社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－MUS－74 (15)
2008－SLP－70 (15)
2008／2／9

Proxy-Agentを用いた音声認識対応ウェブアプリケーション開発フレームワークの提案と実装

中野 鐵兵† 藤江 真也† 小林 哲則†
†早稲田大学

あらまし:
Proxy-Agent[1] に対する拡張機能として開発した音声認識システムと，HTML+JavaScript ベースのウェブアプリケーションとの連携を可能にする枠組みを紹介する．Proxy-Agent は，音声認識システムに対してプラグインの枠組みを導入し，汎用的な機能拡張を可能にするソフトウエアコンポーネントである．提案手法では，Proxy-Agent と音声認識エンジンをネットワーク経由で動的に配信・起動する．また，Proxy-Agent が HTTP ベースのブラウザ連携の枠組みを提供し，ブラウザ側との JavaScript での連携を可能にする．本稿ではさらに，提案手法のプロトタイプとして開発したネットワーク配信型音声認識システムと，それを用いたウェブアプリケーション開発手法について紹介する．

A Proposal of New Development Framework for Web-based Applications Equipped with Speech Interface Using Proxy-Agnet

Teppei NAKANO†, Shinya FUJIE†, Tetsunori KOBAYASHI†
†Waseda University

Abstract:
This paper presents a new development framework for web-based applications equipped with speech interface using Proxy-Agent[1]. Proxy-Agent is a software component that provides supplementary services for speech recognition systems as well as user extensions. An architecture is designed so that Proxy-Agent and speech recognition engines can be delivered, and speech recognition services can be launched in the client side. In this architecture connections between Proxy-Agent and HTML based applications are established by using HTTP services on the Proxy-Agent instance. For the first prototype, we developed a network deliverable speech recognition systems. The development process using the proposed method is also described.

1 はじめに

音声認識機能を利用したウェブアプリケーションの開発を可能にするための新しい枠組みを提案する．インターネットのライフライン化に伴い，従来であればクライアント・サーバ型やスタンドアロン型で開発されていたアプリケーションの多くが，今日ではウェブベースのアプリケーションとして開発されるようになった．これに伴い，音声認識機能を持つアプリケーション開発においても，ウェブベースのインタフェースを用意する要求が増加している．

ウェブベースのアプリケーションに音声インタフェースを付与する取り組みは，音声インタフェースをサポートするようにブラウザを直接拡張する手法をはじめ，様々な手法が検討されてきた [2]．しかしながら今日の音声認識技術では，実用的な精度での音声認識機能の動作には，アプリケーションに特化したエンジンの再構成が必要であり，これをアプリケーション非依存かつエンジン非依存な形で，開発者・利用者ともに過多な負荷を与えることなく実現する枠組みはまだ確立されていない．

Goddeau らは，Win32 プラットフォームで動作するネットスケープのプラグインとして，ウェブページにマイクロフォンのユーザインタフェースを埋め込むことが可能な DIGITAL Voice Plugin を開発し，音声インタフェースを備えたウェブベースのアプリケーションの構築を可能にした [3]．また，Digalakis らは，ブラウザ上で動作する小型プログラムである Java アプレットを用いて，ウェブページに音声入力用インタフェースを埋め込む手法を採用した [4]．これらの手法では，音声入力を含む音声認識の処理の一部をブラウザ上で行い，それ以外の処理を共通の音声認識サーバで実行する．ブラウザ上で動作するプログラムを汎用的にすることで，アプリケーション非依存の共通の枠組みが構築可能となる．しかしながら，アプリケーション毎に使用するエンジンやその構成を自由に変えることは出来ず，実用的な精度の音声認識機能の提供が困難となる．

音声インタフェースを備えたウェブアプリケーションを構築する手法としては，アプリケーションを SALT[5] や XHTML+Voice[6] といった標準的なマークアップ言語を用いて記述し，ブラウザに組み込まれた音声認識エンジン連携の枠組みを利用する手法がある．しかしながらこの手法でも，使用可能なエンジンはブラウザ・環境に依存し，開発者がエンジンの選択や再構成を行うことは出来ない．

開発者が望む音声認識システムを用いたウェブアプリケーションを開発するためには，利用者に特定の音声認識システムのインストールを依頼し，それと連携するための特定の手法を用いてアプリケーションを開発するしかない．しかしながらこの場合でも，利用する音響モデルや言語モデル，パラメー

