Hangman Character Prediction with Soft Targets: A mini-GPT Approach

Huabing Wang

August 9, 2025

Abstract

This report presents a Hangman character prediction mini-GPT achieving 56% success rate using a pure soft targets supervised fine-tuning (SFT) approach. The system employs a two-stage training pipeline: character-level pre-training followed by game-specific fine-tuning with continuous, differentiable soft targets derived from Softmax-smoothed information gain. Both stages utilize the same 12M parameter GPT architecture with 256-dimensional hidden states, trained on 200k words for pre-training and fine-tuning, with an additional 10k words each for validation and testing.

1 Introduction

The Hangman game presents a word-guessing challenge where players must identify a hidden word by guessing individual letters, with limited incorrect attempts allowed. Unlike simple character prediction tasks, Hangman requires:

- **Strategic reasoning**: Optimal letter selection based on information gain and word patterns
- **Context awareness**: Understanding current game state and constraint information
- **Risk management**: Balancing information gain against penalty accumulation
- **Pattern recognition**: Leveraging partial word information and English vocabulary knowledge

Our approach achieves 59% success rate through a sophisticated training pipeline combining character-level pre-training and pure soft targets supervised fine-tuning, without relying on any hard label supervision. The following sections detail our model architecture, training methodology, and the innovative soft targets approach that enables effective learning of Hangman strategies.

2 Model Architecture

The success of our Hangman solver relies on a carefully designed two-stage training pipeline that first establishes linguistic foundations through character-level pre-training, then builds reasoning capabilities through game-specific fine-tuning. Notably, both stages employ the same model architecture, allowing for seamless transfer learning without architectural modifications.

2.1 Model Configuration

Both pre-training and fine-tuning stages utilize identical model architecture, ensuring efficient transfer of learned representations:

This architectural consistency eliminates the need for model scaling between stages, simplifying the training pipeline while maintaining strong performance.

2.2 Two-Stage Training Pipeline

Our training methodology separates the learning process into two distinct but complementary phases. The first phase focuses on understanding the fundamental structure of English words, while the second phase specializes this knowledge for Hangman gameplay.

Parameter	Value
Architecture	GPT-style Transformer
Hidden Dimension	256
Number of Layers	8
Attention Heads	8
Dropout Rate	0.15
Max Sequence Length	64
Learning Rate	1×10^{-3}
Batch Size	512
Parameter Count	$\sim$12M

Table 1: Unified Model Configuration for Both Training Stages

2.2.1 Stage 1: Character-Level Pre-training

Pre-training Significance:

The pre-training stage is crucial for establishing the model's fundamental understanding of English word structure and character arrangement patterns. This phase serves as the linguistic foundation upon which game-specific strategies are later built. During this stage, the model learns:

- **Character co-occurrence patterns**: Which letters frequently appear together (e.g., 'th', 'ch', 'qu')
- **Vowel-consonant relationships**: English phonotactic constraints such as:
  - Vowels often separate consonant clusters
  - Common patterns like CVC (consonant-vowel-consonant)
  - Restrictions on consonant combinations
- **Common subword units**: Frequent prefixes ('un-', 'pre-', 're-'), suffixes ('-ing', '-tion', '-ed'), and roots
- **Position-specific character distributions**:
  - Words rarely start with 'x' or 'z'
  - 'q' is almost always followed by 'u'
  - Common word endings like '-ly', '-er', '-est'
- **Word length patterns**: Typical character sequences for words of different lengths

Pre-training Objective:

The model is trained using standard autoregressive language modeling:

$$\mathcal{L}_{pretrain} = -\sum_{i=1}^{|w|} \log P(c_i | c_1, ..., c_{i-1}) \tag{1}$$

This objective forces the model to predict each character based on preceding context, naturally learning character dependencies and word structure.

Learning Outcomes:

By training on 200k English words, the model internalizes:

1. **Morphological patterns**: Understanding of how words are constructed from morphemes
2. **Phonological constraints**: Implicit knowledge of pronounceable letter combinations
3. **Statistical regularities**: Character frequency distributions and conditional probabilities
4. **Contextual dependencies**: How early characters in a word constrain later possibilities

2.2.2 Stage 2: Hangman-Specific Fine-tuning

Building upon the linguistic foundation established during pre-training, the fine-tuning stage transforms the model into a Hangman player. This phase uses the same architecture but introduces game-specific training data and objectives.

