Vol.2016-SLP-110 No.8
2016/2/6

KaldiにおけるCSJレシピの利用法

篠崎 隆宏[1,a)]　森谷 崇史[1]　田中 智大[1]　渡部 晋治[2]

概要：Kaldi はディープニューラルネットワーク (DNN) ベースの大語彙音声認識に対応した音声認識ツールキットである．Kaldi では音声や統計モデルの操作のための各種コマンドに加えて，様々なコーパスに対してほぼ全自動で認識システムを構築し認識実験を行うためのスクリプト（レシピ）が用意されている．本講演では，日本語話し言葉コーパス (CSJ) のデータを用いて大語彙音声認識を行う CSJ レシピについて紹介する．はじめに全体の動作を解説したのち，チュートリアル形式でスクリプトの説明を行う予定である．チュートリアルでは，短い時間で特徴量の抽出から DNN を用いた音声認識までの手順を試せる軽量版カスタムスクリプトを配布する．Kaldi をインストールしたノートパソコンと CSJ のデータがあれば，その場で実際にスクリプトを動作させながら各ステップの処理内容を習得することが可能である．

CSJ recipe for Kaldi

Shinozaki Takahiro[1,a)]　Moriya Takafumi[1]　Tanaka Tomohiro[1]　Watanabe Shinji[2]

1. はじめに

Kaldi [*1] [1] は国際的に開発が進められているフリーで利用できる高性能な音声認識ツールキットである．様々な認識アルゴリズムがサポートされており，大規模な音声データを効率よく扱うことが可能である．ツールキットは C++ で記述されたコマンド群，Kaldi 内で使用する openFST 等他のツールキットのインストーラ，様々なコーパスに対応して認識システムの構築から評価までをほぼ自動で行うレシピと呼ばれるスクリプト群などから構成されている．

大語彙日本語音声認識用のレシピとして，我々は日本語話し言葉コーパス (CSJ) [2] 用のスクリプトを開発し Kaldi のレポジトリから一般公開している [3]．本レシピは deep neural network(DNN) に基づいた認識アルゴリズムを用いるとともに，システムの各種メタパラメタをスーパーコンピュータを用いた大規模な進化計算により最適化したものであり，非常に高い認識精度を実現している [4]．利用者は Kaldi をインストールし CSJ のデータベースを用意すれば，スクリプトを一つ起動するだけで高性能なシステムを構築し評価することが可能である．なお，CSJ データベースは国立国語研究所より有料で配布されており [*2]，別途購入が必要である．

本講演ではレシピの動作の仕組みについて解説し，チュートリアル形式の実習を行う．実際のシステムの構築には GPU を用いた大規模な計算が必要となるが，実習はノートパソコンを用いて学習や認識のプロセスを短い時間で一通り試せるようにカスタマイズした軽量版レシピを配布して行う．本軽量版レシピはまた，各ステップを独立して試せるように工夫しており，Kaldi の各種コマンドを容易に理解できるように配慮したものである．

本講演は聴講のみの参加の場合は事前準備は不要であるが，チュートリアルに参加する場合は下記で説明するように Kaldi がインストールされたノートパソコンと CSJ のサブセットを参加者が用意する必要がある．Kaldi のインストールについては時間の制約からチュートリアル内では行わない．レシピの実行には Linux 環境が必要となる．Windows や MacOS がインストールされたノートパソコン上で本軽量版レシピを試す場合は，VirtualBox をホスト

[1] 東京工業大学
Tokyo Institute of Technology, Yokohama, Kanagawa 226–8502, Japan
[2] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139-1955, USA
[a)] www.ts.ip.titech.ac.jp

[*1] http://kaldi-asr.org/

[*2] http://pj.ninjal.ac.jp/corpus_center/csj/

表 1 チュートリアルで使用する CSJ データ（要持参）

音声	書き起こし
A01F0055.wav	CSJ/{DVD1 or MORPH}/SDB/core に含まれる
A01F0067.wav	全短単位長単位混合形式形態論データ
A01F0097.wav	*.sdb ファイル 201 講演推奨
A01F0122.wav	（左記の音声データに対応する sdb ファイルを含み，
A01F0132.wav	ある程度用意していただければ大丈夫です.）

