航空宇宙実習＋人工知能実習２０２５

テーマ：工場環境における製品番号等読み上げ音声認識

目的：
・工場環境下で製品番号などを読み上げて音声認識したい.
・音声認識ソフトの限界があるので解決したい.

条件：
・製品番号はアルファベットや数値，記号でできている
・文脈はないので１文字ずつ識別するしかない
・ノイズは結構ある
・PCと安価なマイクでできないか？

学習面の方針：
・人工音声と人工的なノイズの合成でダミーデータを作成する
・文字起こしのモデルを利用できるようになる
・ノイズと発話のバランスを変えた条件で性能の違いを把握する
・ノイズ除去の仕組みを把握する
・ヘッドセットマイクを利用して音声を収集できるようになる
・評価方法（定性評価，定量評価）を理解する
・オフライン（音声収集，認識，評価が別々）での実験を理解する

目的に対する方針：
・実環境で音声を収集して，人工データでの実験を置き換える
・特定の文字や品番でのエラーを把握する
・１００種類以上の製品番号の読み上げの評価を行う
・複数話者でデータを収集する
・ノイズなし（会議室環境）とノイズあり工場環境の２条件で比較する
・オフラインからリアルタイムで動作するために必要な条件を探索する

１．工場のノイズを作る

何を目指しているか？
人工的なノイズを作って，音声ファイルと組み合わせた音源を作る.
その音源から，ノイズを分離して，音声のみ取り出せるか，実験したい.
なので，まずはノイズだけのデータを作る.

プログラム
01_create_factory_noise.ipynb

実際に生成：４種類のノイズを生成
noise = generate_factory_noise(dur_s=20.0, mains=60.0, levels=(0.6, 0.7, 0.3, 0.4), seed=1234)
sf.write("factory_noise_01.wav", noise, SR)

noise = generate_factory_noise(dur_s=20.0, mains=60.0, levels=(0.6, 0.6, 0.3, 0.4), seed=12354)
sf.write("factory_noise_02.wav", noise, SR)

noise = generate_factory_noise(dur_s=20.0, mains=60.0, levels=(0.6, 0.4, 0.4, 0.2), seed=1234)
sf.write("factory_noise_03.wav", noise, SR)

noise = generate_factory_noise(dur_s=20.0, mains=60.0, levels=(0.6, 0.7, 0.4, 0.5), seed=12354)
sf.write("factory_noise_04.wav", noise, SR)

作成したノイズを聞いてみよう.

パラメータを理解しよう.
・乱数，周期，作成時間.

２．人工音声で製品番号などの読み上げ音声を作る

何を目指しているか？
人工的な音声で製品の読み上げを行う.
これで正解がわかった正しい音声ファイルができる.
これと１で生成したノイズと合わせてたい.

プログラム
02_create_AIvoice_productsname.ipynb

実際に生成：３０種類以上の製品番号（架空）を読み上げ．code_???.wavで保存

```
# 4) 製品コード例
codes = [
    "AB-1234",
    "XZ-98765",
    "P000123",
    "K-4501",
  （略）
   "LX-76543",
    "JH-21098"
]
for code in codes:
    text = speak_code_japanese(code)  # 「エー・ビー…」形式に整形
    out = os.path.join(OUTPUT_DIR, f"code_{code.replace('-', '_')}.wav")
    tts_to_wav(text, out)
print("製品コードの生成完了．")
```

作成した音声を聞いてみよう.

パラメータを理解しよう.
・日本語風に読み上げている．速度は？製品コードは？

３．人工音声とノイズを組み合わせた音源を作ってみよう

何を目指しているか？
１と２で作った音声を，いろいろなバランスで合成したい.
そうすると，実際に工場でマイクに向かって話しているような音声が作れる.

プログラム
03_create_VoiceAndNoise.ipynb

実際に生成：
```
# 試しにファイルを明示的に指定して実行
speech_file = "./tts_samples/code_AB_1234.wav"  # 生成済みTTS音声
noise_file  = "./noise/factory_noise_01.wav"            # 生成済み工場ノイズ
output_file = "mixed_AB_1234_0dB.wav"

mix_with_snr_and_padding(speech_file, noise_file, output_file, snr_db=0,
pad_sec=1.0)
```

その後，総当たりで合成.

作成した音声を聞いてみよう.
./mixedフォルダーに作成されている．１２００種類できたはずです.

パラメータを理解しよう.
・前後のパディング，合成のバランスは？

４．音声ファイルの文字起こしをやってみよう

何を目指しているか？
文字起こしとは？
作成した音声ファイルの文字起こしで比較してみよう.
ノイズの影響はあるか確認してみたい.

プログラム
04_voice_recognition.ipynb

実際に生成：
segments, info = model.transcribe("./mixed/
code_AB_1234__factory_noise_01__snr0p0dB_pad1.0s.wav")
print("".join(seg.text for seg in segments))
mix_with_snr_and_padding(speech_file, noise_file, output_file, snr_db=0,
pad_sec=1.0)
出力：
A B' 1 2 3 4

いろいろなファイルで試してみよう
どれもうまくいくでしょうか？

パラメータを理解しよう
・モデルが　small, medium, large　とあります.

