Class Notes: Attention Mechanisms in Neural Networks

1. Motivation

Traditional RNNs compress entire sequences into a single vector, creating an information bottleneck. Attention mechanisms allow models to dynamically focus on different parts of the input, improving performance on tasks like translation, summarization, and question answering.

2. Core Components

Given hidden states $H = [h_1, \ldots, h_T]$:

- **Keys (K)**: Represent what each position contains.
- **Queries (Q)**: Represent what we are looking for.
- **Values (V)**: Contain the actual content to retrieve.

Projected via learnable matrices:

$$Q = HW^Q, \quad K = HW^K, \quad V = HW^V$$

3. Scoring Functions

(a) **Additive (Bahdanau, 2014)**:

$$e_{ij} = v_a^\top \tanh(W_q q_i + W_k k_j)$$

(b) **Multiplicative (Luong, 2015)**:

$$e_{ij} = q_i^\top k_j$$

or $q_i^\top W k_j$.

(c) **Scaled Dot-Product (Transformer, 2017)**:

$$e_{ij} = \frac{q_i^\top k_j}{\sqrt{d_k}}$$

4. Attention Weights and Context Vector

Softmax normalization:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^{T} \exp(e_{ij'})}$$

Context vector:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} v_j$$

5. Multi-Head Computation in Parallel

For h heads, each head has its own projection matrices W_h^Q, W_h^K, W_h^V and reduced dimensionality $d_k = d_{model}/h$.

1. Compute $Q_h = QW_h^Q$, $K_h = KW_h^K$, $V_h = VW_h^V$ for all heads in parallel.
2. For each head, compute attention weights $A_h = \text{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_k}}\right)$.
3. Compute head outputs $O_h = A_h V_h$.
4. Concatenate all O_h along the feature dimension and project with W^O to get the final output.

This parallelization allows the model to focus on different relational patterns simultaneously, such as syntactic dependencies in one head and semantic relations in another.

6. Impact at Decoding Step t

At decoder step t:

- The query q_t comes from the decoder's current hidden state.
- For Bahdanau or Luong attention, each encoder hidden state h_s becomes a key k_s and value v_s.
- The score $e_{t,s}$ measures relevance between q_t and each k_s.
- Softmax over s produces attention weights $\alpha_{t,s}$ that sum to 1.
- The context vector $c_t = \sum_s \alpha_{t,s} v_s$ emphasizes encoder positions s most relevant to generating the next token at step t.
- In multi-head settings, this relevance is computed in multiple subspaces, allowing c_t to integrate multiple perspectives from the encoder.

7. Training Considerations

- Masking for autoregressive tasks.
- Dropout on attention weights.
- Memory complexity $O(T^2)$; use efficient variants for long sequences.

8. Example: Self-Attention Step

For $H \in \mathbb{R}^{4\times 8}$:

1. Project to Q, K, V with $d_k = d_v = 4$.
2. Compute $QK^\top \in \mathbb{R}^{4\times 4}$.
3. Scale, apply softmax row-wise to get weights.
4. Multiply by V to get new token representations.

9. Summary Table

Variant	Q Source	K Source	Scoring	Use Case
Bahdanau	Decoder RNN	Encoder RNN	Additive MLP	Seq2Seq
Luong	Decoder RNN	Encoder RNN	Dot/General	Seq2Seq
Self-Attn	Same sequence	Same sequence	Scaled Dot	Transformers
Multi-Head	Same sequence	Same sequence	Scaled Dot (multi)	Rich relations