juliusbook）、チュートリアル等
- ユーザフォーラム

31

開発履歴

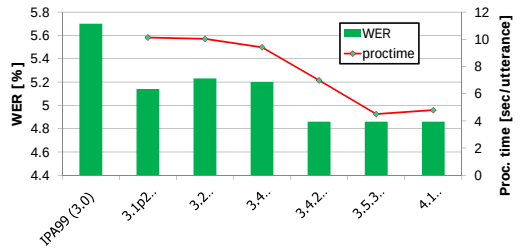
- 1.0 (1998.02)
- 2.2 (1999.10)
  - 探索アルゴリズム改善, 32k, 第1パス単語間triphoneの近似計算
  - 第2パスにビームを導入
- 3.0 (2000.02)
  - PTM, Gaussian pruning, 64k 対応, アラインメント
- 3.1p2 (2001.02)
- 3.2 (2001.08)
  - ショートポーズセグメンテーション
- 3.3 (2002.09)
  - モジュールモード, 複数文法同時認識, SS, マルチパス版
- 3.4 (2003.10)
  - サーチ調整, 信頼度計算, クラスN-gram, バイナリHMM
- 3.4.2 (2004.5) CSRC最終版
- 3.5 (2005.11)
  - GMMによる棄却, 単語グラフ出力, 信頼度枝刈り
  - MFCC HTK互換性拡張, MAP-CMN (3.5.1)
- 3.5.2 (2006.07)
  - 計算速度 20% 前後アップ, 文法の最小化, SLF2DFA
- 4.0 (2007.12)
  - Julian 統合, モジュール化, ライブラリ化, マルチデコーディング
  - 任意長N-gram, 孤立単語認識, CN, GMM-VAD, Decoder VAD,
- 4.1 (2008.10)
  - プラグイン対応, マルチストリーム対応, MSD-HMM対応, CVN, VTLN
  - ドキュメント刷新, Juliusbook 作成
- 4.2 (2011.05)
  - Microsoft Visual C++ 対応など

IPAツールキット (2000)
CSRC最終版 (2003)
ISTC最終版 (2008)

基本的アルゴリズムの確立
サーチの改善
逐次音声認識
単語信頼度
単語グラフ出力
入力棄却
互換性向上
各部の最適化
アーキテクチャの刷新
モジュール化
言語制約拡張
マルチデコーディング
プラットフォーム拡張

32

バージョンごとの性能変遷



Model: Gender-dependent triphone (2000x16), 20k-word 3-gram
Test set: JNAS IPA98 Test set (23 male, 23 female, 200 utterances)
CPU: Pentium 3GHz

言語モデル

概要

- 単語認識
- 記述文法
  - 有限状態オートマトン(FSA)
  - 文脈自由文法(CFG、LR)

  Julius: BNF記法 → FSA (右再帰のみ)
- 単語 N-gram モデル
- クラス N-gram モデル
- マルチモデル

単語認識

- 現在の実用システムの大半
- 発話単位(文全体)を1単語として登録
  - (例)「あっち向いて」「ほい！」
- 単語辞書の記述

```
0       z e r o
1       i ch i
look    a q ch i m u i t e
way     h o i
bye     s a y o: n a r a
```

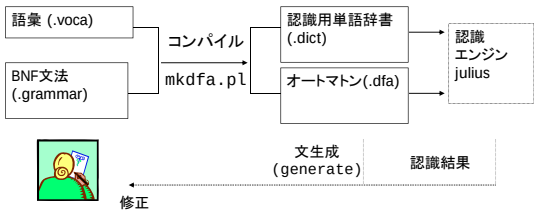
- 読み・音素列
  - 形態素解析器
  - ひらがな→音素列:yomi2voca.pl (Juliusに付属)

実演1 コマンド認識

- あいさつ
- 移動指示「上」「下」「右」「左」
- 動作指示

記述文法による連続音声認識

タスクの語彙と文法を規定



文法・語彙ファイル

- 文法ファイル
  - BNF記法で書き換え規則を定義
  - 非終端記号（単語カテゴリ）のみ

```
S : NS_B  OBJ_P DIR_P PLEASE NS_E
```

