Transformers II

Transformers

- We will continue to look at
  - Encoder-only/decoder-only models
  - In-context Learning
  - Parameter-efficient transformers
  - Low-rank adapters
  - Model distillation
  - Sentence Transformers
  - Reinforcement learning with transformers

Encoder-Decoder Transformers

- Originally, transformers had an encoder and decoder
- The encoder for input sequence and decoder for output sequence
- For machine translation, the decoder produces the text in the foreign language
- For language modelling, the decoder produces the next word in the sequence
  - Decoder output is input shifted by one word

Encoder-Only Transformers

- In encoder-only transformers, the decoder is removed
- These models produce an output for the sequence
- For example, in BERT the output is a vector for each token in the input sequence

---

Decoder-Only Transformers

- In decoder-only transformers, the encoder is removed
- This is used by models such as GPT
- They are focused on generation and trained on next word prediction

---

Fine-tuning Transformers

- Transformers are typically trained on a large corpus of text

- The model is then fine-tuned on a specific task
- In order to fine-tune, we take the model and then add a task-specific head
- The model is trained as usual, but the weights are initialized from the pre-trained model

---

In-content Learning

- In-context learning is a technique to adapt a pre-trained model to a specific task
- It does not require fine-tuning the entire model (i.e., updating the weights)
- Transformers are capable of in-context learning if trained on a diverse set of tasks

---

Prompt engineering

- Techniques for prompt engineering:
  - Zero-shot learning
  - Few-shot learning
  - Chain of thought
  - Prefix tuning

Parameter-efficient Transformers

- Transformers are very large models
- Most don't fit on a single GPU
- Reducing the number of parameters is possible
- Trade-off between model size and performance

Model distillation

- Model distillation is a technique to train a smaller model to mimic a larger model
- The smaller model is trained on the output of the larger model
- DistilBERT was able to reduce the size of BERT by 40% while maintaining 97% of the performance

---

Parameter-efficient Fine-tuning

- Language models such as GPT-3 have billions of parameters

- This makes them very expensive to train
- Parameter-efficient transformers are designed to be smaller and faster to train

---

Low-Rank Adapters (LoRA)

- Low-Rank Adapters (LoRA) are a technique to allow efficient training of large models
- The idea is to add a small adapter layer to each transformer layer
- This adapter layer calculates a low-rank delta, ΔW, that is added to the weights of the self-attention layer