OS 上にインストールしそのゲスト OS として Linux をインストールするのが手軽である．

以下では，第 2 章でチュートリアル参加の事前準備の方法を説明する．チュートリアルに参加する場合は，本章の内容が事前に準備されている事が前提となる．第 3 章では，Kaldi レポジトリより公開している，CSJ レシピの構成と動作について説明する．第 4 章では，チュートリアルで用いる軽量版レシピの動作について説明する．第 5 章で，他のいくつかのレシピについて簡単に紹介する．最後に，第 6 章でまとめを行う．

2. 実習用環境のセットアップ

チュートリアルは，Linux ディストリビューションの一種である 64bit 版 CentOS6.6 に Kaldi がインストールされている環境を前提に行う．チュートリアル用軽量版レシピの実行では GPU は不要であり，Core i5 程度の CPU と 1.5Gbyte 程度のメモリが利用できる Linux があれば動作する．Windows や MacOS が動作するホスト PC 上に VirtualBox をインストールしてその中で Linux を動作させる場合は，ホスト PC のスペックとして 4GByte 程度以上のメモリと 30GByte 以上のディスク空き容量が必要である．以下では，VirtualBox 上に実行環境をセットアップする方法について説明する．ゲスト OS のインストール手順以降については，PC に直接 Linux をインストールする場合と同じである．また，軽量版レシピでは CSJ のデータのうち表 1 に示す 5 講演分の音声データと対応する書き起こしデータが必要になる．スクリプトを動作させる場合，参加者がこれらのデータを持参する必要がある．

なお CSJ の全データを用いて最高性能の DNN 音響モデルの学習を行う場合は，より高いスペックのハードウエアが必要となる．例えば Core i7CPU と GeForce GTX 970 GPU および 32GByte のメモリを搭載し，CentOS が直接インストールされたパソコンを用いた場合，全学習工程を終えるのに約 24 日必要である．

2.1 VirtualBox と VirtualBox 内への CentOS のインストール (所要時間：約 1 時間)

パソコン上に VirtualBox をインストールし，さらに VirtualBox 内に CentOS をインストールする手順は以下のとおりである．Kaldi の実行に必要なライブラリ等についてもインストールが必要である．

(1) 念のため，事前にパソコンのバックアップを行う．
(2) VirtualBox をダウンロードし[*3]，インストーラの指示に従いインストールする．
(3) 64bit 版 CentOS6.6 のインストール用 ISO イメージをダウンロードする[*4]．
(4) VirtualBox の「仮想マシン」メニューから「新規」を選択し，指示に従って仮想マシンを作成する．仮想マシンの名前は任意だが，以下では vbkalditest とする．OS タイプは Linux，バージョンは RedHat(64bit) を選択する．仮想マシンに割り当てるメモリサイズは 1.5Gbyte 以上を指定する．ディスクサイズは 30Gbyte 以上を指定する．
(5) VirtualBox のメイン画面で作成した仮想マシン vbkalditest を選択し，「光学ドライブ」で「ディスクイメージを選択」を選択し，先ほどダウンロードした CentOS の ISO イメージファイルを指定する．起動ボタンを押して，仮想マシンを起動する．
(6) 仮想マシン内で，CentOS のインストーラが起動するので，指示に従ってインストールする．ストレージデバイスのデータについて警告がでるが，仮想ディスクはブランク状態なのでデータの廃棄を選択する．（ディスクサイズが仮想マシン作成時に指定したものと同じであることを確認すること）．ホスト名は任意だが，以下では suzukake とする．インストールが終わると再起動を求められるので，CentOS のインストーラに表示されている再起動ボタンを押す．
(7) インストールした CentOS が起動する．初回起動時，ライセンスの同意や，初期設定画面が表示されるので，指示に従い進む．作成するユーザー名は任意だが，以下では kalditest とする．
(8) ログイン画面が表示されたら，先ほど作成したアカウント (kalditest) でログインする．
(9) ネットワーク接続を有効化するため，CentOS 内で「システム」→「設定」→ 「ネットワーク接続」を選択する．system eth0 を選択し，「編集」ボタンを押して「自動接続する (A)」にチェックを入れ，「適用」ボタンを押す．
(10) 端末を立ち上げ，以下のコマンドを入力して OS を最新版に更新する．

```
[kalditest@suzukake ~]$ su
[root@suzukake kalditest]$ yum upgrade
```

(11) OS のアップデートが完了したら，一回リブートする．OS が再起動したら，再度 kalditest でログインし端末を立ち上げ su コマンドを実行して root ユーザーになり，Kaldi で必要な数値計算ライブラリをインストールする．数値計算ライブラリにはいくつか選択肢があ