The key advantage of maintaining architectural consistency is that the model retains all learned linguistic knowledge while acquiring game-specific reasoning capabilities. The fine-tuning process focuses on:

- Learning to interpret game states (partial words, guessed letters, error counts)
- Mapping linguistic knowledge to decisions
- Optimizing letter selection based on information gain
- Balancing exploration and exploitation in letter guessing

2.3 Enhanced Character Tokenizer

A critical component of our system is the specially designed tokenizer that bridges the gap between raw text and model input. The tokenizer's vocabulary is carefully crafted to support both general character sequences and Hangman-specific patterns.

The tokenizer uses a 128-token vocabulary specifically designed for Hangman:

Token Category	Tokens
Control Tokens (IDs 0-3)	`<pad>`, `<bos>`, `<eos>`, `<unk>`
Base Characters (IDs 4-29)	a-z (26 tokens)
Special Characters (IDs 30-34)	_, `,`, `:`, `|`, `space`
Custom Tokens (IDs 35-37)	`[SEP]`, `[MASK]`, `[CLS]`
Reserved (IDs 38-127)	For future prompt-specific tokens

Table 2: Tokenizer Vocabulary Structure

The large reserved space (90 tokens) allows for future extensions, such as specialized prompt tokens or game-specific markers, without requiring tokenizer retraining.

3 Dataset and Training Data Generation

3.1 Dataset Split

Our experiments utilize a carefully curated dataset of English words divided into three non-overlapping sets:

Dataset	Size	Purpose
Training Set	200k words	Pre-training and SFT training
Validation Set	10k words	SFT hyperparameter tuning
GRPO Set	5k words	For Future PPO/GRPO tunning
Test Set	10k words	Final evaluation

Table 3: Dataset Distribution

The test set remains completely unseen during both pre-training and fine-tuning phases, ensuring unbiased evaluation of the model's generalization capabilities.

3.2 SFT Training Data Generation

The quality and diversity of training data are crucial for developing effective Hangman strategies. Our data generation process creates realistic game scenarios that cover the full spectrum of possible game states, ensuring the model learns to handle both easy and challenging situations.

3.2.1 Soft Targets Generation Strategy

At the heart of our training approach is the innovative use of soft targets that provide continuous, differentiable learning signals. Unlike traditional hard labels that only indicate the "correct" answer, soft targets encode the relative value of different letter choices.

The SFT training data generator creates multiple game states for each word, simulating complete Hangman gameplay scenarios. The key innovation is the use of Softmax-smoothed continuous scores for candidate letters:

$$\text{Score}(c|s) = \frac{\exp(\text{count}(c)/T)}{\sum_{c' \in U} \exp(\text{count}(c')/T)} \tag{2}$$

where U is the set of unguessed characters in the word, $\text{count}(c)$ is the occurrence count of character c in the word, and T is the temperature parameter (default 1.0).

This formulation ensures that characters appearing multiple times in the word receive higher scores, reflecting their greater information value, while maintaining a smooth, differentiable objective.

3.2.2 Game State Simulation

To ensure comprehensive coverage of possible game scenarios, our data generator creates diverse game states through systematic sampling. This approach guarantees that the model encounters and learns from the full range of situations it might face during actual gameplay.

For a word with N unique characters, the generator creates training samples with total guess counts uniformly distributed from 0 to $N + 5$:

- **Total guesses range**: $[0, N + 5]$ where N = unique characters in word
- **Correct guesses**: Up to $N - 1$ (always leaving at least one character for prediction)
- **Wrong guesses**: Up to 5 (game limit is 6 wrong guesses)
- **Distribution**: Uniform sampling across the entire range

3.2.3 SFT Data Enhancement Strategy

Empirical analysis revealed that Hangman difficulty varies significantly with word length. Word with shorter length are usually harder to guess due to limited success space. To optimize performance on the most common and challenging cases, we implement a data enhancement strategy for SFT:

$$P_{\text{enhanced}}(w) = \begin{cases} \alpha \cdot P_{\text{original}}(w) & \text{if } L_{\min} \leq |w| \leq L_{\max} \\ (1 - \rho) \cdot P_{\text{original}}(w) & \text{otherwise} \end{cases} \tag{3}$$

where ρ is the target ratio for enhanced words (default 0.95), and $[L_{\min}, L_{\max}]$ is the enhancement range (default [5, 13]).

3.2.4 Training Sample Format

Each training sample encodes a complete game state along with the soft scoring of available letter choices:

```
{
  "guess": {
    "prompt": "_an___na[SEP]a,n[SEP]0/6",
    "completion": "i"
  },
  "label": "ianthina",
  "soft_targets": {
    "t": 0.211942,    # softmax(1/1.0)
    "i": 0.576117,    # softmax(3/1.0)
    "h": 0.211942     # softmax(1/1.0)
  }
}
```

Example explanation:

- Previous guesses: Letters 'a' and 'n' have been attempted
- Current mistakes: 0/6 incorrect guesses made
- Soft Targets: represent a probability distribution over the next optimal letter choice

This distribution suggests that 'i' is the choice (57.6% confidence) which can reveal more postions, while 't' and 'h' are reasonable alternatives (21.2% each).

