Guess The Word

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

REPORT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MODULE CODE: MOD004553

MODULE LEADER: Mahdi Maktab-Dar-Oghaz

--

STUDENT: SIMONE CAMPO

SID: 1911840

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TABLE OF CONTENTS** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.0 SYNOPSIS ………………………………………………… 3 2.0 INSTRUCTION SET ……………………………………… 4

2.1 HOW TO RUN ….…………...………………….……. 4

2.2 HOW TO PLAY ……………………….………….….. 4

3.0 GAME COMPLEXITY……………………………………… 7

3.1 CODE OVERVIEW………………………..…………… 7

3.2 EASY MODE IMPLEMENTATION ………………… 7

3.3 HARD MODE IMPLEMENTATION ………...……… 8

4.0 FUTURE DEVELOPMENT PLAN …..…………………… 12

**1.0 SYNOPSIS** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hello everyone!

This is the report for the guess the word game. Your goal is to guess the word that a sort of AI agent thought.

Watch out though! The AI agent is smart, and he has developed a few techniques to make it difficult for you to guess its word. The AI agent will tell you how many letters its word is composed, and you have to guess the letters that the word can be composed by. Be careful as you do not have unlimited rounds.

In this brief report you will discover how to run the game, the instructions and how the AI agent is programmed to behave.

Good luck and have fun!

**2.0 INSTRUCTION SET** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2.1 How to run**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Guess the Word is a console based game written in Python. Therefore, you have to instal python interpreter and Conda environment in order for you to be able to run it.**

**Assuming you successfully installed Python interpreter on your machine, you can easily cd in your directory from the Terminal, and run the following command: ‘python GuessTheWord.py’**

**2.2 How to Play**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

There are two possible level of difficulties of this game, which you have to choose at the beginning (Figure 1). The main difference between those lies in the ability of the AI algorithm to find techniques to make it difficult for you to guess the word. You won’t be noticing anything, but the AI model plays smarter!

Text

Description automatically generated

Figure 1: Difficulty Menu

Regardless the level of difficulty, the AI agent will tell you:

* How many letters the word is composed by
* The amount of rounds available (twice the number of letters).
* The list of discovered words, so that you know which words you’ve already attempted
* The current state of the word (i.e., the pattern)

An example of the interface is displayed in Figure 2.

Text

Description automatically generated with medium confidence

Figure 2: Sample Initial Interface

Some input validation has been implemented so that you cannot provide symbols and multi-characters (e.g., “ab”). Additionally, if you suggest a letter that has already been discovered, then that letter would be ignored, and you will need to insert a new letter.

After a few rounds, the interface will look something like Figure 3.

Text

Description automatically generated

Figure 3: Interface of Game initiated

If the letter you guess happens to be in the word the AI agent thought, then the ‘\_’ symbol will be replaced with that letter in real time. No matters if you suggest letters in lower-case or upper-case. The game will always use the lower-case version. Additionally, the list of discovered words is alphabetically ordered so that you can easily figure out if a letter has already been used.

If you manage to guess the word before the end of the available rounds, the AI solution will congrats with you with a message (Figure 4).

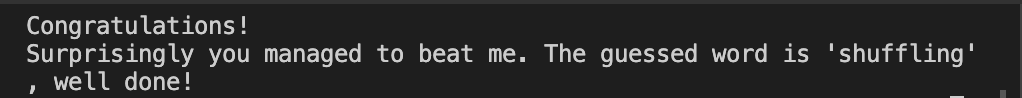


Figure 4: Game Won

Likewise, if you do not manage to guess the word, the AI agent will tell you the thought word (Figure 5).

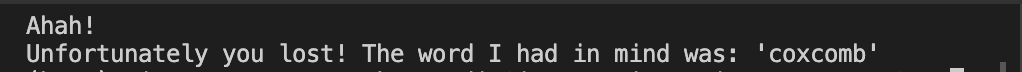


Figure 5: Game Lost

**3.0 Game Complexity**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3.1 Code Overview**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**The Python code is written following the Objrect Oriented Program (OOP) structure. Indeed, we have three pre-defined classes:**

* **Game, which initialises the list of available words loaded from the dictionary.txt file**
* **Player, which is the core class of the program defines the methods that the AI agent does to group words, set patterns, and choose the word family. Each function is explained through in text comments in the program**
* **GuessTheWord (Main Class), defines the thread of the game, it’s responsible for getting user input and calling the relevant functions.**

**3.2 Easy Mode implementation**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

As stated in Section 2.2, the only difference between EASY and HARD mode lies in the way the program decides which word Family to pick. In the easy mode, the AI agent only considers the number of words available within the word family, which has therefore a weight of 100% in the decision-making process.

**3.3 Hard Mode implementation**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In the HARD mode, the AI solution considers three factors before choosing the word family. The total weight is split in the following way:

|  |  |
| --- | --- |
| Factor | Weight |
| The number of words available in every word family. | 60% |
| The number of undiscovered words the pattern has. Indeed, a word with 5 unknown letters is expected to require more guesses than a word with only 2 unknown letters. | 30% |
| The maximum number of uniquely possible letters that are required to guess that word. Due to the nature of the game, these letters are not yet discovered, then the more they are, the less is likely a word will have duplicates of that letters, and therefore more guesses required to guess. This parameter is not as important as the others because of the different assumptions. | 10% |

The maximum percentage of every factor is assigned to the word family that performs the best. The weight of the other word families is calculated taking in consideration the best performances with the following formula:

**currentFamilyScore : maximumFamilyScore = x : maximumAssignableScore**

A function called setWeights calls the following functions in this order:

First Factor

1. getMaximumNumberOfWords()

This function returns the maximum number of words found in one of the word families

This value will be used as a point of reference to calculate the weights for all the other families.

1. setWorFamilyWright(maxItems, weights)

This function takes the maximum number of words found in one of the word families, and calculates the weight of all the word families in the current state.

Second Factor

1. getMaxUndiscoveredLetters()

This function returns the maximum number of undiscovered letters found in one of the word families (i.e., less letters are known, the better it is). This value will be used as a point of reference to calculate the weights for all the other families.

1. setMaxUndiscoveredWeigths(maxOccurrences, weights)

This function takes the maximum number of undiscovered letters found in one of the word families, and calculates the weight of all the word families in the current state.

Third Factor

1. getMaxNumOfPossibleLetters()

This function returns the maximum number of possible letters that can be used to complete the one of the words within a word family (i.e., less letters are known, the better it is)

This value will be used as a point of reference to calculate the weights for all the other families.

1. getMaxNumOfPossibleLetters(maxPossibleLetters, possibleLetters, weigths)

This function exploits the returned values from the getMaxNumOfPossibleLetters function, and assign the weights to the word families.

Final Stage

Finally, the weight dictionary will contain key-value pairs where:

* key: pattern of every word family
* value: its weight

setWeights function will then pick the word family with the highest pattern.

Text

Description automatically generated

Figure 6: Example of Weight dictionary

**4.0 FUTURE DEVELOPMENT PLAN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The game has been tested with different boundary cases and, even though there is no system in the world that comes with no bugs, all the major ones were identified and fixed. As a result, the Game run smoothly and without any issues.

Regarding the functionality, the AI model works well, and it is difficult to beat it unless you’re lucky to have the number of rounds close to the number of letters in the alphabet.

Even though there would be other solutions to come up with more accurate weights for word families, I would focus the improvement on the performances and the time complexity of the algorithm itself. It works well considering we’re dealing with a dictionary of 100.000 words. However, there is always room of improvement.