Julius 音声認識エンジンを基盤技術とする リアルタイム病気診断システム

Infrastructure technology and Julius of speech recognition engine Real-time disease diagnosis system

韓 宇輝† 亀田 弘之†
Han Yuhui Hiroyuki Kameda

音声認識の進歩とともに機械との対話を実現する音声インタフェースへの期待が高まっている．電話応対システムやカーナビの音声インタフェースの一部では，今や実用に足る有用性を得たと言えよう．しかし，音声インタフェースが日常社会に普及した例はまだ少ない．本研究では，これらの音声認識インタフェース部の構成と中小企業への技術移転を目的に，リアルタイム病気診断システムの開発を行った．

1． はじめに

音声は我々にとって最も手軽な情報伝達手段である．このため，機械やコンピュータを音声で操作したいという要望は以前から強く，音声認識技術の実用化に期待が寄せられてきた．このような背景があったにも関わらず，現在の音声認識技術は，高価な専用装置を必要としながらも認識性能が不十分なままである．その結果，中小企業は製品開発にそれらの技術を採用されることが難しい．

しかし，近年のハードウェア技術，ソフトウェア技術の急速な進歩によって，不特定話者を対象とする安価な音声認識ソフトウェアが登場し，音声認識技術が実用化段階に達しつつある．そこで筆者らは，これらの音声認識インタフェース部の構成と中小企業への技術移転を目的に，リアルタイム病気診断システムの開発を行った．

本研究では，開発の前段階としてタスク適応言語モデルの構築法を検討する．大語彙連続音声認識（Julius 図 1）では，必要な認識精度を得るのに，認識対象と合致した単語 N-gram 言語モデルが必要である．その構築には大規模なテキストコーパスが必須で多大な人手を要するので，本研究では，半自動的に言語モデルを構築でき，開発のコスト削減に貢献する手法を採った．その手順は，（1）コーパスの自動作成とトピック依存 N-gram 言語モデルの構築（2）モデル融合でのタスク操作（3）文法の適用によるモデル高精度化の三段階から構成．

一方，リアルタイム病気診断システムは，一問一答形式(例:発熱して，咳も出ると言うと．システムがリアルタイムであなたが風を引いたというメッセージが出てきた．)の音声認識インタフェースを持ち，データベースと連携し，リアルタイムな病気診断サービスを提供する．

†東京工科大学, Tokyo University of Technology

また，音声認識インタフェースの開発では，継続的運用による経験とデータの蓄積が不可欠である．本システムは音声認識エンジンの言語モデルとデータベースを，中小企業の求める個別のアプリケーションに応じて変更することができる．

図 1 Julius のホームページ

2． 音声認識エンジン Julius とは

Julius[1] は大語彙連続音声認識を実行するソフトウェアである．これまでの音声認識の研究で培われてきたアルゴリズムや計算効率化手法をおよそ実装しており，高精度かつ効率よく認識処理を行うことができる．Julius の最も大きな特徴は汎用性・可搬性にある．音響モデルや言語モデルの仕様は標準的なフォーマットを採用しており，置き換えたり修正したりすることが容易である．これにより，大語彙のディクテーションから小語彙の数字認識まで，様々な使用環境や目的に応じたシステムを構築することが簡単にできる．また言語モデルも，大規模な統

計モデルである単語 N-gram，文パターンを人手で与える記述文法，辞書のみを用いる単語認識といったように多様な形式をサポートしており，使用環境に応じて適切なモデルを使い分けることができる．さらに，ソースコードが公開されているので，ユーザ独自の修正や拡張も可能である．

アプリケーションとの連携についても，Julius ではネットワークを介した音声入力や結果のリアルタイムな送受信，プラグインによる機能拡張，認識エンジンライブラリの埋め込みなど，様々な形態をサポートしている．また Microsoft SAPI 対応版も公開されている．逆に，話者の事前登録学習（エンロールメント）や自動適応学習などの機能は現在実装されていない．また，単語を簡単に登録するようなインタフェースも用意されていないため，システム開発者は単語辞書・文法ファイルなどを直接操作する必要がある．

