Transformers and Multi-Head Attention Mechanism

Revolutionizing NLP with Parallel Contextual Understanding

Introduction to Transformers

Imagine reading a paragraph and understanding the relationship between every pair of words simultaneously. Traditional RNNs read sequentially—one word at a time—like reading a letter by letter. **Transformers**, however, look at the entire sentence at once, capturing all word interactions in parallel. This makes them fast and extremely good at modeling language.

Key ideas:

- **Self-attention**: Each word pays attention to every other word to build rich representations.
- **Parallel processing**: Unlike RNNs, Transformers process all words simultaneously, enabling faster training on large datasets.

Transformers consist of stacked layers of **multi-head attention** and **feed-forward networks**, with normalization and residual connections ensuring stability.

Self-Attention Mechanism

Self-attention computes how much each word should focus on others when encoding context. For a sequence of length N, self-attention produces an N×N matrix of attention scores.

Steps:

1. **Query, Key, Value**: For each word, compute three vectors:
    - Query (Q): What am I looking for?
    - Key (K): What information do I have?
    - Value (V): The actual information.
2. **Score computation**: Compute dot-product of Q and K for each word pair, then scale:
$$extscore(Q_i, K_j) = ?racQ_i \cdot K_j\sqrt{d_k}$$
3. **Softmax**: Apply softmax across each row to get attention weights summing to 1.
4. **Weighted sum**: Multiply weights by V vectors and sum to get the updated representation for each word.

Multi-Head Attention

Instead of one attention operation, **multi-head attention** runs multiple attention “heads” in parallel, each learning different aspects of language patterns.

Process:

1. **Project** Q, K, V into h different subspaces.
2. Compute self-attention in each subspace (head) independently.
3. **Concatenate** the output of all heads.
4. **Project back** to the original dimension.

Why multiple heads? Each head can focus on different relationships: syntactic, semantic, positional, etc., enriching the model’s understanding.

Short Example (Pseudo-Code)

```
# Assume input X shape: (batch_size, seq_length, d_model)
# h = number of heads, d_k = d_model / h

# 1. Linear projections for queries, keys, values
Q = linear_q(X)  # -&gt; shape (batch, seq_len, d_model)
K = linear_k(X)
V = linear_v(X)

# 2. Split into h heads
Q_heads = split(Q, h)  # -&gt; (batch, h, seq_len, d_k)
K_heads = split(K, h)
V_heads = split(V, h)

# 3. Scaled dot-product attention per head
def attention(Q, K, V):
    scores = matmul(Q, K.transpose(-1,-2)) / sqrt(d_k)
    weights = softmax(scores, axis=-1)
    return matmul(weights, V)

head_outputs = [attention(Q_heads[i], K_heads[i], V_heads[i]) for i in range(h)]

# 4. Concatenate and final linear projection
multi_head = concat(head_outputs, axis=-1)  # -&gt; (batch, seq_len, d_model)
output = linear_out(multi_head)
```

This pipeline captures multiple perspectives on word relationships and fuses them.

Discussing the Output

After multi-head attention, each word’s representation has combined insights from every other word across multiple heads. This rich embedding then passes through a position-wise feed-forward layer, normalization, and residual connections, resulting in contextualized embeddings ready for classification, translation, or other tasks.

Reflection and Best Practices

Key Takeaways:

- **Transformers** enable parallel context modeling, breaking free from sequential processing limits.
- **Self-attention** captures pairwise interactions among all words.
- **Multi-head attention** provides diverse contextual views.

Common Pitfalls:

- **Computational cost**: Attention scales quadratically with sequence length; consider efficient variants for long texts.
- **Positional encoding**: Since Transformers lack recurrence, add positional information to retain word order.
- **Overfitting**: Large models may overfit; use dropout, regularization.

Real-World Applications:

- **Machine translation**: BERT, GPT, and T5 models.
- **Text summarization**: Generating concise summaries.
- **Question answering**: Extracting precise answers from documents.
- **Language generation**: Chatbots and text completion.

This document provides a concise, beginner-friendly overview of Transformers and multi-head attention. Download the PDF for a fully formatted, ready-to-publish guide.