Trexquant Hangman Challenge

Atul Kumar Pandey

June 18, 2025

Overview

The Trexquant Hangman Challenge involves building an intelligent solver for the Hangman game using a machine learning model. The core idea is to train a neural network to predict missing letters in a word masked with underscores. This project uses a BiLSTM-based model combined with regex-based fallback mechanisms to guess letters with increasing accuracy.

Dataset Creation

The dataset was generated by masking words from a large dictionary. Each word was permuted by replacing random subsets of its characters with underscores. The input features were sequences of integers (mapping `a-z → 1--26`, `_ → 27`, and padding as 0) of length 35. The target was a 26-dimensional binary vector representing the presence of letters in the original word.
Augmentations included:

- Random letter masking
- Consonant-only masking (vowels preserved)
- Vowel prior generation using frequency distributions based on word length

Model Training

The model was trained using TensorFlow on a binary cross-entropy loss. The dataset was split into training and validation using an 0.98:0.02 split. Training was done using SGD optimizer with a learning rate of 0.001 for 8 epochs and batch size 128.

Model Architecture

The model used was a two-layer Bidirectional LSTM with the following architecture:

- `Embedding(input_dim=64, output_dim=64, input_length=35)`
- `BiLSTM(128, return_sequences=True, dropout=0.2)`
- `BiLSTM(64, return_sequences=False, dropout=0.2)`
- `Dense(48, activation='relu')`
- `Dropout(0.20)`
- `Dense(26, activation='sigmoid')`

The output is a 26-dimensional sigmoid layer representing probabilities for each letter.

Fallback Function

The fallback system activates when the model fails to give confident predictions. It relies on:

- Regular expression matching with partially known words
- Letter frequency analysis across matched candidate words
- Chunk-based substring analysis to guess likely letters
- Global letter frequency fallback as a last resort

These steps prioritize unguessed letters, penalize repeated guesses, and avoid over-guessing vowels when vowel ratio is high.

Guess Function

The complete guessing pipeline involves:

1. **Vowel Guessing Rule:** If many tries remain and vowel ratio is low, use vowel priors.
2. **Model Prediction:** Use the BiLSTM model to get 26 probabilities. Guess highest unguessed letter above threshold.
3. **Fallback:** If model gives low confidence predictions, use regex-based fallback guess.

Results

A 50-game practice evaluation was run for different model confidence thresholds. The following table summarizes the average accuracy:

Threshold	**Accuracy (%)**
0.55	48
0.60	52
0.65	58 (most consistent)

The model achieved its best accuracy of **58%** at a threshold of 0.0.65.

Conclusion: The combination of neural prediction and strategic fallback handling can effectively tackle the Hangman game with high accuracy.