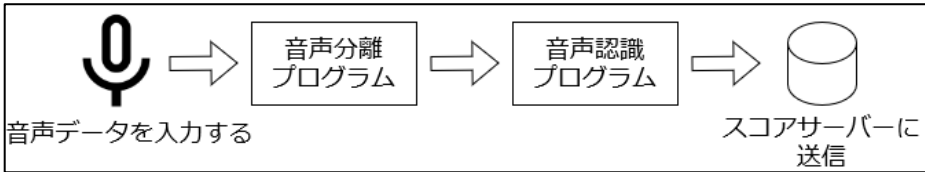


部 門	競 技 部 門	No.1 登録番号	30032
-----	---------	-----------	-------

No.2	1) 予定開発期間：6か月 2) 予定開発人数：3人																																								
	<table border="1"> <thead> <tr> <th></th> <th>4月</th> <th>5月</th> <th>6月</th> <th>7月</th> <th>8月</th> <th>9月</th> <th>10月</th> </tr> </thead> <tbody> <tr> <td>問題分析</td> <td colspan="2" style="text-align: center;">←→</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>設計</td> <td colspan="3" style="text-align: center;">←→</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>実装</td> <td></td> <td colspan="6" style="text-align: center;">←→</td> </tr> <tr> <td>試用・トレーニング</td> <td></td> <td colspan="6" style="text-align: center;">←→</td> </tr> </tbody> </table>		4月	5月	6月	7月	8月	9月	10月	問題分析	←→							設計	←→							実装		←→						試用・トレーニング		←→					
		4月	5月	6月	7月	8月	9月	10月																																	
	問題分析	←→																																							
	設計	←→																																							
実装		←→																																							
試用・トレーニング		←→																																							

No.3	<p>実現方法</p> <p>1) 音声の解析アルゴリズム</p> <p>下記の2つの手法のうち、試用した結果から、正答率・回答時間が優れている方を採用する。大まかな処理の流れとしては図1のとおりである</p> <p>案1)</p> <p>問題データは複数の音声データを合成したものであるため、複数のチャンネル（複数のマイク）を用いる線形フィルタリングではなく、単一のチャンネルでも実現が可能な、非線形フィルタリングを用いて、読みデータと問題データを比較して音源を識別する。読みが混ざったデータから、最も含まれている可能性の高い読みの音源データを打ち消す処理をかける。これを最後の一つになるまで繰り返し、結果を取り出す。このとき、打ち消した音源が正しかったのか検証する処理を掛けて誤っていた場合は修正する。</p> <p>案2)</p> <p>プログラムの構造として、音声分離部分と分離した音声の分類を行う部分の2つにわけたものを採用する。この2つを並列実行させて、処理の高速化を目指す。音声分離部分には Conv-TasNet を使用し、時間領域での特徴抽出、分離、分離した信号の出力をニューラルネットワークによって行う。ここで抽出・分離された音声データを音声認識プログラムにかけて最終的な回答を出力する。音声認識部のトレーニングデータには 0.5秒間隔で読みデータを区切ったものを使用し、札の読み全体を1つの単語として認識・判別できるようにする。</p>
	 <p>図1 音声処理の流れ</p>

No.4	<p>2) その他（独創的なところ）</p> <p>アプリケーションには GUI を作成し、ファイル取り込みや受信を簡便に行えるようにする。またリアルタイムで推測した結果を表示し、送信するかさらに分割データを取り寄せるか選択ができるようにする。さらに、人間の聴力による最終確認が行えるように、重なり合った音源データと、推測を行う PC 側で合成を行ったデータを随時スペクトログラムなどとして表示、対比させることで明らかな推測の誤りを検出できるようにする。</p> <p>最終結果の送信確認画面の表示などの各種 GUI の表示には Qt5 か WebView を用い実装を行う。問題データの推論を行うプログラムにサーバーの役割を持たせ、GUI を表示させるプログラムをクライアントとする設計を取ることで、実装の分担を容易に行えるようにする。</p> <p>また、上記の案2において、モデルデータの作成・学習には Python3 を使用し GPU などの各種アクセラレータを用いて行う。学習は推論の実行には C++ または Rust を使用し、できるだけ高速に推論を行えるようにする。</p>
------	--

No.4	<p>開発環境</p> <p>OS: Windows、Linux</p> <p>エディタ: Visual Studio Code</p> <p>使用言語: Python3、C++、Rust</p>
------	--