*3 https://www.virtualbox.org/
*4 https://www.centos.org/

るが，以下では SSE3 に対応した atlas をインストールする例を示す．

```
$ yum install atlas-sse3-devel
```

(12)CSJ レシピを使用する際は日本語のテキスト処理のために nkf が必要になるため，インストールする．

```
$ yum install nkf
```

(13)もし GPU を使用する場合は，cuda [*5]をインストールする．(cuda に対応した GPU が必要である))

2.2 CentOS への Kaldi のインストール (所要時間：約 1 時間)

CentOS に Kaldi をインストールする手順は以下のとおりである．Kaldi のインストールはユーザー権限または root 権限で行う．以下ではユーザー権限でホームディレクトリの下にインストールする場合について示す．

(1) ユーザーアカウント (kalditest) でログインし，端末を立ち上げる．git コマンドにより，kaldi のソースレポジトリからソースコード一式をダウンロードする．以下で<URL>は，https://github.com/kaldi-asr/kaldi.git である．

```
[kalditest@suzukake ~]$ cd ~
[kalditest@suzukake ~]$ git clone <URL>
```

インストール手順の詳細は各ディレクトリにある INSTALL ファイルに書かれている．

(2) kaldi をコンパイルする．

```
[kalditest@suzukake ~]$ cd kaldi
[kalditest@suzukake ~]$ cd tools
[kalditest@suzukake ~]$ make
[kalditest@suzukake ~]$ cd ../src
[kalditest@suzukake ~]$ ./configure
[kalditest@suzukake ~]$ make depend
[kalditest@suzukake ~]$ make
```

(3) 以上の操作で kaldi が依存しているツール群も大部分自動でインストールされるが，一部ライセンスの制約等により手動インストールが必要なものもある．CSJ レシピでは SRILM ツールキット [*6] を使用するので，別途パッケージを入手した上でインストールする．SRILM のパッケージを kaldi/tools ディレクトリにコピーし，kaldi/tools 以下に用意されている SRILM 用のインストーラを実行すればよい．

```
$ ./install_srilm.sh
```

(4) tools/openfst-1.3.4 のディレクトリの設定が，デフォルトでは他人 (other) がアクセスできない設定になっている場合がある．必要に応じてパーミッションを変更する．

```
$ cd ~/kaldi/tools
$ chmod o+rx openfst-1.3.4
```

(5) インストールが正しく行えたか，いくつかコマンドを引数なしで実行して確認する．最初に，各コマンドに path を通すため path.sh を読み込む．

```
$ cd ~/kaldi/egs/csj/s5/
$ source path.sh
$ copy-feats
$ nnet-copy
```

3. Kaldi 用 CSJ レシピ

Kaldi CSJ レシピ (egs/csj/s5:以下，これをトップディレクトリとする) は CSJ のデータを用いて大語彙音声認識システムの構築と評価をほぼ自動で行う．システムパラメタは Switchboard コーパス用のレシピ (egs/swbd/s5c) を参考にしつつ大規模な進化計算により最適化してある．図 1 に CSJ レシピの処理の流れを示す．認識評価には CSJ 標準評価セット 1，2，3(各 10 講演) を用いており，認識処理は重み付き有限状態トランスデューサ (WFST) をもとに OpenFst により行っている．現在の本レシピの認識性能を表 2 に示す．最新の認識結果はトップディレクトリ下の RESULTS ファイルに記述されている．以下に図 1 に従って学習の手順を示す．

0. モデルを作成するデータの準備

始めに CSJ の短単位長単位混合形式形態論データ (sdb ファイル) の時間ラベルと形態素情報をもとに，発話単位の書き起こしラベルと語彙リストを作成する．これらの処理はすべて local/csj_make_trans 内のスクリプトで行われている．そして得られた書き起こしラベルと語彙リストをもとに，3-gram 言語モデルを学習する．この言語モデルの学習には SRILM ツールキットが必要である．なおこの言語モデルの学習には，CSJ に含まれる全講演データ (約 750 万語) を用いており，語彙サイズは 72k である．また，音響モデルの学習データ量は「学会講演」(967 講演) と「その他の音声」(19 講演) の計 986 講演のみを用いた場合と，対話形式を除く全講演 (3212 講演) を用いた場合の 2 通りのパターンを用意している (デフォルトは 986 講演)．これらの学習データ量の選択は local/csj_data_prep.sh で行うことが可能である．なお，各学習セットを用いたときの計算時間は 986 講演の場合に約 15 日，3212 講演の場合に 24 日であった．計算環境は Core i7-5820K CPU と GeForce GTX 970 GPU，および 32GByte のメモリを搭載したパソコンである．最後に WAV 形式の音声データから MFCC 特徴量を抽出し，Kaldi 独自のアーカイブ形式で保存する．特徴量は MFCC の他に FBANK と PLP も選択可能である．