出てきた文字の比較
・関数：evaluate_product_codeで評価
#　判定する関数.　　正解と文字起こしした文字列を渡す
evaluate_product_code("AB-1234", tmp)

（評価結果）
{'正解コード': 'AB-1234',
 '予測コード': 'AB-1234',
 '全体一致': True,
 '文字正解数': 7,
 '文字総数': 7,
 '文字正解率': 1.0}

５．複数の音声ファイルの文字起こしをやってみよう

何を目指しているか？
フォルダーに保存された音声ファイルを一括で処理したい.
それをエクセルファイルに保存して，あとで評価したい.
５はご，ごー，号，なとど認識される．Aは，A，えー，えい　などと認識される.
それらを辞書で修正しながら文字にしたい.
文章とちがって文脈がないので（前後の出現文字に非依存）辞書が重要.

プログラム
05_recognition_wavfolder.ipynb

実際に生成：

```
rows = []
for fname in tqdm(audio_files, desc=f"Transcribing
({os.path.basename(AUDIO_DIR)})", unit="file"):
    fpath = os.path.join(AUDIO_DIR, fname)
    raw = transcribe_one(fpath)
    fixed = normalize_code_like(raw)   # ← 半角英数＋ - / に正規化
    rows.append({
        "file": os.path.basename(fname),  # ← basename
        "raw_text": raw,                  # 文字起こし直後の文字列
        "fixed_text": fixed,              # 辞書（正規化）後の文字列
    })
```

```
Transcribing (): 100%|██████████████████████| 240/240 [04:34<00:00,  1.14s/file]
```

raw_textはモデルが出力する文字列．fixed_textは辞書で修正後の文字列.

	file	raw_text	fixed_text
0	code_AB_1234__factory_noise_01__snr0p0dB_pad1....	A B DASH 1 2 3 4	AB-1234
1	code_AB_1234__factory_noise_02__snr0p0dB_pad1....	A B DASH 1 2 3 4	AB-1234

結果は音声ファイルのフォルダーにエクセル形式で保存：
ファイル名：transcripts_norm.xlsx

パラメータを理解しよう

```
# ===== 設定 =====
AUDIO_DIR = "./mixed_small/"        # 読み込むフォルダに変更
ASR_MODEL = "small"          # tiny/base/small/medium などモデルの大きさ
```

６．正解と評価してみよう

何を目指しているか？
ファイル名に保存された製品番号から正解を作成.
先につくった文字起こしの結果（辞書で修正後）と正解を比較.
- E列: 全体一致（True/False）
- F列: 文字正解数
- G列: 文字総数
- H列: 文字正解率（正解数 ÷ 総数）

を計算してエクセルで列ごとに保存．他の条件との比較に備える.

プログラム
06_evaluation_results.ipynb

評価結果：
raw_textはモデルが出力する文字列．fixed_textは辞書で修正後の文字列.

file	raw_text	fixed_text	正解コード	全体一致	文字正解数	文字総数	文字正解率
code_AB_1234__factory_noise_01__snr0p0dB_pad1.0s.wav	A B DASH 1 2 3 4	AB-1234	AB-1234	TRUE	7	7	1
code_AB_1234__factory_noise_02__snr0p0dB_pad1.0s.wav	A B DASH 1 2 3 4	AB-1234	AB-1234	TRUE	7	7	1
code_AE_77701__factory_noise_01__snr0p0dB_pad1.0s.wav	AE-777701	AE-777701	AE-77701	FALSE	7	8	0.875
code_AE_77701__factory_noise_02__snr0p0dB_pad1.0s.wav	AE-DASH 777701	AE-DASH777701	AE-77701	FALSE	3	8	0.375
code_AE_77701__factory_noise_03__snr0p0dB_pad1.0s.wav	AE-DASH 77701	AE-DASH77701	AE-77701	FALSE	4	8	0.5
code_AE_77701__factory_noise_04__snr0p0dB_pad1.0s.wav	AE-DASH 777701	AE-DASH777701	AE-77701	FALSE	3	8	0.375

音声ファイルのフォルダーにtranscripts_eval.xlsx　として保存されるよ.

次のステップへ：
・平均の正答率を出してみよう．他にも評価方法を考えて実装してみよう.
・このフォルダーに保存された音声の評価結果の代表値を定めよう.
・そして，他の条件との比較に備えよう.
・失敗している音声の原因を推定しよう：
　　外的要因：ノイズの混入？　そもそも発話が悪い？
　　内的要因：認識文字の誤り？　辞書の間違い？　評価法の誤り？
・その対策を考えてみよう.

７．ノイズ除去してみよう：音声ファイルの前後でノイズ抽出

何を目指しているか？
やはりノイズはよくないのでは？と思いノイズ除去を試してみる.
ノイズ除去後の音声ファイルを作成して，それを５と６と同じ方法で評価できるはず.
そうすれば，ノイズ除去の効果が計測できる.
まずは簡単なスペクトル減算法でノイズ除去した音声ファイルを作ってみる.

