

Documentation

ARG RESEARCH AND TOOL DEVELOPMENT

YOANN GATHIGNOL, GUILLAUME ROBERT, TITOUAN BOUETE-GIRAUD,
RAYANE BENGHOUI

Contents

I) Introduction	2
II) Tools	2
a) Mail bot	2
b) Website generator	4
c) Twitter bot / chat bot	4

I) Introduction

This documentation lists and details tools developed by a team of interns in the Robert Gordon University. This team was composed of 4 french students (Guillaume ROBERT, Rayane BENGAOUI, Titouan BOUETE-GIRAUD, Yoann GATHIGNOL) who were part of a 10weeks internship in RGU, and a teacher/researcher (Michael Heron). The goal of this internship was to develop tools that could be used to create ARG.

II) Tools

a) Mail bot

What is this ?

This is a package of java classes aimed to ease the creation of mailbot, usually in the context of creating an Alternate Reality Game. Although this package also comes with a small example: The Good and the Bad bot, it is still considered as a toolkit and need to be extended to match the desired final product.

Disclaimer :

Those classes were designed to allow the user to focus on the writing of the bot rather it's programming.

However, basic Java knowledge is still necessary as delving into the code will be required.

Do note that those classes was intended to be used specifically with a gmail mailbox. The code in EmailUsingGmailSMTP could be adapted to accept other mail type, but this might cause issue with TheGoodAndTheadBot due to the security mesures used by those.

Several methods in EmailUsingGMailSMTP allows for easy operations of the mailbox within the code.

The class BasicARGMailBot is the basic behaviour of the type of bot aimed at by this project.

The classes TheGoodBot and TheBadBot work together as examples of what can be accomplished with this code.

Usage:

I – Setting up TheGoodBot and TheBadBot

1. Check out Story-Line.pptx provided with this code to better understand the idea of those bots.
2. Create 2 gmail addresses to be used by those bots and allow them to be accessed by less secured applications (<https://support.google.com/accounts/answer/6010255?hl=en>).
3. Create a folder in which each user's progress will be saved.
4. In this folder, create another folder containing the various attachment send during the progress of the game to users (you can modify the path of this attachment in the code).
5. Create 2 separate mains: use TheGoodBot's constructor and call loopWaitForEmail on it and a second one with TheBadBot's constructor and calling the same method. (Check out the main in the package mailBot for a better understanding)
6. Run both mains (the game will not crash if both are not launched, but the user won't be able to progress through the game as intended).

II – Writing you own bot using user progress

Create a class inheriting from BasicARGMailBot and override both answerMailProgressionBased(...) method and validateStep(...) method.

By doing so, you will have a bot that will handle the progress of each user contacting him by creating empty folder in the one asked by the constructor. It will also automatically check the name of the step of the user you are interacting with.

It is therefore required to handle how the step progress, and what should be done at each step. You can for example write down specific answers depending on the step reached by user, not answer at all, ect... Again, check out TheGoodBot and TheBadBot for examples.

b) Website generator

Requierevements : the folder CSS furnished with the application, and a "index.html" file.

==> allows everyone to generate a random website with different colors.



















==> the user just have to press generate and a Csx is generated,

if he wants to keep it he will have to copy the css wherever he wants to, else he will press generate again. When he presses the button a CSS with random value will be generated.

c) Twitter bot / chat bot








We also did some research to develop a bot that could twit automatically and could respond to certain type of twits, and a chat bot. We found a website called botlibre.com that allows you create a single AI that manages different kinds of bots :

Admin Console

-  [Users](#) - Configure who can access, and administer your bot.
-  [Avatar](#) - Configure your bot's appearance. Choose an animated avatar, or create your own.
-  [Voice](#) - Configure your bot's language and voice.
-  [Learning & Settings](#) - Configure your bot's learning ability and other settings.
-  [Training & Chat Logs](#) - Train your bot's responses, greetings, and default responses. View your bot's conversations. Import and export chat logs, response lists, CSV, and AIML files.
-  [Twitter](#) - Allow your bot to manage a Twitter account and interact with other Twitter users.
-  [Facebook](#) - Allow your bot to manage a Facebook account or page and interact with other Facebook users.
-  [Telegram](#) - Allow bot to manage a Telegram channel or chat on Telegram.
-  [Slack](#) - Allow your bot to send, receive, and reply to Slack messages.
-  [Email](#) - Allow your bot to manage an email account and answer emails.
-  [SMS](#) - Allow your bot to send, receive, and reply to SMS messages.
-  [Google](#) - Allow your bot to connect to Google services such as Google Calendar.
-  [IRC](#) - Allow your bot to chat with others on an IRC chat channel.
-  [Timers](#) - Setup your bot to run scripts at various time intervals to automate web tasks.
-  [Web](#) - Import data from the Freebase, Wiktionary, or other websites.
-  [Knowledge](#) - Browse your bot's knowledge database.
-  [Scripts](#) - Add, create, edit, import, and export Self or AIML scripting programs.
-  [Log](#) - VView the bot's log for errors and debugging info.