タの設定等を厳密にコントロールすることは困難である．また連携の手法も環境依存になりやすく，汎用的な枠組みとしては利用できない．さらに利用者の負荷が大きく，企業のイントラネットで使用するシステムのような，使用環境を比較的コントロールできる環境でないとこの枠組みの導入は困難である．

そこで本研究ではこれらの問題を解決し，ウェブアプリケーションの開発者に対して，自由な音声認識エンジンの選択とアプリケーション特化の再構成を可能にしながらも，実行環境に対して非依存な形でブラウザ連携を可能にする手法を提案する．提案手法では，アプリケーションのネットワーク配信技術とProxy-Agent技術[1]を合わせて用いることで，クライアント環境に対するエンジンの動的な配信と，標準的なウェブ技術のみを用いたブラウザ連携を可能にする．本研究ではこのように動作する音声認識システムをネットワーク配信型音声認識システムと呼び，一般のウェブサイト構築に利用可能な枠組みとして実現する．

本稿では，まず2節でウェブアプリケーション用音声認識システムのアーキテクチャとその分類を行う．3節にて提案手法を実現する上での解決が必要な課題を述べ，4節にて提案手法について述べる．さらに5節にて提案手法のプロトタイプとして実際に開発したシステムについて述べ，その設計と提案手法を用いた際の開発方法を紹介する．

2 ウェブアプリケーション用音声認識システムのアーキテクチャ

本節では，ウェブアプリケーションとの連携が可能な音声認識システムのシステムアーキテクチャについて述べ，その分類を行う．また，その中で提案手法がどこに位置づけられるかを述べ，その特徴を明確にする．

Digalakis et al.[4]では，ウェブベースの音声認識システムのアーキテクチャを，音声認識の関連処理が実行されるホストを基準として，Server-only Processing, Client-only Processing, Client-Server Processing, の3通りに分類した．しかしながら，ウェブ関連の技術が大きく発展した今日では，この分類だけではシステムの特徴を十分に表現できない．そこで本研究では，以下の3つの軸を用いてアーキテクチャの分類を行う．

t-axis thin client vs thick client (or fat client)

d-axis independent vs dependent

c-axis unconstrained vs constrained

t-axisは，前提とする実行環境の敷居の低さを示す軸であり，この軸の上位に位置するアーキテクチャでは，携帯端末等の低スペックなマシン環境でも利用可能であるという特徴を持つ．またd-axisは，ブラウザやオペレーティングシステム等の実行環境，及び特定のアプリケーションに対する依存の小ささを示す軸であり，この軸の上位に位置するアーキテクチャでは，アプリケーション非依存に多くの実行環境で利用可能である．c-axisは，音声認識エンジンの性能や機能・再構成に対する制限・制約の少なさを示す軸であり，この軸の上位に位置するアーキテクチャであれば，利用する音声認識エンジンをアプリケーションに特化して自由に再構成し，かつその性能や機能を十分に活用できる．図1は，これらの3軸を用いてアーキテクチャの特徴を示す空間を抽象化した図である．A〜Gのプロットは，具体的なアーキテクチャのマッピングを表す．以降，具体的なアーキテクチャの例に関して述べる．

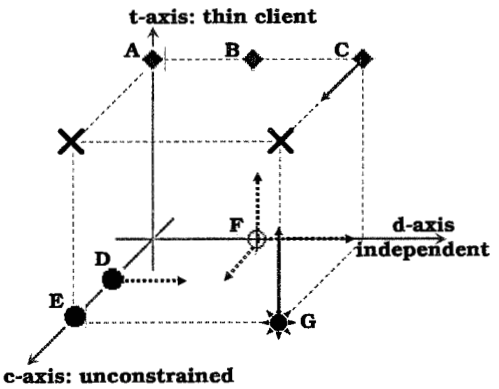

図 1: Architecture Mapping. [A,B,C] Distributed speech recognition systems. [E,F] Standalone speech recognition systems. [F] Embedded speech recognition systems. [G] Network deliverable speech recognition systems (proposed model). Arrows mean possibilities of their extension.

