```
□ Seq2Seq with Attention: Gradient Checklist

□ Обозначения:
- H: Hidden size
- I: Input size = Output_size + Context_size (в декодере)
- C: Context size (2 * encoder hidden size, если BiLSTM)
- A: Attention vector size
- D: Output size (vocab embedding dim)

□ Decoder: Output Layer
• Y_t = p__ @ W_output.T + B_output             # [1×D] = [1×(H+C)] × [(H+C)×D] + [1×D]
• dY_t = Yˆ_t - Y_true_t                         # [1×D]
• dW_output += dY_t.T @ p__                      # [D×1] × [1×(H+C)] = [D×(H+C)]
• dB_output += dY_t                              # [1×D]

□ LayerNorm (на p)
• p_ = (p - mean) / sqrt(var + eps)              # Нормализованное p
• p__ = p_ * γ + β                               # Выход после LayerNorm
• dGamma += dY_t @ W_output * p_                 # [1×(H+C)]
• dBeta  += dY_t @ W_output                      # [1×(H+C)]
• dX = (d * γ - mean_d - xˆ * mean_d_xˆ) / std   # [1×(H+C)]
```

```
□ Decoder LSTM
• dS_t = dX[:H] + от attention                          # [1×H]
• dContext_t = dX[H:] + от attention                    # [1×C]

• dO_t = dS_t * tanh(C_t) * O_t * (1 - O_t)
• dC_t = _dC_t + dS_t * O_t * (1 - tanh(C_t)^2)
• dC˜_t = dC_t * I_t * (1 - C˜_t^2)
• dI_t = dC_t * C˜_t * I_t * (1 - I_t)
• dF_t = dC_t * C_{t-1} * F_t * (1 - F_t)

• dGates = concat(dF, dI, dC˜, dO)                      # [1×4H]
• dW += x_t.T @ dGates                                 # [I×4H]
• dU += h_{t-1}.T @ dGates                             # [H×4H]
• dB += dGates                                         # [1×4H]
```

```
□ Attention (Dot-product)
• α_t = softmax(score(s_{t-1}, h_j))             # [N×1]
• context = Σ α_j * h_j                         # [1×C]

• dContext_t.dot(h_k) → dAlpha_k                # scalar
• dE_tj = Σ_k dAlpha_k * α_k * (δ_jk - α_j)     # softmax grad
• dU_tj = dE_tj * attention_vector              # [1×A]
• dPreact = dU_tj * (1 - tanh²)                 # [1×A]
• dW_e += dPreact.T @ h_j                       # [A×C]
• dW_d += dPreact.T @ s_{t-1}                   # [A×H]
• dV   += u_tj.T * dE_tj                        # [A×1]
```

```
□ Encoder LSTM (forward и backward)
• Тот же процесс, как и в декодере
• dO_j = dH_j * tanh(C_j) * O_j * (1 - O_j)
• dC_j = dH_j * O_j * (1 - tanh²(C_j)) + _dC
• Остальные — аналогично декодеру

• dGates_j = concat(dF, dI, dC˜, dO)            # [1×4H]
• dW_enc += x_j.T @ dGates_j                   # [I×4H]
• dU_enc += h_{j-1}.T @ dGates_j               # [H×4H]
• dB_enc += dGates_j                           # [1×4H]

□ Конец чеклиста
```