4 Soft Targets Training

The SFT phase employs a novel pure soft targets approach that completely eschews hard labels in favor of continuous probability distributions. This methodology provides richer learning signals and better captures the nature of Hangman gameplay.

4.1 Training Methodology

Our training process directly optimizes the model to match the soft target distributions:

Algorithm 1 Soft Targets Training

Input: Game state s, soft targets P_{soft}
$x = \text{Tokenizer}(s)$
$\text{logits} = \text{Model}(x)$
Extract logits at prompt end position
Filter to a-z character logits only
$P_{model} = \text{Softmax}(\text{logits}/T)$
$\mathcal{L} = \text{KL}(P_{soft}||P_{model})$

4.2 Loss Functions

We experimented with two loss formulations:
Primary loss (KL Divergence):

$$\mathcal{L}_{KL} = \sum_{c=a}^{z} P_{soft}(c) \log \frac{P_{soft}(c)}{P_{model}(c)} \tag{4}$$

Alternative loss (Cross-Entropy):

$$\mathcal{L}_{CE} = -\sum_{c=a}^{z} P_{soft}(c) \log P_{model}(c) \tag{5}$$

5 Evaluation Methodology

5.1 Test Protocol

The evaluation follows a rigorous protocol to ensure fair assessment of model performance. The test procedure, as implemented in `sft_simulator.py`, simulates complete Hangman games:

1. **Sequential word selection**: Words are read from the test set in order, ensuring each word is tested exactly once
2. **Game simulation**: For each word, the model plays a complete Hangman game with up to 6 wrong guesses allowed
3. **State formatting**: Game states are formatted as `"_an__na[SEP]a,n[SEP]0/6"` for model input
4. **Character prediction**: The model outputs a probability distribution over a-z characters

6 Results and Analysis

Our experimental evaluation demonstrates the effectiveness of the pure soft targets approach for learning Hangman strategies.

6.1 Performance Metrics

The following table summarizes the key performance metrics achieved by our system:

Metric	Value
Hangman Success Rate	59%
Training Data	200k words $\times$ 4 samples
Validation Data	10k words (for hyperparameter tuning)
Test Data	10k words (unseen during training)
Pre-training Method	Character-level language modeling
Fine-tuning Method	Pure soft targets

Table 4: Final Performance Summary

6.2 Key Success Factors

Analysis of our results reveals several critical factors contributing to the model's performance:

1. **Two-stage training**: Character understanding $\rightarrow$ reasoning
2. **Continuous soft targets**: Differentiable information gain scoring
3. **Comprehensive game simulation**: Uniform coverage of game states
4. **Word length enhancement**: 95% of training data from 5-13 character words

7 Technical Innovations

This work introduces several technical innovations that advance the state of game-playing with language models.

7.1 Continuous Information Gain

The Softmax-smoothed scoring function represents a key innovation:

- Fully differentiable objective
- Temperature-controlled distribution sharpness
- Natural probability interpretation
- Stable gradient flow

8 Limitations and Future Work

While our system achieves competitive performance, several limitations point to promising directions for future research.

8.1 Current Limitations

- 59% success rate indicates room for improvement
- No dynamic strategy adaptation during gameplay
- Single-turn prediction without game tree search
- Fixed information gain metric (character count)

8.2 Potential Enhancements

- Reinforcement learning (PPO or GRPO) for multi-step planning
- Beam search for exploring multiple guess sequences
- Ensemble methods combining multiple strategies
- Integration with word frequency databases
- Dynamic scoring based on letter positions and patterns

9 Conclusion

This work demonstrates the effectiveness of soft targets training for game-playing tasks. By combining character-level pre-training with innovative soft target fine-tuning, we achieve 59% success rate on Hangman.

Key contributions include:

1. A novel training data generation strategy with continuous, differentiable targets based on information gain
2. Successful application of character-level pre-training for establishing linguistic foundations
3. Demonstration that complex reasoning can emerge from carefully designed soft supervision signals

The system provides a solid foundation for decision-making under uncertainty, demonstrating that language models can learn game-playing strategies through pure soft supervision. Our results suggest that similar approaches could be applied to other games and decision-making tasks where optimal actions depend on partial information and probabilistic reasoning.