2.1 分散型音声認識システム

分散音声認識技術を利用したアーキテクチャでは，その実行条件にハイスペックなマシン環境を必要としない (図1(A)(B)(C))．この形態では，ブラウザ上で動作する小型プログラムが音声入力や音声認識の前処理を行い，それ以降の処理がネットワーク上のサーバで実行される．そのため，実行環境に与える負荷が小さい反面，処理の遅延や負荷集中の問題等，避けることの出来ない問題が存在する．また，サーバ共有型の分散音声認識システムを利用する場合，アプリケーション毎のエンジンの独自構成は困難であり，必ずしも利用する音声認識エンジンの機能を十分に活用できない．

このアーキテクチャでは，ブラウザ上で動作するプログラムの種類や組み込み方により，その利用可能範囲が異なる．例えばDIGITAL Voice Plugin[3]のような，特定ブラウザのプラグインとして音声入力用コンポーネントを組み込むことを前提としたアーキテクチャは，汎用的な利用が困難となる (図1(A))．しかしこのような問題は，Digalakis et al.[4]のように，実行環境に対する依存が小さいJavaアプレット等の技術を採用することである程度解決する (図1(B))．この方式では，アプレットが音声入力からMFCCの抽出までを行うことで，狭帯域のネットワークにおいても高精度な音声認識が可能となる．

このようなブラウザに音声インタフェースを埋め込むためにアプレットを採用する手法は，アプレットが音声入力をサーバにそのまま渡すだけの単純なモデル [7] を含め，今日でも様々なアプリケーションで採用されている．Gruenstein らは，AJAX[8] 技術との組み合わせにより，マルチモーダル対話インタフェースを備えたインタラクティブな地図アプリケーションを開発し，公開している [9][10]．また西村らは，一般向けに公開されている音声対話アプリケーションの，ウェブ経由での利用を可能にした Web たけまるくんを公開している [11]．

さらに西村らはウェブシステムに対して音声入力インタフェースを拡張する枠組みとして，w3voice を公開・運用している [12]．ここでは，アプレットとの連携用 API を JavaScript ライブラリとして利用可能にすることで，その枠組みのより広い範囲での利用を可能にした (図 1(C))．また，音声認識サーバ用のライブラリも合わせて公開することで，独自サーバの運用が可能ならば，自由に再構成した音声認識システムの利用も可能にした (図 1(D) 矢印)．

2.2 スタンドアロン型音声認識システム

クライアントの実行環境で直接音声認識エンジンを動作させ，その API を環境依存な方法で直接用いるアーキテクチャでは，自由な音声認識エンジンの再構成とその全ての機能の制限無しの利用が可能である (図 1(D))．この形態では，例えば，利用者に特定の音声認識システムのインストールを要求し，ブラウザとの連携を VBScript + ActiveX コンポーネントを用いて実現する．Windows SAPI 等の標準的な API を用いることで，開発者による自由なエンジンの再構成や，API で定義されていない機能呼び出しが困難になるが，環境非依存性が若干向上する (図 1(E) 矢印)．

2.3 組み込み型音声認識システム

ブラウザに組み込まれた，音声インタフェースを備えたウェブアプリケーション構築の枠組みを利用するアーキテクチャでは，マークアップ言語のみを用いたアプリケーション開発が可能となる (図 1(F))．この形態では，アプリケーションは SALT[5] や XHTML+Voice[6] といったマークアップ言語を用いて記述される．これらのマークアップ言語は標準化が進められているが，実際にはその実行環境は十分に普及しているとは言えず，プラグインや特定のブラウザの利用が必須となる．また，直接 API を呼び出せず，様々な制約が存在する．なお，ブラウザがエンジンを実際に組み込んでいるものから，OS 依存のエンジンを呼び出すもの，場合によってはネットワーク経由で認識処理を実行するもの (図 1(F) 矢印) まで考えられるが，エンジンの再構成や開発の自由度の向上には寄与しない．

2.4 ネットワーク配信型音声認識システム (提案手法)

音声認識エンジンをネットワーク経由で動的に配信・起動し，そこで起動したエンジンとの連携を前提とした HTML を記述することで，スタンドアロン型と同等に少ない制限で，同時に環境依存の小さい音声認識アプリケーションの構築が可能となる (図 1(G))．この形態では，配信される音声認識システムが，ブラウザとの連携手段を合わせて提供する．特にこの連携手段を環境非依存な技術を用いて提供することで，標準的なウェブ構築技術だけを用いた高機能な音声認識システムの構築が可能となる．

3 ネットワーク配信型音声認識システム実現上の課題

ネットワーク配信型音声認識システムでは，音声認識エンジンがクライアント側に動的にダウンロードされ，クライアント端末上でブラウザとの連携を行う．この枠組みを実現するためには，以下の課題の解決手段が必要となる．

