Self-Supervised Learning and Large Pretrained Models

Contextualized Representations

- **Contextualized representations** are **embeddings** that **depend on the context**;
- Words can have different meanings depending on the context;
- **ELMo** is a model that learned **context-dependent embeddings**;
  - **E**mbeddings from **L**anguage **Mo**dels;
  - **ELMo** is a **bidirectional LSTM** model - BiLSTM;
  - Save all parameters at all layers;
  - Then, for your downstream task, tune a scalar parameter for each layer. and pass the entire sentence through this encoder.

Pretraining and Fine-Tuning

- Recalling the **language modeling** task: given a sequence of words, predict the next word: $p_\theta(y_t|y_1, \ldots, y_{t-1})$;
  - There is lots of data available for this task **- unlabeled data -** just raw text;
  - This is called **unsupervised pretraining** or **self-supervised learning**;
- **Pretraining** can be very effective by serving as parameter initialization; there are three **architectures**:
  - **Decoder-only - GPT -** Generative Pretrained Transformer;
    - Language modeling;
    - Used in sequence generation;
  - **Encoder-only - BERT**;

    * Bidirectional context;
    * Used in classification or sequence tagging;
  - **Encoder-decoder - T5**;
    * Bidirectional context and sequence-to-sequence.

Pretrained Decoders

- For **decoder-only** models, the **pretraining** task is **language modeling**;
- **Fine-tuning** is done by training a classifier on the last hidden state:

$$h1, \ldots, h_L = \text{Decoder}(x_1, \ldots, x_L) y = \text{softmax}(Ah_L + b)$$

- Where A and b are the **classifier parameters**;
- There are two common choices for **fine-tuning**:
  - Freeze the pretrained model an **train only** A **and** b;
  - Or **fine-tune everything**, letting gradients backpropagate through the pretrained model.

Pretrained Encoders

- Encoders get **bidirectional context**, so we can't do language modeling - the **pretraining** task is **masked language modeling**;
- The key idea is to replace a fraction of the input tokens with a special **mask** token, and then **predict** the **original** tokens:

$$h1, \ldots, h_L = \text{Encoder}(x_1, \ldots, x_L) y = \text{softmax}(Ah_i + b)$$

Pretrained Encoder-Decoder

- For **Encoder-decoder** models we can do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted by the decoder:

$$h1, \ldots, h_T = \text{Encoder}(x_1, \ldots, x_T) h_{T+1}, \ldots, h_L 2T = \text{Decoder}(x_{T+1}, \ldots, x_{2T}) y_i = \text{softmax}(Ah_i + b)$$

- The **encoder** portion benefits from **bidirectional context**;
- The **decoder** portion benefits from **unidirectional context**;
- **T5** is an example of this architecture;
  - Uses **span corruption** instead of **masking** - randomly selects a span of text and replaces it with a **sentinel** token;

Adapters and Prompting

- Adapters are an alternative to **fine-tuning**, which is **computationally expensive**;
- Instead of fine-tuning the entire model, a **small set** of task-specific parameters (an **adapter**) is added to each layer of the pretrained model;
- Adapters can be used to adapt a pretrained model to a **new task** without **fine-tuning**.

Few-Shot Learning

- **Few-shot learning** is a **supervised learning** task where we have **very few labeled examples**;
- Powerful models can do this via **prompting**;