プログラム
07_noise_reduction.ipynb

設定（ディレクトリの指定）：

```
IN_DIR  = "./mixed"      # ノイズ入りの合成WAV
OUT_DIR = "./denoised"   # ノイズ除去後の保存先
SR      = 16000
PAD_SEC = 1.0            # 先頭/末尾の無音（ノイズ見本として使う）
```

実際に生成：

```
def denoise_nr(in_wav, out_wav, sr=SR, pad_sec=PAD_SEC, prop_decrease=1.0):
    """
    noisereduce（スペクトルゲート）でノイズ除去。
    先頭・末尾の pad_sec をノイズプロファイルとして使用。
    """
    y, _sr = librosa.load(in_wav, sr=sr, mono=True)
    n_pad = int(sr * pad_sec)
    # ノイズ見本（前後の無音パートを結合）
```

でやってるよ．１２００ファイルの処理でもあっという間！

ノイズ除去の結果を聞いてみよう
./denoised　フォルダーに作成されている.

ノイズ除去の効果を評価しよう：
・先の５，６の手順で評価して，ノイズを含んだ場合との結果を比較する.

８．ノイズ除去してみよう：深層学習で音声分離（demucs）

何を目指しているか？
やはりノイズはよくないのでは？と思いノイズ除去を試してみる.
ノイズ除去後の音声ファイルを作成して，それを４と同じ方法で評価できるはず.
そうすれば，ノイズ除去の効果が計測できる.
まずは簡単なスペクトル減算法でノイズ除去した音声ファイルを作ってみる.

プログラム
08_noise_reduction_demucs.ipynb

実際に生成：
```
def denoise_nr(in_wav, out_wav, sr=SR, pad_sec=PAD_SEC, prop_decrease=1.0):
    """
    noisereduce（スペクトルゲート）でノイズ除去。
    先頭・末尾の pad_sec をノイズプロファイルとして使用。
    """
    y, _sr = librosa.load(in_wav, sr=sr, mono=True)
    n_pad = int(sr * pad_sec)
    # ノイズ見本（前後の無音パートを結合）
```

で
demucs_denoise_folder(MIXED_DIR, DENOISED_DIR)
やってるよ．１２００ファイルの処理でもあっという間！

認識結果を見てみよう
denoised_demucs　フォルダーに作成されている.

ノイズ除去の効果を評価しよう：
・先の５，６の手順で評価して，ノイズを含んだ場合との結果を比較する.

９．ノイズ除去なし／除去あり（単純）／除去あり（深層学習）比較

何を目指しているか？
３つの音声データができたので，それを比較.
定量比較を目指す.

プログラム
09_recognition_wavfolder.ipynb

設定（ディレクトリの指定）：
読み込むフォルダーを設定して実行
AUDIO_DIR = "./mixed_small/"

# ===== ディレクトリ一括 → DF作成 =====
audio_files = sorted([f for f in os.listdir(AUDIO_DIR) if f.lower().endswith(".wav")])
assert audio_files, f"{AUDIO_
でそのフォルダーの中に評価用のエクセルファイルができる.
mixed, denoised, denoised_demucs の３条件がある.

また，以下の設定で読み込むエクセルファイルと結果保存用のファイル名を指定します.
条件ごとに繰り返します.
IN_XLSX = "./mixed_small/transcripts_norm.xlsx" # 読み込み元（必要に応じて変更）
OUT_XLSX = "./mixed_small/transcripts_eval.xlsx" # 出力先（上書きしたければ IN_XLSX と同じにしてOK）

評価しよう：
それぞれの結果を把握して比較してみよう.
特定の文字列が失敗しているのか？
ノイズ除去の効果はあるのか？
定量的に調べよう：問題を仮定してその結果を調べる　を繰り返す.
ー＞仮説をつくりそれを検証する方法で効果や原因を明らかにする．具体例も出す.

１０．リアルデータでの実験と評価

何を目指しているか？
人工データでやってみて，おおよその手順と問題を把握できた.
つぎはリアルなデータでやってみたい.
ノイズの種類や音声とのバランスは実環境で計測したい.
マイクの性能にも依存であるが，ヘッドセットはまぁ一般的.
実際に読み上げてみて，音声ファイルを作成．その認識を行ってみる.
音声分離の結果も主観評価.

目標：
・静かな環境での読み上げ文字起こしの精度を明らかにする.
・工場環境での読み上げの文字起こしの精度を明らかにする.
・工場環境のノイズに対応した文字起こしができるのか可能性を示す.

手順（案）：
・読み上げる製品番号を確定
・PCで録音ができるように準備
・読み手も複数人準備
・静かな環境でデータ収集，工場環境でデータ収集
・ノイズ除去（２種類）あり／なしで性能評価
・特定の文字の聞き取りが悪いかもしれない

拡張：
・この方法はオフラインで実行であるが，それをリアルタイムで実行するには？
・作業者の確認方法：再度読み上げ？