岡工の先生チャットボットの制作

西田　橙波　森　彩華
山本　みゆな

１．研究概要

私たちが実際に岡山工業高校（以下岡工）に入学し､生活する中で気付いたことがある。それは、どの先生も教育熱心で優しく、親しみやすいということだ。そこで、岡工の先生を基にしたキャラクターのゲームを制作すれば、外部の方に先生の魅力を伝えることができるのではないかと考えた。

コンピュータが人と自然な会話ができることを目的として､Python を用いた人工無能のチャットボットを制作した。また、ユーザがより楽しめるよう、ビジュアルノベルを参考にした、ゲーム風の UI にした。

２．研究の具体的内容

（１）目的

岡工では生徒主体のオープンスクールが開催されている。そのため、生徒の雰囲気が分かりやすい反面、岡工の先生の魅力があまり伝わっていないのではないかと考えた。そこで、岡工に在籍する先生を基に、性格を反映したチャットボットを作成し､誰でも気軽に、先生との疑似的な会話が行えるようにするためにプログラムを開発することにした。

（２）開発環境

プログラムを作成するテキストエディタには Visual Studio Code（図１）、GUI の設計には QtDesigner を使用した。

プログラム言語は AI 開発でよく使用されている Python を選択した。Python とはプログラミング言語の一種で、Web 開発や人工知能など汎用性が高い言語である。人気のある言語であるため、参考書籍の多さや、シンプルで読みやすいことから、今回使用する言語とした。

QtDesigner は GUI を視覚的に設計するためのツールで、画面の構成に使った。ボタンやテキストボックスなどの要素を、視覚的に配置し、簡単にウィンドウを作成できる。作成した UI は Python のコードと連携できる。

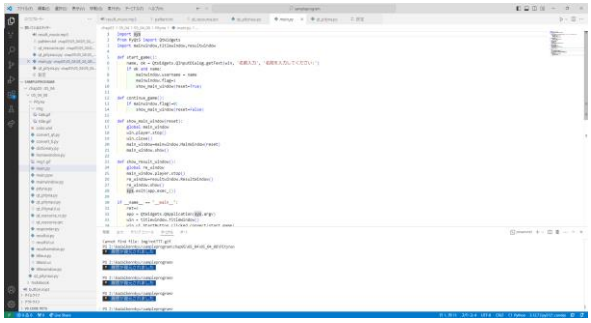

図１　コーディング画面

（３）開発の進め方

ユーザが入力した言葉に対する返答など、基本的なプログラムは参考文献をベースにして進め、その後自分達でタイトル画面などの機能を追加した。主に表１の年間計画をもとに作業を進めた。

１学期には、基本的な応答プログラムや、画面の設計を行い、感情を実装した。１学期後半から２学期にかけては、タイトル・リザルト画面・名前呼び機能など、よりゲームらしくなる要素の追加をした。３人で作業するため、それぞれ作業を分担し、手の空いた人は、パターン・ランダム辞書の内容を追加する作業を行った。想定よりも早く完了したため、11 月、12 月には、報告書や展示準備と同時進行で、年間計画にはなかった音楽の挿入を行った。

表１　年間計画

３～４月	応答プログラム・基本画面設計
５月	感情の実装
６～７月	タイトル・リザルト画面
９月	名前呼び機能
10 月	エンディング分岐機能
11～12 月	岡工祭展示・報告書作成

（４）チャットボットの設定

チャットボットの設定を決め、それに基づいた返信や表情画像を用意した。今回は、岡工の先生をモチーフにしたチャットボットを作成するために、表２をキャラクター設定とした。昨年度岡工情報技術科２年の担任を務めていた横田先生に実際に取材し、得られた情報などを参考にしている。（許可取得済み）

表２　キャラクター設定

名前	縦田（先生）
所属	情報技術科
担当教科	工業情報数理、PLC、計測実習
部活動	マイコン同好会
趣味	釣り
好物	ラズベリーパイ
性格	生徒想いで優しく、穏やか。 教育熱心で真面目。

（５）UI

図２　メイン画面

図２はメイン画面である。図２に表示した番号と対応した機能を表３に示した。

表３　メイン画面解説

No.	名称	詳細
1	会話ログ	会話ログを表示する
2	メッセージウィンドウ	縦田先生の返信を表示する
3	入力欄	ユーザの入力欄
4	話すボタン	３に入力した文字列を送信

（６）会話の仕組み

```
118    def button_talk_slot(self):
119        """[話す]ボタンのクリック時に呼ばれるメソッド"""
120        if not self.user_name:
121            self.prompt_name_input()
122            return
123
124        value = self.ui.LineEdit.text()
```

