Vol.2015-SLP-108 No.3
2015/10/30

HTML5対応クラウド音声認識プラットフォーム

鎌土 記良[1,a)]　藤村 滋[1]　岩瀬 義昌[2]　青野 裕司[1]　政瀧 浩和[1]　山田 智広[1]　大津谷 亮祐[2]

概要：我々はこれまで，PC やモバイル端末を問わず，HTML5 対応 Web ブラウザのみでサーバークライアント型音声認識機能を実現する方式の提案と，その実用性について報告を行ってきた．我々が提案する方式では，クライアントで実行される JavaScript を介して音声認識機能の利用が可能となるため，ブラウザさえインストールされていれば端末を問わず音声認識機能をネットワークを介して利用することができる．具体的には，クライアントのブラウザ上で実行される JavaScirpt にてマイクからの音声データの取得と圧縮を行い，Web サーバへのストリーミング送信を実施する．サーバー側では，送信音声に対し音声区間検出器 (Voice Activity Detector; VAD) と音声認識処理を行う．これによりクライアントでの計算コストを削減し，Web ブラウザのみでの音声認識を実現する．本システムは NTT アイティ社の協力の元，SpeechRec for Browser としてサービスインした．昨年からは，NTT ドコモのドコモ・デベロッパー・サポートへ，今年は NTT コミュニケーションズの SkyWay からの利用ができるようになり，徐々にその利用が広まりつつある．そこで，本稿では，これまで我々が提案してきたシステムの構成について，再度詳細を述べた上で，昨今のブラウザや端末の対応状況確認を実施し，最後に提案システムの具体的な利用方法について述べる．

キーワード：音声認識，Web アプリケーション，Web ブラウザ，WebSocket，Media Stream，Web Audio API

Introduction of noise-robust ASR platform based on HTML5

Noriyoshi Kamado[1,a)]　Shigeru Fujimura[1]　Yoshimasa Iwase[2]　Yushi Aono[1]
Hirokazu Masataki[1]　Tomohiro Yamada[1]　Ryosuke Otsuya[2]

Abstract: We propose a browser-based speech recognition system using HTML5 in a broad sense and report its performance in actual use. Our proposed method enables browsers in PCs and mobile devices to use speech recognition function by client-side JavaScript code. Unlike traditional Web applications, there is no need to install specific application or browser plug-in. The flow of processing, at first getting the streaming audio data from the microphone of the client, and the client device transmits its streaming data to the speech-recognition server by the WebSocket protocol. In consideration of the quality of mobile broadband, by using client-side JavaScript, compression for audio data is also performed, and the Voice Activity Detector(VAD) and the speech recognition decoder are implemented to the server because of the reduction of the computational cost of the client. The proposed system has been started to provide as a SaaS "SpeechRec for Browser" from last year by NTT-IT. This service is not only available from DoCoMo developer support of NTT DoCoMo from last year, is now available from the SkyWay of NTT Communications. In this paper, we explain the architecture of our proposed system and support status of browsers and devices again based on this kind of scenery. And we also report the audio compression performance in the client and the quality of actual use in mobile broadband. In the result, even now the proposed method has adequate quality as using speech recognition system.

Keywords: Voice Recognition, Web Application, Web Browser, WebSocket, Media Stream, Web Audio API

© 2015 Information Processing Society of Japan

1. はじめに

近年，Web ブラウザの高機能化が進み，その機能を活かしたサービスの成長は目覚ましい．Web ブラウザの高機能化はデスクトップ・ノート PC 上のブラウザが先行していた．しかし，現在はスマートフォンやタブレット端末上のブラウザにおいても同等の高機能化も進みつつある．

Web ブラウザの高機能化は，W3C と WHATWG が仕様策定および標準化を進めてきた HTML5 に基づいており，HTML5 は 2014 年 10 月 28 日 (米国) に W3C によって正式な勧告として仕様が公開された．従来は，インタラクティブ性の高いアプリケーションは OS から提供される機能の元で実現することが主流であったが，HTML5 の登場により，現在ではブラウザもアプリケーションを実現するプラットフォームとしての地位を固めたといっても過言ではない．

このような背景と並行し，従来は VoIP （Voice over IP）の一部であった，比較的 HTTP と親和性の高い音声・映像通信技術を，HTML5 （正確には HTTP）に取り込もうという動きが出てきた．この取組は，WebRTC (Web Real Time Communication) と呼ばれ，ブラウザ上でプラグイン無しに映像や音声等のリアルタイム双方向通信を実現する，WebRTC の議論の中には，ブラウザを通してリアルタイムに音声や映像のストリームを取得するための仕組みも含まれている．これは，スマーフトフォンやタブレットといった端末の違いを問わずに，Web ブラウザ上で動作するアプリケーションにより音声入力機能や音声対話機能を手軽に実現できる可能性を示している．ブラウザ上のアプリケーションとして実現することにより，従来では OS や端末ごとに異なる実装が必要であった音声入力インターフェースを，統一的な実装で実現することが可能となる．我々はこの点に着目し，様々な HTML5 対応ブラウザ上で呼び出すことのできる音声認識技術の開発を行った [1].

