研究班番号【109】

機械学習を用いたカウンセリングロボットの開発

Development of a counseling robot with a machine learning

情報班:田中　継慈・高　忠希・山田　聖也
TANAKA Keiji and KOH Choonghee and YAMADA Seiya

abstract

The purpose of this study is to develop a voice interactive counseling robot. The programming language used was Javascript. As a result, we developed a listening counseling robot. However, because of the limitations of the answers, future work is to use machine learning to enable a variety of responses.

要約

本研究の目的は音声対話型カウンセリングロボットの開発である。プログラム言語はJavascriptを使った。結果として、我々は傾聴型のカウンセリングロボットを開発した。しかし、回答に限界があるから、機械学習を使って様々な対応を可能にすることが今後の課題である。

1. はじめに

人に話しにくい悩みでもロボットに対してなら話しやすいと考え、音声対話型カウンセリングロボットの開発をすることにした。

2. 作成方法

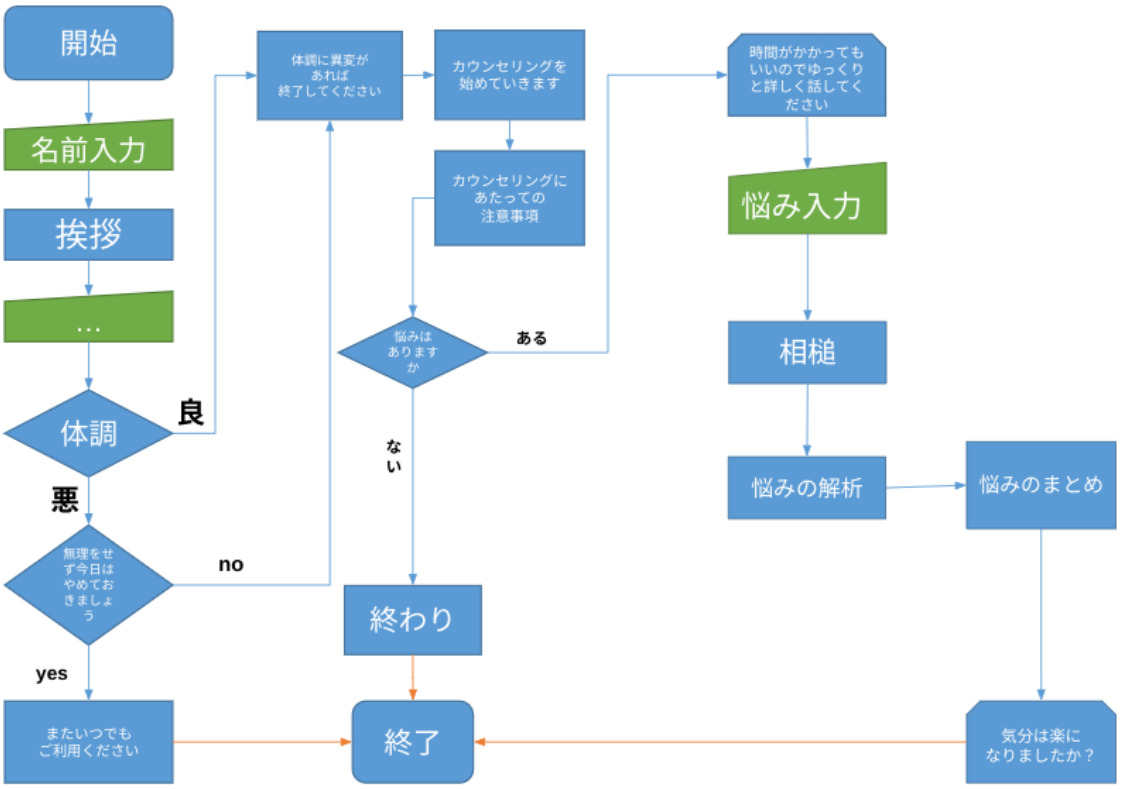
プログラム言語はJavascriptを用いJsfiddle上で実行した、APIはSpeechSyinthesis、SpeechRecognitionを用いた。また作成の前にフローチャートを作った。

3. 結果

利用者の話をロボットが聞くという傾聴型のカウンセリングロボットを作成した。このロボットを使うことで、利用者は傾聴を通して自分で悩みに対する答えを出すことができる。そのためこのロボットが悩みの回答を出すということはなく、あくまで聞き役に徹するようにした。当初の目標としていた機械学習を用いることは技術的、時間的な面で断念した。