- 音声認識エンジン配信
- 異なる音声認識エンジンの共存
- リソース共有
- ブラウザ連携
- オフライン動作

音声認識エンジンの配信の問題は，配信対象となるモジュールが音声認識エンジン本体であるという点で，分散音声認識システムの際の前処理モジュールの配信とは，必要となる手法が異なる．すなわち，配信されるデータには音響モデルや言語モデル等大容量のモジュールも含まれる．また，それらを毎回配信するのは現実的ではなく，キャッシュの枠組みを備える必要がある．

さらに，複数のウェブアプリケーションで利用可能な枠組みとして考えた場合，利用可能な音声認識エンジンを 1 つに固定することは望ましくない．そのため，複数の異なる音声認識エンジンを利用可能にする枠組みが求められる．さらにその際，アプリケーション毎に同一のモジュールを別々に配信するのではなく，異なるアプリケーション・エンジン間でのリソースを共有する枠組みが求められる．

ブラウザ連携の枠組みとしては，複数のエンジンが同時に利用可能な，環境非依存な共通の枠組みを提供する必要がある．各々のエンジンが競合する同一の連携手段を提供するだけでは，1 つのエンジンが一度使われると別のエンジンが利用できなくなるという問題が発生する．さらに，標準的な API だけでなく，エンジン独自の API も利用できることが望ましい．

さらに，インターネット接続が切断している際のサービス提供の枠組みが備えられる事が望ましい．今日のウェブベースのアプリケーションの普及に伴い，Google Gears[15] や Adobe AIR[16] のように，インターネット接続が利用できないときでもウェブアプリケーションを利用可能にする枠組みが求められている．ネットワーク配信型の音声認識システムにおいても同様の枠組みの検討が必要であり，ネットワークの常時接続を前提とすることは出来ない．

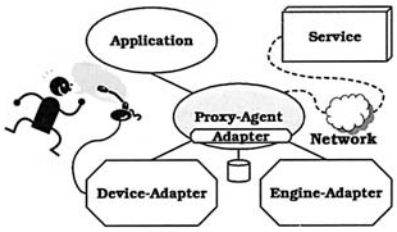

図 2: Proxy-Agent Overview

4 Proxy-Agent を用いたネットワーク配信型音声認識システム

前節で述べた課題を解決するための基本アプローチとして，Proxy-Agent を利用したネットワーク配信型音声認識システムの構築手法を提案する．

Proxy-Agent とは，アプリケーションプログラム，音声認識エンジン，入力デバイス，の間に配備されたソフトウエアコンポーネントであり (図 2)，以下の枠組みを提供する．

Extension capability 全ての拡張機能が Proxy-Agent に対するプラグインとして拡張可能

Networking capability ネットワーク経由でのサーバ連携が可能

Monitoring capability 実利用環境における実行時の情報が取得可能

Upgrade capability 実利用環境におけるプラグインの継続的な更新が可能

Sharing capability 言語リソースやコンポーネントの実装，フレームワーク等の共有が可能

Proxy-Agent の拡張機能として前述の枠組みを実装することにより，Proxy-Agent に対応した音声認識エンジンがネットワーク配信型音声認識システムとして動作可能になる．

Proxy-Agent を用いることで，エンジン配信を，Proxy-Agent 本体の配信と，Proxy-Agent に対するエンジンの配信との 2 段階で対応することが可能となる．具体的には，Proxy-Agent を音声認識システムの実行基盤として配信し，Proxy-Agent の拡張機能としてエンジンの配信機能を実現する．これにより，複数の異なる音声認識エンジンの共存を可能にすると同時に，リソースの共有も可能となる．また，ブラウザ連携の枠組みも Proxy-Agent の拡張機能として実装することで，複数のエンジンが共通の枠組みを同時に利用可能となる．さらに，オフライン動作に必要な情報を Proxy-Agent が保存する枠組みを提供することで，ネットワーク非接続時の動作も可能となる．

