音声言語情報処理 5－6
（1995．2．4）

クライアント・サーバ構成のHMM-LR連続音声認識システムとその応用

山田智一　野田喜昭　井本貴之　嵯峨山茂樹

NTTヒューマンインタフェース研究所

〒238-03 神奈川県横須賀市武1-2356
tomokazu@nttspch.hil.ntt.jp

クライアント・サーバ構成により実現したHMM-LR法に基づく連続音声認識システムとその応用について述べる。近年の計算機の処理能力向上などにより、音声認識はソフトウェアのみによって実現できる可能性が高まりつつある。このような状況を積極的に採り入れ、ソフトウェアによる音声認識システムを構築した。また、効率的な開発・利用環境を目指し、クライアント・サーバ型のシステムとした。本システムの構成に関する基本的な考え方、導入した要素技術、インタフェース等の報告に加え、実際にこのシステムを利用したマルチモーダル入力ツールの例として、作図ソフトへの応用を報告する。

An HMM-LR Continuous Speech Recognition System based on a Client Server System Architecture and its Applications

Tomokazu YAMADA　Yoshiaki NODA　Takashi IMOTO
Shigeki SAGAYAMA

NTT Human Interface Labolatories

1-2356, Take, Yokosuka-Shi, Kanagawa, 238-03 Japan

This paper describes an HMM-LR continuous speech recognition system based on a client server system architecture and its applications. Rapidly increasing computational power of standard workstations have made speech recognition possible to be realized by pure software engineering. Under such circumstances, we construct a software-based speech recognition system. The system is also based on a client server system architecture, which is used to get an effective environment for both application's development and utilization. Basic motivations for the system architecture, introduced techniques, interfaces and an implementation for a drawing application are described.

1 はじめに

従来、音声認識技術は専用のハードウェア装置などによって提供されることが多かった。しかし近年の汎用ワークステーション等の処理能力向上により、タスクによってはソフトウェアのみでも実時間処理できる段階に近づいている。ソフトウェアのみによって実現することにより、専用のハードウェア等を作成していた場合に比べ、開発に要する時間は大幅に短縮され、開発コストも削減できる。このような理由から、ソフトウェアによる実現形態は今後の音声認識技術の実用化において一つの主流になっていくと考えられる。

このような状況を踏まえ、我々はソフトウェアにより実現した音声認識サーバ・システムを構築した。HMM-LR法に基づく音素同期型のプログラムをクライアント・サーバ・システム(以下CSS)として再構成し、具体的なアプリケーション部分とサーバ部分とを完全に分離して、アプリケーションに依存しない音声認識サーバとして汎用性を高めた。CSS型として構成することで、音声認識機能をブラックボックスとして扱うことができる。これにより、音声認識アプリケーションの作成者は特に音声認識に関する専門知識を持つ必要はなくなり、アプリケーションのタスクに特化したヒューマンインタフェース等の開発に専念することができる。また、音声認識サーバとアプリケーションとは同時に独立して開発することができ、効率的である。

本稿では、本システムの構成、要素技術などについて述べた後、音声認識アプリケーション作成の例として、本システムの作図ソフトへの応用例を示す。

2 音声認識サーバのシステム構成

2.1 ソフトウェアとCSSによる実現

本システムの目標の一つは、音声認識の専門知識がなくても、容易に音声認識機能を持ったアプリケーションが開発できる環境を提供することである。そこでまず本システムの構築にあたり、ソフトウェアのみによって実現し、CSS型として構成することとした。このような構成をとることの利点には以下のようなものがあげられる。

ソフトウェアによる実現

- 開発サイクルの短縮
- 開発コストの削減
- 変更・修正などの容易さ

クライアント・サーバ・システムとしての実現

- アプリケーションと音声認識サーバの独立した開発が可能
- 音声認識サーバに修正やバージョンアップが生じた場合、アプリケーションには一切変更の必要がなく、資産が無駄にならない
- 一つの音声認識サーバで同時に複数のクライアントに対応できる(図1)

