DSAI SET 5

euron

Recurrent Neural Networks (RNNs), LSTMs, and GRUs

1. What are Recurrent Neural Networks (RNNs), and how do they differ from traditional feedforward neural networks?

Answer:

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data. Unlike traditional feedforward networks, which process inputs independently, RNNs maintain a hidden state that allows them to capture temporal dependencies by using connections that cycle back into the network.

Each time step in an RNN takes an input and a hidden state from the previous time step:

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$

where:

- Ht is the hidden state at time t,
- xt is the input at time t,
- Wh and Wx are weight matrices,
- b is a bias term.

However, standard RNNs suffer from vanishing and exploding gradient problems when processing long sequences, making them inefficient for capturing long-range dependencies.

---

2. How do LSTMs improve upon standard RNNs?

Answer:

Long Short-Term Memory (LSTM) networks are a specialized type of RNN that address the vanishing gradient problem using a memory cell and three gates: the **input gate, forget gate, and output gate**.

- **Forget Gate**: Decides what information to discard.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- **Input Gate**: Decides what information to update.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

- **Memory Update**:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

- **Output Gate**: Decides what information to pass to the next hidden state.

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

$$h_t = o_t \odot \tanh(C_t)$$

This gating mechanism allows LSTMs to selectively retain long-range dependencies in sequences.

---

3. What is a Gated Recurrent Unit (GRU), and how does it compare to LSTM?

Answer:

A Gated Recurrent Unit (GRU) is a simplified variant of an LSTM that reduces computational complexity while maintaining performance.

GRUs use only **two gates**: the **reset gate** and the **update gate**, defined as:

- **Reset Gate:**

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$$

- **Update Gate:**

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)$$

- **Hidden State Update:**

$$\tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Unlike LSTMs, GRUs do not maintain a separate memory cell (Ct), making them more computationally efficient but slightly less expressive.

Attention Mechanisms & Transformer Architecture

4. What is an attention mechanism, and why is it important in sequence models?

Answer:

An attention mechanism allows models to focus on relevant parts of an input sequence while processing each element. It is particularly useful in tasks like machine translation, where different words have varying levels of importance depending on the context.

The attention mechanism computes a weighted sum of input features:

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

where:

- Q(query), K (key), and V (value) are matrices derived from the input sequence,
- dk is the dimensionality of the key.

Self-attention mechanisms, as used in Transformers, allow models to relate different positions in a sequence regardless of their distance.

---

5. Explain the Transformer architecture introduced by Vaswani et al. in "Attention is All You Need."

Answer:

The **Transformer architecture** is a deep learning model designed to replace recurrent networks for sequence tasks. It consists of:

- **Multi-Head Self-Attention**: Computes attention multiple times in parallel.
- **Positional Encoding**: Adds sequence order information.
- **Feed-Forward Networks**: Fully connected layers after self-attention.
- **Layer Normalization & Residual Connections**: Stabilize training.

Transformers eliminate the sequential nature of RNNs, allowing parallel computation and significantly improving efficiency.

BERT, GPT, and Large Language Models (LLMs)

6. What is BERT, and how does it differ from GPT?

Answer:

BERT (**Bidirectional Encoder Representations from Transformers**) is a **bi-directional** language model trained using **Masked Language Modeling (MLM)**. It captures context from both left and right simultaneously.

GPT (**Generative Pretrained Transformer**) is an **auto-regressive** model trained using **causal language modeling**, predicting one token at a time from left to right.

BERT excels in **understanding tasks** (e.g., NER, classification), while GPT is better for **generative tasks** (e.g., text generation, summarization).

---

7. How does fine-tuning work in BERT?

Answer:

Fine-tuning in BERT involves:

1. Adding a task-specific **head** (e.g., classification, QA).
2. Training on labeled data with a smaller learning rate.
3. Using **transfer learning**, where pre-trained embeddings are adjusted for a specific task.

NLP Applications

8. What are some challenges in Named Entity Recognition (NER)?

- Ambiguity: "Apple" can mean a company or fruit.
- Data imbalance: Rare entities occur less often.
- Multi-word entities: "New York Times" must be recognized as one entity.

---

9. How does text classification using Transformers work?

- Tokenize text using **WordPiece/BPE**.
- Pass through **pre-trained BERT/GPT model**.
- Extract **[CLS]** token embedding for classification.

---

10. Explain the process of machine translation using Transformers.

- Encode source text to embeddings.
- Apply self-attention to capture dependencies.
- Decode using **beam search or greedy decoding**.

---

11. How does extractive text summarization work with BERT?

- Encode document.
- Rank sentences by **importance scores**.
- Select top-ranked sentences.

---

12. How does abstractive text summarization differ?

- Generates **new text** rather than selecting.
- Uses **Seq2Seq models** like **T5**.

---

13. Explain question answering using BERT.

- Pass context + question to BERT.

- Use start/end token classification.

---

14. What is contrastive learning in NLP?

- Used in **sentence embeddings** (SimCSE).
- Maximizes similarity for positive pairs.

---

15. How do we handle Out-of-Vocabulary (OOV) words?

- **Subword tokenization** (BPE, WordPiece).
- **Character-level models** (CharRNN).

---

16. How do Retrieval-Augmented Generation (RAG) models work?

- Retrieve relevant documents.
- Generate responses conditioned on retrieval.

---

17. What are LoRA and QLoRA in fine-tuning LLMs?

- LoRA: **Low-rank adaptation**, updating only small matrices.
- QLoRA: **Quantized LoRA**, reducing memory footprint.