文献 [1] で，我々は，実現したシステムの構成について述べ，ブラウザや端末の対応状況，クライアントとなる複数のブラウザ上でのリサンプリングと音声データの圧縮処理速度の計測，モバイル回線での音声認識処理時間の計測を行い，静音環境下における実用性について述べた．そして，文献 [2] で本システムの雑音環境下での認識実験を行い，その有効性を示した．

本システムは，NTT アイティ社より，SpeechRec for Browser として昨年よりサービス提供を開始した．本サービスを利用し，NTT ドコモのドコモ・デベロッパー・サポート [3] と NTT コミュニケーションズの SkyWay [4] でブラウザ上から高精度な音声認識を提供することができるようになった．本稿では，再度，提案システムについてその詳細をより詳しく述べると共に，その利用方法と現在の周辺技術のサポート状況を踏まえた報告を行う．

[1] 日本電信電話株式会社
Nippon Telegraph and Telephone Corporation
[2] NTT コミュニケーションズ
NTT Communications Corporation
[a)] kamado．noriyoshi@lab．ntt．co．jp

2. 提案システム

2.1 概要

本節では，従来の Web ブラウザ上での音声認識システムと我々の提案システムとの差異について述べる．図 1 に、提案システムの構成を示す．

従来の取り組みとしては，ユーザの発話を入力とする Web アプリケーションを構築するために，専用のブラウザやブラウザを用いた手法が主流であった．例えば，専用のボイスブラウザを用いて Web アプリケーションを実行することを前提とし、Microsoft を中心に，音声処理向けの HTML 拡張 タグ仕様，Speech Application Language Tags(SALT) [6] が SALT Forum より提案されていた．また，同様の規格として W3C では VoiceXML [7] という標準 XML フォーマットの標準化が進められていた．また，プラグインを前提として方法としては，一般的なブラウザで動作させられる方法として，Flash や Java アプレットを用いる方法がある．WebGALAXY [8] や以前の w3voice [9], [10] は Java applet を用いた音声認識機能の実装を行っている．これらは PC ブラウザであれば事前のインストー ルは要求しないが，現状 AndroidTM や iOS などで Java アプ レットをサポートするブラウザは少なく，これらの方式をその ままモバイル端末で利用するのは難しい．これらの従来の取り組みから，端末の Web ブラウザのみで音声認識機能を組み込んだ Web アプリケーションを利用することができ，かつアプリケーション開発者が音声認識サーバを用途に応じて自由に選択できる方式は実現されていなかった．HTML5 では，ワンソース・マルチプラットフォームを利点の一つとして上げているため，HTML5 上での音声認識ライブラリとして，提案システムのように Web ブラウザの種類やプラグインに依存せずに動作する Web アプリケーションライブラリの実現が求められていた．

提案システムと同様に，Web ブラウザに対してプラグインなどの特別なソフトウェアの追加なしに，HTML や JavaScript の記述のみで音声認識機能を利用可能とする方法としては (1) Google が独自に Google Chrome ブラウザに実装した x-webkit-speech，(2) W3C の Web Speech API で議論された Web Speech API （W3C の標準化プロセスを経ていないため，W3C 正式勧告となることはない），(3) 西村らの w3voice [5] がある．しかし，(1) 及び (2) はブラウザ内にほぼ全ての実装が隠蔽されており，認識サーバーの切り替えや内部的な音響パラメータの調整が不可能であり，ユーザーの幅広い要望に応えることができない．また，(3) は様々なブラウザで音声認識を利用可能とするための着想は同一であると考えるが，我々はさらにストリーミングでの音

