UDYAM'24

ANNUAL TECHNICAL FEST OF ELECTRONICS ENGINEERING SOCIETY

MOSAIC PS2

Hangman Game

Hangman is a classic word-guessing game typically played between two players where one player thinks of a word and the other player tries to guess it by suggesting letters within a certain number of attempts. In this context, we'll focus on the computer generating a word for the player to guess.

Here's a brief outline of how the game is typically played:

- **Word Selection:** The computer selects a word from a predefined list of words. In our case, we have a dataset of English words.
- **Displaying Hidden Word:** The computer displays the number of letters in the word as blanks, representing each letter with an underscore "". For example, if the word is "hangman," the computer displays " _ _ _ _ _ _ _".
- **Guessing Letters:** The player guesses letters one at a time, trying to figure out the word. If the guessed letter is in the word, all occurrences of that letter are revealed. If not, the computer keeps track of the incorrect guesses.
- **Game Termination:** The game ends when the player correctly guesses the word or runs out of attempts (commonly represented by drawing a hangman figure).

The goal of this problem statement is to develop an algorithm/model capable of playing the Hangman game effectively with a good score. Your algorithm has to suggest the next letter to guess based on the current state of the partially revealed word. The implementation should be able to handle unseen words during testing and provide accurate suggestions to maximize the chances of winning the game.

The task for participants is to implement the *'suggest_next_letter_sol'* function in the provided *'your_solution.py'* file, similar to that implemented in *'guess.py'*. This function should take into account the current state of the word (with known letters revealed and unknown letters represented by dashes) and suggest the next letter to guess. The function should be designed to make intelligent decisions based on the available information to maximize the chances of successfully guessing the word within the given number of attempts. Do not make any changes in any other files except *'your_solution.py' file*, however, you are encouraged to read the rest of the code to understand how the game works.

ELECTRONICS ENGINEERING SOCIETY

The statistical modeling approach gives around 18% accuracy. So try to improve this accuracy significantly using your approach.

Dataset:

The dataset provided for this problem statement consists of 50,000 English words. It's important to note that the dataset is self-contained, with no external sources allowed during training. This restriction ensures that the model learns solely from the provided data, enhancing its ability to generalize and make informed decisions during gameplay.

Evaluation:

Your algorithm will undergo evaluation on a distinct test dataset consisting of 15,000 words. This test dataset is entirely separate from the provided training dataset, ensuring that the model's performance generalizes well to unseen data. Participants should strive to develop algorithms that not only attain high accuracy on the training dataset but also demonstrate robust performance on unseen words during evaluation.

Commands related to the game:

- ***python hangman.py:*** This command executes the Hangman game based on the participant's implementation of word guessing in ***your_solution.py***. It allows participants to assess the performance of their solution in automatically playing the game.
- ***python hangman.py --play True:*** Running this command enables participants to manually play the Hangman game. This interactive mode helps participants understand the mechanics of the game and how their implementation affects the gameplay.
- ***python hangman.py --sample True:*** This command plays the Hangman game based on the solution provided in ***char_level_rnn.ipynb***. It serves as a reference implementation and can be used by participants to compare their solutions against a baseline.

Rules:

- No external dataset is allowed.
- Participants have to submit a detailed report explaining their approach.
- You must submit the your_solution.py file and your model(checkpoint), along with any other files required to run the code.
- Don't cheat, and have fun!