tdmelodic

Hideyuki Tachibana

2022 年 07 月 04 日

目次

Tokyo Dialect MELOdic accent DICtionary

これは、ニューラルネットワークにより、日本語（東京方言）の大規模なアクセント辞書を自動生成するモジュールです。

このモジュールの目的は、日本語の大規模アクセント辞書を自動生成することです。そのために UniDic と NEologd という既存の二つの辞書を利用します。UniDic では正確なアクセント情報が提供されていますが、扱える語彙がやや限定されています。一方 NEologd は非常に大規模な語彙を扱っている一方、アクセント情報を提供していません。

第1章　予備知識

1.1 日本語ピッチアクセント

日本語はピッチアクセント言語です。英語では単語中の各音節の強弱に差をつけることで単語にアクセントをつける一方、日本語では単語中の各拍子の高さに差をつけることによりアクセントを表現します。アクセントは、ほかの多くの言語でそうであるように、日本語の話し言葉においてもとても大切です。

日本語のアクセントには方言ごとにかなりの違いがあります。このモジュールでは、現在最も影響力な大きな方言のひとつである、現代東京方言のアクセントを扱います。

1.2 東京方言のアクセントについて

1.2.1 上昇と下降（アクセント核）

現代東京方言のアクセントモデルでは、ピッチの急激な変化を引き起こす 2 種類の契機が想定されています。

- 上昇　[　ここでピッチを上げる
- 下降　]　ここでピッチを下げる（これはアクセント核ともいう）

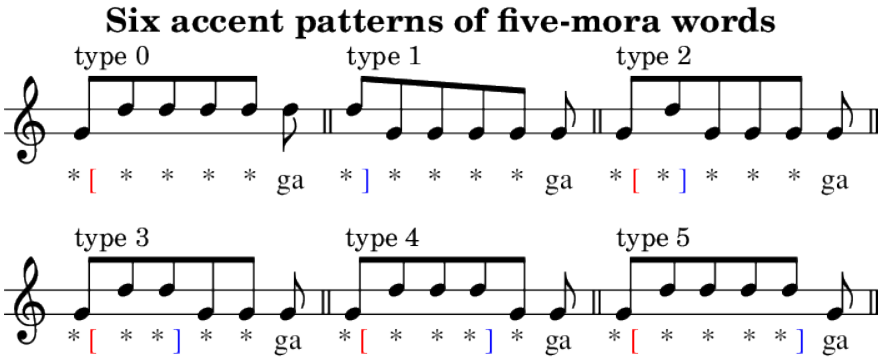
これらのアクセント契機は好き勝手に配置できるわけではなく、一定の制約に従う必要があります。たとえば、原則として下降] は一単語中に多くても一回しか出現しません。また、上昇 [は単語の冒頭にしか現れません。このような制約を満たしうるアクセントパターンの種類は、n 拍の単語の場合、以下のような $n+1$ 種類のパターンに限定されます。

- `*[****`
- `*]****`
- `*[*]***`
- `*[**]**`
- `*[***]*`
- `*[****]`

1.2.2 アクセント核位置の数字表現

アクセント核の位置を数字で表すことでアクセントパターンを表記する方法がよく使われています。新明解国語辞典や UniDic などでは、この数字式アクセント型表記法が使われています。

- 0 型:　途中でピッチが下がらない単語（下降　]　がない単語）
- n 型 ($n > 0$): n 拍目の直後にピッチの下り目（アクセント核）がある単語。



1.2.3 平板型、頭高型、中高型、尾高型

- 0 型は **平板**（へいばん）型とも呼ばれます。
- 1 型は **頭高**（あたまだか）型とも呼ばれます。
- 2 型から $n-1$ 型は **中高**（なかだか）型とも呼ばれます。
- n 型は **尾高**（おだか）型とも呼ばれます。

例えば

- **0 型（平板型）**
  - 野球 `ya[kyuu`, パソコン `pa[sokon`, 鉛筆 `e[npitsu`, 緑茶 `ryo[kucha`, りんご `ri[ngo`, 渋谷 `shi[buya`
- **1 型（頭高型）**
  - サッカー `sa]Qkaa`, ジュース `ju]usu`, 猫 `ne]ko`, メロン `me]ron`, 金魚 `ki]ngyo`, 新橋 `shi]mbashi`
- **2 〜 n-1 型（中高型）**
  - バドミントン `ba[domi]nton`, 折り紙 `o[ri]gami`, カブトムシ `ka[buto]mushi`, 冷蔵庫 `re[ezo]oko`, 池袋 `i[kebu]kuro`
- **n 型（尾高型）**

  - 足 a[shi], 紙 ka[mi], 花 ha[na], 海苔 no[ri], 米 ko[me], 光 hi[kari], 犬 i[nu], 馬 u[ma]