図３　会話について

図３のプログラムは、ユーザが「話す」ボタンをクリックした時に呼ばれる button_talk_slot メソッドである。まず、self.user_name が空であれば、ユーザに名前の入力を促す prompt_name_input メソッドが呼ばれ、処理が終了する。もし、ユーザ名が設定されていれば、self.ui.LineEdit.text()からユーザ名を取得し、value に格納する。

```
126        if not value:
127            self.ui.LabelResponce.setText('どうしたの?')
```

図４　入力欄が空の場合

入力欄に何もない場合(value=空の値)、「どうしたの？」と返信を LabelResponse（メッセージウィンドウ）に表示する（図４）。

```
134            self.ui.LabelResponce.setText(response)
135            self.putlog('> ' + value)
136            self.putlog(self.prompt() + response)
137            self.ui.LineEdit.clear()
```

図５　ログ表示

```
 99    def prompt(self):
100        """縦田先生のプロンプトを生成するメソッド"""
101        p = self.tateta.get_name()
102        return p + '> '
```

図６　返信ログ

入力欄にユーザの文字列がある場合、生成した返信(response)を LabelResponse（メッセージウィンドウ）に表示する。また、ログ(会話ログ)の処理もここで行う（図５）。

putlog メソッドを使い、ログに「>」と「value（ユーザの入力文字列)」を表示する。そして、prompt で取得した「返信のパターン」＋「＞」＋「縦田先生の返信」を表示する（図６）。

リセットしないと誤入力が増えたため、LineEdit.clear()で入力欄をリセットしている。

縦田先生との会話を可能にするために、「ランダム応答」「パターン応答」の２種類を組み合わせた。基本的にパターン応答が優先される。

パターン応答はパターン辞書(図７)を作成し、その辞書にマッチした会話をすると決まった返答を返す。

```
退屈です　　じゃあ…マイコンについて語ろうか
お腹す(いた|きました)　はい、ラズベリーパイ
こんにち[はわ]　こんにちは
```

図７　パターン辞書一例

tab で区切り、正規表現（例：退屈です）と応答フレーズ（例：じゃあ…マイコンについて語ろうか）に分けている。

正規表現とは、ユーザの入力した文字列を辞書に当てはめるために用意した特別なパターンのことである。

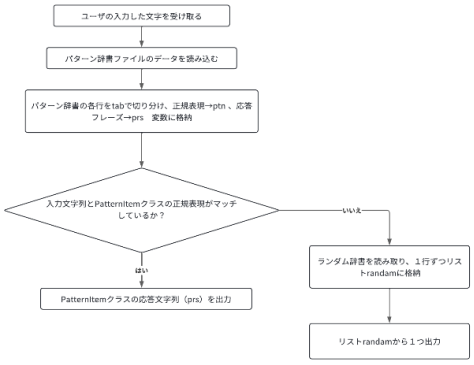

図８　パターン応答のフローチャート

図８はパターン応答のフローチャートである。

パターン辞書の内容を増やせば増やすほど多くの言葉に対応し、より円滑な会話が可能になる。また、「こんにちは」や「こんにちわ」などの表記ゆれ対策には、Python の re モジュールにあるメタ文字（表４）という特殊な表現方法を用いることで対応した。 []の中で、指定した言葉が一つでも含まれていれば、パターンにマッチする。

表４　メタ文字表

メタ文字	意味
［○●］	［］の中でどれか一文字
［○○｜●●］	｜で区切られたうちの一つ

ランダム応答（10%）は、ユーザが話す内容に関係なく事前に用意したランダム辞書（図９）の単語を返す。主に、ユーザの発言がパターン辞書のどれにもマッチしない場合に返すようにした。

```
マイコン...
マイクロコンピューター...
ラズベリーパイ...
コンデンサー...
時間たつの早いね
ん？マイコン同好会に入りたい？
```

図９　ランダム辞書

（７）感情の実装

プログラム内で感情を状態(数値)として考え、実装することにした。そこで、機嫌値というステータスを作り、縦田先生に疑似的に四段階の感情を実装させた。ある特定の言葉に反応・時間経過によって、初期値０から変動するようにした。

```
78  class Emotion:
79      """ 縦田先生の感情モデル
80
81      Attributes:
82          pattern (PatternItemのlist): [PatternItem1, PatternItem2, PatternItem3, ...]
83          mood (int): 縦田先生の機嫌値を保持
84      """
85      # 機嫌値の上限／下限と回復値をクラス変数として定義
86      MOOD_MIN = -100
87      MOOD_MAX = 100
88      MOOD_RECOVERY = 0.5
```

