横山研究室演習VIレポート
「人工無脳の制作」

2011SE286 渡邉将匡
2014SE003 赤羽里帆
2014SE048 木村孝大
2014SE067 水野竣太郎
2014SE081 大久保雄飛

1.概要

今回，演習の講義を利用して5人1組となって人工無脳を作成した．開発するにあたり，「人工無脳を動かすためのプログラム制作」「人工無脳の返答パターンを形成する辞書制作」の2部門に分かれる事にした．具体的には，

プログラム開発：木村，大久保
辞書制作：渡邉，赤羽，水野

という分担を行い，開発した．

2.開発に使用したツール

今回は人工無脳をAndroidアプリとして実装することを目標とし，人工無脳を動かす根幹となるプログラムにはKotlinを使用した．アプリの画面などのUIを司る部分は，Androidアプリ開発ではxmlが通常使われるため，これに則ってxmlを使用した．

辞書を開発するにあたり，Googleが提供するGoogleスプレッドシートを使用して作成し，完成した辞書をテキストファイルとして保存したのち，csvファイルに変換した．人工無脳プログラムの実装にはAndroid Studioを利用し，プログラムのバージョン管理はGitHubを利用した．なお，作成した辞書はcsvファイルに変換した後，プログラムと同様GitHubで管理した．
人工無脳の辞書を作成する方法に関する情報自体はソースコードのような人工無脳を直接動かす存在ではないため，今回は成果物として扱っていない．

3.開発する人工無脳の目標

人工無脳は，人工知能と同様に入力された文章に対応した返事を出力する機能を保有している．ただし，人工知能とは異なり，人工無脳はユーザとの会話を通じて学習する事がない．今回，開発チームは人工無脳の基本的な機能に加えて，以下の機能を追加する事にした．

・感情を1次元のパラメータとして持ち，入力によってその感情パラメータが変動する．
・同じ入力でも，感情パラメータの数値によって返事が変動する．

今回は以上の機能を実現させる事を目標にして，Android開発を行った．

4.設計
4.1 辞書設計

第一正規化表

正規表現	機嫌が悪い時の反応	機嫌が普通の時の反応	機嫌が良い時の反応	好感度の増減	機嫌が悪い時の語尾	機嫌が普通の時の語尾	機嫌が良い時の語尾

第二正規化表

正規表現	機嫌が悪い時の反応	機嫌が普通の時の反応	機嫌が良い時の反応	機嫌の増減(Int)

語尾ID(Int)	語尾	機嫌

二つの表の間に関連はない.

辞書の設計はデータベースの設計に則り，列挙されたカラムを第三正規化した.列挙されたテーブルが2つであり，関連がなかったためER図は作成していない.

4.2 プログラム設計

要求としては文献[1]にて説明されている人工無脳を実現したいということであり，具体的には，以下のとおりである.

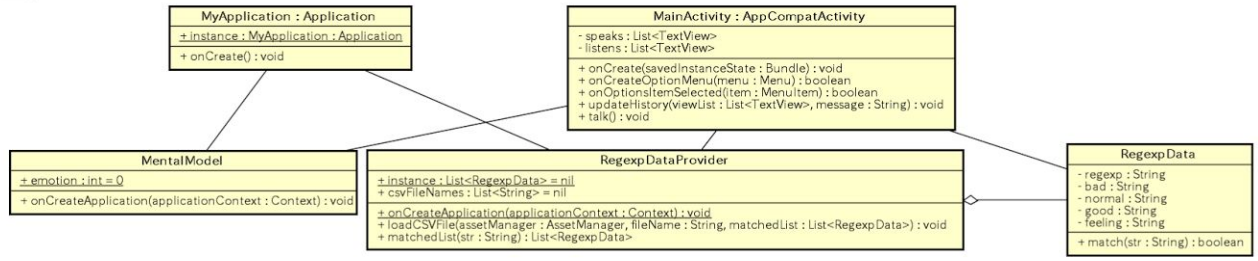
要求仕様書
・感情のような機能を持つ.
・ユーザーの入力を正規表現にマッチするか判定する.
・判定が真の場合は感情に応じて応答を返す.
　　良い，普通，悪いの3パターン
・判定が真の場合は感情を増減させる.
・判定させたい正規表現と３種類の応答及び感情の増減値をcsvファイルで持つ.
・csvファイルを選択する.
・入力と応答の履歴を5つまで表示する.