1.2.4 平板型と尾高型は何が違うのか

平板型と尾高型は一見すると同じに見えるかもしれません。しかし、これらは後続する助詞などのピッチに違いが生じます。例えば主格などの格助詞「が」が後続する場合以下のようになります

- はなが
  - 鼻が　は [なが
  - 花が　は [な] が
- ひかりが
  - ひかり（新幹線）が　ひ [かりが
  - 光が　ひ [かり] が
- はしが
  - 橋が　は [し] が
  - 端が　は [しが
  - 箸が　は] しが
- はが
  - 葉が　は [が
  - 歯が　は] が

ただし、後続の助詞が常にこのように振舞うわけではありません。たとえば属格（所有格）などを表す連体格助詞「の」は、直前のアクセント核の影響を受けにくいことが知られています。（ただしこれにも例外はあります。）

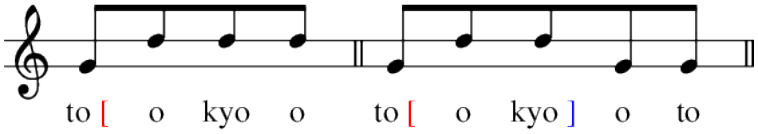
- はなの
  - 鼻の　は [なの
  - 花の　× は [な] の　○は [なの
- ひかりの
  - ひかり（新幹線）の　ひ [かりの
  - 光の　× ひ [かり] の　○ひ [かりの
- はしの
  - 橋の　× は [し] の　○は [しの

  - 端の　は [しの
  - 箸の　は] しの
- はの
  - 葉の　は [の
  - 歯の　〇は] の　× は [の

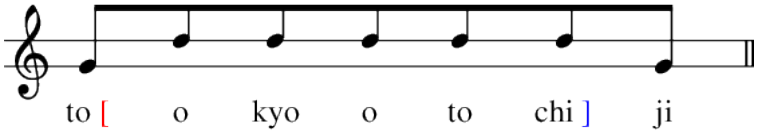
1.2.5 複合語

複合語のアクセント規則は少々複雑です。例えば、

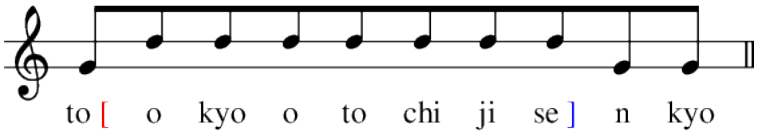
- 東京　と [うきょう
- 東京都 と [うきょ] うと



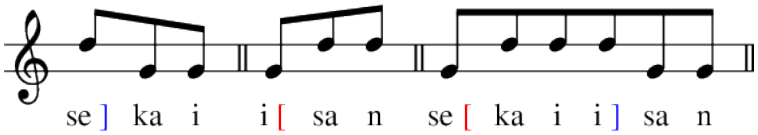
- 東京都知事　と [うきょうとち] じ



- 東京都知事選挙　と [うきょうとちじせ] んきょ

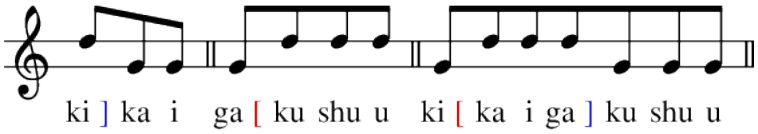


- 世界　せ] かい
- 遺産　い [さん
- 世界遺産　せ [かいい] さん



- 機械　き] かい
- 学習　が [くしゅう

- 機械学習　き [かいが] くしゅう



1.2.6 tdmelodic のロゴについて

日本語には、アクセントがないと区別がつかないような単語のペアが多くあります。例えば、

- ふじ
  - 富士　ふ] じ
  - 藤　ふ [じ
- さけ
  - 鮭　さ] け
  - 酒　さ [け
- はし
  - 端　は [し
  - 橋　は [し]
  - 箸　は] し

これでお分かりのように、tdmelodic のロゴは、アクセントの異なる二つの「ふじ」にちなんだものです。

1.3 文献案内

- ウィキペディア
  - Wikipedia - Japanese pitch accent[1]
- 教科書
  - 松森, 新田, 木部, 中井, 日本語アクセント入門, 三省堂, 2012
- 辞書
  - OJAD (Online Japanese Accent Dictionary)[2]

[1] https://en.wikipedia.org/wiki/Japanese_pitch_accent
[2] http://www.gavo.t.u-tokyo.ac.jp/ojad/

– **NHK** 日本語発音アクセント新辞典, NHK 出版, 2016

– 金田一, 秋永, 新明解日本語アクセント辞典 第 **2** 版, 三省堂, 2014

第2章　docker イメージをビルド

2.1 コードとデータのダウンロード

2.1.1 事前準備

お手元の Unix 系処理系（Ubuntu や MacOS など）に、Git, Docker, MeCab (libmecab-dev など) をセットアップしてください。