$$W_{new} = W_{old} + \Delta W$$

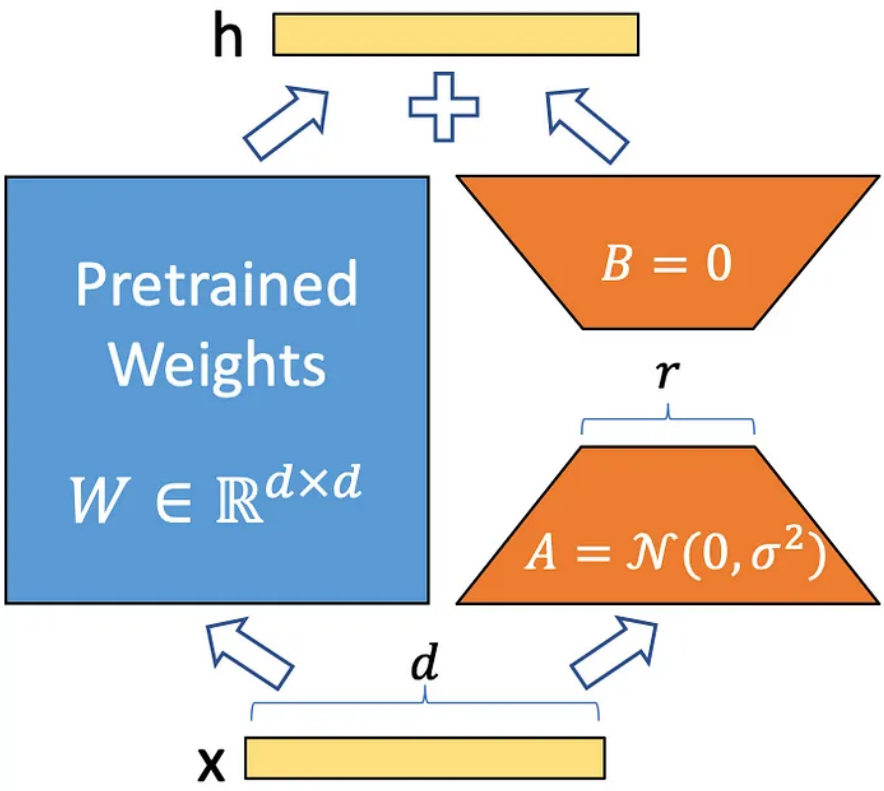

Figure 1: Low-Rank Adapters

Low-Rank Adapters (LoRA)

- The delta is a product of two low-rank matrices with $r \ll d$.
- Faster to train
- Smaller memory footprint
  - Not less than inference on the full model!

Quantization

- Quantization is a technique to reduce the precision of the weights
- Instead of 32-bit floating-point numbers, we use 8-bit integers
- This reduces the memory footprint and speeds up training

Quantization Precision Levels

- Floating-point numbers consist of a sign bit, exponent, and mantissa

$$\text{float} = (-1)^s \times 2^e \times 1.m$$

NB Mantissa is a binary fraction

Quantization Precision Levels

Bits	Sign	Exponent	Mantissa
64	1	11	52
32	1	8	23
16	1	5	10
8	1	3	4

Affine Quantization

We can quantize to integers using the following formula:

$$x_q = \text{round}(\frac{x}{s} + Z)$$

Where s is the scale factor and Z is the zero-point

This allows us to quantize down to 8, 4 or even 2 bits.

Quantization

- Quantization can be done in two ways:
  - Post-training quantization
  - Quantization-aware training

Post-training Quantization

- Post-training quantization is done after the model is trained
- The weights are quantized to a lower precision
- Further training on the quantized model can improve performance

---

Quantization-aware Training

- During forward passes, weights and activations are simulated as quantized (e.g., 8-bit integers)
- Stored as floating-point numbers during training.
- Gradients are calculated in full precision.
- Enables higher accuracy and fast inference.

Sentence Transformers

- Sentence transformers are models that take a sentence as input and produce a vector representation
- These vectors can be used for tasks such as semantic similarity, clustering, and classification

---

Siamese Networks

- Siamese network, uses two similar examples x_1 and x_2.

$$h_1 = f(x_1)$$

$$h_2 = f(x_2)$$

$$\text{similarity} = \text{cosine}(h_1, h_2)$$

Triplet loss

- Triplet loss, uses a positive example x_p and a negative example x_n.

$$\text{loss} = \max(0, \text{margin} - \text{cosine}(h_a, h_p) + \text{cosine}(h_a, h_n))$$

Sentence Transformer Output

- Mean pooling: average of the token embeddings
- Max pooling: maximum of the token embeddings
- CLS token: the output of a special token `[CLS]` in the transformer

---

Using Sentence Transformers

- Textual similarity: cosine similarity between two vectors
- Classification: add a new network to predict class from vector
- Clustering: use k-means or hierarchical clustering
- Information Retrieval: use vectors to retrieve similar documents; often using vector database such as FAISS

Reinforcement Learning with Transformers

- Combines reinforcement learning with human-provided feedback
- Used to train models such as GPT

RLHF

1. Pre-train a model on a large corpus
2. Collect human feedback
    - Model generates answers
    - Human annotators rank results

3. Train a reward model
    - Predict a numerical reward from human feedback
4. Fine-tune the model using reinforcement learning
    - Use Proximal Policy Optimization (PPO) or other RL algorithms

---

Why Reinforcement Learning?

- Align model with human values
- Handle ambiguous or subjective tasks
- Mitigate harmful behaviour

Summary

- Transformers are a powerful architecture for NLP
- Most transformers are encoder-only or decoder-only
- Transformers can be fine-tuned for specific tasks
- Parameter efficient transformers can be trained more easily
- Sentence transformers are used for semantic similarity and other tasks
- Reinforcement learning can make transformers more human-like