Julius はこのように自由度が高いので，ディクテーションにとどまらず，講演や会議などの自動書き起こし，会話ロボット・エージェント，家電の操作，次世代カーナビなど様々な用途に利用されている．その反面，自由度が高いということは，ある程度中身を理解して自分で組み立てる必要があることを意味する．例えていうなら，市販のソフトが既製のパソコンのようなものであるのに対して，Julius はカスタムメイドのパソコンに対応する．パーツを理解した上で，自分の目的に合致した仕様のものを選んで構成する必要がある．自分の要求にあう仕様のパーツがなければ，自分で作成することもできるのである．

2.1 HMM 音響モデル

音響モデルは，音素の並びを統計的に学習したものであり，HMM（Hidden Markov Mode; 隠れマルコフモデル）によるモデル化が広く使われる[2]．HMM は特徴ベクトル時系列の確率モデルであり，自己遷移を持つ複数の状態間を遷移することで，音声のような長さの一定しない時系列信号を効率良くモデル化することが可能である．HMM の学習には EM（Expectation Maximization）アルゴリズムと呼ばれる最尤パラメータ推定手法が用いられる．

音響モデルは，通常，単純に音素ごとにモデル化を行ったモノフォン（monophone）モデルである．しかし，音素は同じ音素であっても調音結合の影響により後続する音素によって変化する．この対処として前後の音素環境を考慮した三つ組み音素をモデル単位として利用する．その音素環境依存モデルのことをトライフォン（triphone）モデルと呼ぶ．前後の音素環境を考慮するためトライフォンモデルは高い認識精度を得ることができるが，音素の組合わせにともなって HMM のモデル数が増大するため，認識に必要な計算量は大量になる．また，膨大な数のすべてのトライフォンに対して十分な学習用音声データを確保することは困難である．そこで，状態ごとに音響的特徴が近いトライフォンを共有し，モデル数を削減した状態共有トライフォンが用いられる．一方，HMM では状態ごとに混合正規分布を用いて出力確率分布を構成するため，異なるモデルや状態間で混合正規分布のための正規分布を共有することで効率的なモデルを構成することができる．これを Tied-Mixture モデルと呼ぶ．特に中心音素が同一であるトライフォンの間だけで分布を共有する手法は，PTM（Phonetic Tied-Mixture; 音素内タイドミクスチャ）モデル[3] と呼ばれる．PTM モデルは，モノフォンモデルから出力確率分布を，状態共有トライフォンモデルから状態共有構造を抽出し，出力確率分布を状態共有構造に重みづけを行い作成される．

以後，本研究で用いる音響モデルは，特に断りがない限りこの PTM モデルで統一するものとする．使用した音素数は 5 種類の母音と約 20 種類の子音に長母音，拗音などを加えた合計 43 音素である．音響モデルの学習には HTK[4] を使用し，ファイルフォーマットは HTK 形式のものに準ずる．

2.2 単語辞書と **N-gram** 言語モデル

音響モデルが話者性や音声入力環境などの音声認識における音響的特徴を担うものであるに対し，言語モデルと単語辞書は，言い回しなどの文章表現や認識対象単語などの言語的特徴を定めるものである．言い換えれば，言語モデルは音声認識システムにおいて認識対象となるタスクを決定する要素である．言語モデルと単語辞書の中で定義されていない文章表現や単語を音声認識では受理することは困難なので，音声インタフェース内で使用することもできない．多様な言語的特徴を柔軟に受理できる言語モデルを使用することが求められる．

