Sequence-to-Sequence Networks, RNNs, and Attention Mechanisms

Building Blocks for Advanced NLP Tasks

1. Sequence-to-Sequence (Seq2Seq) Configurations

Imagine translating English sentences to French. You input a whole English sentence and expect an entire French sentence back. This requires different sequence configurations:

a. 1-to-1

- **Description**: Single input, single output.
- **Examples**: Classifying sentiment from a sentence ("I love it!" ⇒ Positive).

b. 1-to-Many

- **Description**: Single input, sequence output.
- **Examples**: Image captioning (one image ⇒ sequence of words).

c. Many-to-1

- **Description**: Sequence input, single output.
- **Examples**: Sentiment analysis (sentence ⇒ sentiment label).

d. Many-to-Many

- **Description**: Sequence input, sequence output.
- **Examples**: Machine translation, speech recognition.
  - **Synchronized**: Input and output aligned timestep-wise (video labeling).
  - **Asynchronous**: Input length ≠ output length (translation).

2. Recurrent Neural Networks (RNN), LSTM, and GRU

2.1 RNNs

Concept: Process sequences step by step, carrying a hidden state. Like reading a sentence word by word, remembering context.

Equations:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$y_t = W_{hy}h_t + b_y$$

- **Problem**: Vanilla RNNs struggle with long-range dependencies due to vanishing/exploding gradients.

2.2 LSTM (Long Short-Term Memory)

Concept: Adds gates to control information flow, remembering long-term dependencies.

Key Gates and Equations:

- **Forget gate**: $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
- **Input gate**: $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
- **Cell candidate**: $\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$
- **Cell state**: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- **Output gate**: $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
- **Hidden state**: $h_t = o_t * \tanh(C_t)$

LSTMs manage long-term context effectively.

2.3 GRU (Gated Recurrent Unit)

Concept: A simpler alternative to LSTM with combined gates.

Equations:

- **Update gate**: $z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)$
- **Reset gate**: $r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$
- **Candidate**: $\tilde{h}_t = \tanh(W_h[r_t * h_{t-1}, x_t] + b_h)$
- **Hidden state**: $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

GRUs offer similar performance with fewer parameters.

3. Attention Mechanisms

Why Attention?

When translating long sentences, Seq2Seq encoders must squash all information into one vector. Attention lets the decoder look at all encoder states, focusing on relevant parts per output token.

3.1 Bahdanau (Additive) Attention

Score computation:

$$e_{t,s} = v^T \tanh(W_h h_s + W_d d_{t-1} + b_{att})$$

Attention weights:

$$\alpha_{t,s} = \frac{\exp(e_{t,s})}{\sum_{s'} \exp(e_{t,s'})}$$

Context vector:

$$c_t = \sum_s \alpha_{t,s} h_s$$

Decoder then uses c_t and its state d_{t-1} to produce output.

3.2 General and Dot-Product Attention

- **Dot-product**: $e_{t,s} = d_{t-1}^T h_s$
- **Scaled dot-product**: $\frac{d_{t-1}^T h_s}{\sqrt{d}}$ to stabilize gradients.

Reflection

Key Takeaways:

- Seq2Seq supports various input/output sequence configurations.
- RNNs handle sequences but struggle with long dependencies; LSTM and GRU mitigate this with gating.
- Attention allows dynamic focus on encoder outputs, greatly improving translation and other tasks.

Common Pitfalls:

- Forgetting to initialize hidden states properly.
- Not masking padded tokens in attention.
- Overlooking computational cost of attention on long sequences.

Real-World Applications:

- Machine translation (Google Translate).
- Text summarization.
- Chatbots and dialogue systems.

This document provides a clear, beginner-friendly overview of Seq2Seq configurations, RNN variants, and attention mechanisms. Use the included equations and explanations directly—no manual formatting needed.