図 10　感情モデルの初期設定

図 10 は縦田先生の感情モデルの初期設定である。

MOOD_MIN（最小値）=-100

MOOD_MAX（最大値）=100

MOOD_RECOVERY（変動値）=0.5

```
122         # 機嫌値を徐々に戻す処理
123         if self.mood < 0:
124             self.mood += Emotion.MOOD_RECOVERY
125         elif self.mood > 0:
126             self.mood -= Emotion.MOOD_RECOVERY
```

図 11　会話について

self.mood は現在の機嫌値を示している。ユーザの入力を処理する際に変動する。もし、機嫌値が 0 以下なら、0.5 ずつ増加し、0 に戻す。0 以上であれば、0.5 ずつ減少させる（図 11）。

```
141         # 機嫌値moodの値を機嫌変動値によって増減する
142         self.mood += int(val)
143         # MOOD_MAXとMOOD_MINと比較して、機嫌値が取り得る範囲に収める
144         if self.mood > Emotion.MOOD_MAX:
145             self.mood = Emotion.MOOD_MAX
146         elif self.mood < Emotion.MOOD_MIN:
147             self.mood = Emotion.MOOD_MIN
```

図 12　会話について

142 行目の int(val)（発言によって変動する値）の追加により、現在の機嫌値が増減する。

そのとき、追加後の機嫌値が最大（最小）値を超える場合、最大（最小）値ちょうどにし、決められた値を超えないようにする（図 12）。

表 5　感情の変化

機嫌値	感情	変化
31 以上	喜(happy)	笑顔。好意的。
30-10	普通(talk)	デフォルト
-10-30	悲(empty)	悲しそうな顔
-30 未満	怒(angry)	怒り。素っ気ない。

表 5 に機嫌値による感情の変化について示した。

```
94      def change_looks(self):
95          """縦田先生の表情を変えるメソッド"""
96          em = self.tateta.emotion.mood
97          if -10 <= em <= 30:
98              self.ui.LabelShowImg.setPixmap(QtGui.QPixmap(":/re/img/talk.gif"))
99          elif -30 <= em < -10:
100             self.ui.LabelShowImg.setPixmap(QtGui.QPixmap(":/re/img/empty.gif"))
101         elif em < -30:
102             self.ui.LabelShowImg.setPixmap(QtGui.QPixmap(":/re/img/angry.gif"))
103         elif 30 < em:
104             self.ui.LabelShowImg.setPixmap(QtGui.QPixmap(":/re/img/happy.gif"))
```

図 13　感情変化

em = self.tateta.emotion.mood で、縦田先生の現在の機嫌値を em に代入。if 文で、現在の値がどの感情に分類されるか判別し、それに対応した表情の画像を表示する（図 13）。

```
5   class Tateta(object):
6       """ 縦田先生の本体クラス
7
8       Attributes:
9           name (str): Titynaオブジェクトの名前を保持
10          dictionary (obj:`Dictionary`): Dictionary(辞書データ管理)オブジェクトを保持
11          res_random (obj:`RandomResponder`): RandomResponder(ランダム応答)オブジェクトを保持
12          res_pattern (obj:`PatternResponder`): PatternResponder(パターン応答)オブジェクトを保持
13      """
14      def __init__(self, name):
15          """ Tatetaオブジェクトの名前をnameに格納
16              Responderオブジェクトを生成してresponderに格納
17
18          Args:
19              name(str)   : Titynaオブジェクトの名前
20          """
21          # Tatetaオブジェクトの名前をインスタンス変数に代入
22          self.name = name
23          # Dictionary(辞書辞書データ管理)を生成
24          self.dictionary = dictionary.Dictionary()
25          # Emotion(感情)を生成
26          self.emotion = Emotion(self.dictionary.pattern)
27          # RandomResponder(ランダム応答)を生成
28          self.res_random = responder.RandomResponder(
29                  'Random', self.dictionary.random)
30          # PatternResponder(パターン応答)を生成
31          self.res_pattern = responder.PatternResponder(
32                  'Pattern', self.dictionary)
```

図 14　会話について

図 14 が定義した Tateta オブジェクトである。名前・辞書データ・応答フレーズ・感情をそれぞれ生成、保持する。

図 15　感情変化

図 15 は縦田先生の 4 つの感情を示す表情の画像である。

15##マイコン同好会に入りたい(です|)　大歓迎！|ロボットコンテストに出よう！

図 16　ユーザの発言による変化

図 16 は、ユーザの発言によって機嫌値が変化する特殊なパターンの一例である。「マイコン同好会に入りたい（です)」と発言した場合、機嫌値が 15 上昇（15##）する。

