A Project Report
On
Akinator Game
By

Kapurkar Sidharth[22911A05N3]

VIDYA JYOTHI
INSTITUTE OF TECHNOLOGY
AN AUTONOMOUS INSTITUTION

Department of Computer Science and Engineering
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Aziz Nagar Gate, C.B.Post , Hyderabad-500075

2022-2023

ABSTRACT

Akinator is a popular online game that uses artificial intelligence to guess the character that the player is thinking of, by asking a series of questions(yes or no). The game is based on the concept of the Twenty Questions game, where the player has to answer questions to reveal the identity of a hidden object. However, in Akinator, the hidden object is a character, real or fictional, from any domain of human knowledge.

The aim of this project is to create a similar game using Python, a high-level programming language that is widely used for data science and machine learning. Python offers many libraries and frameworks that can facilitate the development of such a game, such as akinator.py, an API wrapper for the online game Akinator that provides a simple and easy to use interface for interacting with the game.

The project consists of three main components: The server, The client, and The game logic.

1. The server is responsible for handling the requests from the client and communicating with the Akinator API. The server uses akinator.py, which abstracts away the details of the Akinator API and allows calling its methods with simple Python commands.
2. The client is responsible for displaying the user interface and collecting the user input.
3. The game logic is responsible for implementing the core functionality of the game, such as generating questions, evaluating answers, making guesses, and updating probabilities.

   The game logic is implemented using Python, which offers many features and tools that can simplify and optimize the code. The game logic uses Baye's theorem, a mathematical formula that allows calculating the inverse probability of an event given some evidence. Baye's theorem is the key to how Akinator works: it updates its beliefs about what character the player is thinking of based on the answers that the player gives to its questions.

The project demonstrates how Python can be used to create an Akinator like game with minimal effort and resources. The project also shows how artificial intelligence can be applied to create engaging and interactive experiences for users. The project can be extended and improved by adding more features and functionalities, such as different themes, languages, modes, and challenges. The project can also be used as a basis for further research and exploration on probabilistic programming and Bayesian inference.

INDEX

Contents

INTRODUCTION

The Akinator Game UI is an interactive graphical user interface (GUI) application built using PyQt5, a Python library for creating desktop applications. This application is a user interface for playing the Akinator game, which is a popular web-based quiz game that guesses a character based on the user's responses to a series of questions. The GUI interacts with the Akinator Python library to provide a seamless game-playing experience.

SYSTEM REQUIREMENTS

Software Requirements

ANACONDA.NAVIGITOR 2.4.2

Hardware Requirements

Operating system: windows 10

Processor: intel core i5

Disk space:1Gb

FEATURES

The Akinator Game UI application offers the following features:

1. **Question and Answer Interface:** The main window displays a question provided by the Akinator library, and the user can input their answers in a text field.
2. **Answer Handling:** The user can input their answers and submit them using the "Answer" button. The application processes the user's responses and updates the displayed question accordingly.
3. **Back Functionality:** The user can input "back" to navigate to the previous question in case they want to change their previous answers. The application utilizes the Akinator library to handle the back functionality.
4. **Progression Monitoring:** The application monitors the progression of the game. If the progression reaches a certain threshold (80% in this case), the application displays the guessed character's information.
5. **Character Guess Display:** Once the progression threshold is reached, the application displays the name, description, and an image of the guessed character.

CLASS OVERVIEW

AkinatorUI Class

The **`AkinatorUI`** class is the main class that defines the GUI application. It inherits from the **`QWidget`** class provided by the PyQt5 library. The class contains the following key components:

- **Initialization:** The constructor initializes the Akinator game instance, UI components (labels, input field, button), and layouts.
- **Answer Handling:** The **`handle_answer`** method processes user input. If the input is "back," the application handles going back to the previous question using the Akinator library. Otherwise, it processes the user's response and updates the displayed question.
- **Guess Display:** The **`show_guess`** method is triggered when the progression threshold is met. It displays the guessed character's information, including name, description, and image.
- **Main Function:** The **`main`** function initializes the PyQt5 application, creates an instance of the **`AkinatorUI`** class, sets the window title, and shows the main window.