2.1.2 コードのダウンロード

作業ディレクトリを作成し、GitHub からコードをダウンロードしてください

```
WORKDIR=/path/to/your/work/dir
cd $WORKDIR
git clone --depth 1 https://github.com/PKSHATechnology-Research/tdmelodic
```

2.1.3 UniDic の辞書ファイルをダウンロード

国立国語研究所のサイト[3]から UniDic の辞書ファイルをダウンロードしてください。複数のバージョンが公開されていますが、このモジュールでは `unidic-mecab_kana-accent-2.1.2_src.zip` を使います。

```
wget https://ccd.ninjal.ac.jp/unidic_archive/cwj/2.1.2/unidic-mecab_kana-accent-2.1.2_src.zip
cp unidic-mecab_kana-accent-2.1.2_src.zip ${WORKDIR}/tdmelodic
```

注意：このファイルは後ほど再利用します。ダウンロード先のサイトに負荷をかけすぎることのないよう、何度もダウンロードするのは避けてください。ダウンロードした zip ファイルはローカルのどこかに保管しておくことを推奨します。

[3] https://ccd.ninjal.ac.jp/unidic/

2.2 Docker ビルド

以下のコマンドで docker のイメージをビルドしてください。これには数分かかります。

```
cd ${WORKDIR}/tdmelodic
docker build -t tdmelodic:latest . # --no-cache
```

2.3 動作確認

もし興味があれば、以下のコマンドを試してみてください。このプロセスは飛ばしても構いません。

```
you@machine:~$ docker run --rm tdmelodic:latest /bin/bash -c "echo 深層学習 | mecab -d \`mecab-
↪config --dicdir\`/unidic"
深層	シンソー	シンソウ	深層	名詞-普通名詞-一般	0
学習	ガクシュー	ガクシュウ	学習	名詞-普通名詞-サ変可能	0
EOS
```

```
you@machine:~$ docker run -it --rm tdmelodic:latest
root@docker:~/workspace$ echo 深層学習 | mecab -d `mecab-config --dicdir`/unidic
深層	シンソー	シンソウ	深層	名詞-普通名詞-一般	0
学習	ガクシュー	ガクシュウ	学習	名詞-普通名詞-サ変可能	0
EOS

root@docker:~/workspace$ python3

>>> from tdmelodic.nn.lang.mecab.unidic import UniDic

>>> u = UniDic()
[ MeCab setting ] unidic='/usr/lib/x86_64-linux-gnu/mecab/dic/unidic'
[ MeCab setting ] mecabrc='/usr/local/lib/python3.8/dist-packages/tdmelodic/nn/lang/mecab/my_
↪mecabrc'

>>> u.get_n_best("深層学習", "しんそうがくしゅう", 3)
 ([[{'surface': '深層', 'pron': 'シンソー', 'kana': 'シンソウ', 'pos': '名詞-普通名詞-一般', 'goshu': '
漢', 'acc': '0', 'concat': 'C2'}, {'surface': '学習', 'pron': 'ガクシュー', 'kana': 'ガクシュウ',
↪'pos': '名詞-普通名詞-サ変可能', 'goshu': '漢', 'acc': '0', 'concat': 'C2'}], [{'surface': '深',
↪'pron': 'シン', 'kana': 'シン', 'pos': '接頭辞', 'goshu': '漢', 'acc': '', 'concat': 'P2'}, {
↪'surface': '層', 'pron': 'ソー', 'kana': 'ソウ', 'pos': '名詞-普通名詞-一般', 'goshu': '漢', 'acc
↪': '1', 'concat': 'C3'}, {'surface': '学習', 'pron': 'ガクシュー', 'kana': 'ガクシュウ', 'pos': '名
詞-普通名詞-サ変可能', 'goshu': '漢', 'acc': '0', 'concat': 'C2'}], [{'surface': '深', 'pron': 'フカ
↪', 'kana': 'フカイ', 'pos': '形容詞-一般', 'goshu': '和', 'acc': '2', 'concat': 'C1'}, {'surface
↪': '層', 'pron': 'ソー', 'kana': 'ソウ', 'pos': '名詞-普通名詞-一般', 'goshu': '漢', 'acc': '1',
↪'concat': 'C3'}, {'surface': '学習', 'pron': 'ガクシュー', 'kana': 'ガクシュウ', 'pos': '名詞-普通名
詞-サ変可能', 'goshu': '漢', 'acc': '0', 'concat': 'C2'}]], [0, 1, 2], 9)

>>> Ctrl-D

root@docker:~/workspace$ exit
you@machine:~$
```

第3章　UniDic ユーザー向け辞書生成

注意：この作業は数時間から数日かかる可能性があります。

3.1 ベースになる NEologd 辞書の準備

3.1.1 NEologd 辞書ファイルをダウンロード

まず、以下のようなコマンドで NEologd の辞書をダウンロードしてください。

```
WORKDIR=/path/to/your/work/dir
cd $WORKDIR # move to the working directory
git clone --depth 1 https://github.com/neologd/mecab-unidic-neologd/
```