音声認識をCSS型として実現することは新しいことではなく、近年、広く行なわれるようになってきた[1][2]。これは現状でソフトウェアにより音声認識を実現するには、まだ比較的高速(かつ高価)なワークステーションが必要となるため、これをサーバに利用して複数クライアントに対応できるようにし、クライアントには低価格のワークステーションやパソコンを利用してアプリケーション当たりの単価をある程度抑えられることが理由としてあげられる。また、マルチメディア環境下で音声を入力手段の一つとして提供する場合などには、音声認識部がモジュール化されており、他のアプリケーションから容易に利用できることが必要で、このような点でも有効な構成方法であると考えられる。高速WS1台+複数のPC(または安価なWS)というシステム構成は、現時点での妥当な実現形態の一つだと考えられる。

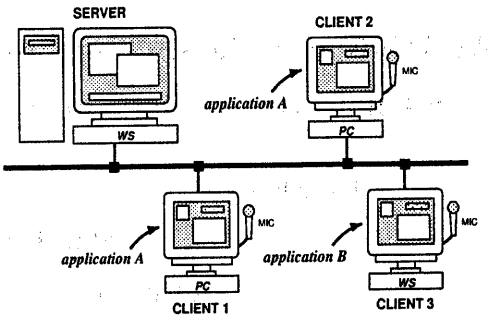

図1 CSS構成下での利用形態

2.2 音声認識API

このように、CSS型のソフトウェア・システムとして音声認識システムを構築するわけだが、効率的な開発環境を提供するには、どのようなインタフェースで音声認識アプリケーションと音声認識サーバを結ぶかが重要である。我々は、X windowにおける

Xlibに対応した音声認識のアプリケーション・プログラム・インタフェース(API)をC言語ライブラリとして用意し、これを介してアプリケーションの作成を行なう環境を提供する。音声認識APIを介したアプリケーションの実装は、以下のような利点を持つ。

- 音声認識機能をブラックボックスとして扱える
- プロトコルを隠蔽でき、その変更に伴う修正を吸収することができる

これにより、音声認識に関する専門知識を持たない人でも容易に音声認識機能を持ったアプリケーションが作成できるようになる。また、プロトコル等に変更が生じても、ライブラリを介して作成されていれば、アプリケーションには一切変更の必要がない。将来的にPC等の処理能力が向上すれば、ローカルに接続することで同一計算機上でアプリケーションとサーバを実行したり、CSS型ではなく一つのプログラムとして構成することも考えられる。このような場合にも、音声認識APIを利用していれば、音声認識処理部との接続形態を意識する必要はなく、同じようにアプリケーションを作成できる。

クライアントからサーバへ送る、認識処理依頼、単語リストや文法の設定、雑音・回線適応化、ビーム幅の設定など、すべての要求をAPIとして用意している。

2.3 要素技術の統合

本システムの元になったHMM-LRに基づく音素同期型のシステムは、音素環境独立のHMMを用いた不特定話者用の連続音声認識システムで、数万単語の処理が可能である[3]。今回、CSS型として再構築するに際し、利用環境まで含めて考慮した上で、以下に述べる種々の認識アルゴリズム、要素技術を統合した。