5 プロトタイプ開発

Proxy-Agent を用いて配信型音声認識システムのプロトタイプを開発した．本節ではその詳細を示す．

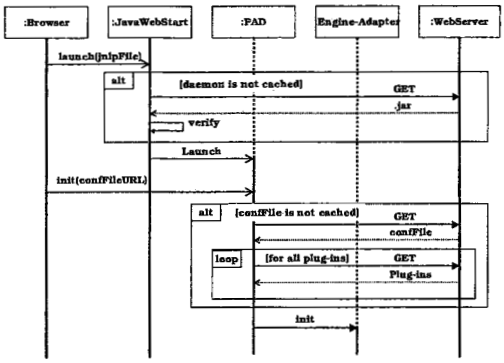

図 3: Sequence diagram of network delivery. [1. When user clicks the link to the Proxy-Agent Daemon jnlp file, in which required jar files are written, Java Web Start process (jws) is called by the operating system. 2. if the click is the first time, or the version is updated, jws start downloading the libraries written in the jnlp file. 3. PAD is launched by the jws. 4. JavaScript running in the browser calls init method of PAD. 5,6. PAD downloads all required plug-ins. 7. Engine is prepared.]

5.1 配信機能の実装

Proxy-Agent のデーモンプログラム (PAD) を Java Web Start を用いて配信可能にした．Java Web Start は，java ベースのアプリケーションのネットワーク配信を可能にする技術であり，J2SE 1.4 以降がインストールされている環境ならば設定なしで利用可能である．HTML に記載されたリンクからのプログラム起動が可能であり，必要なライブラリが自動的にダウンロードされる．キャッシュ機構を備え，一度ダウンロードしたプログラムは次回起動時には再度ダウンロードすることなくプログラムが起動される．また，オフライン起動も可能であり，ネットワークに接続しない状態でも利用可能である．

音声認識エンジンの配信は，Proxy-Agent が備えるプラグイン配信の枠組みを用いて必要なプラグインをダウンロードする．必要なプラグインは，アプリケーション毎に用意されたエンジンの構成ファイルに従って決定され，一度ダウンロードしたプラグインはキャッシュされる．また，複数のエンジンの共存が可能であり，それらが同一のプラグインを利用するならば，それらは共有される．これらの配信に関するシーケンスを図 3 に示す．

5.2 ブラウザ連携

HTTP のサービスをローカル・ループバック・アドレスで提供する，簡易ウェブサーバ機能を Proxy-Agent のプラグインとして用意した．すなわち，音声認識エンジンとブラウザとの連携は，Proxy-Agent が提供する HTTP のサービスを経由して行われる．また，ブラウザ側からの接続を可能にするために，JavaScript のライブラリを用意した．この枠組みを用いると，ブラウザ側には HTML+JavaScript だけで全ての必要な処理が記述できる．通常，クロスド

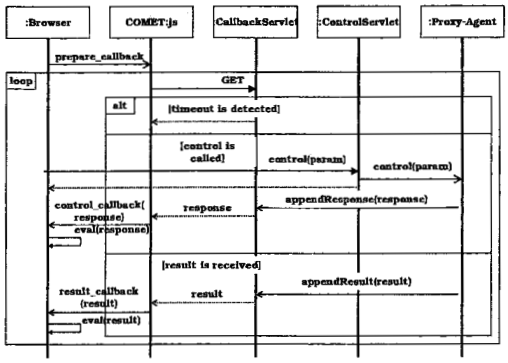

図 4: Sequence diagram of JavaScript library. [1. After the scripts are loaded, prepare_callback is called to setup the connection for callback (as a COMET client). 2. COMET client calls the server and waits for the response. This call is retured when (A)Timeous is detedted, (B)Control response is received, (C)Processing result is received. 5,6. response or result are append to the callback queue. Callback servlet generate the JSON data representing the data, and return it to the COMET client. 7. COMET client send callback method of the JavaScript written in the HTML. 8. JSON data is executed via eval method.]

メインの制限により，JavaScript を用いたローカルの HTTP サービスとの通信は利用できない．しかしながら今回の実装では，JSONP[13] 方式でライブラリを構築したことでこの問題を回避した．また，音声認識エンジンからの結果や状態を通知するために，COMET[14] 方式でコールバックの枠組みを実装した．シーケンスを図 4 に示す．

JavaScript のライブラリでは，以下のインタフェースを用意した．

- 認識処理の開始要求
- 認識処理の停止要求
- Engine-Adapter に対する処理要求（制御・設定等，任意の機能の呼び出しが可能）
- データに対するタギング

