月刊：音声認識システムを作る

Shigeki Karita

April 7, 2018

Contents

Abstract

D 言語で作る大規模音声認識システム

1 音声認識の概要

みなさんは音声認識のことを知ってますか？私は，例えば大学の授業といった場面でも，音声認識の全体像を把握することは難しいだろうと思っています．その理由は長年の地道な改良によって統計・数学的な背景が多岐に渡り

すぎたせいだと思います．そんな中で 2010 年代辺りから音声認識に新たな派閥ができました．

- 従来：重み付き有限状態トランスデューサ (WFST)[?] を核とした階層的なモデルを複合した大規模なシステム [?] (Kaldi, HTKなど)
- 最近：音声からテキストへの変換を直接モデル化した単純なシステム (ESPnetなど)

この流れで言うと，「ああ，この人は最近の音声認識はシンプルだから，そっちの解説を始めるんだな」と思うでしょう．たしかに私は普段，後者の新しい方式を研究しており，紹介しやすいものですと様々な最先端技術を実装する OSS プロジェクト ESPnetにも貢献しています．

ところがどっこい，この解説記事では前者しか扱いません．無謀にも地道に古典的な音声認識システムをコツコツ作っていこうという話です．普通に作っても面白くないので，既存の音声認識ライブラリは使わず，純粋に D 言語だけで一から作ろうと思います．

1.1 なぜ作るのか

作る理由としては，いままで何冊も音声認識に関する本や論文を読みましたが，Kaldi の認識スクリプトの最初から最後まで実際は何が動いているかは私はわかりません．というのも歴史的に様々な先端技術 (音響モデル，言語モデル，WFST) を複合しているため，全てを理解している人は少ないと思います．

D 言語を採用した理由は実行速度やメモリ使用量の点で有利だからです．例えば Python や Haskell で書くとリスト処理などパイプラインの実装は楽そうです．しかし経験的には実行速度が遅かったり，多くのメモリを消費します [?]．結局 WFST の演算などコアな部分は C/C++で書くことになるでしょう．それならば C/C++と同レベルのバイナリを作れて，標準ライブラリなどでリスト処理などアルゴリズムが充実した D 言語が適していると思いました．

1.2 大まかなプラン

これから作るものの大雑把なリストです．性能評価の実験では，無料で入手できる大規模なTEDLIUM コーパス(有名人の CreativeCommons な講演 TED talk の音声と字幕を元にしたデータセット) を使います．

1. 音声特徴量: まさに音声認識の秘伝のタレ．音声は巨大なので認識しやすく STFT などで変換します．最終的に MFCC 特徴量を作ります．
2. 音響モデル: 音声特徴量をいきなりテキストへ変換するのは難しいので，音声特徴量から音素へ変換するモデルを作ります．最終的にオーソドッ

クスな隠れマルコフモデル (HMM) と混合正規分布モデル (GMM) を作ります.

3. 発音辞書: 英語辞書とかに乗ってる，単語と音素列の対応を羅列したデータです．音韻学者を呼ばないとつくれないので，出来合いのデータに対して WFST への変換ツールを作ります.

4. 言語モデル: TEDLIUM コーパスについてくる ARPA 形式のデータを変換するか，データセットから学習します.

個人の趣味プロジェクトかつ，大規模な方を選んでしまったので月 1 くらいのペースでランダムに更新できればなと思います．半年くらいで何か認識できるようになれば嬉しいくらいのペースです.

1.3 実装における情報源

- The HTK Book [?] - 歴史的な経緯などもまとまった随一のドキュメントだと思います.
- Kaldi `http://kaldi-asr.org/doc/` - 2018 年現在，最もアクティブな音声認識ツールキットでしょう．文章は読みにくいけど，コードにもコメントが多い印象.
- [?] - OpenFST のサイトにも紹介されている音声認識における WFST の基礎的なアルゴリズムの説明.
- [?] - 同じく OpenFST のサイトに紹介されている WFST アルゴリズムのサーベイ論文

2 Kaldi で学ぶ音声の階層構造と統計モデル

まず実際に既存の音声認識ツールキット (Kaldi) が何をしているのか解説したいと思います．Ubuntu では次のようにダウンロード・コンパイルします．とりあえずニューラルネット音響モデルはまだ使わないので CUDA は必要ないです.

```
sudo apt-get install subversion libopenblas-dev libgfortran-7-dev libblas-dev
sudo apt-get liblapacke-dev checkinstall
sudo ln -s /usr/lib/x86_64-linux-gnu/libopenblas.so /usr/local/lib/
git clone https://github.com/kaldi-asr/kaldi.git
git checkout 1a1e265ae8386910a3967010c845fbd29ddb25e4
cd kaldi/tools
```

```
make -j4
cd ../src
./configure --shared --use-cuda=no --openblas-root=/usr/local
make -j4
```

とりあえず我々がターゲットにしている TEDLIUM コーパスのレシピ (※実験のスクリプトのこと) を，動かしてみましょう．とりあえず本記事の対象である triphone の GMM-HMM 音響モデルの評価 (stage 10) まで走ったら exit して大丈夫です．

```
cd kaldi/egs/tedlium/s5_r2
./run.sh |& tee log
```

このスクリプト run.sh をひらくと，大まかな処理の段階わけがされています．

```
if [ $stage -le 0 ]; then
  local/download_data.sh
fi
```

大まかに各ステージでやっていることを説明すると，こんな感じです．

- stage 0-1. TEDLIUM コーパスと言語モデルのダウンロードと整備 (私の環境では 5 時間かかりました)
- stage 2. 発音辞書の準備 (WFST 化)
- stage 3-5. 言語モデルの準備 (未知語などの前処理，学習，WFST 化)
- stage 6. MFCC 特徴量の抽出，入力データの正規化
- stage 7. データセットの縮小 (最も短い 10000 発話を選択)
- stage 8. 初期モデルとして音素単位の HMM 音響モデルを縮小データで学習
- stage 9. 初期モデルで作ったアライメントを元に，3 組音素 HMM(tri1) を全データで学習
- stage 10. tri1 をデコードして認識性能の評価

音素 HMM とは大雑把に言うと音素ごとに一つ HMM を用意して，それぞれの音声の確率をモデル化しています．また，「アライメント」とは音声の各時刻がどの音素なのかという情報で，「3 組音素」とは単純に現在の音声フレームに対する音素だけでなく前後の音素は何だったのかという情報も合わせた単位で HMM を作っています．

3 音声特徴量

4 音響モデル

5 発音辞書

6 言語モデル

7 重み付き有限状態トランスデューサ

8 認識デコーダ

9 系列学習

References

[1] Takaaki Hori and Atsushi Nakamura. Speech Recognition Algorithms Using Weighted Finite-State Transducers. *Synthesis Lectures on Speech and Audio Processing*, 9(1):1–162, jan 2013.

[2] Mehryar Mohri. Weighted Automata Algorithms. pages 213–254. 2009.

[3] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech Recognition with Weighted Finite-State Transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[4] T. Shinozaki, S. Furui, Y. Horiuchi, and S. Kuroiwa. Pipeline decomposition of speech decoders and their implementation based on delayed evaluation. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–4, Dec 2012.

[5] Steve Young, Mark Gales, Xunying Andrew Liu, and Phil Woodland. The HTK Book. 3.5(alpha1), 2015.

Emacs 25.2.2 (Org mode 9.1.8)