単語辞書は，単語のエントリ表記と出力上の表記及び音素記号列から構成される．単語辞書ファイルの例を図 2 に示す．ファイルのフォーマットは HTK 形式である．言語モデルの学習元であるコーパス中に出現する異なり単語数は，膨大な数になってしまうため，計算量や記憶量などシステムの実装上の問題から使用する単語数を制限する必要がある．一般にコーパスに出現する頻度が高い単語を上位数千から数万のオーダで限定して単語辞書に使うことが多い．コーパスに出現した単語でもこの辞書に登録されてない単語やその読み（音素記号列）は未知語として扱われ，音声認識することはできない．このため，単語辞書の内容は認識性能に大きな影響を与える．

音声認識の言語モデルには，文脈自由文法などの有限状態ネットワークで記述された記述文法やコーパスから統計的な手法によって確率推定を行う統計的言語モデルが用いられる．通常，自動販売機などの狭い認識対象に限定しても良いタスクの場合には，文法記述型の音声認識を用いることが多い．しかし，記述文法では，システムはあらかじめ想定された文法内の発話のみしか受理できず，発話の文章表現や語尾などの発話様式までも限定されてしまう．認識対象語彙が比較的小さくなる事や複雑な文法を開発者が記述する必要があるなどの問題点も知られている．一方，統計的言語モデルを用いた大語彙連続音声認識では，認識結果を開発者があらかじめ決定的に定義することは難しく，一見するとアプリケーションに組み込むのには向かない．しかし，柔軟に様々な発話を受理することが可能であるため利用する価値は高い．そこで，本研究で開発する音声インタフェース内では統計的言語モデルを採用することにした．

現在，音声認識において最も良く利用される統計的言語モデルは単語 N-gram モデルである．単語 N-gram モデ

ルは単語連鎖のマルコフモデルで構成され，単純なモデルでありながら効果が大きい[5].

```
</s> [] silE
<s> [] silB
、+、+79/0/0 [、] sp
日報+ニッポー+2/0/0 [日報] n i q p o:
日没+ニチボツ+2/0/0 [日没] n i ch i b o ts u
日本+{ニッポン/ニホン}+12/0/0 [日本] n i q p o N
日本+{ニッポン/ニホン}+12/0/0 [日本] n i h o N
```

図 2：単語辞書ファイルの例

2.3 N-gram 言語モデルにおけるタスク適応手法

前述のように N-gram 言語モデルはコーパスから単語の並びを統計的に学習し，大語彙を容易に扱うことができる．このため，多様な文章表現を柔軟に受理可能である．しかし，コーパスに含まれる語彙や語尾様式，発話表現などの言語的特徴は作成された言語モデルにも反映される．音声認識で高い認識精度を得るには，入力音声と言語モデルの言語的特徴が近くなければならず，一つの言語モデルであらゆるタスクの語彙や表現をすべて網羅することは難しい．

例えば，IPA（情報処理振興事業協会）の「日本語ディクテーション基本ソフトウェアの開発」プロジェクト[30]では，言語モデルの学習に新聞記事コーパスを用いた．新聞記事は容易に大量のデータを収集できるが，完成した言語モデルは新聞記事の書き言葉を反映したものになり，日常会話で使用するような話し言葉を対象としたタスクでの認識には適さない．話し言葉を認識するためには話し言葉のテキストコーパスからの言語モデルの学習が必要であり，現在も話し言葉テキストコーパスの整備が求められている[35].

また，音声インタフェースで想定されるタスクは多種多様であるため，それら個々のタスクに適した言語モデルを提供できることが望ましい．ところが，認識対象に合致する大量の学習元コーパスが常に得られるとは限らない．言語モデルの学習用テキストの作成には，対象タスクの話題に応じたコーパス収集の他にも，収集テキストの整形などコストが高い作業を要する．そのため，想定されるあらゆるタスクごとに言語モデルを準備するのは量的に限界があり現実的ではない．

