Report of Deep Learning for Natural Language Processing

MengCao ZY2403801

buaa_cm@buaa.edu.cn

Abstract

This report conducts a comparative study of two deep-learning-based text-generation methods: an LSTM language model and a GPT-2 Transformer model. These models are applied to the task of generating wuxia novel text using the Jin Yong corpus. We detail the architectures of both models, describe the experimental setup, and analyze the trade-offs between generation quality, training efficiency, and resource consumption.

Introduction

Since the emergence of ChatGPT, interest in large pretrained generative models has surged. However, practical experience with these models often lags behind the theoretical hype. This work focuses on two representative architectures—recurrent (LSTM) and attention-based (Transformer)—to demonstrate how to: (1) train or finetune each model on the Jin Yong novels corpus, (2) generate sample wuxia text, and (3) compare their strengths and weaknesses. By focusing on these models, we aim to provide a clear understanding of their operational principles and practical performance in a specific domain.

Methodology

LSTM Language Model

LSTM (Long Short-Term Memory Network), an improved RNN, addresses the long-term dependency problem by using a **cell state** and **gating mechanisms** to selectively retain or forget information over sequences.

1. Cell State (*Ct*)

The cell state acts as a "conveyor belt" for long-term information storage, updated linearly to preserve gradients and avoid vanishing issues.

2. Gating Mechanisms

Three gates control information flow: - Forget Gate (f_t): Decides what to discard from the previous cell state C_{t-1}: $f_t = \sigma\big(W_f[h_{t-1}, x_t] + b_f\big)$ - Input Gate (i_t) & Candidate State ($\tilde{C}_t$): - Selects new information to store: $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ - Generates candidate new state: $\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$ - Output Gate (o_t): Controls the hidden state h_t output from the cell state:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \quad h_t = o_t \odot \tanh(C_t)$$

3. Cell State Update

Combines forgotten old information and new input: $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$

Principle: Embedding → multi-layer LSTM → fully connected softmax.

Training: Truncated BPTT with sequence length 100, batch size 2048, learning rate 1e-3, 10 epochs.

Generation: Autoregressive sampling with temperature or multinomial strategies.

GPT 2 Transformer Model

The Transformer, introduced in *"Attention Is All You Need"* (Vaswani et al., 2017), revolutionizes sequence processing by replacing recurrent layers with **self-attention mechanisms**, enabling parallel computation and modeling of global dependencies efficiently.

1.Self-Attention Mechanism

The core of Transformer lies in scaled dot-product attention, which computes relationships between all pairs of tokens in a sequence to capture contextual dependencies:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Query (Q), Key (K), Value (V): Linear projections of the input sequence, determining token relevance (Q-K similarity) and information aggregation (V).
- Scaling ($\sqrt{d_k}$): Mitigates gradient vanishing from large dot-product values in high dimensions.

2.Multi-Head Attention

Parallelizes self-attention across h heads to capture diverse sub-space relationships:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\dots,\text{head}_h)W^O, \quad \text{head}_i$$

$$= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

Each head learns distinct attention patterns, and outputs are concatenated and linearly transformed for final use (e.g., in encoder/decoder layers).

3.Encoder-Decoder Architecture

The Transformer consists of symmetric encoder-decoder stacks:

- Encoder: Processes the input sequence through:

Self-Attention: Each token attends to all tokens in the input (including itself).

Feed-Forward Network: Applies two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Residual connections and layer normalization stabilize training:

$$\text{LayerNorm}\left(x + \text{Attention}(x)\right)$$

- Decoder: Generates output sequence by adding cross-attention (attending to encoder outputs) between self-attention layers:

$$\text{CrossAttention}(Q = \text{decoder\_prev}, K = V = \text{encoder\_output})$$

4.Positional Encoding

Since Transformers lack recurrence, positional encodings inject sequential order into token embeddings using sinusoidal functions:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i}/d_{\text{model}}\right), \quad PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i}/d_{\text{model}}\right)$$

Where pos is the token position, i the dimension, and d_{model} the embedding dimension.