1. GMM-HMM 最尤学習 (学習セットのサブセット)

GMM-HMM の学習ステップでは，はじめに 13 次元の

[*5] https://developer.nvidia.com/cuda-zone
[*6] http://www.speech.sri.com/projects/srilm/

表 2 各 CSJ 標準評価セットの WER [%]

	Training Step	Training set (Academic etc.)				Training set (All)			
		Eval1	Eval2	Eval3	AVG	Eval1	Eval2	Eval3	AVG
GMM	ML	15.3	14.0	17.3	15.5	15.7	13.4	15.5	14.8
	+ bMMI	14.0	12.2	16.0	14.1	14.0	11.5	13.3	12.9
	+ f-bMMI	13.9	12.3	15.6	13.9	14.0	11.3	13.0	12.8
DNN	Cross entropy optimization	11.9	10.2	13.9	12.0	11.6	9.2	10.4	10.4
	sMBR sequential training	11.3	9.4	12.5	11.1	10.6	8.7	9.3	9.5
	+ Lattice re-generation	10.9	9.2	12.2	10.7	10.3	8.6	9.1	9.4

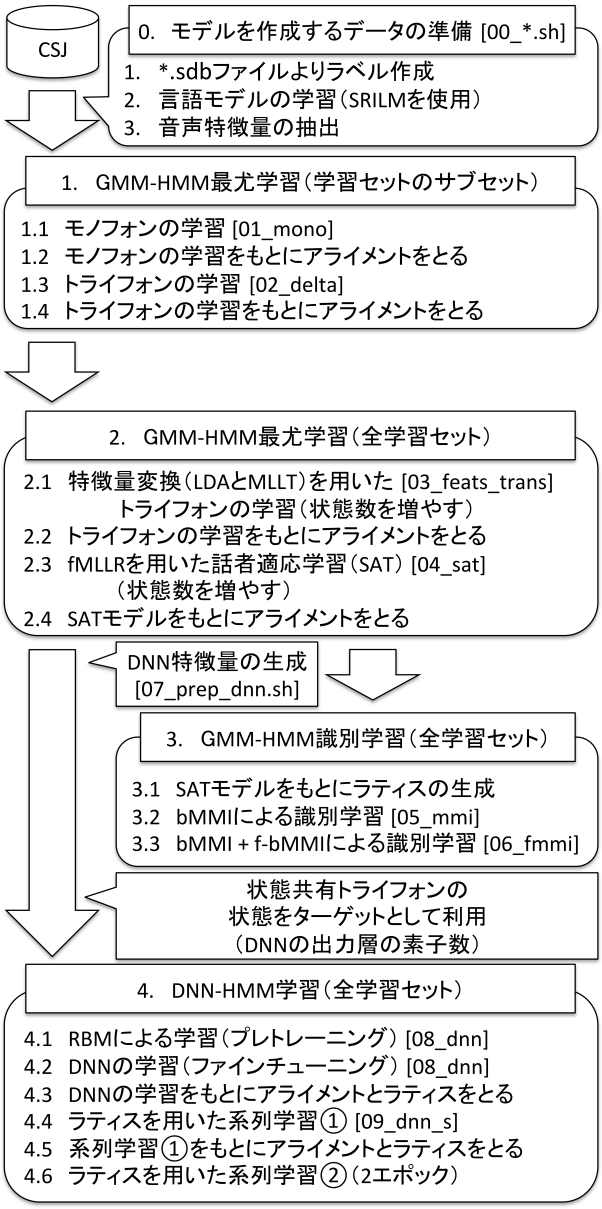

図 1 CSJ レシピのフローチャート.