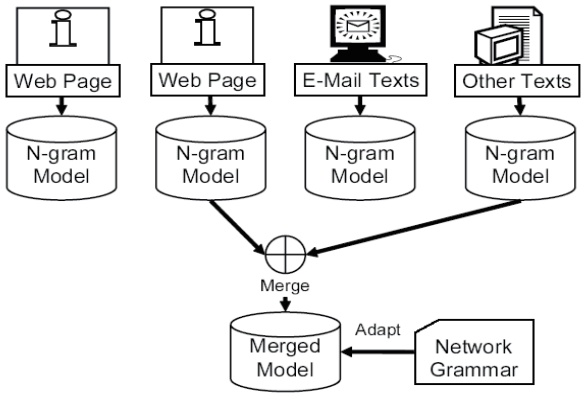
以上のような理由から，効率良く低コストに認識対象タスクに適した N-gram 言語モデルを作成する手段の確立が求められてきた．そこで，本研究では，タスク適応 N-gram 言語モデルの作成過程のほとんどを自動化可能なモデル構築の手順を開発する．その手順は以下の三段階から構成される．概略を図 3 に示す.

1. N-gram モデルの学習に必要なテキストを自動収集する．このとき，テキストはインターネットの代表的なアプリケーションである World Wide Web（以下，Web と略）の Web ページやメール，その他のテキストコーパスから抽出する．収集した学習用のテキストが，ある特定のトピックに偏ったコーパスになるなら，構築される N-gram モデルもそのトピックに依存したモデルとなる．

2. 認識対象タスクに関連するトピックを持つモデルを作成した前述のものの中から選択，互いに融合する．以上の手順により言語モデルのタスク操作を実現する．

3. 高い認識精度を得るために，別途作成したネットワーク記述文法を融合モデルに適用し，タスク適応 N-gram 言語モデルは完成する．

図 3 タスク適応 N-gram 言語モデルの構築手順



3 モジュールモード

本研究では，Julius を音声認識サーバーとして動かすことができる．モジュールモードで起動された Julius は，クライアントアプリケーションと TCP/IP で接続し，認識結果や音声イベントをリアルタイムに送出し，またクライアントからの命令を受信して処理する．

“-module” をつけることで Julius をサーバーモード（モジュールモード）で起動できる．その後，パッケージに含まれるサンプルクライアント“jcontrol”を使って，Julius に接続できる．接続後，音声入力を行うと認識結果や音声イベントが XML 形式でクライアントに随時送信される．例を図 4 に示す．<RECOGOUT>...</RECOGOUT>が認識結果である．認識失敗時は，<RECOGOUT> ... </RECOGOUT> の代わりに<RECOGFAIL/> が出力される．そのほか，エンジンの起動・中断（<STARTPROC/>, <ENDPROC/>），1 回の認識処理の開始・終了（<STARTRECOG/><ENDRECOG/>）など様々な情報がリアルタイムに出力される．

```
<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="382" MSEC="3820"/>
<RECOGOUT>
  <SHYPO RANK="1" SCORE="-6888.637695" GRAM="0">
    <WHYPO WORD="silB" CLASSID="39" PHONE="silB" CM="1.000"/>
    <WHYPO WORD="上着" CLASSID="0" PHONE="u w a g i" CM="1.000"/>
    <WHYPO WORD="を" CLASSID="35" PHONE="o" CM="1.000"/>
    <WHYPO WORD="白" CLASSID="2" PHONE="sh i r o" CM="0.988"/>
    <WHYPO WORD="に" CLASSID="37" PHONE="n i" CM="1.000"/>
    <WHYPO WORD="して" CLASSID="27" PHONE="sh i t e" CM="1.000"/>
    <WHYPO WORD="下さい" CLASSID="28" PHONE="k u d a s a i" CM="1.000"/>
    <WHYPO WORD="silE" CLASSID="40" PHONE="silE" CM="1.000"/>
  </SHYPO>
</RECOGOUT>
```

図 4 認識サーバから送信される情報の例

クライアントからはコマンドを送ることができる．jcontrol のプロンプトに pause と入力すると，Julius を一時停止できる．一時停止した Julius は resume で再開できる．他に，文法や辞書を Julius に送ったり，複数の文法を用いた同時認識を制御するなど，多くの機能が提供されている．