次にカウンセリングのプロセスについて記す。実行ボタンを押して始めると、名前を入力し、ロボットが読み上げる。そして、身体的な体調に異変がないかどうかを聞いて、カウンセリングに入る。まず、どんな悩みか大雑把に話させて次にいつどこで起こったかという状況を質問する。そして、利用者は自分の気持ちを整理できるまで話しを続ける。その間ロボットは相槌を言葉が切れるごとに打つ。利用者がもう終わっても良いと判断し、その旨をロボットに伝えることで、ロボットが悩みの自己解決ができたかどうかを尋ねて終了する。

プログラムの詳細について記す、使ったAPIは前述した2つで、SpeechSyinthesisはロボットの質問の際の音声合成に、SpeechRecognitionは利用者の言葉を文字起こしする音声認識に使用した。軸となっているコードは条件をつけるifとSpeechSyinthesisである。工夫した点は、名前を入力するときに利用者が一人称や語尾を付け加えても名前だけを認識できるようにreplaceというコードを使って名前のみをとりだせるようにした。また、変数を使ってロボットの相槌が不規則に変化するようにした。



以下プログラミング

```
const taRecognized = document.getElementById("recognized");
var SpeechRecognition = SpeechRecognition || webkitSpeechRecognition
const recognition = new SpeechRecognition();
nyuryoku = 0;
count = 0;

const name_question = new SpeechSynthesisUtterance("お名前をお聞かせください");
speechSynthesis.speak(name_question);

recognition.lang = "ja-JP";
recognition.continuous = true;
recognition.interimResults = false;

recognition.onresult = function(event) {
    for (let i = event.resultIndex; i < event.results.length; i++) {
    var random = Math.floor( Math.random() * 4 );

    if (event.results[i].isFinal) {
      const result = event.results[i][0].transcript;
      taRecognized.innerHTML += (result + "¥n");
      if (nyuryoku == 0) {
```

```
          if (result.indexOf('') != -1) {
            let name = result.replace(/です|でござる|でやんす|やで|私の|僕の|俺の|名前は/g, "");
            const uttr = new SpeechSynthesisUtterance(name + "さんですね。よろしくおねがいします
");
            speechSynthesis.speak(uttr);
            const nayami_question = new SpeechSynthesisUtterance("体調はいかがですか");
            speechSynthesis.speak(nayami_question);

            nyuryoku = nyuryoku + 1

          }


        } else if (nyuryoku == 1) {
          if (result.indexOf('はい')!= -1 ||result.indexOf( 'よい') != -1||result.indexOf( '大丈夫') != -1||
result.indexOf('グッド') != -1|| result.indexOf('悪くない')!= -1 || result.indexOf('まあまあ')!= -1
||result.indexOf ('そこそこ')!= -1 ||result.indexOf ('完璧') != -1  ||result.indexOf ('いい') != -1) {

            const start = new SpeechSynthesisUtterance("それでは、はじめましょう。悩みはありますか。
");
            speechSynthesis.speak(start);
            nyuryoku = nyuryoku + 1

          } else if (result.indexOf('無理') != -1) {

            const start = new SpeechSynthesisUtterance("今日はやめておきましょう。またいつでもご利
用ください。");
            speechSynthesis.speak(start);
            nyuryoku = nyuryoku + 1
          }

        } else if (nyuryoku == 2) {
          if (result.indexOf('あります')!= -1 ||result.indexOf ('ある') != -1|| result.indexOf('聞いて') !=
-1) {

            const serihu = new SpeechSynthesisUtterance("わかりました。簡潔に話してください");

            speechSynthesis.speak(serihu);


            nyuryoku = nyuryoku + 1
          } else if (result.indexOf('ない') != -1) {

            const serihu = new SpeechSynthesisUtterance("わかりました。おわります。またいつでもご利
用ください。");
            speechSynthesis.speak(serihu);
            nyuryoku = nyuryoku + 1
          }


        } else if (nyuryoku == 3) {
```

```
        if (result.indexOf('') != -1) {

          nayami = result.replace(/です|でござる|でやんす|やで|私の|僕の|俺の|悩みは/g, "");
          const serihu = new SpeechSynthesisUtterance("なるほどあなたのなやみは" + nayami + "ですね" );
          speechSynthesis.speak(serihu);