MFCC とその動的特徴量（Δ と ΔΔ）を用いてモノフォンを学習し，ついでトライフォンを学習する．音素は 3 状態の left-to-right 型 HMM であり，無音は 4 状態でモデル化している．音素と無音の各状態数は utils/prepare_lang.sh のオプションで変更することができる．トライフォンの状態共有を行うための質問リストは，Kaldi のプログラムにより自動生成される．この段階では，処理の高速化のため一部の学習データのみを用いている．

2. GMM-HMM 最尤学習（全学習セット）

次に，そのモデルを用いて全学習データのアライメントをとり，モデルの再学習を行う．特徴量には，9 フレームの静的 MFCC を結合した 117 次元の特徴量を線形判別分析（LDA）を用いて 40 次元に圧縮した上で，最尤線形変換（MLLT）を適用したものを用いる．ついで，特徴量空間最尤線形回帰（fMLLR）を用いた話者適応学習（SAT）を行う．なお，トライフォンの学習では学習ステップが進むごとに逐次ガウス混合数とトライフォンの状態数を増やしている．SAT モデルを用いた認識処理では評価データに対して教師なし適応を行っている．この SAT で得られたアライメントと状態共有決定木を用いて DNN-HMM 学習を行う．

3. GMM-HMM 識別学習（全学習セット）

最尤学習で得たモデルをもとに学習データのアライメントとラティスをとり，さらに識別学習を行う．識別学習では，このラティスによる認識仮説と正解ラベルの相互情報量を最大化するブーステッド最大相互情報量（bMMI）[5] 基準を用いた後，ブーステッド特徴量空間最大相互情報量（f-bMMI）[5] 基準も用いて学習を行う．なお，この学習で得られる全ての結果は DNN-HMM 学習には用いていない．

4. DNN-HMM の学習と認識

GMM-HMM の最尤学習をもとに，DNN-HMM の学習を行う．特徴量には LDA により次元圧縮した 40 次元の特徴量に fMLLR を適用したものを用いる．これらの変換行列は GMM-HMM 最尤学習で得られたものである．はじめに，制限付きボルツマンマシン（RBM）を用いたプレトレーニングと，クロスエントロピーを誤差尺度とした誤差逆伝播法によるファインチューニングを行う．次に学習した DNN をもとにアライメントとラティスを生成し，状態レベルのベイズリスク最小化（sMBR）による系列学習 [6] を行う．さらにこの系列学習の結果からアライメントとラティスを再生成し，系列学習を 2 エポック行う．

4. 軽量版カスタムレシピを用いた実習

本章ではチュートリアルに用いるカスタムレシピ (s5_demo) の内容を記載する．このカスタムレシピは通

常の CSJ レシピの軽量版となっており，CSJ の小さなサブセットを用いて Kaldi の根幹となる部分を重点的に説明する．チュートリアルに参加する場合は CSJ という名前のディレクトリを s5_demo の下に作成し，表 1 に示すデータ（*.wav，*.sdb）を全てこのディレクトリに置くことで下記の実習の準備が整う．カスタムレシピ内の各実行スクリプトおよびディレクトリは図 1 中の [*] にそれぞれ対応している．s5_demo 内のスクリプト（*_synops.sh）は，各ステップで中心的な役割を担うコマンドの概略を記述している．また，付録に研究を行う上で便利だと思われる Kaldi のコマンドを一部記す．

4.1 GMM-HMM 学習

GMM-HMM 学習ではモノフォンの学習 [01_mono] とトライフォンの学習 [02_delta] 部分を重点的に説明する．これは以降の GMM-HMM 最尤学習でも上記の学習手順が根幹となっているためである．モノフォンの学習では初期アライメントの作成と音響モデルの更新方法を，トライフォンの学習では状態共有決定木の作成手順を中心に説明する．

4.2 DNN-HMM 学習

DNN-HMM 学習では RBM による学習と DNN の学習 [08_dnn] 部分を重点的に行う．RBM による学習では入力層 (GB-RBM) および中間層 (BB-RBM) のメタパラメタの設定から Contrastive Divergence 法による学習と各層の結合方法を説明する．DNN の学習では出力層に対応する各状態の事前確率の計算方法，出力層および DNN の作成と誤差逆伝播法による学習方法を説明する．

5. 他のレシピの紹介

筆者らは，CSJ 以外でも，主に以下にあげる遠隔発話音声認識タスクにおいて，Kaldi レシピの作成もしくはメインテナンスを精力的に行っている．

- kaldi/egs/chime2: 2nd CHiME challenge track 2 [7]
- kaldi/egs/chime3: 3rd CHiME challenge [8]
- kaldi/egs/reverb: REVERB challenge [9]
- kaldi/egs/ami: AMI task [10]