次に，以上で出てきている曖昧な用語などをデータ辞書で定義した.

データ辞書

応答 = ユーザの入力と正規表現をパターンマッチさせてマッチした場合に返す文字列
RegexpData = 正規表現 + 応答(悪い) + 応答(普通) + 応答(良い) + 機嫌の増減値

Ｃｌａｓｓ図



5.実装

前節における設計を元にkotlinで実装を行った．kotlinは関数型のプログラミング言語であり，JetBrain社によって開発された．構文はScalaの影響を強く受けており，Scalaと同様にJavaと完全な互換があり，自動変換可能である．現在はAndroidアプリケーション開発の公式プログラミング言語として扱われており，今後広く使われていくことが予想される．kotlinの特徴として，コンパイル時にnullチェックされる．また，nullを許容する型と許容しない型の２つにわけられており，実行時にnullに起因した例外が起こりづらくなっている．

実装に際して，androidアプリであることから，アプリを閉じたり開いたりした場合の処理が必要になった．そのため，MyApplicationクラス，RegexpDataProviderクラス，MentalModelクラスの３つをシングルトンとして扱うこととした．kotlinでシングルトンを実装するために，kotlinの機能であるobject及びcompanion objectというクラスに類似した構文を用いた．これらはそのインスタンスが必ず一つだけであることが保証されたオブジェクトである．

正規表現とのパターンマッチの実装に際して，複数の正規表現とマッチする可能性があったため，仕様として，複数マッチした場合は，マッチした要素のリストの先頭を使用することとした．

xmlの実装においては，アプリの機能として必要なテキストの入力フィールドと出力フィールドを配置した．そのほかに付加的に実装したものとして，Navigation Drawerを挙げる．これはAndroidアプリにおいて，通常時画面外に配置してあるものをスワイプ操作により引き出すという機能である．一般的に，Drawer部にはメニューなどを配置することが多い．今回は人工無脳との会話のログを5応答保持するものとしてDrawer部を利用した．

ソフトウェア全体の成果物はhttps://github.com/yokoyama-lab/Chatbotにあり，その中でも辞書の成果物は
https://github.com/yokoyama-lab/Chatbot/tree/develop/app/src/main/assetsにある．

6.考察

今回の開発にあたり，GitHubでバージョン管理を行った事により，バージョン管理を何処でも行う事ができ，複数人が同時に，かつ安全に行う事ができた．そしてAndroid Studioを利用して実装を行ったことにより，Githubでのバージョン管理において手軽にデータの更新を行うことができた．

また，辞書のデータ作成自体をGitHubではなくGoogle Driveを用いた事によって，グループ内の人間であれば誰でも編集出来るようになった．

7.課題点

今回の演習では，以下の課題点があった．

・開発したプログラムのテストを全体では行えなかった．
・作成した辞書のデータ量が不十分である．
・語尾表の存在意義が見出せなかった．
・テスト用の辞書を作成していなかった．

今回開発した人工無脳のプログラムは，開発者自身によるテストは行われたものの，チーム全員でプログラムの挙動を確認する機会を設けられなかった．

人工無脳に返答のレパートリーを与える辞書のバリエーションが少ない以上，予期していない入力を受けた時に適切な返答が出来なくなるという問題を回避する事が出来ない．その観点から，人工無脳に求められる機能が十分であったとは言いがたい．

今回は語尾表を作成する事を目標の一つに加えていたが，語尾の意義および適切な語尾とは何かを明確にする事が出来なかった事が原因で語尾表を作成する事が出来なかった．語尾表に関する問題は2点の原因が考えられる．

（1）辞書の開発を行う過程で語尾表に関する理解が出来なかった．

（2）語尾表は反応表と関連がないため，存在しなくてもプログラム上問題がなかったせいで，（1）の理由も含め，実装が進まなかった．

最も大きかった課題点は，テストケース専用の辞書を作成しなかった事であろう．今回はプログラム上で感情パラメータ導入，ホワイトボックステストでも問題はなかったものの，テスト用に極端に感情パラメータが変動する辞書を制作しなかったため，ブラックボックステストを行う事が出来なかった．

以上の観点から，今回は人工無脳を完成させる事は出来たものの，工学的に作成出来たとは言い難い．

参考文献

[1]秋山　智俊，2014，「恋するプログラム　Rubyでつくる人工無脳」，株式会社マイナビ