Week 5: Attention Is All You Need - Transformers

Discovery-Based Learning Exercises (Student Version)

Learning Objectives

By the end of this session, you 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

---

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	_____
studied	4	_____
hard	5	_____
and	6	_____
completed	7	_____
all	8	_____
assignments	9	_____
passed	10	_____
the	11	_____
exam	12	_____

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

1.2 Parallelization Challenge

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

Answer: _____

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

Discovery

You've identified the key limitation of RNNs: sequential processing prevents parallelization. What if we could process all positions simultaneously?

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.

For the sentence "The cat sat", draw arrows showing which words should connect:

The cat sat

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

Connections = _____

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	___
sat looks at →	cat	___
sat looks at →	sat	___
Total		Should sum to 1.0

Think: How would you compute these scores mathematically?

2.3 The Three Roles

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

1. **Q**_____: The word asking "who is relevant to me?"
2. **K**___: The word answering "here's what I offer"

3. **V**____: The word providing "here's my actual information"

Hint

Think: Query, Key, Value - like a database lookup!

2.4 The Attention Formula

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

1. Compute similarity: $Q \times K^T =$ _____
2. Scale by: $\frac{1}{\sqrt{---}}$ (why scale?)
3. Apply _____ to get weights that sum to 1
4. Multiply by _____ to get final output

Discovery

Congratulations! You just invented self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

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 _____ (noun/verb)
- Semantic: "bank" means _____ (financial/shore)
- Position: "bank" is the _____ word

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

3.2 Multi-Head Design

Think: What if we had multiple attention mechanisms running in parallel, each looking for different patterns?

Exercise: Design multi-head attention:

1. Number of parallel attentions: _____ (typically 8-16)
2. Each head size: $\frac{d_{model}}{\text{_____}}$
3. How to combine outputs: _____

Discovery

Multi-head attention lets the model attend to different types of information simultaneously - syntax, semantics, position, etc.

4 Part 4: The Position Problem (10 minutes)

4.1 Order Blindness

Exercise: Self-attention treats these as identical. Why is this a problem?

1. "The cat chased the mouse"
2. "The mouse chased the cat"

Both have the same words, same attention scores between words...

Q: What information is self-attention missing?

4.2 Encoding Position

Think: How can we tell the model about word positions?

Exercise: Evaluate these approaches:

Approach	Pros	Cons
Add position number [1,2,3...]	Simple	_____
Learn position embeddings	Flexible	_____
Use sin/cos waves	_____	Complex

4.3 Sinusoidal Encoding

Q: Why use sine and cosine for positions?

Properties to achieve:

- Each position should be unique
- Model should understand relative positions
- Should work for any sequence length

Exercise: Fill in the position encoding formula:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\text{-----}}}\right)$$

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

Discovery

Sinusoidal position encodings allow the model to learn relative positions and generalize to longer sequences than seen during training!

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

5.1 Layer Design

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

1. _____ (computes attention)
2. _____ (adds shortcut)
3. _____ (normalizes)
4. _____ (processes each position)
5. _____ (another shortcut)
6. _____ (normalizes again)

5.2 The Feed-Forward Network

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

Think about:

- Attention combines information from different positions
- FFN processes _____

5.3 Residual Connections

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

Benefits:

1. Gradient flow: _____
2. Information preservation: _____
3. Training stability: _____

5.4 Stack and Scale

Q: If one layer is good, what about many layers?

Model	Layers	Parameters
BERT-Base	12	110M
GPT-2	___	1.5B
GPT-3	___	175B

Discovery

Transformers scale exceptionally well - more layers and parameters consistently improve performance!

6 Part 6: Advantages Analysis (10 minutes)

6.1 Parallelization

Exercise: Compare processing time:

100-word sequence:

- RNN: 100 sequential steps = ____ time units
- Transformer: ___ parallel step(s) = ____ time unit(s)

Speedup factor: ____×

6.2 Long-Range Dependencies

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

- RNN: ____ steps (through all intermediate)
- Transformer: ____ step(s) (direct connection)

6.3 Interpretability

Think: With attention weights, what can we visualize?

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 = # YOUR CODE: Q x K^T

    # Step 2: Scale
    scores = scores / # YOUR CODE

    # Step 3: Softmax
    weights = # YOUR CODE: softmax(scores)

    # Step 4: Apply to values
    output = # YOUR CODE: weights x V

    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))
```

8 Reflection Questions

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

Q: What tasks beyond NLP could benefit from transformers?

Q: What are potential limitations of transformers?

Summary

Today you discovered:

1. Self-attention enables parallel processing
2. Multi-head attention captures different relationships
3. Positional encoding provides sequence information
4. Transformers scale better than RNNs

These concepts you "invented" power ChatGPT, BERT, and virtually all modern NLP systems!

Next Week: We'll explore pre-trained language models and the revolution they started!