Week 5: Attention Is All You Need - Transformers

Discovery-Based Learning Exercises (Instructor Version)

Learning Objectives

By the end of this session, students will:

- Discover why RNNs have fundamental limitations
- Reinvent self-attention from first principles
- Design the multi-head attention mechanism
- Build a complete transformer architecture
- Understand positional encodings

Teaching Note

This session uses discovery-based learning. Let students struggle with problems before revealing solutions. The "aha!" moments are crucial for deep understanding.

---

1 Part 1: The Sequential Processing Problem (10 minutes)

1.1 The Waiting Game

Exercise: Let's process this sentence with an RNN. Mark the dependencies:

"The student who studied hard and completed all assignments passed the exam"

Word	Step	Depends on previous steps?
The	1	No
student	2	Yes (step 1)
who	3	**Answer: Yes (steps 1-2)**
studied	4	**Answer: Yes (steps 1-3)**
hard	5	**Answer: Yes (steps 1-4)**
and	6	**Answer: Yes (steps 1-5)**
completed	7	**Answer: Yes (steps 1-6)**
all	8	**Answer: Yes (steps 1-7)**
assignments	9	**Answer: Yes (steps 1-8)**
passed	10	**Answer: Yes (steps 1-9)**
the	11	**Answer: Yes (steps 1-10)**
exam	12	**Answer: Yes (steps 1-11)**

Q: Can we process "passed" before we process "assignments"? Why or why not in an RNN?

Answer: No! In an RNN, each step depends on the hidden state from the previous step. This creates a sequential bottleneck - we must process words 1-9 before we can process word 10.

1.2 Parallelization Challenge

Think: You have 12 GPUs. How many can you use simultaneously to process this sentence with an RNN?

Answer: **Answer: Only 1! RNN processing is inherently sequential.**

Q: What if you could look at ALL words at once instead of sequentially?

Answer: We could use all 12 GPUs in parallel, achieving 12x speedup! This is the key insight behind transformers.

Teaching Note

Emphasize the parallelization limitation - this is the primary motivation for transformers. Modern GPUs have thousands of cores that RNNs can't fully utilize.

2 Part 2: Inventing Self-Attention (15 minutes)

2.1 The Direct Connection Idea

Exercise: Instead of passing information step-by-step, let's connect every word directly to every other word.

Answer: Students should draw all possible bidirectional arrows: The↔cat, The↔sat, cat↔sat, plus self-connections.

Q: How many connections did you draw? For a sentence with n words, how many connections would we need?

Connections = **Answer: n^2 (including self-connections)**

Teaching Note

This quadratic complexity is important - it's both the strength (all connections) and weakness (computational cost) of transformers.

2.2 Computing Relevance

Exercise: For each word pair, assign a relevance score (0-1):

When processing "sat", how relevant is each word?

Query: "sat"	**Key**	**Relevance Score**
sat looks at →	The	**Answer: 0.1**
sat looks at →	cat	**Answer: 0.7**
sat looks at →	sat	**Answer: 0.2**
Total		1.0

Teaching Note

Students' exact numbers will vary - that's fine! The key insight is that "cat" (the subject) should have high relevance to "sat" (the verb).

2.3 The Three Roles

Q: Each word needs to play three roles. Can you identify them?

1. **QAnswer: uery**: The word asking "who is relevant to me?"
2. **KAnswer: ey**: The word answering "here's what I offer"
3. **VAnswer: alue**: The word providing "here's my actual information"

2.4 The Attention Formula

Exercise: Design the attention mechanism. Fill in the steps:

1. Compute similarity: $Q \times K^T =$ **Answer: attention scores**
2. Scale by: $\frac{1}{\sqrt{d_k}}$ where $d_k =$ **Answer: dimension of keys** (why scale? **Answer: To prevent softmax saturation**)
3. Apply **Answer: softmax** to get weights that sum to 1
4. Multiply by **Answer: V** to get final output

3 Part 3: Why Multiple Heads? (10 minutes)

3.1 Different Types of Relationships

Exercise: Consider the sentence: "The bank by the river bank"

First "bank" should attend to different words for different reasons:

- Syntactic: "bank" is a **Answer: noun**
- Semantic: "bank" means **Answer: financial institution**
- Position: "bank" is the **Answer: second** word

Q: Can a single attention pattern capture all these relationships?

Answer: No! Different relationships require different attention patterns. This motivates multi-head attention.

3.2 Multi-Head Design

Exercise: Design multi-head attention:

1. Number of parallel attentions: **Answer: 8-16** (typically 8)
2. Each head size: $\frac{d_{model}}{n_{heads}}$ where $n_{heads} =$ **Answer: 8-16**
3. How to combine outputs: **Answer: Concatenate then linear projection**

Teaching Note

Explain that each head learns different patterns: one might focus on syntax, another on semantics, another on position, etc.

4 Part 4: The Position Problem (10 minutes)

4.1 Order Blindness

Q: What information is self-attention missing?