© 2015 Information Processing Society of Japan

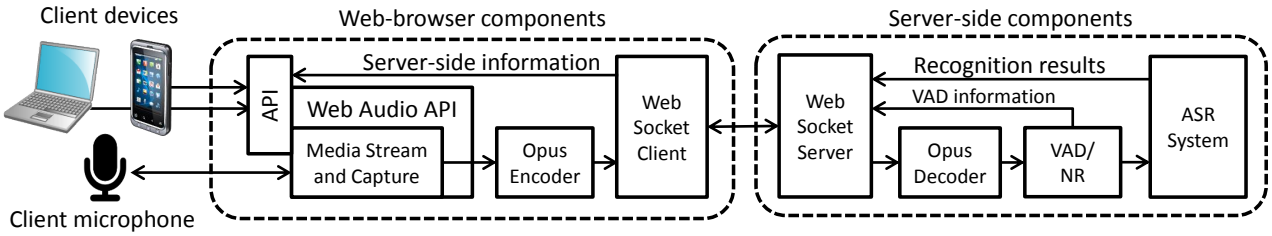

図 1　Proposed ASR system based on Web-browser components.

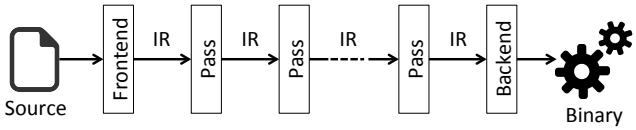

図 2　LLVM の構造

声送信，音声の圧縮符号化やハンズフリー化に必要な音声区間検出機能の実装を加えている．

2.2　内部構成とその詳細

まず，マイクデバイスからの音声ストリームの取得は，現在 W3C で標準化に向けた取り組みが進められている Stream API (W3C 文書名: Media Capture and Streams) と Web Audio API の仕様に従い，ブラウザへの依存を避けている．更に，IETF (Internet Engineering Task Force) によって開発されたオープンな圧縮フォーマット Opus をブラウザに依存しない標準的な JavaScript 記法にて再実装することで，ブラウザ非依存で軽量かつ高精度な音声符号化を実現している．

この再実装には，LLVM (Low Level Virtual Machine) [11] を用いた C/C++ プログラムの JavaScript への高度なトランスレータである Emscripten [12] を用いた．Emscripten における LLVM では，一般的なコンパイラと同じく，プリプロセッサ，字句解析，構文解析，意味解析，最適化とコード生成という順序で C/C++のコードを処理する．この流れに通常のコンパイラと大きな違いは無い．入力ソース・コードを意味解析するまでが Frontend で，その後は最適化の過程で Pass というユニットを通り，その間は IR (Intermediate Representation) という最適化に適した中間言語で Backend まで処理される．Backend で IR を最終ターゲットである JavaScript へと変換する．IR は変換元の C/C++ や JavaScript の言語体系とは全く無関係で非常に最適化に適した言語体系となっているため，変換元や変換先の制約に囚われず，高度な最適化技術を導入することができる．このため，Emscripten により変換された JavaScript は，JIT 上で動作するにも関わらず，非常に高速に動作する．

このように変換された Opus 音声符号化のライブラリは C/C++ で提供されているライブラリと等価な機能が利用できるため，ビットレートはブラウザ側で調整可能であり，様々な回線状況に応じて柔軟な運用が可能となる．

また，クライアント側では音声の取得と送信のみを行い，計算コストの大きい発話区間検出/雑音抑圧（VAD/NR）部 [13] は，クライアントから送信された音声データから雑音等を除去して音声区間を抽出する音声区間検出処理を行い，音声認識には音声区間のデータのみを音声認識処理に用いることができるようになっている．通常，雑音抑圧や音声区間検出処理においては，音声符号化による音声信号の劣化が悪影響を及ぼすことが考えられるため，クライアント側で符号化を行った後に，サーバー側でこれらの処理は行うことは好ましくない．この点については，[2] にて実験的に問題ない点が確認できている．

また，提案システムでは音声認識処理に VoiceRex [14] を採用している．圧縮デコーダと VAD，音声認識処理を構築した Web サーバとの間は WebSocket プロトコル上で MessagePack を用いた独自のプロトコルを使用している．用途に応じて音声認 識サーバを切り替える際は，WebScoket の接続を URL を指定 して切り替えることができる．Web Speech API とは異なり，音声認識サーバをアプリケーション開発者が選択する自由度が ある構成となっている．

提案の構成では，音声区間検出処理がサーバに配置されているため，クライアントから音声データ送信の終了を自動で行うことができない．そこで，サーバで実施される音声区間検出処理において，音声区間から雑音区間に遷移したタイミングで音声区間検出処理の終了をクライアントに通知する．クライアントではこの通知を受けて音声デー タの取得と送信を終了する．音声区間と判定された時間区間に該当する音声データのみを VoiceRex に入力し，あらかじめ設定した音響モデルと言語モデルに基づいて音声データをテキスト化する．最終的な認識結果をクライアントに送信し，Web アプリケーション内で利用することができる．

