**Seq2Seq + Attention (BiLSTM Encoder + Bahdanau Attention + LSTM Decoder)**

## Архитектура

* **Encoder:** двунаправленный LSTM (BiLSTM)
* **Attention:** Bahdanau (additive)
* **Decoder:** однонаправленный LSTM
* **Вход в декодер:** конкатенация эмбеддинга токена и контекстного вектора из attention
* **Нормализация:** LayerNorm после конкатенации, дальше масштабирование и сдвиг:  
  [ \_t = (x\_t) a + b ]

## 1. Forward Pass

1. **Encoder (BiLSTM)**  
   Для каждого входного токена ( x\_j ^{d\_{in}} ): [ \_j = *f(x\_j,* {j-1}), \_j = *b(x\_j,* {j+1}), h\_j = [\_j;,\_j]^{2h}. ]
2. **Attention (Bahdanau)**  
   На шаге ( t ) декодера, имеем скрытое состояние ( s\_{t-1}^h ) и ( {h\_j}): [ e\_{tj} = v^(W\_s,s\_{t-1} + W\_h,h\_j + b\_e), \_{tj} = , c\_t = *j* {tj},h\_j. ]
3. **Decoder Input**  
   Эмбеддинг предыдущего токена (e\_t^d) и контекст (c\_t^{2h}): [ x\_t =

* , \_t = (x\_t),a + b. ]

1. **Decoder (LSTM)** [ (s\_t,m\_t) = (*t, (s*{t-1},m\_{t-1})), ] где (s\_t^h, m\_t^h).
2. **Output & Loss**  
   [ z\_t = W\_{out},s\_t + c\_{out}, \_t = (z\_t), L\_t = -y\_t^(\_t). ]

## 2. Backward Pass (шаг (t))

### 2.1 Output Layer

[ ^{(y)}\_t = \_t - y\_t,  
 = ^{(y)}\_t,s\_t^,  
 = ^{(y)}\_t,  
 {(s)}*t = W*{out},^{(y)}\_t. ]

### 2.2 Decoder LSTM

Обозначения:  
(a\_t = W\_{dec}*t + U*{dec}s\_{t-1} + b\_{dec})  
разбиваем на гейты (ai,af,ao,ag).

Локальные ошибки гейтов: [ ^{(o)}\_t = (^{(s)}\_t (m\_t)) (o\_t(1-o\_t)), ^{(m)}\_t = ^{(s)}\_t o\_t (1-^2(m\_t)), \ ^{(g)}\_t = (^{(m)}\_t i\_t) (1-g\_t^2), ^{(i)}\_t = (^{(m)}\_t g\_t) (i\_t(1-i\_t)), ^{(f)}\_t = (^{(m)}*t m*{t-1}) (f\_t(1-f\_t)). ]

Собираем в один вектор (^{(gates)}\_t^{4h}).

Градиенты параметров: [ = ^{(gates)}\_t,\_t^, = {(gates)}*t,s*{t-1}, = ^{(gates)}\_t. ]

Градиент по нормализованному входу: [ {()}*t = W*{dec},^{(gates)}\_t. ]

### 2.3 LayerNorm

Обозначим (d = d+2h). Пусть (^{()}\_t^d).  
Стандартно: [ ^{()}\_t = ^{()}\_t a, = ^{()}\_t \_t, = ^{()}*t, ] [ ^{(x)}*{t,i} = . ]

Разделяем (x\_t=[e\_t;c\_t]): [ ^{(e)}*t = [^{(x)}\_t]*{1:d}, ^{(c)}*t = [^{(x)}\_t]*{d+1:d+2h}. ]

### 2.4 Attention (Bahdanau)

Из (^{(c)}*t) в (^{()}*{tj}): [ ^{()}*{tj} = (^{(c)}t)^h\_j. ] Через softmax: [ ^{(e)}{tj} = k ^{()}{tk}(*{j=k}-*{tk}). ] Далее (r*{tj}=(u\_{tj}), u\_{tj}=W\_s h\_j+W\_h s\_{t-1}+b\_e): [ ^{(u)}\_{tj} = {(e)}*{tj},v(1-r*{tj}2), ] Параметры: [ = *j ^{(u)}*{tj} h\_j^, = *j {(u)}{tj} s{t-1}, = j ^{(u)}{tj}, = j ^{(e)}{tj} r*{tj}. ]

## 3. Encoder (BiLSTM) Gradients

Общая градиентная составляющая для (h\_j): [ ^{(h\_j)} = ^{(c)}*t* {tj} + W\_s{(u)}\_{tj}. ] Разделяем на директную и обратную ветки: [ ^{(*j)} = [^{(h\_j)}]*{1:h}, ^{(*j)} = [^{(h\_j)}]*{h+1:2h}. ]

### 3.1 Прямая ветка (())

Для каждого (j=1..N): Локальные гейты ({(i,j)},{(f,j)},{(o,j)},{(g,j)}) (аналогично декодеру), после: [ = \_j ^{(,j)} (x\_j{enc}), = *j ^{(,j)} (s*{j-1}{enc,}), = \_j ^{(,j)}. ]

### 3.2 Обратная ветка (())

Аналогично, но (s\_{j+1}^{enc,}) вместо (s\_{j-1}), и суммирование по (j).

## 4. Accumulation

Все (L\_t/) аккумулируются по (t=1..T): [ = \_{t=1}^T . ]