3.1.2 NEologd 辞書ファイルの抽出と、パッチによる微修正

次に `unxz` コマンドで、NEologd の単語リスト（CSV ファイル）を抽出します。

```
# if your system has the unxz command
unxz -k `ls mecab-unidic-neologd/seed/*.xz | tail -n 1`
# otherwise
docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    unxz -k `ls mecab-unidic-neologd/seed/*.xz | tail -n 1`
```

これにより、`mecab-unidic-user-dict-seed.yyyymmdd.csv` といったファイル名の CSV ファイルが生成されます。次に、ここで抽出した CSV ファイルにパッチを当てます。これにより作業ディレクトリ配下に `neologd_modified.csv` というファイルが作られます。

```
docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    tdmelodic-neologd-preprocess \
    --input `ls mecab-unidic-neologd/seed/mecab-unidic-user-dict-seed*.csv | tail -n 1` \
    --output neologd_modified.csv \
    --no-rmdups --no-rm_wrong_yomi
```

`--no-rmdups`, `--no-rm_wrong_yomi` などのオプションは、あるカテゴリーに属する単語を辞書から除去するかどうかを指定するためのものです。これらのオプションは以下のコマンドにより確認できます。

```
docker run --rm tdmelodic:latest tdmelodic-neologd-preprocess -h
```

3.2 推論

注意：**この処理はかなりの時間がかかります！**（参考までに、筆者の MacBookPro では 2 時間半、Linux サーバーでは 5 時間かかりました。）

では、アクセント辞書を生成しましょう。ここでは、NEologd の辞書ファイルに掲載されている全ての単語について、機械学習ベースの手法により、アクセント情報を推定します。以下のコマンドにより、各単語にアクセント情報を付与した新しい辞書が生成されます。

```
docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    tdmelodic-convert \
    -m unidic \
    --input neologd_modified.csv \
    --output ${WORKDIR}/tdmelodic_original.csv
cp ${WORKDIR}/tdmelodic_original.csv ${WORKDIR}/tdmelodic.csv # backup
```

3.3 後処理

以下のスクリプトにより、単語生起確率（条件付確率場におけるユニグラム・コスト）の微修正などの後処理を行います。

```
cp ${WORKDIR}/tdmelodic.csv ${WORKDIR}/tdmelodic.csv.bak
docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    tdmelodic-modify-unigram-cost \
    -i tdmelodic.csv.bak \
    -o tdmelodic.csv
```

第4章　IPADIC ユーザー向け辞書生成

注意：この作業は数時間から数日かかる可能性があります。

4.1 ベースになる辞書の準備

4.1.1 IPADIC のダウンロード

はじめに、https://taku910.github.io/mecab から IPADIC をダウンロードしてください。

```
WORKDIR=/path/to/your/work/dir
cd $WORKDIR # move to the working directory
cp /path/to/your/download/dir/mecab-ipadic-2.7.0-XXXX.tar.gz $WORKDIR
tar zxfv mecab-ipadic-2.7.0-XXXX.tar.gz
```

ダウンロードした圧縮ファイルを展開して得られたディレクトリの中を `ls` などのコマンドで見てみると、多数の CSV ファイルと設定ファイルがあるのが確認できます。これらの辞書ファイルの文字コードは EUC-JP になっていますが、これを以下のコマンドにより UTF-8 に変換します。もし `nkf` コマンドがある場合は以下のコマンドを実行してください。

```
find ./mecab-ipadic-2.7.0-* -type f -name "*.csv" | xargs -I{} nkf -w --overwrite {}
```

もし `nkf` がない場合は Docker を使って以下のようなコマンドで変換できます。

```
docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    find ./mecab-ipadic-2.7.0-* -type f -name "*.csv" | xargs -I{} nkf -w --overwrite {}
```

4.1.2 NEologd のダウンロード

同様に、NEologd の辞書ファイルをダウンロードしてください。

```
cd $WORKDIR # move to the working directory
git clone --depth 1 https://github.com/neologd/mecab-ipadic-neologd/
```

次に `unxz` コマンドで、NEologd の単語リスト（CSV ファイル）を抽出します。もし `unxz` コマンドがインストールされている場合は以下のようなコマンドを実行してください。

```
find ./mecab-ipadic-neologd/seed/ -type f -name "*.xz" | xargs -I{} unxz -k {}
```

unxz がない場合は、docker を使用して以下のコマンドで同様のことができます。

```
find ./mecab-ipadic-neologd/seed/ -type f -name "*.xz" | xargs -I{} \
   docker run --rm -v $(pwd):/root/workspace tdmelodic:latest unxz -k {}
```

