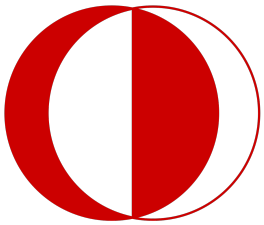

Middle East Technical University

Electrical and Electronics Engineering Department

EE443 - Project Proposal

December 2021

Student Name : M. Murat Kaynarca
Student ID : 2031003

Contents

1 Introduction

In this report, I will try to elaborate the project I have worked on in the summer of 2021.

The starting point of this project is based on a game series, Fallout. A post-apocalyptic role playing game based on nuclear war in an alternative timeline to our universe. My 3 dearest friends, Alperen Keleş (graduated from METU CENG, now TA at UMD, CP), Bilal Yavuz (graduated from METU CENG, now software engineer in Microsoft) and Semih Kafkas (undergraduate in METU CE) and I tried to find a solution to a mini game, that takes place inside the game. We have processed several wrong and horrible ideas multiple times and eventually we have acquired a fulfilling result that we all have agreed on.

In this report I will start with describing the mini game. Then I will proceed to explain my solution to the mini game.

2 Fallout Terminal Game

In the Fallout series, the terminal is placed around with several uses. It can disable the security systems in the building or open a door and such. An example of the terminal can be seen in Figure 1.

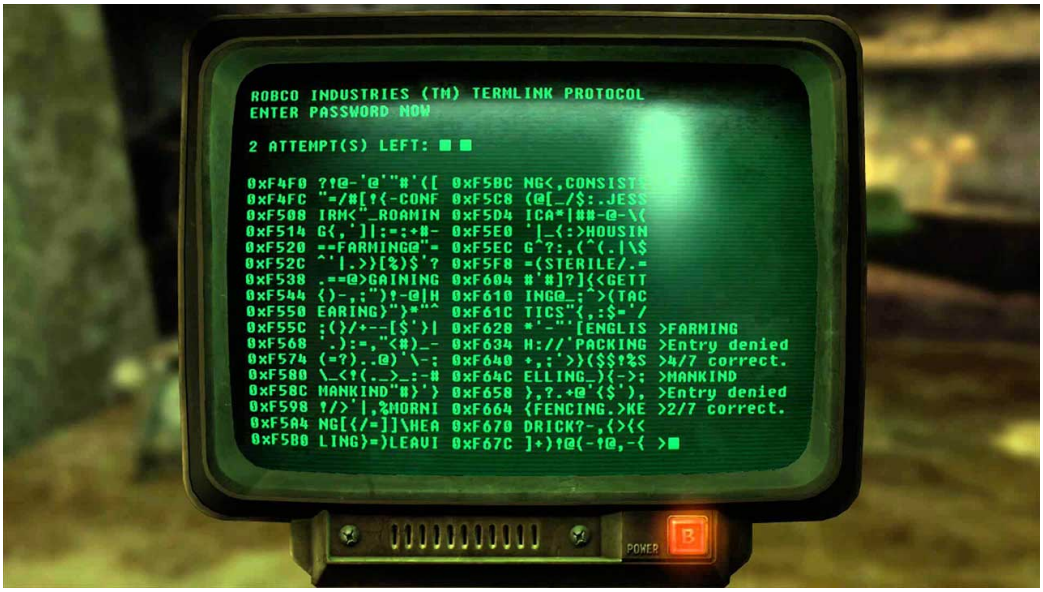

Figure 1: The Fallout Terminal

As you can see from the screen, there are several words of same length in the screen. Some of them are contoured with white lines in Figure 2. In the right side there are previous entries and their "correctness".

Figure 2: The Fallout Terminal properties

The game can be summarized as follows.

- There is a list of words (usually around 15 words) with same number of characters (depending on the difficulty), which are shown in the center of the screen.
- One of them is the password.
- If we click to a word, the game tells us how many correct letters in the correct place.
- If we click the password, we unlock the terminal i.e. win the game.
- There is limited number of guesses that we can take, or we got locked out of the terminal i.e. lose the game.