4 提案システム

以下著者らが開発したリアルタイム病気診断システムについて述べる．

4.1 開発環境

- Windows 7 64bit
- JDK 6.0
- Eclipse 3.7
- Seasar2[6]
- MySQL 5.5
- Julius 4.1.5

4.2 システム概要

本システムの処理プロセスは図 5 に示す．

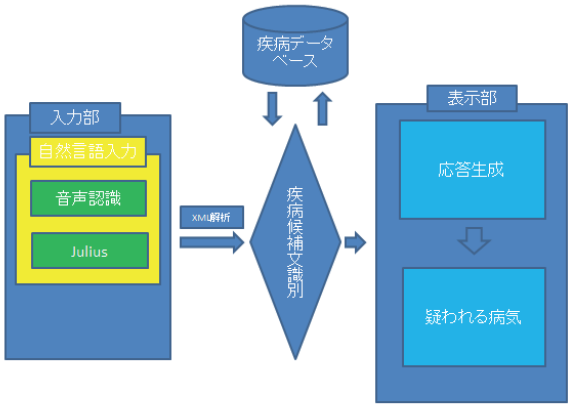


図 5 処理プロセスの例

本システムは一問一答形式(例:発熱して，咳も出ると言うと．システムがリアルタイムであなたが風を引いたというメッセージが出てきた．)の音声認識インタフェースを持ち，データベースと連携し，リアルタイムな病気診断サービスを提供する（図 5 メイン画面）．

本システムの音声インタフェースは，これまで述べてきた統計的言語モデルを用いる大語彙連続音声認識を基盤に開発された．統計的言語モデルを使うことで，人手では記述を網羅することが困難な文法ベースの音声認識と比較して，言い回しなどの文章表現の多様性を吸収しながら柔軟に認識できるのは前述の通りである．音声インタフェースのプログラム構成は音声認識部とその認識結果を用いた応答生成，応答音声を発声する応答合成部から成る．以下では，音声認識と応答生成の詳細を述べる．

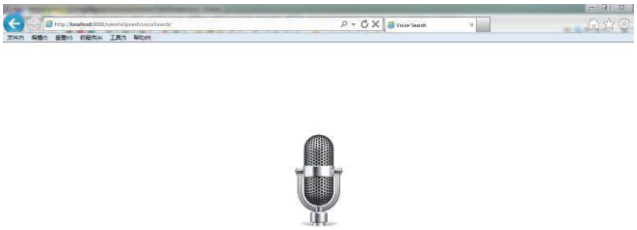


図 5 メイン画面

4.2.1 音声認識部

大語彙連続音声認識プログラムには Julius を用い，認識対象のタスクに適した言語モデルを得るため，前述の手順に従ってタスク適応 N-gram 言語モデルを構築した．以下に手順を述べる．

まず，下記の 2 種類のテキストをモデル学習用に収集し，各々について 2-gram 及び逆向き 3-gram モデルを作成した．

1. Web 検索を用いて収集した goo ヘルスケアホームページ内の Web ページテキスト
2. 人手で収集した本システムを想定した質問文テキスト

各モデルの語彙は，「1. Web ページテキスト」に関しては出現頻度の高いものから上位 4 万語であり，「2. 想定質問文テキスト」では出現したすべての単語である．

次に生成した各モデルを N-gram モデルの融合ツールを用いて融合した．融合重み λf, λg には，0.5 を用いた（融合割合 1:1）．以降，このモデルを融合言語モデルと呼ぶ．

続けて，融合言語モデルにネットワーク記述文法の単語間制約を適用して Ngram 確率を強化した文法適用言語モデルを作成した．適用に用いた文法の異なり単語数は 441 である．文法では定義されているが融合言語モデルには含まれない 43 単語に関して，単語辞書中の未知語クラスのエントリに対して 43 単語の出力表記と読み（音素記号）を与えることで文法適用言語モデルに追加した．