この処理により、多数の CSV ファイルが ./mecab-ipadic-neologd/seed/ 配下に生成されます。

4.2 推論

注意：この処理はかなりの時間がかかります！

では、アクセント辞書を生成しましょう。ここでは、NEologd の辞書ファイルに掲載されている全ての単語について、機械学習ベースの手法により、アクセント情報を推定します。以下のコマンドにより、各単語にアクセント情報を付与した新しい辞書が生成されます。

4.2.1 IPADIC

```
find ./mecab-ipadic-2.7.0-*/ -type f -name "*.csv" | xargs -I{} \
    docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
        tdmelodic-convert -m ipadic --input {} --output {}.accent
```

もしくは、以下のコマンドでも同様に辞書を生成できます。

```
cat ./mecab-ipadic-2.7.0-*/*.csv > ipadic_all.csv
docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    tdmelodic-convert -m ipadic \
        --input ipadic_all.csv \
        --output ipadic_all.csv.accent
```

4.2.2 NEologd

まず、もし必要であれば以下のコマンドで前処理を行ってください。(-h により前処理のオプションを表示できます。)

```
find ./mecab-ipadic-neologd/seed/ -type f -name "*.csv" | xargs -I{} \
    docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
        tdmelodic-neologd-preprocess -m ipadic --input {} --output {}.preprocessed
```

次に、以下のコマンドによりアクセントを推定します。

```
find ./mecab-ipadic-neologd/seed/ -type f -name "*.csv" | xargs -I{} \
    docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
        tdmelodic-convert -m ipadic --input {}.preprocessed --output {}.accent
```

以上の作業により、アクセント情報が付与された辞書ファイル `*.csv.accent` が得られます。

もしくは、以下のようなコマンドでも辞書ファイルが得られます。

```
cat ./mecab-ipadic-neologd/seed/*.csv > neologd_all.csv

docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    tdmelodic-neologd-preprocess -m ipadic \
        --input neologd_all.csv \
        --output neologd_all.csv.preprocessed

docker run --rm -v $(pwd):/root/workspace tdmelodic:latest \
    tdmelodic-convert -m ipadic \
        --input neologd_all.csv.preprocessed \
        --output neologd_all.csv.accent
```

第5章　UniDic-tdmelodic を MeCab 辞書として活用

5.1 UniDic-tdmelodic のインストール

前の章で作成した tdmelodic.csv ファイルの中身を UniDic のデフォルトの辞書 (lex.csv) にコピーし、適切なコマンドラインオプションを与えてインストール用スクリプトを実行することで、tdmelodic (tdmelodic-upadic) をインストールできます。

まず、ファイルのパスを指定します。

```
WORKDIR=/path/to/your/work/dir
UNIDIC_ZIP_PATH=/path/to/your/unidic/file/unidic-mecab_kana-accent-2.1.2_src.zip
TDMELODIC_CSV=${WORKDIR}/tdmelodic.csv
```

次に、UniDic の zip ファイルを展開します。

```
cd ${WORKDIR}
cp ${UNIDIC_ZIP_PATH} .
unzip unidic-mecab_kana-accent-2.1.2_src.zip
```

次に、辞書ファイルを結合します。

```
cd ${WORKDIR}/unidic-mecab_kana-accent-2.1.2_src
cp lex.csv lex_bak.csv # backup
cat ${TDMELODIC_CSV} >> lex.csv
```

最後に、tdmelodic をインストールします。

```
./configure --with-dicdir=`mecab-config --dicdir`/tdmelodic
make
make install
```

5.2 UniDic-tdmelodic の使用例

いくつかの例をご覧ください。

5.2.1 例 1

```
echo 一昔前は人工知能のプログラミング言語といえば Common Lisp や Prolog だった。 | \
    mecab -d `mecab-config --dicdir`/tdmelodic/
```

```
一昔　ヒトムカシ　ヒトムカシ　一昔　名詞-普通名詞-一般　　2,3
前　マエ　マエ　前　名詞-普通名詞-副詞可能　　1
は　ワ　ハ　は　助詞-係助詞
人工知能　ジ [ンコーチ] ノー　ジンコウチノウ　人工知能　名詞-固有名詞-一般　　@
の　ノ　ノ　の　助詞-格助詞
プログラミング言語　プ [ログラミングゲ] ンゴ　プログラミングゲンゴ　プログラミング言語　名詞-固有名詞-一般　　@
と　ト　ト　と　助詞-格助詞
いえ　イエ　イウ　言う　動詞-一般　五段-ワア行　仮定形-一般　0
ば　バ　バ　ば　助詞-接続助詞
Common Lisp　コ [モンリ] スプ　コモンリスプ　Common Lisp　名詞-固有名詞-一般　　@
や　ヤ　ヤ　や　助詞-副助詞
Prolog　プ [ロログ　プロログ　Prolog　名詞-固有名詞-一般　　@
だっ　ダッ　ダ　だ　助動詞　助動詞-ダ　連用形-促音便
た　タ　タ　た　助動詞　助動詞-タ　終止形-一般
。　　。　補助記号-句点
EOS
```