As an example let's say that we have the following list as our words, taken from the actual Fallout game in Novice difficulty (Table 1).

SEEDY
OWNED
BADLY
CHEAT
WORTH
LAWNS
HAVEN
HATES
GATES
HANDY
HUMOR
ALARM
NEVER
THANK
THROW
DYING

Table 1: Initial word list

Let's click "SEEDY". The game tells us that it has 0 likeness with the password. Thus password is among the words that has 0 likeness with the word "SEEDY". The likeness is calculated as follows in Tables 2 and 3.

S	E	E	D	Y
O	W	N	E	D
-	-	-	-	-

Table 2: Calculation of likeness of "SEEDY" and "OWNED", which results with 0

S	E	E	D	Y
C	H	E	A	T
-	-	+	-	-

Table 3: Calculation of likeness of "SEEDY" and "CHEAT", which results with 1

We do the same operation with every word thus our word list is reduced to Table 4.

OWNED
BADLY
WORTH
LAWNS
HAVEN
HATES
GATES
HUMOR
ALARM
THANK
THROW
DYING

Table 4: Reduced word list

Let's click on "THANK". The game says that it has likeness of 1 with the password. Reduced word list as follows in Table 5.

LAWNS
ALARM

Table 5: 2^{nd} reduced word list

As we can observe from the Table 5, the password is either "LAWNS" or "ALARM". Let's say that we have clicked on "ALARM". The game says that we have found the password.

3 Solution

In the thought processes we have always asked the wrong question such as "How do we find the password?", which is the fundamental mistake in our thought process. In the beginning we have always worked with deterministic methods, which led us to false solutions. After some time, we realized our mistake and approached the problem with probabilistic methods and focused on picking best next word. The method is as follows.

- Let's say that we have a word list containing n words, with length m.
- We create an nxn similarity matrix. We calculate each word's similarity with every word, including itself. Every row and every column represents a word. The value in i^{th} row, j^{th} column indicates the similarity of i^{th} word with respect to j^{th} word.
- Every word has similarity levels in between zero and m (similarity with itself). Thus we have n words, spread through 0 to m, gives us a probabilistic distribution.
- With this distribution, we can deduce the probability of the number of words that can be eliminated if we pick this word and if the likeness level of that word comes out as that likeness level.

Let's say that we have the same word list as previous example, with the password as ALARM again.

The similarity matrix as follows in Table 6.

	SEEDY	OWNED	CHEAT	BADLY	WORTH	LAWNS	HAVEN	HATES	GATES	HANDY	HUMOR	ALARM	NEVER	THANK	THROW	DYING
SEEDY	5	0	1	1	0	0	0	0	0	2	0	0	1	0	0	0
OWNED	0	5	0	0	0	0	1	1	1	1	0	0	1	0	0	0
BADLY	1	0	5	0	0	1	1	1	1	2	0	0	0	0	0	0
CHEAT	1	0	0	5	0	0	0	0	0	0	0	0	0	1	1	0
WORTH	0	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0
LAWNS	0	0	1	0	0	5	1	2	2	1	0	0	0	1	0	1
HAVEN	0	1	1	0	0	1	5	3	2	2	1	0	2	0	0	0
HATES	0	1	1	0	0	2	3	5	4	2	1	0	1	0	0	0
GATES	0	1	1	0	0	2	2	4	5	1	0	0	1	0	0	0
HANDY	2	1	2	0	0	1	2	2	1	5	1	0	0	0	0	0
HUMOR	0	0	0	0	0	0	1	1	0	1	5	0	1	0	1	0
ALARM	0	0	0	0	0	0	0	0	0	0	0	5	0	1	0	0
NEVER	1	1	0	0	0	0	2	1	1	0	1	0	5	0	0	0
THANK	0	0	0	1	0	1	0	0	0	0	0	1	0	5	2	1
THROW	0	0	0	1	1	0	0	0	0	0	1	0	0	2	5	0
DYING	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	5

Table 6: Similarity matrix for the word list

The distribution of the similarity levels for each word can be examined in Table 7. Rows indicate the word, columns indicate the similarity level.