- **音素環境依存HMM** [4]
  認識性能向上のために音素環境を考慮したHMM(triphone)を利用する。モデル、状態、基底分布、特徴量の4階層を共有化し、計算量の増加を抑えている。
- **環境依存HMM用LRパーザ** [5]
  環境独立の音素を終端記号とするLRテーブルを利用し、最後に予測された音素ではなく、それより前に予測された音素を中心音素として設定する簡単な方法により、音素環境依存のHMMを扱うことができる(3章参照)。
- **オン・デマンド処理による高速化**
  4.1節で述べるオン・デマンド処理により、音声の入力の開始と同時に認識処理を始めることを可能としている。従来よりも音声を入力し始めてから結果が得られるまでにかかる時間を短縮することができる。
- **前向き尤度によるヒューリスティック関数を利用したビーム探索** [6]
  従来利用していた後向き尤度によるヒューリスティック関数を利用したビーム探索は、音声の入力が終了しないと行なうことができなかった。前向き尤度によるヒューリスティック関数を利用したビーム探索は、これと等価な認識性能を得ることができ、加えて処理時間は削減される。
- **単語リストによる語彙の指定**
  本システムはHMM-LR法に基づいているので、文脈自由文法(CFG)による語彙の定義が可能である。しかし実際に音声認識が用いられる場面を考えると、認識性能と処理時間の制約から、現時点では単語認識に利用される機会が多いと考えられ、この場合の語彙定義をCFGで行なうのは、利用者には煩わしい。そこで、単語認識を行なう場合には、簡単な単語のリストで語彙の設定ができるインタフェースを採用した。単語リストは認識対象語彙の発音をローマ字、または仮名で記すだけの簡単なものである。サーバ内部で文脈自由文法に変換され、LRテーブルが作成される(3.1節参照)。
- **雑音適応化** [7]
  図1のように複数のクライアントから音声認識サーバを利用する場合など、そのクライアントの設置場所毎に環境雑音が違い、認識性能にも悪影響を及ぼす。クライアントとサーバの接続後、数秒間クライアント環境の雑音を取り込み、NOVO合成法によって雑音に対するモデルの適応化を行ない、処理性能を改善する。
- **話者適応化** [8]
  クライアント毎に利用する話者も異なる。話者に対する適応化によっても、認識性能を向上させることができる。ここではMAP/VFSによる話者適応化で、クライアントとサーバの接続後、数単語を発声してもらうことで認識性能を改善できる。

これらの要素技術のいくつかについて、以降の章で詳述する。

3 音素環境依存HMMに対応したLRパーザ

音素環境依存のHMMを扱うためのLRパーザの実現方法について説明する。

3.1 単語リストによる設定に伴う制約

2.3節で述べたように、本システムは単語認識用の語彙設定方法として簡単な単語リストによる設定方法を用意している。単語リストは、単語の発声内容をローマ字、または仮名で記したものをリストにしただけの単純なもので、サーバに送られるとほぼ1対1で対応する文脈自由文法に変換され(図2参照)、それからLRテーブルを構築する。このような理由から、アプリケーションによっては頻繁にLRテーブルの再構築が行なわれるので、これに時間のかかる方法では問題がある。

単語リストの例

```
  ...
いけぶくろ                    | 池袋
ちゅううおおく　やえす        | 中央区八重洲
  ...
```

変換された文脈自由文法の例

```
S → word
  ...
word → i k e b u k u r o
word → ch u u o o k u pause y a e s u
  ...
```

図2 単語リストからCFGへの変換例

環境依存モデルをHMM-LR法に基づくシステムで扱うための方法は、すでにいくつか提案されているが[9] [10] [11] [12] [13]、これらのうち、LRテーブル自体を音素環境依存のモデルに対応したものに変換する方法は、一般にテーブル作成に時間がかかり、本システムで利用するには問題がある。そこで、本システムではテーブルは音素環境独立のモデル用のものを利用し、パーザレベルで対処する方法をとることにした。パーザレベルで対処する方法として文献[10]の方法があるが、この方法は、後続音素を知るための先読み処理に冗長なパージング動作があり効率的でない。ここではその冗長なパージング動作を省き、実現も容易な、以下のような新しい方法を導入した。