以上に概要を示した本システムの構成により，幅広いタスクに応じた音声認識サーバーをクラウド経由で提供することで，ユーザーニーズに柔軟に応えることが可能となる．

3.　対応ブラウザ状況

提案システムの構成に従って音声認識サーバを構築し，音声の取得から認識結果の取得までの動作検証を行った．各プラットフォーム毎の動作検証の結果を表 1 に示す．試験環境は前稿からアップデートされた，より新しい環境を選択した．

© 2015 Information Processing Society of Japan

表 1　各プラットフォーム毎の動作状況.

Devices(OS)	Browser(Version)	Status
MacBookAir 11-inch, Mid 2011 (Mac OS X 10.7.5)	Chrome(v45.0)	✓
	Firefox(v41.0)	✓
	Opera(v32.0)	✓
	Safari(v8.0.8)	
	Safari(v8.0.8) with Temasys WebRTC Plugin	
Let’s note CF-J10 (Windows7 Professional)	Chrome(v45.0)	✓
	Firefox(v41.0)	✓
	Opera(v32.0)	✓
	Internet Explorer (v9.0)	
	Internet Explorer (v9.0) with Temasys WebRTC Plugin	
	Internet Explorer (v10.0)	
	Internet Explorer (v10.0) with Temasys WebRTC Plugin	
	Internet Exploler(v11.0)	
	Internet Exploler(v11.0) with Temasys WebRTC Plugin	
Nexus 4 (Android v4.4.2)	Chrome(v45.0)	✓
	Firefox(v28.0)	✓
	Opera(v20.0)	✓
Xperia Z (Android v4.2.2)	Chrome(v45.0)	
	Firefox(v28.0)	✓
	Opera(v32.0)	✓
	Android Browser(v4.2.2)	
Xperia Z3 Compact (Android v4.4.4)	Chrome(v45.0)	
	Firefox(v28.0)	
	Opera(v32.0)	
iPhone 5 (iOS 8)	Chrome(v45.0)	
	Safari(v8.0)	
	Opera Mini(v11.0.0)	
Nexus 10 (Android v4.4.2)	Chrome(v45.0)	✓
	Firefox(v28.0)	✓
	Opera(v20.0)	✓
Galaxy Tab 10.1 (Android v4.4.4)	Chrome(v45.0)	
	Firefox(v28.0)	✓
	Opera(v20.0)	✓
	Android Browser(v4.0.4)	
iPad Air 2(iOS 9)	Chrome(v45.0)	
	Safari(v9.0)	
	Opera Mini(v11.0.0)	

全体を通し，現存するすべてのプラットフォームで提案システムの動作は確認できなかったが，改めて，音声認識を実現する 1 つのソースで，複数のプ ラットフォームで動作可能であることが確認できた．Firefox では前回の検証 [1] においては使用した Opus エンコーダが動作しなかったため，圧縮処理を除く音声認識機能の動作検証を行ったが，今回の検証においては Emscripten のバージョンアップに伴い，Opus エンコーダの再トランスレートと一部のソース修正により動作が確認でき，対応プラットフォームの幅が広がったことが確認できた．

また，Xperia Z における Chrome と Xperia Z3 Compact でのブラウザの動作に関しては，音声ストリームの取得に成功し,Web Audio API の ScriptProcessorNode で音声データを取得しようとしているものの，入力音声とは対応しないランダムなデータが格納されたり，そもそものデータがコールバックに渡らないため，意図しない認識結果が返ったり，認識がタイムアウトしたりする．バックエンドで共通に使用している OpenSL ES の使用方法か，オーディオデバイスドライバの問題と思われる．

今回検証したプラットフォームのパターン 32 種類のうち 16 パ ターンで動作が確認できた．表 1 より，Mac OS X と Windows7 の PC 上では，Chrome，Firefox で共通して動作が確認できた.OS にプリインストール されているブラウザ (Internet Exploler,Safari) では動作が確認できなかった．今回は，前回の条件に追加して Temasys 社の WebRTC プラグインの動作も確認したが，createMediaStreamSource