4.2.2 応答生成部

音声インタフェースの応答生成は，ユーザの一発話ごとの音声認識結果をもとに，あらかじめ用意された応答候補文から一文を選択し，応答として返すものである．今回，システムの設計に際して，ユーザに時間遅れ無しに応答が返せることを重視し，簡潔なプログラム構成になるように心がけた．応答内容に関しては，ユーザの関心を維持することを重視して，なんらかの情報を持った応答を返すことを主眼とした．ユーザがシステムに対する関心を失うとフィールドテストやデータ収集には好ましくないためである．具体的には，「わかりません」や「もう一度お願いします」などのインタラクションを阻害する応答を極力生成しないようにするとともに，質問に関連した話題を広くカバーできるよう応答候補文を作成した．

以下では，応答生成の具体的な手順について説明する．図 6 は，あらかじめ用意した応答候補文の例である．

図 6　応答候補文の例

これら候補からの応答の選択には，音声認識結果とあらかじめデータベースに登録した用例テキストとの形態素マッチングによる．用例テキストは，本システムの過去ログや生駒市役所の業務記録などから作成した質問

文の形態素列である．現在，309 文の用例テキストが登録されている．

すべての用例テキストには，前述の応答候補文の中からその質問の応答として相応しいものへの対応が定義されている．具体的には，図 7 に示す．

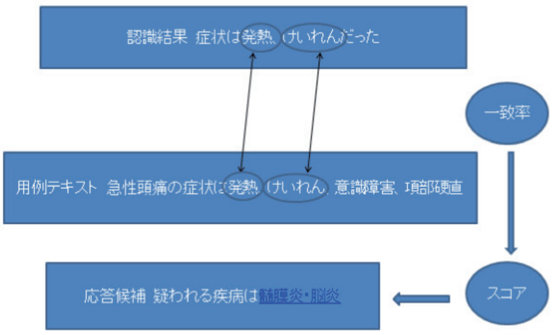


図 7 用例ベースのスコア計算

用例テキストとの形態素の一致数を求め，用例テキストの単語数で割り一致率を求める．その一致率をスコアとする．このとき，カウントに用いるのは質問の意図理解に重要な名詞，動詞，形容詞などの自立語形態素のみである．すべての用例テキストに対して一致率を算出し，最も高いスコアに対応する応答候補文を応答とする．複数の候補のスコアが同じ場合は，応答はそれら候補の中からランダムに選択される．本手法では，キーワードリストを必要とする応答生成である．

4.2.3 XML 解析部

図 4 に示す，認識サーバから送信される情報が図 8 に示す解析部による，前述 4.2.2 の応答生成部に渡す．

```
public class JuliusOutputParser {
    String recognition_output;
    public JuliusOutputParser()
    {
            recognition_output="";
    }
    public Boolean addLine(String line)
    {
            if(line.equals("</RECOGOUT>"))
            {
                    return true;
            }
            line=line.trim();
            if(line.startsWith("<WHYPO"))
            {
               int pos=line.indexOf('"', 13);
              String word=line.substring(13,pos);
                    recognition_output+=word+" ";
            }
            return false;
    }
    public String getRecognitionOutput()
      {
            String ret=recognition_output;
            recognition_output="";
            ret=ret.replaceAll("<s>", "");
                ret=ret.replaceAll("</s>", "");
            ret=ret.trim();
            return ret;
    }
}
```

図 8 XML 解析部

4.2.4 Julius への通信部

Julius は-module を指定することでモジュールモードで起動する．モジュールモードでは，起動後，クライアントからの TCP/IP 接続待ちとなる．デフォルトのポート番号は 10500 であるが，-module portnum のようにポート番号を変更することができる．

クライアントからの接続を受けると，Julius は音声認識可能な状態となる．音声入力に対して音声認識を行いながら，クライアントへ認識結果を送信する．また，音声入力の開始・終了などのイベントを随時送出する．クライアントからは，認識の一時停止や再開といった動作制御，認識用文法の追加や削除などが行える．また，複数モデルによるマルチデコーディング時は，認識処理インスタンスごとの一時停止なども行える．