Cf.

```
echo 一昔前は人工知能のプログラミング言語といえば Common Lisp や Prolog だった。 | \
    mecab -d `mecab-config --dicdir`/unidic/
```

```
一昔　ヒトムカシ　ヒトムカシ　一昔　名詞-普通名詞-一般　　2,3
前　マエ　マエ　前　名詞-普通名詞-副詞可能　　1
は　ワ　ハ　は　助詞-係助詞
人工　ジンコー　ジンコウ　人工　名詞-普通名詞-一般　　0
知能　チノー　チノウ　知能　名詞-普通名詞-一般　　1
の　ノ　ノ　の　助詞-格助詞
プログラミング　プログラミング　プログラミング　プログラミング-programming　名詞-普通名詞-サ変可能　　4
言語　ゲンゴ　ゲンゴ　言語　名詞-普通名詞-一般　　1
と　ト　ト　と　助詞-格助詞
いえ　イエ　イウ　言う　動詞-一般　五段-ワア行　仮定形-一般　0
ば　バ　バ　ば　助詞-接続助詞
Common　Common　Common　Common　名詞-普通名詞-一般　　0
Lisp　Lisp　Lisp　Lisp　名詞-普通名詞-一般　　0
や　ヤ　ヤ　や　助詞-副助詞
Prolog　Prolog　Prolog　Prolog　名詞-普通名詞-一般　　0
だっ　ダッ　ダ　だ　助動詞　助動詞-ダ　連用形-促音便
た　タ　タ　た　助動詞　助動詞-タ　終止形-一般
。　　。　補助記号-句点
EOS
```

5.2.2 例 2

```
echo 横浜市中区日本大通 | mecab -d `mecab-config --dicdir`/tdmelodic
```

```
横浜市中区日本大通　ヨ [コハマ] シナ [カ] クニ [ホンオオド] オリ　ヨコハマシナカクニホンオオドオリ　横浜市中区日本大通　名詞-固有名詞-地名-一般　　@
EOS
```

```
echo 横浜市中区日本大通 | mecab -d `mecab-config --dicdir`/unidic
```

```
横浜	ヨコハマ	ヨコハマ	ヨコハマ	名詞-固有名詞-地名-一般	0
市中	シチュー	シチュウ	市中	名詞-普通名詞-一般	0,2
区	ク	ク	区	名詞-普通名詞-一般	1
日本	ニッポン	ニッポン	日本	名詞-固有名詞-地名-国	3
大通	ダイツー	ダイツウ	大通	名詞-普通名詞-一般	3,0
EOS
```

5.2.3 例 3

```
echo 980hPa | mecab -d `mecab-config --dicdir`/tdmelodic/
echo 15mm | mecab -d `mecab-config --dicdir`/tdmelodic/
echo 4月10日 | mecab -d `mecab-config --dicdir`/tdmelodic/
```

```
980hPa	キュ]ーヒャクハ[チジュウヘクトパ]スカル	キュウヒャクハチジュウヘクトパスカル	980hPa	名詞-固有名詞-一般
	@
EOS
15mm	ジュ[ウゴミリメ]ートル	ジュウゴミリメートル	15mm	名詞-固有名詞-一般	@
EOS
4月10日	シ[ガツトオカ	シガツトオカ	4月10日	名詞-固有名詞-一般	@
EOS
```

第6章　IPADIC-tdmelodic を MeCab 辞書として活用

6.1 IPADIC-tdmelodic のインストール

前の章で作成した `*.csv.accent` ファイルを IPADIC のディレクトリ配下に配置し、適切なコマンドラインオプションを与えてインストール用スクリプトを実行することで、`tdmelodic-ipadic` をインストールできます。

```
# paths
WORKDIR=/path/to/your/work/dir
NEOLOGD_DIC_DIR=${WORKDIR}/mecab-ipadic-neologd/seed
IPADIC_DIR=${WORKDIR}/mecab-ipadic-2.7.0-XXXX
```

```
# copy
for f in `ls ${NEOLOGD_DIC_DIR}/*.csv.accent`
do
    target=`basename $f`
    target=${target%.accent}
    cp $f $IPADIC_DIR/$target
done

for f in `ls ${IPADIC_DIR}/*.csv.accent`
do
    target=`basename $f`
    target=${target%.accent}
    cp $f $IPADIC_DIR/$target
done
```

```
# install
cd ${IPADIC_DIR}
./configure --with-dicdir=`mecab-config --dicdir`/tdmelodic-ipadic
make
make install
```