3.2 環境依存モデルの決定方法

評価すべき環境依存モデルの決定方法を図3で説明する。これは、あるLRテーブルに基づいて、最初の4音素までの予測で生成される候補を木構造で示した例である。各パス上に示されているのがパーザによって予測された音素で、/#/は文頭の無音を表す。本システムは音素同期で処理を行ない、すべての候補に対して1音素予測するたびに、HMMに基づく尤度によってビーム探索による候補の枝刈りが行なわれる。図の一番下に、各ステップで評価を行なうtriphoneモデルを、本システムでの方法と、永井らによる文献[10]の方法の場合について示す。中心音素Cの先行音素をP、後続音素をSとして、[PCS]のように示している。

木の一番下のパスにある#-a-k-aという音素列からなる候補を例に説明する。これは、ステップ1で/#/を、ステップ2で/a/を、ステップ3で/k/を、というように音素が予測された候補である。音素環境独立のHMMを用いた場合には、各ステップで予測された音素に対応するHMMによって尤度計算を行ない、ビーム探索に基づく候補の枝刈りを行なった。これに対し、永井らの方法では、例えばステップ3では、予測された音素/k/を中心音素とし、先読みによって後続音素が/a/であることを調べて、評価すべきtriphoneモデルを[$_{a}$k$_{a}$]であると特定した(すでに予測されている先行音素/a/を知るには、予測された音素を逐次格納しておくスタックを用意すればよい)。先読み操作では、パーザが使うのと同じLRテーブルを用いて、ほぼパージングと同様の動作によって後続音素/a/を調べている。しかしこの情報はtriphoneモデルを特定するのに用いた後に捨てられ、ステップ4で新たにパーザによって/a/が予測される。このため、同一の音素をそれぞれ2回予測するような動作が行なわれることになり、効率的で

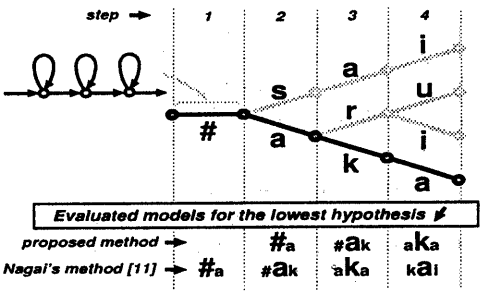

図3 パーザの予測音素と評価すべき音素環境依存HMMの関係

ない。

この点を改善するため、我々は先読みは行なわず、最後に予測された音素を含むすでに予測された音素の系列から、評価すべきモデルを特定することにした。例えばステップ3で/k/が予測されると、これを右側の環境とし、ステップ2で予測された/a/を中心音素とし、ステップ1で予測された/#/を左側の環境とするtriphoneモデル[#a_k]を評価すべきモデルとして特定する。永井らの方法と比べると、同一のtriphoneモデルを評価する時期が1ステップずつ遅れた形になっている。ステップ1では音素仮説を生成するのみで、尤度計算等は行なわれない。一見、仮説が多く生成され過ぎるように思われるが、永井らの方法でも、先読みをした時点で部分仮説が生成されるので、効率にはほとんど差がない。

3.3 LRテーブルで環境依存性に対処する方法との比較

一般にLRテーブルの変更によって音素環境依存のHMMに対処する場合は、図3の例と対応づけた図4で説明すると、ステップ1で[#_a]、ステップ2で[#a_k]、ステップ3で[_ak_a]の、各triphoneモデルに対応する終端記号が予測される。終端記号自体が異なるので、例えば[#_a]と[#_s]はステップ1から別の候補として生成される。本システムでの方法では、音響尤度評価の計算開始時期は遅れるが、その後のステップ2、ステップ3、ステップ4で生成される候補の数は、テーブルで対処する場合のステップ1、ステップ2、ステップ3で生成される候補の数と等しく、ビーム探索の効率に差はない。

ここではtriphoneの場合で説明したが、我々の提案する方法ではbiphoneの場合でも、またtriphoneより長い環境を考慮に入れたモデルを扱う場合でも、ビーム探索、及び仮説の枝刈りを開始する時期を遅らせるステップを変更すればよく、この処理は一般