          const serihu2 = new SpeechSynthesisUtterance("それはいつのときですか？");
          speechSynthesis.speak(serihu2);


          nyuryoku = nyuryoku + 1
        }

      } else if (nyuryoku == 4) {

        if (result.indexOf('')!= -1) {
            nayami = result.replace(/です|でござる|でやんす|やで|私の|僕の|俺の/g, "");

          const serihu = new SpeechSynthesisUtterance("そのときの状況を詳しく仰ってください");
          speechSynthesis.speak(serihu);

          nyuryoku = nyuryoku + 1
        }

      } else if (nyuryoku == 5) {
            const outou = new SpeechSynthesisUtterance("話すことがなくなったら言ってくださいね");
            const aiduti1 = new SpeechSynthesisUtterance("うん、うん");
        const aiduti2 = new SpeechSynthesisUtterance("そうなんですね");
        const aiduti3 = new SpeechSynthesisUtterance("はい");
        const aiduti4 = new SpeechSynthesisUtterance("そうですか");

            if(result.indexOf('終わり')!= -1||result.indexOf('話すことがなくなり')!= -1||result.indexOf('終了')!= -1||result.indexOf('以上')!= -1){
            const serihu = new SpeechSynthesisUtterance("悩みの自己解決はできましたか。");
        speechSynthesis.speak(serihu);
            nyuryoku = nyuryoku + 1
        return;

            }else if (result.indexOf('')!= -1) {
        if (count == 0){

            speechSynthesis.speak(outou);
            count = count + 1

            }else {
            if (random == 0){
                        speechSynthesis.speak(aiduti1);
                }else if (random == 1){
```

```
                    speechSynthesis.speak(aiduti2);
                }else if (random == 2){
                    speechSynthesis.speak(aiduti3);

            } else {
                speechSynthesis.speak(aiduti4);

                    }
                }
            }

        }else if(nyuryoku == 6) {
            if (result.indexOf('はい')!= -1||result.indexOf("うん")!= -1||result.indexOf("良い")!=
-1||result.indexOf("いい")!= -1|| result.indexOf('できました')!= -1) {
            const serihu = new SpeechSynthesisUtterance("お役に立てて良かったです");
          speechSynthesis.speak(serihu);
        }else if(result.indexOf('いいえ')!= -1||result.indexOf("いや")!= -1) {
            const serihu = new SpeechSynthesisUtterance("そう、ですか、申し訳ないです");
          speechSynthesis.speak(serihu);
        }
        const serihu = new SpeechSynthesisUtterance("これでカウンセリングを終了します。ご利用い
ただきありがとうございました。");

        speechSynthesis.speak(serihu);
        console.log(nayami);
        nyuryoku = nyuryoku + 1

      }
    }
  }
}

recognition.onstart = function(event) {
  console.log(event);
}
recognition.onerror = function(event) {
  console.log(event);
}
recognition.onend = function(event) {
  console.log(event);
}
```

4. 考察

今後の展望として、予め用意している会話では不自然だから、AIを導入して回答の幅を増やし、様々な場面に対応できることが必要である。また、音声合成に関しては発音や抑揚が人間とはかなり違っており、改善したい。今回は傾聴という手法だったが、ロボットが回答を出すというカウンセリング法が悩みの解決には早い場合もあるから、悩みに適切な回答を出すロボットもAIを利用して開発する研究を進めたい。

5. 結論

当初は機械学習を用いた音声対話型カウンセリングロボットの開発を目標としたが、時間的、技術的な問題で利用者に自己解決を促すカウンセリングロボットを開発した。しかし、回答に限界があるから、機械学習を使って様々な対応を可能にすることが今後の課題である。

6. 参考文献ならびに参考Webページ

カオナビ人事用語集　編集部(2022)　『傾聴とは？【意味・目的ややり方を簡単にわか　りやすく解説】』
https://www.kaonavi.jp/dictionary/keicho/

金城俊哉2016.『Python プログラミング逆引き大全　400の極意』
庄司美沙2019.『Pythonで動かして学ぶ自然言語処理入門』

7. 謝辞

この研究では、大阪工業大学の宮脇健三郎先生と杉川智先生にご教授いただき進めることができました。誠にありがとうございました。