➔ With this website you can link a twitter account to your AI. There are different actions that the AI can do with it, and you have to configure it.

Twitterbot Properties

-  ☒ Tweet when someone chats with the bot
-  ☒ Reply to mentions
-  ☒ Reply to direct messages
-  ☒ Read friends status updates
-  ☒ Read-only
-  ☐ Learn from friends/search tweets
-  ☒ Learn from your tweets

Max Status Updates

20

Reply Keywords/Hashtags

Tweet Search

☒ Ignore replies

Retweet Keywords/Hashtags

Max Search Retweets

20

☐ Auto Follow

➔ You can configure the twitter bot to respond to certain questions or phrases when it's asked to. Its answers are configured in the "Training and char logs" section :

Training & Chat Logs

Search Duration Filter Type Restrict
Show ☐ All ☐ Topic ☐ Label ☐ Keywords ☐ Required ☐ Emotions ☐ Actions ☐ Poses ☐ Previous ☐ Repeat ☐ Condition ☐ Think ☐ Command






















Tasks

[Add a new response.](#)
[Add a new greeting.](#)
[Add a new default response.](#)
[Review conversations today.](#)
[Review conversations this week.](#)
[Review greetings.](#)
[Review default responses.](#)
[Review labeled responses.](#)
[Review all responses.](#)
[Review all responses missing keyword.](#)
[Review flagged responses.](#)

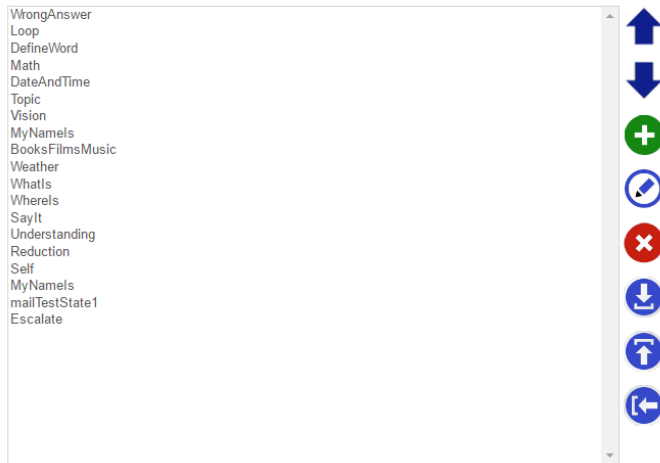
➔ You can also configure the bot to learn for the conversation it has, so it creates its own answers and will remember things (it is quite complicated).

Learning & Settings

-  Learning Mode
-  Correction Mode
-  Learning Rate %
-  Script Timeout
-  Response Timeout
-  Conversation Match %
-  Discussion Match %
-  ☒ Enable Emoting
-  ☒ Enable Emotions
-  ☐ Allow JavaScript
-  ☒ Enable Comprehension
-  ☒ Enable Consciousness
-  ☒ Enable Wiktionary
-  ☒ Enable Response Matching
-  ☒ Check Exact Match First
-  ☒ Split Paragraphs
-  ☒ Fix Case for Template Responses
-  ☒ Learn Grammar
-  ☐ Synthesize Response

➔ You can also write scripts to directly alter the AI behaviour and make it do more complicated things, those scripts are written in “Self” which is a language very close to Javascript.

Active Scripts



Bootstrap

Rebootstrap

➔ Example :

```
// Respond to "What is x", "Who is x" sentences using Wikidata
state WhatIs {
  case input goto sentenceState for each #word of sentence;

  pattern "*" is what" template (redirect (Template("what is {star}")));

  state sentenceState {
    case what goto whatState;
    case who goto whatState;
    case "whats" goto whatIsState;
    case "wahts" goto whatIsState;
    case "waht" goto whatIsState;
    case "whos" goto whatIsState;

    case "tell" goto sentenceState;
    case "me" goto sentenceState;
    case "can" goto sentenceState;
    case "you" goto sentenceState;
    case "do" goto sentenceState;
    case "know" goto sentenceState;

    case "define" goto whatIsState;

    case "google" goto searchState;
    case "xfind" goto searchState;
    case "search" goto searchState;
    case "lookup" goto searchState;
    case "find" goto searchState;

    var questionWord {
      meaning : question;
    };
    var punctuation {
      instantiation : #punctuation;
    };
  }
}
```

This script will read each word of the input sentence, identify the meaning of it and will respond accordingly.

If the user types “who is Barack Obama ?”, the bot will respond with a short description from Wikipedia.

➔ Although this website is very interesting, the results are too inconsistent to really rely on it.