Transformer-Based Hangman Solver

Dhairya Kantawala

IIT Bombay

April 29, 2025

Overview

- Problem: Predict missing letters in Hangman using deep learning.
- Solution: Enhanced Transformer model trained with masked language modeling.
- Focus: Mathematical formulation of model, training, and guessing strategy.

Data Encoding

- Characters mapped: a–z (1–26), _ (mask, 27), PAD (0)
- Each word: $x = (x_1, x_2, \dots, x_{20})$, $x_i \in \{0, 1, \dots, 27\}$
- Random mask: 30%–50% characters $\rightarrow$ _

Model Architecture

- Embedding: $E \in \mathbb{R}^{28\times512}$, Position: $P \in \mathbb{R}^{1\times20\times512}$
- Input: $H_0 = \mathrm{LayerNorm}(E(x) + P)$
- Transformer Encoder: 12 layers, 16 heads/layer, $d_{model} = 512$
- Feedforward: 2048 units, GELU activation
- Output: Linear layer to 26 logits (letters)

Transformer Block (Math)

$$
\begin{aligned}
\text{Attention}(Q, K, V) &= \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \\
\text{MultiHead}(Q, K, V) &= \mathrm{Concat}(\text{head}_1, \ldots, \text{head}_{16})W^O \\
\text{FFN}(x) &= W_2 \cdot \mathrm{GELU}(W_1 x + b_1) + b_2 \\
H_{l+1} &= \mathrm{LayerNorm}(H_l + \text{FFN}(H_l))
\end{aligned}
$$

Training Objective

- Masked Language Modeling: Predict masked letters.
- Loss: Cross-entropy only at masked positions.
- For position i masked:

$$\mathcal{L}_i = -\sum_{c=1}^{26} y_{i,c} \log p_{i,c}$$

- Total loss: Average over all masked positions.

Optimization

- Optimizer: AdamW
- Learning rate schedule: Cosine decay with warmup

$$\text{lr} = \begin{cases} \alpha \cdot \frac{\text{step}}{\text{warmup}} & \text{if step} < \text{warmup} \\ \alpha \cdot 0.5(1 + \cos(\pi \cdot \frac{\text{step}-\text{warmup}}{\text{total}-\text{warmup}})) & \text{otherwise} \end{cases}$$

- Gradient clipping: $\|\nabla\| \leq 1.0$

Guessing Strategy

- For input word x, model outputs logits $z_{i,c}$ for each masked position i.
- Softmax: $p_{i,c} = \frac{e^{z_{i,c}}}{\sum_{j=1}^{26} e^{z_{i,j}}}$
- Weighted sum over masked positions:

$$w_i = \frac{1/(i+1)}{\sum_{j=0}^{n-1} 1/(j+1)}$$

$$p_c = \sum_i w_i \cdot p_{i,c}$$

- Guess: $\arg\max_{c \notin \text{guessed}} p_c$

Results

- Achieved $\sim 88\%$ win rate on Dictionary words.
- Achieved $\sim 92\%$ win rate on common English words.
- Achieved $\sim 48\%$ win rate on tricky sientific words.
- Model generalizes to unseen words and patterns.
- Demonstrates power of Transformers for character-level tasks.

References

- Vaswani et al., "Attention is All You Need", 2017
- PyTorch Documentation
- Assignment Source Code