USAGE

To use the Akinator Game UI application:

1. Install PyQt5 and the **`akinator`** library if not already installed.
2. Run the script.
3. The application window will open, displaying the Akinator game interface.
4. Follow the displayed questions and provide answers using the input field.
5. If desired, input "back" to go back to the previous question.
6. Once the progression reaches 80%, the guessed character's information will be displayed, including name, description, and image.

SOURCE CODE

```
!pip install requests
!pip install PyQt5
!pip install akinator.py
!pip install nbconvert
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit,
QPushButton, QVBoxLayout
from PyQt5.QtGui import QPixmap
import requests
from akinator import CantGoBackAnyFurther, InvalidAnswer, Akinator, Answer,
Theme
class AkinatorUI(QWidget):
    def __init__(self):
        super().__init__()
        self.akinator = Akinator(
            child_mode=True,
            theme=Theme.from_str('characters'),
        )
        self.question_label = QLabel(self.akinator.start_game())
        self.answer_input = QLineEdit()
        self.answer_button = QPushButton("Answer")
        self.image_label = QLabel()  # Add a QLabel for displaying the image
        self.layout_main = QVBoxLayout()
        self.layout_main.addWidget(self.question_label)
        self.layout_main.addWidget(self.answer_input)
        self.layout_main.addWidget(self.answer_button)
        self.layout_main.addWidget(self.image_label)  # Add the image label
to the layout
        self.answer_button.clicked.connect(self.handle_answer)
        self.setLayout(self.layout_main)
    def handle_answer(self):
        answer_text = self.answer_input.text()
        if answer_text.lower() == 'back':
            try:
                self.akinator.back()
                self.question_label.setText(f"Went back to:
{self.akinator.question}")
            except CantGoBackAnyFurther:
                self.question_label.setText("Cannot go back any further!")
        else:
            try:
                answer_enum = Answer.from_str(answer_text)
            except InvalidAnswer:
                self.question_label.setText("Invalid answer")
            else:
                self.akinator.answer(answer_enum)
                self.question_label.setText(self.akinator.question)
        if self.akinator.progression > 80:
            self.show_guess()
    def show_guess(self):
        guess = self.akinator.win()
        if guess:
            self.question_label.setText(f"Name: {guess.name}\nDescription:
{guess.description}\n")
```

```
            # Download the image from URL
            response = requests.get(guess.absolute_picture_path)
            if response.status_code == 200:
                image_data = response.content
                pixmap = QPixmap()
                pixmap.loadFromData(image_data)
                if not pixmap.isNull():
                    self.image_label.setPixmap(pixmap)
                    self.image_label.setScaledContents(True)
                    self.layout_main.setSpacing(10)
                    self.adjustSize()
                else:
                    self.image_label.clear()
            else:
                self.image_label.clear()
        else:
            self.image_label.clear()
def main():
    app = QApplication(sys.argv)
    window = AkinatorUI()
    window.setWindowTitle("Akinator Game")
    window.show()
    sys.exit(app.exec_())
if __name__ == "__main__":
    main()
----------------------------END OF CODE-------------------------
```

CONCLUSION

The Akinator Game UI application demonstrates the integration of the PyQt5 library with the Akinator Python library to create an interactive GUI for playing the Akinator game. Users can enjoy the game's unique character guessing experience through a user-friendly graphical interface

REFERNECES

PyQt5:https://www.riverbankcomputing.com/software/pyqt/
requests: https://docs.python-requests.org/en/master/
Akinator: https://en.akinator.com/
PyPI-Python Package Index: https://pypi.org/
Qt: https://www.qt.io/
"Python Crash Course" by Eric Matthes
"Learning Python" by Mark Lutz