6.2 UniDic-tdmelodic の使用例

いくつかの例をご覧ください。

6.2.1 例 1

```
echo 一昔前は人工知能のプログラミング言語といえば Common Lisp や Prolog だった。 | \
    mecab -d `mecab-config --dicdir`/tdmelodic-ipadic
```

```
一昔      名詞, 一般,*,*,*,*, 一昔, ヒトムカシ, ヒ [ト] ムカシ
前        名詞, 副詞可能,*,*,*,*, 前, マエ, マ] エ
は        助詞, 係助詞,*,*,*,*, は, ハ, ワ
人工知能          名詞, 固有名詞, 一般,*,*,*, 人工知能, ジンコウチノウ, ジ [ンコーチ] ノー
の        助詞, 連体化,*,*,*,*, の, ノ, ノ
プログラミング言語        名詞, 固有名詞, 一般,*,*,*, プログラミング言語, プログラミングゲンゴ, プ [ログラミングゲ] ンゴ
と        助詞, 格助詞, 引用,*,*,*, と, ト, ト]
いえ      動詞, 自立,*,*, 五段・ワ行促音便, 仮定形, いう, イエ, イ [エ]
ば        助詞, 接続助詞,*,*,*,*, ば, バ, バ
Common Lisp       名詞, 固有名詞, 一般,*,*,*,Common Lisp, コモンリスプ, コ [モンリ] スプ
や        助詞, 並立助詞,*,*,*,*, や, ヤ, ヤ
Prolog   名詞, 固有名詞, 一般,*,*,*,Prolog, プロログ, プ [ロログ
だっ      助動詞,*,*,*, 特殊・ダ, 連用タ接続, だ, ダッ, ダ] ッ
た        助動詞,*,*,*, 特殊・タ, 基本形, た, タ, タ
。        記号, 句点,*,*,*,*,。,。,。
EOS
```

6.2.2 例 2

```
echo 横浜市中区日本大通 | mecab -d `mecab-config --dicdir`/tdmelodic-ipadic
```

```
横浜市中区日本大通        名詞, 固有名詞, 地域, 一般,*,*, 横浜市中区日本大通, ヨコハマシナカクニホンオオドオリ, ヨ [コハマ]
シナ [カ] クニ [ホンオード] ーリ
EOS
```

6.2.3 例 3

```
echo 980hPa | mecab -d `mecab-config --dicdir`/tdmelodic-ipadic
echo 15mm | mecab -d `mecab-config --dicdir`/tdmelodic-ipadic
echo 4 月 10 日 | mecab -d `mecab-config --dicdir`/tdmelodic-ipadic
```

```
980hPa   名詞, 固有名詞, 一般,*,*,*,980hPa, キュウヒャクハチジュウヘクトパスカル, キュ] ウヒャクハ [チジュウヘクトパ]
スカル
EOS
15mm     名詞, 固有名詞, 一般,*,*,*,15mm, ジュウゴミリメートル, ジュ [ウゴミリメ] ートル
EOS
4 月 10 日 名詞, 固有名詞, 一般,*,*,*,4 月 10 日, シガツトオカ, シ [ガツトオカ
EOS
```

第7章　一件ずつ推論するモード

辞書全体を一度に推論するのではなく、個々の単語について一つずつアクセント推定したいケースもあるでしょう。このページではそのためのツールを紹介します。

7.1 s2ya: 表層形 → 読み＆アクセント

s2ya は、単語の標準的な表記法（表層形）から、その読みとアクセントを推定します。読みの推定には、MeCab と UniDic による最適解を利用します。

- 入力：漢字などによる、標準的な表記法（表層形）
- 出力：読みとアクセント

```
$ echo 機械学習 | docker run tdmelodic:latest tdmelodic-s2ya
```

すると以下のような結果が得られます。

```
キ［カイガ］クシュー
```

以下のようなエイリアスを利用すると便利です。

```
$ alias tdmelodic-s2ya="docker run tdmelodic:latest tdmelodic-s2ya"
```

このコマンドを使って、他の単語も試してみてください。

```
$ echo 深層学習 | tdmelodic-s2ya
シ［ンソーガ］クシュー

$ echo 確率微分方程式 | tdmelodic-s2ya
カ［クリツビブンホーテ］ーシキ

$ echo 電験一種 | tdmelodic-s2ya
デ［ンケンイ］ッシュ

$ echo マルクス・アウレリウス・アントニヌス | tdmelodic-s2ya
マ［ルクスアウレリウスアントニ］ヌス

$ echo IoT | tdmelodic-s2ya
ア［イオーティ］ー
```