PLCやりたいです いいね|10##本貸してあげる

図 17　応答の変化

図 17 は、現在の機嫌値によって、縦田先生が特殊な返答をするパターンの一例である。機嫌値が 10 以上(10##)の時「本貸してあげる」と返答する。

（８）タイトル画面

図 18　タイトル画面

図 18 はタイトル画面である。「はじめから」、または「つづきから」を選ぶことができる。タイトル名は「I&Teacher」で、意味は「私と先生」「わたしと縦田先生」という意味がある。また、アンテナ風のデザインなど、タイトル画面からも、情報技術科らしさが伝わるようにした。

「はじめから」を選んだ場合、最初に名前を入力する。入力された名前と進行状況（機嫌値や会話回数）は、再び「はじめから」を押さない限り保持される。

```
5   def start_game():
6       name, ok = QtWidgets.QInputDialog.getText(win, '名前入力', '名前を入力してください:')
7       if ok and name:
8           mainwindow.username = name
9           mainwindow.flag=1
10          show main window(reset=True)
```

図 19　名前保存

図 19 はユーザの名前を保存するプログラムである。「はじめから」ボタンを押すと、名前を入力するダイアログが現れる。名前を入力し、OK ボタンを押すと、mainwindow.username(name)にユーザの名前が格納される。mainwindow.flag によって、ユーザを設定したというフラグが立ち、ユーザの情報は、再度「はじめから」を押すまで更新されない。

```
48          global username
49          self.user_name = username
50          # 名前をログに表示
51          self.putlog(f"ようこそ、{self.user_name}さん！")
```

図 20　ようこそログ

グローバル変数 username を宣言し、ユーザの名前を格納する。会話ログに「ようこそ、(ユーザの名前）さん！」と表示する（図 20)。

```
127             response = self.pityna.dialogue(value)
128             """名前呼び"""
129             random.randint(1,5)==5
130             response=self.user_name+"さん、"+response
```

図 21　ランダム名前呼び

randam 関数でランダムに１～５の値を取得し、５の時（20%）に、response(縦田先生の返信)の前に、名前を付け足し表示する（図 21)。

図 22　実行例

図 22 はプログラムの実行例である。

（９）リザルト画面

図 23　リザルト画面

図 23 はリザルト画面である。

```
6   class ResultWindow(QtWidgets.QMainWindow):
7       def __init__(self):
8           """初期化処理"""
9           super().__init__()
10          self.ui = resultui.Ui_Dialog()
11          self.ui.setupUi(self)
12          self.initPlayer()
13          t1=str(mainwindow.count)+"回"
14          t2=str(mainwindow.mood)
15          t3=mainwindow.username + "さんの結果は…"
16          self.ui.kaiwa.setText(t1)
17          self.ui.kigenn.setText(t2)
```

図 24　リザルト画面処理（抜粋）

mainwindow.name（ユーザの名前）や mainwindow.count（会話回数）、mainwindow.mood（現在の機嫌値）を文字列に変換し、t1、t2、t3 に代入している。そしてその値を表示することで、リザルト画面にユーザ情報を載せることができる(図 24)。

(10) 動画・音楽

機嫌値によって、終了時に流れる動画(エンディングムービー)が異なり、エンディング分岐を実現させた。また、タイトル・リザルト・メイン画面で音楽を流した。

```
141             if  em > 90:  # 機嫌値が90以上のとき動画を再生
142                 self.flag=999
143                 self.play_video()
144                 self.close()
```

図 25　動画再生

現在の機嫌値を em とし、90 以上の時に動画を再生する。self.flag によって、エンディングを見た（クリア）というフラグを立てる(図 25)。

３．研究のまとめ

（１）展示について

オープンスクールや文化祭で展示した際、岡工の魅力などについても、ユーザが入力すれば返すことができた。動作も異常終了することがなかった。

しかし、「おはよう、そういえば、提出物忘れました」など、１文で２つ以上の意味（“おはよう”と“提出物忘れました”）を持つ文章だと、“おはよう”が優先され、会話がかみ合わないこともあった。

また、ユーザがプログラムの実行ボタンを押さないといけないため、スムーズにタイトル画面に移行しないこともあった。

（２）感想・反省

今回の研究で、人工無能を使った単純な会話の実現を成功させることができた。人工無能とゲームを組み合わせ、ユーザが楽しめるようなものに仕上がったと思う。

反省点は、チームで作業をしているため、それぞれの進捗がバラバラであったことから、スケジュールのずれが発生し、負担が偏ってしまう場面や時間の無駄が発生してしまい、安定して進めることができなかった。他にも、人工無能であるため、複雑な会話などには対応しきれないことも問題点であった。

参考文献

・金城俊哉（2024）「Python プログラミング パーフェクトマスター［最新 Visual Studio Code 対応第４版］」
・令和５年度(2024)課題研究/人工無能によるチャットボットの制作