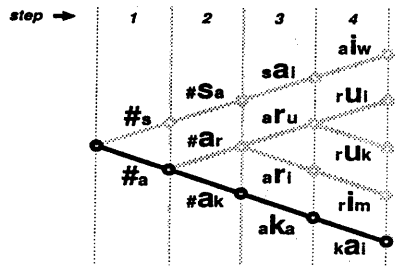

図4 LRテーブルで音素環境依存HMMに対処する方法での候補の生成例

化できる。特にtriphoneよりも長い環境を考慮に入れたモデルを扱う場合には、LRテーブルの変更によって対処する方法では状態数が爆発するので現実的には困難であり、この提案手法が有効である。

本システムでの方法は、認識時にこれらの処理を行なう分、処理速度が低下すると言えるが、実際に実験を行なったところ、認識処理中で評価すべき音響モデルを特定するのに要した時間は、認識処理全体の0.1%未満に過ぎず、問題にならないことが分かった。

4 システムの高速化

本システムを高速化するために採用した2つの項目について説明する。

4.1 オン・デマンド処理

従来のHMM-LR法に基づく音声認識システムは、例えば音声認識サーバの元となったシステム[3]では図5(a)に示されるように、音声データ切り出し→LPC分析→HMMの出力確率計算→後向き尤度計算、と順に処理を行なってテーブル等にデータを保持した後、パーザで音素を予測してtrellis計算と照合を行なう、という形で処理が進んでいった。しかしこれでは音声の入力が終了するまでは、認識の処理が開始できない。我々は、音素同期型のHMM-LR法でも、音声の入力が終了するのを待たずに処理を開始できるように、以下のような方法を実装した(図5(b)参照)。

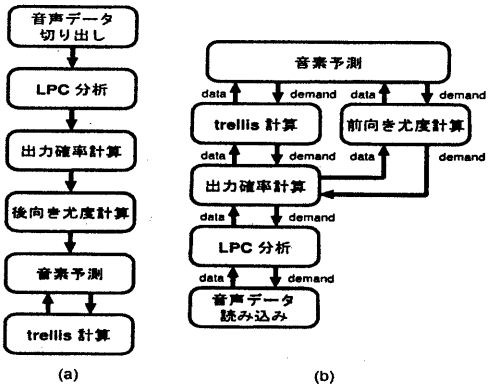

図5 オン・デマンド処理方式と従来方式の処理手順の違い

認識動作はパージングから始まり、音素を予測したところで後述する前向き尤度によるヒューリスティッ

クの値を得ようとする。これが用意されていなければ、trellis計算を実行しようとする。ここで必要な出力確率をテーブルに参照しに行き、もしテーブルに値が入っていなければ、LPC分析された特徴量を読み出そうとする。特徴量も求められていなければ、音声データを読みに行く。このように従来方法でのデータ作成の順番とほぼ逆順に必要なデータを必要となった時点で要求するような、オン・デマンド型の処理を導入した。これにより、音声の発声中も利用可能な音声部分から順に処理を始めることができ、発声終了から結果を得るまでの時間を短縮することが可能となった。

この機能をクライアント・サーバ間で実行するため、音声データはクライアント側でまとめて一度に送るのではなく、細かなブロック単位(可変長)で送る方式とした。

4.2 前向き尤度によるヒューリスティック関数の利用

音声認識サーバの元となったプログラムは、入力音声の終端を検出してから、まず音声区間全体を用いて後向き尤度によるヒューリスティック関数を計算し、これを用いてビーム探索の効率を上げていた[3]。これは部分仮説nに対するトレリス計算によって得られた尤度関数$g_n(t)$と音声終端からの後向きトレリス計算から得られた尤度関数$\hat{h}(t)$から次式によって得られる値をその仮説の評価値として利用するものである。

$$\hat{f}_n = \max_t \{g_n(t) + \hat{h}(t)\} \qquad (1)$$

しかし今回のように音声の終端を待たずに処理を開始する場合、このような後向き尤度による評価値を利用することができない。そこで、以下に述べるような前向き尤度によるヒューリスティック関数を利用したビーム探索を行なう方法を導入した。