- 語彙ファイル
  - 各単語カテゴリに属する単語

```
%OBJ
ボール　　ぼーる
```

39

実演2 タスク文法の例

ロボット操作タスク

- 玉を右に押して
- ブロックを左側に動かして
- 上じゃなくて左です

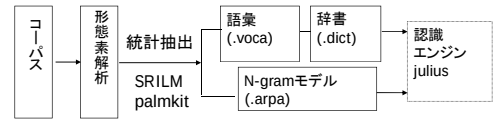
受付案内タスク

- 小泉 さん
- えーと　田中 さん
- あのー　遠山 さんお願いします

40

単語N-gramによる音声認識

文例コーパスから連鎖統計を学習



41

形態素解析

- 日本語は分かち書きされないので不可欠
- Chasen, Mecab, Juman 等
- 修正…区切り・読み
  （例）行って＋オコナッテ、方＋カタ、終了＋シューリョー
- 後処理…数字の位取り
- 「表記＋読み{＋品詞}」で単語を定義
  （例）「本＋ホン{+10}」

42

単語N-gramの構築

- 履歴(N-1)単語から、次単語を予測・評価
  N=2 → bigram,　N=3 → trigram
- 基本的にはN単語連鎖のカウント

$$p(S) = \prod_i p(w_i \mid w_{i-2}, w_{i-1})$$

$$p(w_i \mid w_{i-2}, w_{i-1}) = \frac{count\ (w_{i-2}, w_{i-1}, w_i)}{count\ (w_{i-2}, w_{i-1})}$$

  - 実際にはスムージングが重要

43

文例

その右にある青いやつの前に行って
それを後ろに回ってちょっと押して下さい
そんなには押さない
赤いブロックの右に行って
そうじゃなくって東側
もうちょい右
その位置から押して下さい

44

実演3 単語N-gramモデルの構築

- 形態素解析
- 認識語彙を定義
  - 出現単語の頻度上位 k 単語
- N 単語連鎖をカウント
  - カットオフ
- N-gram 確率推定
  - スムージング

チェック
- ランダム文生成: `generate-ngram`

45

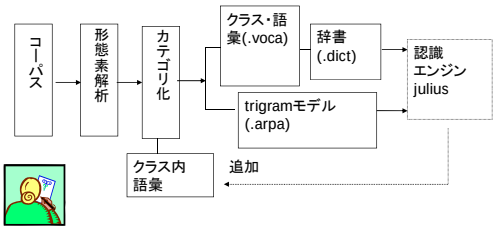
言語モデルの考察1

- 記述文法 （内包的）
  - タスクが複雑だと記述が大変
  - 記述していないパターンは受理できない
  → 話し言葉向きでない
- 単語N-gram （外延的）
  - 大量にデータを集めないと信頼できない
  - コーパスに出てこない単語は受理できない
  → 対話システム向きでない

46

クラスN-gramによる音声認識

コーパスのキーワードをカテゴリ（抽象）化



47

クラス導入（抽象化）されたテキストの例

<s> その+ソノ DIRECTION に+ニ ある+アル COLOR OBJECT の+ノ DIRECTION に+ニ DO て+テ </s>

<s> それ+ソレ じゃ+ジャ なく+ナク て+テ もっと+モット DIRECTION に+ニ あっ+アッた+タ COLOR OBJECT の +ノ DIRECTION に+ニ 行け+イケ </s>

<s> それ+ソレ を+ヲ DIRECTION に+ニ DO て+テ ちょっと+チョット DO て+テ </s>

48

言語モデルの考察2

- クラスN-gram
  - 文法よりも話し言葉に強い
  - カテゴリと語彙でタスク知識を反映
  - 言い回しまでは登録できない
  →やはりかなりの文例（数百文）が必要
- マルチモデル
  - 文法と N-gram の併用
  - 言語モデルごとの認識結果→後段処理で判定

49

実演4 マルチモデル

- ひとつの認識プロセスで並列動作
  - 単語：単一コマンド
  - 文法：複雑な命令、定型的文章
  - N-gram：自由な発話（ディクテーション）
- 認識結果の選択
  - 音響尤度
  - 対話の文脈
  - 意味解釈

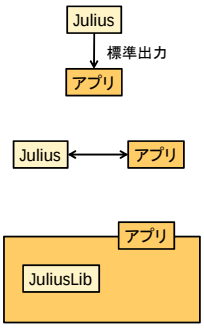
50

アプリケーションとの連携

51

連携方式

- 標準出力
  - 結果のログを解析
- サーバークライアント
  - 双方向にソケット接続
  - 認識結果の送信
  - 命令の受信
- ライブラリ
  - アプリへ組み込み