クライアントとの接続は一対一を想定しており，複数クライアントの同時接続には対応していない．クライアントとのネットワークソケットが切断されると，Julius は認識を停止して，次のクライアントが接続するまで待ち状態となる（図 9 クライアント通信部）．

```
public class JuliusClient implements Runnable {
  private String hostname;
  private int port;
  public JuliusOutputParser parser;
  private BufferedWriter writer;
  private Object mutex;
  public String ret;
  public String getRet() {
    return ret;
  }
  public JuliusClient(String hostname, int port) {
    this.hostname = hostname;
    this.port = port;
    parser=new JuliusOutputParser();
    writer=null;
    mutex=new Object();
  }

  @Override
  public void run()
  {
    Socket connection=null;
    int num_tries=0;
    while(connection==null && num_tries<3)
    {
      try{
        connection=new Socket(hostname,port);
      }catch(Exception e)
      {
        connection=null;
      }
      num_tries++;
      if(connection==null)
        try{
          Thread.sleep(1000);
        }catch(Exception e){}
    }

    if(connection==null)
    {
      System.out.println("Failed to connect to Julius. Please restart the program.");
      return;
    }

    BufferedReader stream=null;
```

```
try{
      stream=newBufferedReader(new InputStreamReader(connection.getInputStream(),"utf8"));
      synchronized (mutex) {
        writer=new BufferedWriter(new OutputStreamWriter(connection.getOutputStream()));
      }
    }catch(Exception e){
      System.out.println("Cannot connect:"+e);
      return;
    }
    while(connection.isConnected())
    {
      String line;
      try {
        line=stream.readLine();
      } catch (IOException e) {
        System.out.println("Error reading socket: "+e);
        break;
      }
      if(line!=null && parser.addLine(line))
      {
        ret=parser.getRecognitionOutput();
        TalkManager.addOnlineTalkContent(ret);
      }else{
        break;
      }
    }

    try {
      stream.close();
      stream=null;
      synchronized (mutex) {
        writer.close();
        writer=null;
      }
    } catch (IOException e) {}
    return;
  }
}
```

図 9 クライアント通信部

以下は Julius をモジュールモードで起動することのコマンド

bin\julius-4.2.1 -C fast.jconf -charconv sjis utf-8 -module

5 データベース

本システムのデータベースは MySQL を採用する．以下は SQL 構築文（図 10）を示す．

```
CREATE DATABASE IF NOT EXISTS sdvoice DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;

CREATETABLEIFNOTEXISTS sdvoice.disease_symptom(
disease_id INT AUTO_INCREMENT NOT NULL PRIMARY KEY COMMENT '疾病 ID',
disease_name VARCHAR(30) NOT NULL COMMENT '疾病名',
disease_symptom VARCHAR(1024) NOT NULL COMMENT '症状'
) ENGINE=InnoDB,COMMENT='疾病設定';
```

図 10 SQL 構築文

テーブルの構造は以下表 1 示す．

```
+----------------+---------------+------+-----+---------+----------------
| Field          | Type          | Null | Key | Default | Extra          |
+----------------+---------------+------+-----+---------+----------------
| disease_id     | int(11)       | NO   | PRI | NULL    | auto_increment |
| disease_name   | varchar(30)   | NO   |     | NULL    |                |
| disease_symptom | varchar(1024) | NO   |     | NULL    |                |
+----------------+---------------+------+-----+---------+----------------
```