5.LoRA (Low-Rank Adaptation)

In our script, **parameter-efficient fine-tuning (PEFT)** is applied to a pre-trained GPT-2 language model using the **LoRA (Low-Rank Adaptation)** technique. LoRA introduces a lightweight and efficient way to adapt large language models by injecting trainable low-rank matrices into specific layers (e.g., attention projection layers) of the model, while keeping the original pre-trained weights frozen. This significantly reduces the number of trainable parameters, making fine-tuning feasible on limited hardware and with less data.

Instead of updating the entire model, LoRA only modifies a small number of parameters associated with the low-rank adapters. This allows for faster training, lower memory usage, and easier deployment while maintaining strong performance. In this case, the fine-tuning task is framed as a **causal language modeling** problem, where the model learns to predict the next token in a sequence using a corpus of classical Chinese martial arts novels.

Experimental Studies

1. Setup

- **Data**: *Heaven Sword and Dragon Saber* (《倚天屠龙记》), encoded in GB18030, with character-level tokenization.
- **Hardware**: Single NVIDIA RTX 4090 GPU (12 GB).
- **Metrics**: Perplexity on the validation set, time per epoch, GPU memory usage, and sample coherence.

LSTM Configuration

- Input sequence length: 100
- Embedding layer dimension: 256
- Hidden layer dimension: 512
- Learning rate: 10−3
- Batch size: 1024
- Epochs: 100

Transformer Configuration

- Learning rate: 3×10−4
- Batch size: 16
- Epochs: 300

2. Results

After training both models, we tested two scenarios: generating short text from short input and long text from long input.

LSTM Generation Results

- **Short Input Prompt**: “无忌哥哥，你可也曾答允了我做一件事啊。” (A sentence from the novel end).

  Output: “张无忌心中一动，喜道：‘师父大撒英雄大会，你何不叫我无忌这淫贼造反作乱，该当何时还知谢主？’”

  Analysis: Sentences are slightly unsmooth, but character names match the original work, indicating the LSTM has learned contextual patterns.
- **Long Input**: A passage from the novel describing 周芷若 (Zhou Zhiruo) and 张无忌 (Zhang Wuji).

  Output: “元兵几个方灰奔的挂血喷出，大声喝道：‘兀那老道，言语不知轻重！武当派转眼全灭。你不怕死，难道这山上余名道人弟子，个个都不怕死么？’这人说话中气充沛，身高膀阔，形相极是威武。张三丰长声吟道：‘人生自古谁无死，留取丹心照汗青’这是文天祥的大关连，但不知何以有犯计去的便连这些人都总镖头尽又了，我十倍也难常了。”张无忌脚步轻捷，向他神色高望，只见那情耸立，别沉着座船。众番僧上得眼有半数，他已”

  Analysis: Longer passages show repetition and logical drift, likely due to limited long-range dependency modeling.

GPT-2 Transformer Generation Results

- **LoRA Finetuning**: Applied Low-Rank Adaptation to freeze most pre-trained weights, updating only 0.35% (442,368/124,882,176 parameters), significantly improving efficiency.
- **Short Input Prompt**: Same as above.

  Output (Greedy Sampling): “张无忌道：‘我们不知道，我们跟你没有甚么？’张无忌道：‘我们不知道’”

  Output (Top-p=0.95, Top-k=50):

  “群雄冷笑道：‘我的朱九真不会说话，你这便会说到的，混的一哥。’私外的向来了这时，窗外冷笑道”

  “一声，手下倒是一笑。张无忌缓步而出。那提起了轻轻微的一笑，问道：‘纷姑娘，你害手件事杀害’”

  Analysis: Outputs are more diverse but occasionally grammatically inconsistent, reflecting the model’s creativity over memorization.