同様にして、文のアクセントも推定できます。

```
$ echo 今日の東京の天気は晴れ | tdmelodic-s2ya
キョ] ーノト [ーキョーノテ] ンキワハレ

$ echo 漢字の音読みには主に呉音と漢音があり、漢音の方が新しい。 | tdmelodic-s2ya
カ [ンジノオンヨミニ] ワオ] モニゴ [オントカ] ンオンガアリ [カ] ンオンノホ] ーガアタラシ] ー

$ echo 現在、西新宿ジャンクションから談合坂サービスエリアまで、およそ四十五分 | tdmelodic-s2ya
ゲ] ンザイニ [シシンジュクジャ] ンクションカラダ [ンゴーサカサ [ービスエ] リアマ] デ] オ [ヨソヨ] ンジュー [ゴ] フン

$ echo 完備なノルム空間をバナッハ空間といい、完備な内積空間をヒルベルト空間という。 | tdmelodic-s2ya
カ] ンビナノ [ルムク] ーカンオバ [ナッハク] ーカントイーカ] ンビナナ [イセキク] ーカンオヒ [ルベルトク] ーカントイウ

$ echo 権利の行使及び義務の履行は、信義に従い誠実に行わなければならない。 | tdmelodic-s2ya
ケ] ンリノコ] ーシオヨビギ] ムノリコーワ [シ] ンギニシ [タガイセージツニオコナワナ] ケレ] バナラナイ
```

警告: 上で見たように、tdmelodic は形式的には文のアクセント予測が可能であり、また学習データにも少量の文章データを使用しています。しかし、tdmelodic はもともと文のアクセント推定を主眼として設計・学習したものではありません。このため、この方法によって推定された文のアクセントはあくまで参考程度のものです。

注釈: `s2ya` の読み予測には UniDic 辞書を使用します。これは、docker イメージの中に入っている辞書が UniDic のみであるためです。もし Neologd など他の辞書を使用して読み推定をしたい場合は、次の `sy2a` をご利用ください。

7.2 sy2a: 表層形＆読み → アクセント

`sy2a` は、単語の表層形と読みから、アクセントを推定するツールです。

- 入力：表層形（漢字など）と読み
- 出力：アクセント

例えば以下のように使用します。

```
$ alias tdmelodic-sy2a="docker run -v tdmelodic:latest tdmelodic-sy2a"
$ echo 機械学習, きかいがくしゅー | tdmelodic-sy2a
キ [カイガ] クシュー
```

他の例

```
$ echo 日本語アクセント, にほんごあくせんと | tdmelodic-sy2a
ニ [ホンゴア] クセント

$ echo 御御御付け, おみおつけ | tdmelodic-sy2a
オ [ミオ] ツケ

$ echo 談合坂 SA, だんごーざかさーびすえりあ | tdmelodic-sy2a
ダ [ンゴーザカサービスエ] リア
```

また、文のアクセントも同様に推定できます。

```
$ echo Wifi に接続できません，わいふぁいにせつぞくできません | tdmelodic-sy2a
ワ [イファイニセ [ツゾクデキマセ] ン

$ echo 国立市の国立大学，くにたちしのこくりつだいがく | tdmelodic-sy2a
ク [ニタチ] シノコ [クリツダ] イガク

$ echo 漢音は、当時の唐の都、長安の音を持ち帰ったものである。，かんおんわとーじのとーのみやこちょーあんのおとおもちかえったものである | tdmelodic-sy2a
カ] ンオンワ [ト] ージノト] ーノミ [ヤコ [チョ] ーアンノオ [ト] オモ [チカエッタモノ] デア] ル
```

注釈: Neologd などの先進的な辞書を使用して読み推定を行いたい場合は、一例として以下のようなコマンドで読みが推定できます。

```
$ TEXT=ラグビー日本代表の試合を見に飛田給に

$ YOMI=`echo $TEXT \
$       | mecab -d \`mecab-config --dicdir\`/mecab-unidic-neologd/ \
$       | sed -e "/^EOS/d" | cut -f 2 | perl -pe 's/\s+//g'`

$ # An alternative approach:
$ YOMI=`echo $TEXT | mecab -Oyomi -d \`mecab-config --dicdir\`/mecab-ipadic-neologd/`

$ # check the result.
$ echo $YOMI
ラグビーニホンダイヒョーノシアイオミニトビタキューニ

$ # accent prediction.
$ echo $TEXT,$YOMI | tdmelodic-sy2a
ラ [グビーニホンダ] イヒョーノシ [アイオミ] ニトビタキュ] ーニ
```

第8章　文献情報

論文等で引用いただく場合は以下の bibtex をご利用ください。

```bibtex
@inproceedings{tachibana2020icassp,
   author    = "H. Tachibana and Y. Katayama",
   title     = "Accent Estimation of {Japanese} Words from
               Their Surfaces and Romanizations
               for Building Large Vocabulary Accent Dictionaries",
   booktitle = {2020 IEEE International Conference on Acoustics,
               Speech and Signal Processing (ICASSP)},
   pages     = "8059--8063",
   year      = "2020",
   doi       = "10.1109/ICASSP40776.2020.9054081"
}
```

論文リンク: [IEEE Xplore][4], [arXiv プレプリント][5]

[4] https://ieeexplore.ieee.org/document/9054081
[5] https://arxiv.org/abs/2009.09679