Answer: Position/order information! Self-attention is permutation invariant - it produces the same result regardless of word order.

4.2 Encoding Position

Exercise: Evaluate these approaches:

Approach	Pros	Cons
Add position number [1,2,3...]	Simple	**Answer: Can't generalize beyond trainin**
Learn position embeddings	Flexible	**Answer: Fixed maximum length**
Use sin/cos waves	**Answer: Generalizes to any length**	Complex

4.3 Sinusoidal Encoding

Exercise: Fill in the position encoding formula:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Answer: (uses sine for even dimensions)

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Answer: (uses cosine for odd dimensions)

Teaching Note

The sinusoidal encoding creates unique patterns for each position and allows the model to learn relative positions through trigonometric identities.

5 Part 5: Building the Complete Transformer (15 minutes)

5.1 Layer Design

Exercise: Design one transformer layer. What components do we need?

1. **Answer: Multi-head attention** (computes attention)
2. **Answer: Residual connection** (adds shortcut)
3. **Answer: Layer normalization** (normalizes)
4. **Answer: Feed-forward network** (processes each position)
5. **Answer: Residual connection** (another shortcut)
6. **Answer: Layer normalization** (normalizes again)

5.2 The Feed-Forward Network

Q: After attention, why do we need position-wise feed-forward networks?

Answer: Attention combines information from different positions. FFN processes each position independently, adding non-linearity and increasing model capacity.

5.3 Residual Connections

Exercise: Why add the input to the output (residual/skip connections)?

Benefits:

1. Gradient flow: **Answer: Prevents vanishing gradients in deep networks**
2. Information preservation: **Answer: Allows model to preserve original information**
3. Training stability: **Answer: Makes deep networks easier to optimize**

5.4 Stack and Scale

Model	Layers	Parameters
BERT-Base	12	110M
GPT-2	**Answer: 48**	1.5B
GPT-3	**Answer: 96**	175B

6 Part 6: Advantages Analysis (10 minutes)

6.1 Parallelization

Exercise: Compare processing time:

100-word sequence:

- RNN: 100 sequential steps = **Answer: 100** time units
- Transformer: **Answer: 1** parallel step(s) = **Answer: 1** time unit(s)

Speedup factor: **Answer: 100**×

6.2 Long-Range Dependencies

Q: How many steps for word 1 to influence word 100?

- RNN: **Answer: 99** steps (through all intermediate)
- Transformer: **Answer: 1** step(s) (direct connection)

Teaching Note

This direct connection is why transformers handle long-range dependencies better than RNNs, which suffer from gradient vanishing over long sequences.

7 Coding Challenge: Build Your Own Attention

```python
import numpy as np

def self_attention(Q, K, V):
    """
    Q, K, V: matrices of shape (seq_len, d_k)
    """
    d_k = Q.shape[1]

    # Step 1: Compute scores
    scores = np.dot(Q, K.T)  # ANSWER

    # Step 2: Scale
    scores = scores / np.sqrt(d_k)  # ANSWER

    # Step 3: Softmax
    exp_scores = np.exp(scores - np.max(scores, axis=1, keepdims=True))
    weights = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)  # ANSWER

    # Step 4: Apply to values
    output = np.dot(weights, V)  # ANSWER

    return output, weights

# Test it!
seq_len, d_k = 4, 8
Q = np.random.randn(seq_len, d_k)
K = np.random.randn(seq_len, d_k)
V = np.random.randn(seq_len, d_k)

output, weights = self_attention(Q, K, V)
print("Attention weights:", weights)
print("Do weights sum to 1?", np.allclose(weights.sum(axis=1), 1))
```

Teaching Note

Have students implement this - it's the core of transformers! The actual implementation is surprisingly simple once they understand the concept.

8 Reflection Questions

Q: Why is the paper titled "Attention Is All You Need"?

Answer: Because transformers completely replace RNNs/CNNs with attention mechanisms. No recurrence, no convolutions - just attention!

Q: What tasks beyond NLP could benefit from transformers?

Answer: Computer vision (ViT), protein folding (AlphaFold), music generation, time series forecasting, etc. Any sequential or structured data!

Q: What are potential limitations of transformers?

Answer:

- **Quadratic memory/compute complexity with sequence length**
- **Require lots of data and compute for training**
- **Lack of inherent inductive biases (need more data)**
- **Difficulty with very long sequences (though improving)**

Teaching Summary

Key concepts students should take away:

1. **Parallelization**: Transformers process all positions simultaneously
2. **Self-attention**: Direct connections between all positions
3. **Multi-head**: Different heads for different relationships
4. **Position encoding**: Necessary for sequence order
5. **Scaling**: Transformers scale better than any previous architecture

Teaching Note

End with excitement about the transformer revolution! These concepts power ChatGPT, DALL-E, and virtually all modern AI systems. Students have just understood one of the most important innovations in AI history.