特に 3rd CHiME challenge において，本レシピはチャレンジの公式ベースラインとして用いられており，また 2015 年 12 月には，[11] の結果に基づいた，新しいベースラインも Kaldi 公式レポジトリに公開されている．

6. まとめ

日本語話し言葉コーパス (CSJ) のデータを用い, 非常に高性能な大語彙日本語音声認識システムを構築し認識評価を行う CSJ レシピについて紹介した．レシピは GMM-HMM を学習した後その結果を用いて DNN-HMM を学習する仕組みとなっており，それら各ステップがどのように動作しているのかについて解説を行った．チュートリアルはノートパソコンを用いて短い時間で各ステップを試せるよう，軽量なカスタム版レシピを用いて行った．CSJ の全データを用いた実際のシステム構築では，GPU を用いた大規模な計算が必要である．CSJ レシピは現在も改良中であり，今後リカレントニューラルネットワーク言語モデルを組み込んだバージョン [12] を公開する予定である．

謝辞 篠崎隆宏，森谷崇史，田中智大は本研究について JSPS 科研費 26280055 の助成を受けた．渡部晋治は MERL より支援を得た．

参考文献

[1] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlıcek, P., Qian, Y., Schwarz, P., Silovskı, J., Stemmer, G. and Veselı, K.: The Kaldi speech recognition toolkit, *Proc. ASRU* (2011).

[2] Furui, S., Maekawa, K. and Isahara, H.: A Japanese national project on spontaneous speech corpus and processing technology, *Proc. ASR'00*, pp. 244–248 (2000).

[3] Moriya, T., Shinozaki, T. and Watanabe, S.: Kaldi recipe for Japanese spontaneous speech recognition and its evaluation, *Fall Meeting of ASJ 2015*, pp. 155–156 (2015).

[4] Moriya, T., Tanaka, T., Shinozaki, T., Watanabe, S. and Duh, K.: Automation of system building for state-of-the-art large vocabulary speech recognition using evolution strategy, *Proc. IEEE ASRU* (2015).

[5] Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B. and Saon, G.and Visweswariah, K.: Boosted MMI for model and feature-space discriminative training, *Proc. ICASSP*, pp. 4057–4060 (2008).

[6] Vesely, K., Ghoshal, A., Burget, L. and Povey, D.: Sequence-discriminative training of deep neural networks, *Proc. Interspeech*, pp. 2345–2349 (2013).

[7] Vincent, E., Barker, J., Watanabe, S., Le Roux, J., Nesta, F. and Matassoni, M.: The second 'CHiME' speech separation and recognition challenge: Datasets, tasks and baselines, *Proc. IEEE ICASSP*, pp. 126–130 (2013).

[8] Barker, J., Marxer, R., Vincent, E. and Watanabe, S.: The third 'CHiME' Speech Separation and Recognition Challenge: Dataset, task and baselines, *Proc. IEEE ASRU*, pp. 504–511 (2015).

[9] Kinoshita, K., Delcroix, M., Yoshioka, T., Nakatani, T., Sehr, A., Kellermann, W. and Maas, R.: The reverb challenge: Acommon evaluation framework for dereverberation and recognition of reverberant speech, *Proc. IEEE WASPAA*, pp. 1–4 (2013).

[10] Hain, T., Wan, V., Burget, L., Karafiat, M., Dines, J., Vepa, J., Garau, G. and Lincoln, M.: The AMI system for the transcription of speech in meetings, *Proc. IEEE ICASSP*, pp. 357–360 (2007).

[11] Hori, T., Chen, Z., Erdogan, H., Hershey, J., Roux, J., Mitra, V. and Watanabe, S.: The MERL/SRI system for the 3rd CHiME challenge using beamforming, robust feature extraction, and advanced speech recognition, *Proc. IEEE ASRU*, pp. 475–481 (2015).

[12] Tanaka, T., Moriya, T., Shinozaki, T., Watanabe, S. and Hori, T.: Implementation and evaluation of RNN language model for CSJ recipe on Kaldi, *to appear in Spring Meeting of ASJ 2016* (2016).