式(1)における$\hat{h}(t)$は、何らかのヒューリスティックに基づいて、音声の終端から後向きに計算して求められる。これと同様のヒューリスティックを用いて、音声の始端から前向きに計算して求めた尤度関数を$\hat{g}(t)$とすると、次式の関係がある。

$$\hat{g}(t) + \hat{h}(t) = \hat{h}(0) = \text{一定} \qquad (2)$$

式(1)と式(2)から、次式が得られる。

$$\hat{f}_n = \max_t \{g_n(t) - \hat{g}(t) + \hat{h}(0)\} \qquad (3)$$

ここで$\hat{h}(0)$は、仮説n、時刻tのいずれにも依存しない定数であり、$\hat{f}_n$によって評価値の比較を行なう場合には、$\hat{h}(0)$を除いた次式を用いても同様の探索結果が得られる。

$$\tilde{f}_n = \max_t \{g_n(t) - \hat{g}(t)\} \qquad (4)$$

よって，前向き尤度$\hat{g}(t)$をヒューリスティック関数として用いる評価値計算方法は，式(1)において$\hat{h}(t)$を全ての部分仮説に共通に用いる場合と等価である(詳細は文献[6]参照)。

前向き尤度によるヒューリスティック関数の値には、既に仮説の尤度計算で得られている値を利用することもできるので、設定によっては後向き尤度による場合よりもヒューリスティック関数の計算に要する時間は少なくてすむ。この方法により、オン・デマンド処理でも従来と同等の効率的なビーム探索を行なうことができる。

5 認識性能

現在のシステムの認識性能を把握するために行なった、単語、及び文節認識実験の結果を示す。不特定話者用HMMの学習には、ATRの5240単語の偶数番目の単語と音韻バランス216単語を16名分用いた。この学習用音声データから、モデルと状態の2階層で共有構造を持つ音素環境依存型HMM(triphone、環境依存クラスタ数1497)と、音素環境独立型HMM(26音素)を作成し、実験を行なった。2階層共有モデルの認識性能は、3階層、4階層共有のモデルとほぼ等しい[4]。環境独立モデルは、比較のための参考モデルとして用意した。パラメータは16次のケプストラム、16次のΔケプストラム、Δパワー(全部で33次元)である。

以下の単語、及び文節認識実験は、話者1名(MMS)のデータを用いて行なった。話者MMSは、学習用データの16名に含まれていない。

5.1 単語認識実験

学習用データに用いなかったATRの5240単語の奇数番目の単語のうち、データ番号の末尾が1の524単語を用いて、不特定話者単語認識実験を行なった。表1に、評価データに対する、ビーム幅と文節認識率の関係を示す。

5.2 文節認識実験

国際会議の問い合わせタスクに対する不特定話者文節認識実験の結果を示す。文節文法は単語数1076、文法数2679で、終端記号は環境独立HMMとして用意された26音素で構成されている。評価用データには、同じ国際会議の問い合わせタスクの278文節(DSB3)

表1 音素環境独立、及び音素環境依存モデルを用いた場合の単語認識率[%]

ビーム幅	環境依存モデル	環境独立モデル
10	92.2 (93.3)	96.4 (97.9)
20	96.9 (98.5)	97.1 (98.9)
50	98.1 (99.8)	97.5 (99.2)
100	98.1 (99.8)	97.5 (99.2)
200	98.1 (99.8)	97.3 (99.2)

(話者MMS. ()内は1～3位の累積認識率)

表2 音素環境独立、及び音素環境依存モデルを用いた場合の文節認識率[%]

ビーム幅	環境依存モデル	環境独立モデル
10	57.6 (66.5)	69.8 (84.5)
20	73.0 (85.6)	73.4 (91.7)
50	81.3 (95.7)	74.5 (95.0)
100	83.1 (97.5)	74.5 (95.7)
200	83.8 (98.6)	74.5 (95.3)