	0	1	2	3	4	5
SEEDY	11	3	1	0	0	1
OWNED	10	5	0	0	0	1
BADLY	9	5	1	0	0	1
CHEAT	12	3	0	0	0	1
WORTH	14	1	0	0	0	1
LAWNS	8	5	2	0	0	1
HAVEN	7	4	3	1	0	1
HATES	7	4	2	1	1	1
GATES	8	4	2	0	1	1
HANDY	7	4	4	0	0	1
HUMOR	10	5	0	0	0	1
ALARM	14	1	0	0	0	1
NEVER	8	5	2	0	0	1
THANK	10	4	1	0	0	1
THROW	11	3	1	0	0	1
DYING	13	2	0	0	0	1

Table 7: Distribution matrix for word list

If we examine the HAVEN, we can see that there are;

- 7 words that has 0 similarity with HAVEN
- 4 words that has 1 similarity with HAVEN
- 3 words that has 2 similarity with HAVEN
- 1 word that has 3 similarity with HAVEN
- 0 words that has 4 similarity with HAVEN
- 1 word that has 5 similarity with HAVEN which is itself

We can deduce that if we pick HAVEN, the game can tell us

- 0 similarity with 7/16 probability
- 1 similarity with 4/16 probability
- 2 similarity with 3/16 probability
- 3 similarity with 1/16 probability
- 4 similarity with 0/16 probability
- 5 similarity with 1/16 probability

With these results, we eliminate;

- 9 words (16 - 7) with 7/16 probability
- 12 words (16 - 4) with 4/16 probability
- 13 words (16 - 3) with 3/16 probability
- 15 words (16 - 1) with 1/16 probability
- 16 words (16 - 0) with 0/16 probability
- 15 words (16 - 1) with 1/16 probability

If we take mean of these results, expected word count that the word HAVEN will eliminate is

$$(9 * \frac{7}{16}) + (12 * \frac{4}{16}) + (13 * \frac{3}{16}) + (15 * \frac{1}{16}) + (16 * \frac{0}{16}) + (15 * \frac{1}{16}) = 11.25$$

If we calculate this "elimination score" for each word the result will be as follows in Table 8.

SEEDY	7.75
OWNED	8.125
BADLY	9.25
CHEAT	6.375
WORTH	3.625
LAWNS	10.125
HAVEN	11.25
HATES	11.5
GATES	10.625
HANDY	10.875
HUMOR	8.125
ALARM	3.625
NEVER	9.25
THANK	8.625
THROW	7.75
DYING	5.125

Table 8: Elimination score table

With this table we can say that if we pick the word HATES which has the highest elimination score with 11.5, we are expected to eliminate 11.5 words.

If we do so;

A	L	A	R	M
H	A	T	E	S
-	-	-	-	-

The game will tell us that it has 0 similarities with the password and we will eliminate 9 words. We keep iterating this algorithm until the password is cracked.

- With the same steps, the best guess for 2nd round is THANK, which has 1 similarity with ALARM and eliminates 4 words.
- With the same steps, the best guess for 3rd round is CHEAT, which has 0 similarity with ALARM and eliminates 1 word.
- With the same steps, the best guess for 4th round is ALARM, which is the exact match for the example.

4 Results

- With this algorithm, over 100 examples in the hardest difficulty (around 15 words, 12 characters), we have solved the puzzle within 4 guesses and never lost the game.
- For testing the algorithm, we tried several extremely hard cases(10000 words, 15 random characters from the set A:G i.e. huge word list with very similar words). The max guesses that the algorithm takes to solve is 8 while it approximately solves within 6 guesses.
- I have written this algorithm in Python and it is available in this GitHub repository.