表 A·1 rspecifier

rspecifier	意味
ark:foo.ark	アーカイブファイル foo.ark を読み込む
scp:foo.scp	スクリプトファイル foo.scp を読み込む
ark:-	標準入力から入力する
ark:gunzip -c foo.gz |	圧縮された foo.gz を読み込む

表 A·2 wspecifier

wspecifier	意味
ark:foo.ark	アーカイブを foo.ark に書き込む
scp:foo.scp	スクリプトを foo.scp に書き込む
ark:-	バイナリ形式で標準出力に出力する
ark,t:-	テキスト形式で標準出力に出力する
ark,t:|gzip -c >foo.gz	テキスト形式の出力を圧縮し foo.gz に書き込む
ark,scp:foo.ark,foo.scp	アーカイブ・スクリプトの両形式で同時に書き込む

付　　録

A.1 Kaldi コマンド

本付録では筆者らが研究を行う際に便利だと思われる Kaldi のコマンドを一部抜粋したものである．Kaldi の多くのコマンドでは，入力ファイルや出力ファイルの指定に共通のインターフェースが用いられている．入力ファイルは rspcecifier を用いて指定し，出力ファイルは wspecifier を用いて指定する．Kaldi ではこれらのインターフェースがスクリプト中のパイプライン処理に多用されている．表 A·1, A·2 [*7]に各インターフェースの表記法を記す．Kaldi において用いられるアーカイブ（ark）形式のファイルは，複数の発話の音声特徴量その他をまとめたものである．またスクリプト（scp）は，アーカイブファイル中の各発話を指定するのに用いられるインデックスファイルである．なお，コマンド一覧中の” \”は改行記号を表す．

A.1.1 バイナリ・テキスト形式の変換

以下では Kaldi で扱われる各種バイナリ形式データをテキスト形式に変換するためのコマンド一覧を示す．テキスト形式からバイナリ形式への変換も同様に可能である．

- 特徴量ファイル:
  copy-feats ark:foo.ark ark,t:destfile または
  copy-feats ark:foo.scp ark,t:destfile
- ラティスファイル（圧縮形式）:
  copy-feats ”ark:gunzip -c foo.lat.ark|” ark,t:destfile
- 単語リスト:
  copy-int-vector ark:words.bin ark,t:destfile
- GMM-HMM モデルファイル:
  gmm-copy –binary=false hmm.mdl destfile
- GMM-HMM 統計量ファイル:
  （本コマンドは，本来統計量ファイルの操作用である）
  gmm-sum-accs –binary=false destfile hmm.acc
- Diagonal GMM モデルファイル:
  gmm-global-copy –binary=false gmm.mdl destfile
- Diagonal GMM 統計量ファイル:
  （本コマンドは，本来統計量ファイルの操作用である）
  gmm-global-sum-accs –binary=false destfile foo.acc
- DNN ファイル（dbn または nnet ファイル）:
  nnet-copy –binary=false foo.(dbn/nnet) destfile
- アライメントファイル:
  ali-to-pdf hmm.mdl \
  ”ark:gzip -c ali.*.gz|” ark,t:destfile
- 決定木ファイル:
  copy-tree –binary=false tree destfile
- 整数形式（テキスト）の認識結果を単語に置換:
  utils/int2sym.pl -f 2: word.txt int.hyp

A.1.2 ファイル情報の表示

音響モデルファイルなどの概要を表示するコマンドを下記に示す．

- GMM-HMM ファイル:
  gmm-info foo.mdl または，hmm-info foo.mdl
- GMM ファイル:
  gmm-global-info foo.dubm
- DNN ファイル:
  nnet-info foo.nnet
- 音響モデルの情報:
  am-info foo.mdl
- 特徴量の次元数:
  feat-to-dim scp:feats.scp - または,
  feat-to-dim ark:feats.ark -
- 決定木ファイル:
  tree-info tree

A.1.3 その他のコマンド

その他，いくつかの基本的なコマンドを以下に示す．

- アライメントの確認:
  show-alignments prones.txt foo.mdl ark:ali.*
- 状態決定木ファイルの質問リスト
  およびリーフノードの PDF 作成:
  draw-tree phones.txt tree | \
  dot -Tps -Gsize=8,10.5 | ps2pdf - tree.pdf
  graphviz のインストールが必要
- 認識率の計算
  compute-wer –text –mode=present \
  ark:ref.txt ark,p:hyp.txt
  (mode の説明)
  present：認識結果に含まれる発話のみを評価
  all：認識結果がない場合，結果が空として集計
  strict：認識結果がない場合エラー終了

[*7] http://www.danielpovey.com/files/Lecture1.pdf