© 2015 Information Processing Society of Japan

メソッドのサポートがされていないために，本手法の動作は確認できなかった．これらのユーザが Web ブラウザで音声認識機能を利用するためには，対応するブラウザをあらためてインストールするか，WebSocket 経由で外部アプリケーションからマイク信号を転送してもらう，WebSpeechAPI など少し手間のかかる方法を採用する必要がある．スマートフォン・タブレットでは，Android では Chrome，Firefox，Opera で動作が確認できた．しかし，OS のバージョンが古い，またはそれに伴ってブラウザのバージョンが 古い場合は，Media Stream and Capture または Web Audio API のどちらかが未サポートであり，動作を確認できなかった．また，Android Browser と iOS 端末上の各ブラウザでは，Media Stream and Capture が未サポートのため動作させることができなかった．これらの問題は，前稿においては非常に大きな問題であったが，現在では Android の標準ブラウザが Chrome となるなどその状況は改善されつつある．

また，最終的には HTML5 (JavaScript) でアプリケーションを記述できれば良いというのであれば，こういったブラウザの実装に依存しないよう，iOS, Anrdoid においては Apache Cordova のような HTML5 とネイティブアプリケーションのハイブリッドライブラリのプラグインとして本機能を実装するという手もある．

4. アプリケーションの記述方法

前述のように，提案システムは，昨年より NTT アイティ社から SpeechRec for Browser としてサービス提供されている．本サービスは，NTT ドコモのドコモ・デベロッパー・サポートと，NTT コミュニケーションズの SkyWay より利用可能となっている．

本稿では，提案法の具体的な利用方法について，今年，SkyWay の付加機能としてリリースされた SkyWay 音声認識機能の利用方法を交えて説明する．SkyWay 音声認識機能では、前述の HTML5 音声認識機能をクライアント（ブラウザ側）でマイク等を通じて入力した自由発話音声を、サーバ（SkyWay 側）と連携して、文字列として受け取る機能となっている．なお，WebSocket が疎通する環境でないと、正常に動作しないため，事前に確認をすること．[*1]

なお，以降で記載するサンプルコードの中で、SkyWay が提供している API キーが必要となる．実際に試用する場合は，SkyWay 公式サイトから開発者登録を行って頂きたい．サンプルでは、利用可能ドメインに「localhost」と書いた，API キーを生成しておくこと．また，サンプルコードは Github での配布も行っているため、そちらも合わせてご利用頂きたい．[*2]

ここで紹介するサンプルは、「音声認識を開始するボタン」と「認識結果を表示するエリア」の 2 要素のみで構成するシンプルな音声認識アプリケーションとする．

図 3　SkyWay の音声認識機能のサンプル

この機能を構成しているサンプルコードは以下の通りとなる．

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <title>SkyWay Speech Recognition Sample</
        title>
</head>
<body>

<!-- 音声認識開始用のボタン、および結果表示用のエリア -->
<button id="start_rec">start</button>
<div id="result"></div>

<!-- 必要なライブラリ群の読み込み  -->
<script src="https://code.jquery.com/jquery
    -1.11.3.min.js"></script>
<script src="https://skyway.io/dist/msgpack.
    codec.js"></script>
<script src="https://skyway.io/dist/
    libspeexdsp.js"></script>
<script src="https://skyway.io/dist/resampler.
    min.js"></script>
<script src="https://skyway.io/dist/speechrec.
    min.js"></script>

<!-- 音声認識用のコードサンプル -->
<script>
SpeechRec.config({
   'SkyWayKey':'3969c30a-2789-4e46-9155-79
       a188256633',
   'OpusWorkerUrl':'libopus.worker.js'
});

$("#start_rec").click(function(){
    SpeechRec.start();
    console.log("音声認識を開始します");
});

SpeechRec.on_proc(function(info){
    console.log(info.volume);
});

SpeechRec.on_result(function(result){
```

*1 WebSocket の疎通確認には，https://www.websocket.org/echo.html が利用できる
*2 https://github.com/nttcom/SkyWay-SpeechRec

© 2015 Information Processing Society of Japan

```
36      console.log(result.candidates);
37      $("#result").text(result.candidates[0].
            speech);
38  });
39  </script>
40
41  </body>
42  </html>
```

SkyWay 音声認識機能を利用するためには、大きく分けて 2 つの処理が必要になる．まず SpeechRec.config 関数を利用して、SkyWayKey および libopus.worker.js へのパスを設定する．サンプルでは，下記のように config を設定している．ここでの API キーは前述の通り，SkyWay の公式サイトから取得したものを入力する．