5.3 アダプタの設計と配備

Proxy-Agent を用いた音声認識システムでは，アプリケーション毎に音声認識エンジンを利用するためのアダプタが用意される．アダプタは Engine-Adapter と Device-Adapter の組み合わせとして定義される (図 2)．Engine-Adapter は，プラグインによる機能拡張の枠組みを有する仮想音声認識エンジンであり，開発者がその構成を自由に設計できる．そのインタフェースは，音声認識エンジンと拡張機能のインタフェースの集合として公開され，提案手法ではそれらを JavaScript を用いて呼び出すことが可能である．これを利用するために，提案手法に従って構築するウェブアプリケーションでは，開発者は HTML(もしくはその生成プログラム) に加え，アプリケーションで利用するアダプタとその

```
<script src="proxy.agent.service.server.url/?
        adapter=config.file.url"></script>
```

図 5: An example of script tag

構成ファイルを用意する．

アダプタは，Proxy-Agent に対するプラグインとして，Eclipse IDE[17] を用いて開発する．基本構成要素はプラグインとして提供されており，一般的な音声認識エンジンであれば，設定ファイルを用意するのが主作業となる．動作可能なアダプタが用意できると，アダプタの実行に必要なプラグインをフィーチャーと呼ばれる Eclipse の配布単位にまとめ，別に用意した構成情報と共にサーバに登録する．このとき，構成ファイルには以下の項目が指定される．

- アダプタのプラグイン ID
- アダプタのフィーチャID と配信用 URL
- 他に依存するフィーチャID と配信用 URL

5.4 HTML 開発

提案手法における HTML 開発は，通常の HTML+JavaScript を用いたウェブサイト開発の手法がそのまま利用可能である．具体的には，まず図 5 のようなスクリプトタグを用意する．ここでは，ソース属性にサービスを利用するためのライブラリの URL と，そのアプリケーションで利用するアダプタ構成ファイルの URL をパラメータとして指定する．また，音声インタフェース用のフレームワーク等，他の拡張ライブラリを利用する際には，パラメータでその利用の設定が可能である．これにより，Proxy-Agent との接続が自動的に行われ，ライブラリが提供する API を用いた音声認識エンジン連携が可能となる．認識結果やその他処理の戻り値は JSON 形式で与えられ，JavaScript で容易な取り扱いが可能である．

6 議論

本節では，ネットワーク配信型音声認識システムに関して，エンジンの選択・再構成の有効性と，エンジン配信の実用性に関して議論を行う．

提案手法では，音声認識対応ウェブアプリケーション開発の枠組みをアプリケーション非依存かつエンジン非依存な形で提供する．そのため，ウェブアプリケーション開発者は，開発するアプリケーションに適した音声認識エンジンの選択と，アプリケーションに特化したエンジンの再構成が可能である．反面，提案手法でウェブアプリケーションを構築する際の作業の増大が懸念される．サーバ共有型の分散型音声認識システムと比較すると，提案手法では新たなアダプタの定義と構成ファイルの作成，プラグインのサーバへのアップロードといった追加の作業が必要となる．特に，アダプタの定義は Eclipse でのプラグイン開発の知識も必要となり，必ずしも簡単な作業ではない．これらの作業は Eclipse の拡張機能としてウィザードを作成する等，自動化が可能な範囲であるが，現時点では開発者の負担が増大する．しかしながら，サーバ共有型の分散音声認識

では，タスクオープン・話者オープンの認識が求められ，今日の技術では実用的な認識率は得られない．また，常時音声入力を受け付けるハンズフリー音声入力や，自然対話インタフェースの構築は困難である．独自サーバによる分散音声認識システムの場合には，アプリケーションに特化した自由な構成が可能となるが，この場合はサーバの設定やメンテナンス等の作業が発生し，開発者の負担は提案手法以上に大きくなる．スタンドアロン型の音声認識システムの場合はこれらの問題の回避が可能であるが，音声認識エンジンの再構成の作業負担を利用者に強いるため，望ましくない．結果として，アダプタ定義の作業負担の増大を差し引いても，提案手法を用いたエンジンの選択と再構成可能性は有効である．

また，モデルデータを含んだ音声認識エンジンをネットワークで配信することの非実用性が懸念される．そこで，[18] で利用した Sphinx4[19] ベースの音声認識システムを，配信可能な形式に変換した．結果としては PAD の容量で 17M 程度，音響モデルデータを含んだエンジンのプラグインの容量で 20M 程度であり，一度だけ配信するという観点では十分現実的な容量であった．もちろん，配信に対する実用的な容量はネットワーク回線に大きく依存するために，この容量を許容範囲とは見られないケースもある．しかしながら，ネットワークの下りの回線速度は今後も引き続き太くなることが確実であり，容量の点で問題となるケースは少ないと考えられる．