(話者MMS. ()内は1～3位の累積認識率)

を用いた。上述の文節文法はこの評価データを生成可能である。表2に、評価データに対する、ビーム幅と文節認識率の関係を示す。

5.3 性能に対する考察

両実験とも、尤度計算を開始する時期を遅らせる新しいパーザの実現法でも、ビーム幅を広くする必要のないことが確認できた。524単語認識に比べると、文節認識実験はかなり難しいタスクであるが、環境依存モデルと4.2節で述べた前向き尤度を用いた効率的な探索手法により、ビーム幅を極端に広げなくても、ある程度の認識性能が得られることが分かった(文節認識実験では、ビーム幅200で認識率はほぼ飽和していた)。

6 音声認識サーバを用いた応用例

本システムを用いて簡単に作成できるアプリケーションの例として、音声認識によるショートカット機能を持つ作図ソフトを作成した。このような応用を例として取り上げた理由は、作図ソフトにおいて音声認識機能を併用すると、作業効率を向上させられることがすでに示されており[14]、音声認識アプリケーションとして好例であること、また作図ツールとしてpdsのソースコードが簡単に入手できることである。音声認識APIを介することにより、X Window上の作図ソフトであるtgifに、極めて短期間で音声認識機能を付加することができた。図6に、音声認識機能を付加した作図ソフトの構成を示す。

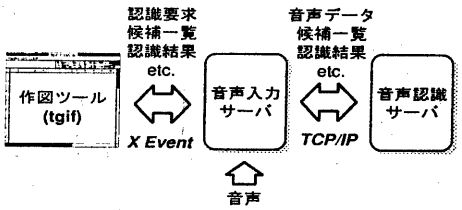

図6 音声ショートカット機能を持った作図ソフトの構成

音声認識サーバは、アプリケーション側での音声の取り込み、及びサーバへの送信を仮定している。しかし音声入出力機構は端末に依存し、作図ソフト自体に組み込むことは機種依存になり過ぎ、汎用性を欠く。むしろキーボードやマウスと同じく、端末の持つ入力デバイスとして考えた方がよい。そこで、まずアプリケーションと音声認識サーバとの間に介在して、各端末毎に音声入力機能を提供する音声入力サーバを作成した。音声入力サーバの機能は以下のようなものである。

- アプリケーション・プログラムからの要求に従い単語リストを音声認識サーバに中継する
- アプリケーション・プログラムからの音声入力要求があり次第、ローカルに接続してある音声入力デバイスから音声データを取り込み、これを逐次、音声認識サーバに送る
- 音声認識サーバが送り返してきた結果文字列の中から第一位候補の文字列を選び、Xのイベントの形で(音声入力要求を送ってきた)アプリケーションに送信する

音声入力サーバを用いるために、アプリケーション(tgif)には、通常のキーボード入力やマウスのクリックのようなイベントの他に、音声入力サーバからのクライアント間メッセージ・イベントも受信し、内容に従って処理を進めるように変更を加えた。

あらかじめアプリケーションが持っていたショートカット機能を利用し、その各項目に対し、音声認識機能による指定が可能となるよう実装した。このことと音声認識APIを利用したことにより、極めて短期間に音声認識機能の組み込みを完了した。実際には音声入力サーバの作成に1週間、tgifアプリケーションの改造には数日を要したのみである。

どのショートカット機能にどのような単語を割り当てるかは、Xのリソースの機能を利用して、ローマ字で指定できるようにした。これにより、単語リストの指定と同じように認識語彙が設定でき、作図ソフトの使用者に応じたカスタマイズが簡単に行なえる。指定可能な機能数は106個である(各機能に複数の語彙を設定可能)。