```
1  SpeechRec.config({
2      'SkyWayKey':'3969c30a-2789-4e46-9155-79
           a188256633',
3      'OpusWorkerUrl':'libopus.worker.js'
4  });
```

SpeechRec.config には，その他のパラメータも設定可能であり，サンプルコードの配布元の GitHub 上に記述されたドキュメントから確認できる．次に各種コールバックの設定を行う．サンプルでは 2 種類のコールバックを設定している．

```
1  SpeechRec.on_proc(function(info){
2      console.log(info.volume);
3  });
4
5  SpeechRec.on_result(function(result){
6      console.log(result.candidates);
7      $("#result").text(result.candidates[0].
           speech);
8  });
```

コンフィグおよびコールバックの設定が完了したら，実際に音声認識を開始 · 停止するための関数を呼び出す．サンプルでは，#start_rec ボタンがクリックされた際に呼び出す SpeechRec.start が音声認識を開始するための処理に該当する．

```
1  $("#start_rec").click(function(){
2      SpeechRec.start();
3      console.log("音声認識を開始します");
4  });
```

なお，開始だけではなく任意のタイミングで音声認識を終了させる SpeechRec.stop 等も利用可能である (サンプルでは省略している)．その他の関数は SpeechRec.config 同様に GitHub 上のドキュメントから確認できるため，そちらを参照願いたい．

5. まとめ

本稿では，我々の開発した，多くの HTML5 対応ブラウザ上で呼び出すことの出来る音声認識システムの詳細と，その利用方法について解説を行った．

提案システムが基礎としている WebRTC の仕様は着実に固まりつつあるが，いくつかのプラットフォームではその実装が適切でないために音声認識機能の完全なワンソース · マルチプラットフォームはまだ実現できていないのが現状である．今後は，より広いプラットフォームでの動作を実現するため，ブラウザの詳細な実装の確認，適切な実装の提案等を行ってゆく予定である．

参考文献

[1] 槙 優一， 鎌土 記良， 藤村 滋， 青野 裕司， 中山 丈二， 阪内 澄宇， 山田 智広， “音声認識機能を有する Web アプリケーションの実装と評価,” 情処技報, vol.159， no.6, pp.1-6, 2014.
[2] 鎌土 記良, 藤村 滋, 青野 裕司, 政瀧 浩和, 阪内 澄宇, “雑音に頑健な HTML5 対応クラウド音声認識プラットフォームの開発とその評価,” 春季音講論, 2015.
[3] NTT ドコモ docomo Developer support
https://dev.smt.docomo.ne.jp/
[4] NTT コミュニケーションズ SkyWay
http://nttcom.github.io/skyway/
[5] 田藤 千弘， 西村 竜一， “HTML5 による音声入力ウェブアプリケーションの開発キット,” 秋季音講論， pp.119–120, 2014.
[6] K. Wang, “SALT: A Spoken Language Interface for Web-based Multimodal Dialog System,” In Proceedings of International Conference of Spoken Language Processing(ICSLP), pp. 22412244, 2002.
[7] M Oshry, et al., “Voice Extensible Markup Language (VoiceXML) Version 2.1,” W3C Recommendation, 2007.
[8] R. Lau, et al., “WebGALAXY-Integrating Spoken Language and Hypertext Navigation,” In Proceedings of EUROSPEECH, pp.883 − 886, 1997.
[9] R. Nisimura, et al., “Speech-to-text input method for Web system using Javascript,” In Proceedings of IEEE Workshop on Spoken Language Technology(SLT), pp.209 − 212, 2008.
[10] R. Nisimura, et al., “Development of Speech In- put Method for Interactive VoiceWeb Systems,” Human-Computer Interaction, Novel Interaction Methods and Techniques, Lecture Notes in Computer Science, Vol. 5611, pp. 710 − 719,2009.
[11] C. Lattner and V. Adve, “LLVM: A compilation framework for lifelong program analysis & transformation,” in Code Generation and Optimization, 2004. CGO 2004. International Symposium on. IEEE, 2004, pp. 75 − 86.
[12] A. Zakai, “Emscripten: an llvm-to-javascript compiler,” in Proceedings of the ACM international conference companion on Object oriented program- ming systems languages and applications companion. ACM, 2011, pp. 301 − 312.
[13] M. Fujimoto, K. Ishizuka, and T. Nakatani, “Study of integration of statistical model-based voice activity detection and noise suppression, ” Proc. Interspeech'08, pp.2008-2011, 2008.
[14] 政瀧他, NTT 技術ジャーナル, no.18, vol.11, pp.15-18, 2006.

© 2015 Information Processing Society of Japan