52

標準出力

- Julius の標準出力を取り込む
  - パイプ等
  - クライアント側で解析・取り出し
- アプリからJuliusの制御はできない

```
sentence1: <s> 赤い ボール を 右 に </s>
wseq1: 7 0 1 2 3 4 8
phseq1: silB | a k a i | b o: r u | o | m i g i |
n i | silE
cmscore1: 1.000 1.000 1.000 1.000 1.000 1.000
1.000
score1: -5139.489258
```

53

サーバークライアント

- Julius ＝音声認識サーバ、アプリ＝クライアント
- 結果をクライアントへ送信
  - 認識結果
  - 音声認識のトリガ開始・終了　等
- クライアントアプリから制御命令を受理
  - 音声入力中断
  - 音声入力再開
  - 言語モデルの入れ替え
  - モデルごとの無効・有効切り替え

54

クライアントへ送信される結果

```
<STARTPROC/>
<INPUT STATUS="LISTEN" TIME="994675053"/>
<INPUT STATUS="STARTREC" TIME="994675055"/>
<STARTRECOG/>
<INPUT STATUS="ENDREC" TIME="994675059"/>
<GMM RESULT="adult" CMSCORE="1.000000"/>
<ENDRECOG/>
<INPUTPARAM FRAMES="312" MSEC="3120"/>
<RECOGOUT>
  <SHYPO RANK="1" SCORE="-6888.637695" GRAM="0">
    <WHYPO WORD="silB" CLASSID="7" PHONE="silB" CM="1.000"/>
    <WHYPO WORD="赤い" CLASSID="0" PHONE="a k a i" CM="1.000"/>
    <WHYPO WORD="ボール" CLASSID="1" PHONE="b o: r u" CM="0.988"/>
    <WHYPO WORD="を" CLASSID="2" PHONE="o" CM="1.000"/>
    <WHYPO WORD="右" CLASSID="3" PHONE="m i g i" CM="1.000"/>
    <WHYPO WORD="に" CLASSID="4" PHONE="n i" CM="1.000"/>
    <WHYPO WORD="silE" CLASSID="8" PHONE="silE" CM="1.000"/>
  </SHYPO>
</RECOGOUT>
```

55

命令

- PAUSE-エンジンを一時停止する.
- RESUME-一時停止状態から復帰してエンジン動作を再開する
- GRAMINFO-エンジンが現在持つ文法情報を返す
- CHANGEGRAM
- ADDGRAM
- その他

56

ライブラリとしての組み込み

- Julius のコア部をライブラリとしてリンク
- 音声認識部の処理手順
  - ライブラリ初期化
  - 音声ストリームを開く
  - 認識のメインループ
- メイン関数はライブラリ内にある
  - 入力終了までループ
    - 制御は返らない
    - コールバックを活用 or スレッド化
  - 認識結果の取得：コールバックを定義

57

コールバックの種類

イベント callback	・起動・停止・入力開始終了・認識開始終了・・・
認識結果 callback	・第1パス，第2パス，途中結果（一定フレームごと） ・単語グラフ・confusion network・アラインメント
割り込み callback	・一定時間おきに割り込み
入力音声 callback	・入力音声やトリガ音声を加工
一時停止 callback	・エンジン一時停止時の処理移行先

58

プログラム例（C言語）

```
#include <julius/juliuslib.h>
void status_recready(Recog *recog, void *dummy) { ..... }
void status_recstart(Recog *recog, void *dummy) { ..... }
void output_result(Recog *recog, void *dummy)  { ..... }
int main(int argc, char *argv[]) {
        Jconf *jconf = j_config_load_file_new(jconf_filename);
        Recog *recog = j_create_instance_from_jconf(jconf);
        callback_add(recog, CALLBACK_EVENT_SPEECH_READY, status_recready, NULL);
        callback_add(recog, CALLBACK_EVENT_SPEECH_START, status_recstart, NULL);
        callback_add(recog, CALLBACK_RESULT, output_result, NULL);
        j_adin_init(recog);
        j_open_stream(recog, NULL);
        j_recognize_stream(recog);
        j_recog_free(recog);
}
```

MS Visual C++ 用のラッパークラスも配布パッケージに同梱