7 むすび

本稿では，音声インタフェースを備えたウェブアプリケーション開発のための新たな枠組みとして，ネットワーク配信型音声認識システムを用いた手法を提案した．また，Java Web Start を用いた Proxy-Agent の配信の枠組みと，Proxy-Agent によるエンジン配信の枠組みを合わせた提案手法の実現方法について述べた．さらに Proxy-Agent の拡張として HTTP ベースのブラウザ連携機能を実装することで，標準的な HTML+JavaScript のみを用いて音声認識システムとの連携を可能にする手法を示した．

今後の予定としては，サービスの一般公開のためのフレームワークの拡充と，セキュリティの観点から必要な機能と枠組みの検討を行う．

謝辞 本研究は，経済産業省，平成 18,19 年度戦略的技術開発委託費「音声認識基盤技術の開発」及び早稲田大学理工学研究所・プロジェクト研究「音声認識技術実用化」の一部として実施されたものである．

参考文献

[1] Teppei Nakano, Shinya Fujie, and Tetsunori Kobayashi. EXTENSIBLE SPEECH RECOGNITION SYSTEM USING PROXY-AGENT. Proc. of ASRU2007, pp.601-606, December 2007.

[2] Bayer, S. Embedding speech in Web interfaces. In Proc. of ICSLP 96, Vol.3, pp.1684–1687, Oct. 1996.

[3] Goddeau, D., Goldenthal, W., and Weikart, C. Deploying Speech Applications over the Web. In Proc. of Eurospeech 97, pp.685–688, Sep. 1997.

[4] Digalakis, V., Neumeyer, L., Perakakis, M. Quantization of cepstral parameters for speech recognition over theWorld Wide Web. In Proc. of ICASSP 98, Vol.2, pp.989–992, May 1998.

[5] Speech Application Language Tags. `http://www.saltforum.org/`

[6] XHTML+Voice. `http://www.voicexml.org/specs/multimodal/x+v/`

[7] Z. Tu and P. Loizou. Speech recognition over the Internet using Java. In Proc. of ICASSP 99, pp.2367–70, March 1999.

[8] Jesse James Garrett. Ajax: A New Approach to Web Applications. `http://adaptivepath.com/ideas/essays/archives/000385.php`, February 2005.

[9] Alexander Gruenstein, Stephanie Seneff, and Chao Wang. Scalable and Portable Web-Based Multimodal Dialogue Interaction with Geographical Databases. In Proc. of Interspeech2006, pp.453–456, September 2006.

[10] Alexander Gruenstein and Stephanie Seneff. Releasing a Multimodal Dialogue System into the Wild: User Support Mechanisms. In Proc. of the 8th SIGdial Workshop on Discourse and Dialogue, pp.111–119, September 2007.

[11] 西村 竜一, 三宅 純平, 河原 英紀, 入野 俊夫. ネットワーク公開試験に向けた音声対話 Web アプリケーションの開発. 日本音響学会 2007 年春季研究発表会講演論文集, pp.17–18, 2007.

[12] 西村 竜一, 三宅 純平, 河原 英紀, 入野 俊夫. 音声入力・認識機能を有する Web システム w3voice の開発と運用, 情報処理学会研究報告, 2007–SLP–68-3, pp.13–18, 2007.

[13] Bob Ippolito. Remote JSON - JSONP. `http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/`, December 2005.

[14] Alex Russell. Comet: Low Latency Data For Browsers. `http://alex.dojotoolkit.org/wp-content/LowLatencyData.html`, March 2006.

[15] Google Gears. `http://gears.google.com/`.

[16] Adobe AIR. `http://labs.adobe.com/technologies/air/`.

[17] Eclipse - an open development platform. `http://www.eclipse.org/`

[18] 熊井朋之, 中野鐵兵, 小林哲則, 石川 泰. 機能構造と連続キーワード入力を利用した音声インタフェースのユーザビリティ評価. 日本音響学会 2007 年秋季研究発表会, 2-3-6, Sep. 2007.

[19] Paul Lamere, Philip Kwok, William Walker, Evandro Gouvea, Rita Singh, Bhiksha Raj, Peter Wolf. Design of the CMU Sphinx-4 decoder. In Proc. of Eurospeech 2003, pp.1181–1184, 2003.