使用者が認識を意図しないような発声にも音声認識機能が働くと、作図の妨げとなる。また、作図ソフトの性格上、マウスは作図作業に占有される。そこで、キーボード上の特定のキーを押すことを、音声入力要求のトリガーとした。

実際に音声ショートカット機能を利用してみると、例えば複数図形を中心線に合わせて揃える機能など、そう頻繁には使わないがたまに必要になる機能などの利用において、キーボード上の数文字からなる(通常の)ショートカットによる指定に比べ、意味のある単語をキーワードにできる音声ショートカットが有効であった。また、先に述べたように、音声入力モードを持たない場合に比べ、状況に応じてマウスによるメニュー選択、音声ショートカットによるコマンド実行などを組み合わせることで、作業効率を向上させられる可能性が示されている[14]。このような効率的な音声認識アプリケーションを短期間で作成することができるのは、本システムの構成法によるところが大きいと考えられる。

7 むすび

今後の音声認識技術の実用化において、アプリケーションを作成する側、アルゴリズムを提供する側の双方にとって効率的なシステム環境を考慮し、CSS型に構成した音声認識サーバ・システムを構築した。種々の音声認識要素技術を統合し、音声認識APIによる効率的な開発環境を示した。本システムを利用した例として作図ソフトへの応用を示し、マルチモーダル入力ツールが容易に構成できることを示した。今後、種々のアプリケーションへ利用し、得られた知見から、更に改良を加えていく。

謝辞

本研究に際し、御討論頂いた音声情報研究部の皆様に深く感謝致します。

参考文献

[1] 永田, 橋本, 竹林: "ワークステーションにおける音声認識機能の検討", 音学講論(春), 2-Q-30(1993).

[2] 小高, 天野, 畑岡: "電話回線とLANを介した音声認識応用の検討", 信学技報, SP94-55(1994).

[3] 南, 山田, 鹿野, 松岡: "番号案内を対象とした大語い連続音声認識アルゴリズム", 信学講論, **J77-A**, 2, pp. 190–197(1994).

[4] 高橋, 嵯峨山: "4階層共有構造の音素HMM", 信学技報, SP94-73(1994).

[5] 山田, 嵯峨山: "環境依存の音素モデルを用いるLRパーザの一実現法", 音学講論(秋), 3-8-8(1994).

[6] 野田, 嵯峨山: "前向きヒューリスティック関数を用いたビーム探索によるHMM-LR音声認識の検討", 音学講論(秋), 3-8-16(1994).

[7] F. Martin, K. Shikano, Y. Minami, and Y. Okabe: "Recognition of Noisy Speech by Composition of Hidden Markov Models", 信学技法, SP92-96(1992).

[8] 高橋, 嵯峨山: "最大事後確率推定と移動ベクトル場平滑化の組合せによる逐次話者・回線適応", 信学技報, SP94-74(1994).

[9] H. Tanaka K. Itou, S. Hayamizu: "Continuous Speech Recognition by Context Dependent Phonetic HMM and an Efficient Algorithm for Finding N-best Sentence Hypotheses", ICASSP92, I-21(1992).

[10] 永井, 北, 嵯峨山: "HMM-LR法における音素文脈依存型LRパーザの検討", 音学講論(秋), 3-8-16(1990).

[11] 永井, 嵯峨山, 北: "音素コンテキスト依存型LRテーブルの生成アルゴリズム", 音学講論(春), 3-5-2(1991).

[12] 永井, 菊池, 嵯峨山, 北: "文脈自由文法から音素コンテキスト依存文法への変換アルゴリズム", 音学講論(春), 3-1-6(1992).

[13] H. Tanaka, H. LI, and T. Tokunaga: "Incorporation of Phoneme-Context-Dependence in LR Table through Constraint Propagation Method", Integration of Natural Language and Speech Processing, pp.15–22(1994).

[14] 志田, 西本, 小林, 白井: "音声・マウス・キーボードを併用した作図システムS-tgifとその評価", 信学技報, SP94-29(1994).