

November 22, 2019

ラベル付けのための音声のテキスト化

沼田 賢一

富山県立大学 情報基盤工学講座

November 22, 2019

本研究の背景

現代, 多くの人がスマートフォンやウェアラブルデバイスを持ち歩くことが一般的であり, 急速な情報技術の発達から, 個人の生活や行動をデータとして取得, 記録することが可能となっている.

このようなスマートフォンやウェアラブルデバイスを使用して取得して得られる人間の活動の記録のデータをライフログデータという. ライフログデータは解析を行うことで, 個人の健康管理に活かしたり, ビジネスとしてターゲティング広告にも使われ社会に活かしたりできると考えられている.

目的

従来の提案手法による行動パターン解析における分類精度の向上を図る。

分類精度向上のための提案手法

- ① 入力信号によるラベル付け
- ② テキストデータのベクトル化

入力信号によるラベル付け

4/16

ライフログデータの収集時に画像のテキスト変換と同様に, 同時に音声マイクを用いて, 現在の行動を認識できるような簡易的なテキスト変換 API を用いて定期的に入力信号として収集することでライフログデータの解析する際に入力信号を教師データとして含めることで分類精度を向上させる.

4/16

テキストデータのベクトル化

5/16

Word2vec は Mikolov らにより発表された単語群のベクトル化手法である。Word2vec を自然言語処理の分野で応用した例に、日本語動詞・形容詞に関する類似度データセットの構築やイベント情報の分類など数多くの利用例がある。本研究では、カメラから取得した画像を画像認識 API を通してテキスト化させたものに対して Word2vec を用いてベクトル化させる。そして家具、動作、食べ物といった 6 つの基準を設けそれらとの類似度の値を求める。

5/16

目的

6/16

今回の目的は、行動識別の時に用いるラベルをマイクを通して音声
をテキスト化したものを保存できるようにすること。
マイクは、購入していないので以下のカメラを用いて音声の入力に
使った。



6/16

- 1 google speech to-text
- 2 julius
- 3 google assistant

60 分を超えると有料

機能	標準モデル (動画と拡張音声通話を除くすべてのモデル)		プレミアム モデル* (動画、拡張音声通話)	
	0～60 分	60 分超～100 万分まで	0～60 分	60 分超～100 万分まで
音声認識 (データロギングなし-デフォルト)	無料	\$0.006/15 秒**	無料	\$0.009/15 秒**
音声認識 (データロギングあり)	無料	\$0.004/15 秒**	無料	\$0.006/15 秒**

julius とは、オープンソースの音声認識エンジンのこと。

Raspberry Pi

音声認識エンジン Julius (ジュリアス)

1 入力された音声データの特徴を見つける



2 ① の特徴に近い文字列を探す



3 ② の文字列と登録しておいた言葉を比較して
近い言葉を割り出す

julius を使える環境を作って、日本語の音響・言語データが収められているディクテーションキットを用いて試してみたが、精度はかなり悪かった

```
<<< please speak >>>Warning: strip: sample 0-95 has zero val
pass1_best: 図 の
sentence1: って 思う 。
pass1_best: だ 。<input rejected by short input>
pass1_best: 明日 、<input rejected by short input>
pass1_best: あの 年
sentence1: 殴る 。
pass1_best: <input rejected by short input>
pass1_best: おはよう 無線
sentence1: おはよう 無線 。
pass1_best: 彼の
sentence1: 帰ろう 。
pass1_best: おはよう 、
sentence1: うん うん 。
pass1_best: かく し 、<input rejected by short input>
pass1_best: だよ 。<input rejected by short input>
pass1_best: 非<input rejected by short input>
pass1_best: と 、
sentence1: S 。
pass1_best: って 、<input rejected by short input>
pass1_best: 二 日 、<input rejected by short input>
pass1_best: と 、<input rejected by short input>
pass1_best: そして 、
sentence1: ストップ 。
```

「おはよう」といった結果の図

精度を上げるには精度のいい辞書データを作らなくてはいけない。
独自辞書を作成するには、読みと音素、構文、語彙といった4つの
要素に対応するファイルを作成しなくてはいけない。

google assistant とは、Google が開発した AI アシスタント。
今回はアシスタントの結果は無視で入力 (認識) だけしてもらう。
使用するために、google cloud platform でプロジェクトを新規作成
して、API を有効化させる。認証が必要なので、OAuth でトークン
を発行する。最後にラズパイで google assistant api の SDK を入れ
て設定する。

実行してみると

```
12:11:21.2129:ERROR:speech_processing_task.cc(646) Detected communication error: Downstream closed prematurely
^C(env) pi@raspberrypi:~$ google-assistant-demo
ON_MUTED_CHANGED:
  ['is_muted': False]
ON_START_FINISHED
Segmentation fault
(env) pi@raspberrypi:~$ google-assistant-demo --project-id my-project-id --device-model-id my-model-id
```

このようなエラーがでてしまった。

調べていくと google assistant API を使うための SDK は google assistant library と google assistant service の 2 種類があったが、google assistant library が今年の 6 月に廃止されてしまった。hotword(ok google や hey siri) で assistant を起動できるのが google assistant library のほうのみあった。hotword で assistant を起動するには、もう一つの google assistant service に含まれるキーボードの入力で assistant を起動するファイルを編集して作らなければいけない。

	google assistant library (廃止)	google assistant service
hotword(ok google)	○	×
push to talk(キーボードで実行)	○	○

今は、assistant を起動するのにキーボード入力を取り除いたところ。
(常に起動している状態)

hotword の検出をさせるために、snowboy という hotword 検出できるソフトを用いて組み合わせている途中。

今後の課題

- 1 snowboy を組み合わせて起動できるようにする。
- 2 hotword を言って google assistant を起動した後、保存する仕組みを作る。
- 3 保存されたものを形態素解析して場所と行動に分ける