表 1 候補文テーブルの構造

画面上に一覧(図 11)，編集(図 12)，追加(図 13)，削除することができる．

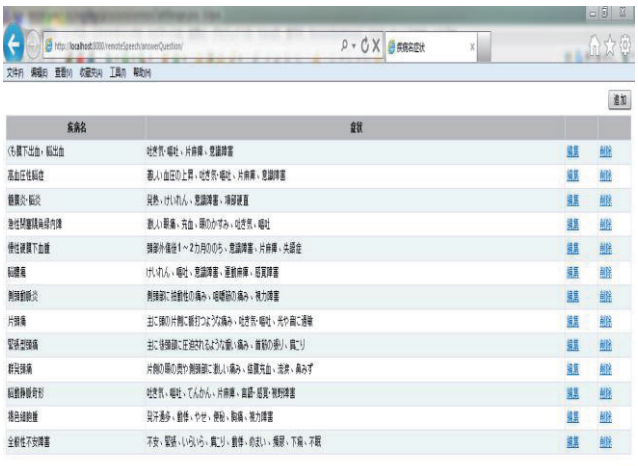


図 11 一覧

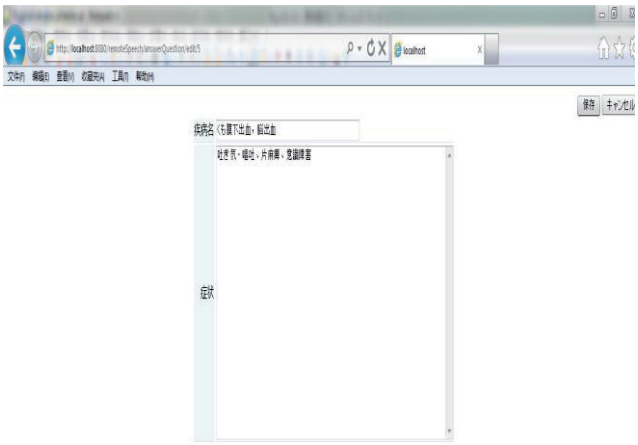


図 12 編集

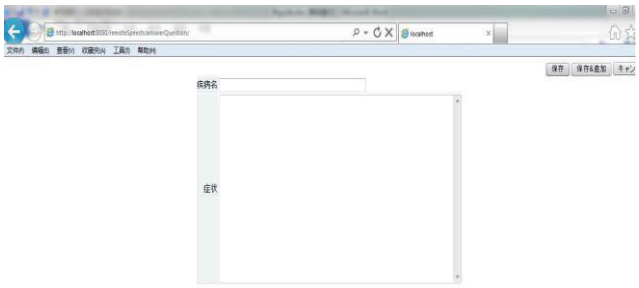


図 13 追加

6 評価

以下では，評価目的，方法，結果等について述べる．

6.1 評価目的

ユーザの発話に対して，システムがどの程度正しく応答できるかを調べる．

6.2 実験システム

前述のリアルタイム病気診断システム．

6.3 被験者

アルバイト先の 15 名会社員．

6.4 評価方法

被験者に音声入力によってリアルタイム病気診断システムを与えてもらった．応答正確率をアンケートにより集計した．

6.5 評価結果

応答正解率は，満足な応答結果が得られた応答文の79%割合である．応答内容が満足なものかは人の主観により判別した．

7 おわりに

本システムの応答正確率はまだ高くなっかた．今後，音声認識の精度向上を効率的に実現できるため，システムの運用で発話ログを収集し，言語モデルの再学習に利用することができるも検討する．そして，本システムの応答形式はテキストだけ表示する．TTS（Text To Speech）で音声応答も期待する．

謝辞

実験に協力してくださった方々にこの場を借りて感謝の意を示します．

参考文献

[1] Juliusライブラリ, http://julius.sourceforge.jp/ (2012/11).
[2] 中川聖一: 確率モデルによる音声認識, コロナ社, 1988.
[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-D-II, no.12, pp.2517- 2525, 2000.
[4] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. Woodland: The HTK Book (for HTK Version 3.2.1), Cambridge University Engineering Department, 2002.
[5] F. Jelinek: “Self-Organized Language Modeling for Speech Recognition,” Language Processing for Speech Recognition, pp.450- 506, Mercel Dekker, Inc., 1990.
[3] Seasar2 ライブラリ,http://www.seasar.org/index.html (2012/11).