- **Long Input**: The same novel passage as the LSTM test.

  Output: “青笑的白被外。张无忌离周芷若被她说话，向周芷若不能高了啦，突然间道：‘我瞧，你说话！’只听得那姓芷若因步声说道：‘我空智之色，说话而瞧得正小儿。’张无忌心想：‘张三丰内力却可是等甚么知晕式豪？来终患小快，今日不禁这个小儿等沉骨，一无不曲成了他，们自己不敢等甚么，你’”

  Analysis: Maintains thematic coherence over longer spans but shows occasional token-level repetition.

  The specific result comparison is shown in Table 1.

Table1 Model Comparison Table

Comparison Dimension	LSTM Language Model	GPT-2 Transformer Model
Parameter Scale	~25.8M (lightweight, suitable for resource-constrained scenarios)	~523M (pre-trained model + LoRA fine-tuning, with only 0.35% tunable parameters)
Training Data	Uses corpus from a single wuxia novel Heaven Sword and Dragon Saber (《倚天屠龙记》)	Based on pre-trained model (Chinese Clue Corpus Small) + fine-tuning on Heaven Sword and Dragon Saber corpus
Training Configuration	- Sequence length: 100, Batch size: 1024, Learning rate: 1e-3, 100 epochs	- Max length: 512, Batch size: 16, Learning rate: 3e-4, 300 epochs
Training time	- Training time: ~2 minutes/epoch	- Fine-tuning time: ~1 minute/epoch
Generation Strategies	Simple Temperature/Multinomial Sampling (Autoregressive Sampling)	Supports Greedy Search, Beam Search, Top-k/Top-p Sampling (stronger controllability)
Typical Output Example	“张无忌心中一动，喜道：‘师父大撒英雄大会，你何不叫我无忌这淫贼造反作乱……’”	“一声，手下倒是一笑。张无忌缓步而出。那提起了轻轻微的一笑，问道：‘纷姑娘，你害手件事杀害’”
Generation Quality	- Short Text: Smooth sentences, character names consistent with the original work - Long Text: Logical drift and high repetition rate	- Short Text: High diversity, occasional unnatural grammar - Long Text: Better thematic coherence, rare word repetition

3.Analysis of Results

Generation Quality

- **LSTM**: Excels at short-range context, producing smoother sentences that resemble the original corpus, but struggles with long-term coherence and logical flow.
- **Transformer**: Generates more diverse and contextually flexible content, with better long-range dependency modeling, though it may produce grammatically unusual sentences due to its reliance on abstract reasoning over direct memorization.

Efficiency & Resources

- **LSTM**: ~25.8M parameters, trains in ~2 minutes/epoch with low GPU memory usage, suitable for resource-constrained environments.
- **Transformer**: ~523M parameters, finetunes in ~1 minute/epoch but requires substantial GPU memory and I/O, reflecting a trade-off between quality and resource demands.

Control & Flexibility

- **LSTM**: Limited to simple sampling strategies (temperature/multinomial).
- **Transformer**: Offers advanced decoding techniques (beam search, top-k/p), enabling finer control over output diversity and randomness.

Future work could explore finetuning larger pretrained models (e.g., GPT-3, T5) and incorporating reinforcement learning with human feedback to further enhance generation quality and grammatical consistency.

Conclusion

- **LSTM**: Fast, resource-efficient, and suitable for rapid prototyping or applications with constrained hardware, though it lacks long-range coherence.
- **Transformer**: Delivers higher-quality, more flexible generation but requires significant computational resources, making it ideal for tasks prioritizing output diversity and contextual depth.

References

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation.

[2] Vaswani, A. et al. (2017). Attention Is All You Need. NeurIPS.

[3] Radford, A. et al. (2019). Language Models are Unsupervised Multitask Learners. OpenAI.

[4] Wolf, T. et al. (2020). Transformers: State-of-the-Art Natural Language Processing. EMNLP.