Пример: Генерация кода с использованием LSTM

Входные данные

Формат и структура:

- Последовательности токенов ограниченного языка программирования:

  Примеры:
  ```
  ['FOR', 'i', '0', '10', 'PRINT', 'i']
  ['IF', 'x', '>', '5', 'THEN', 'ASSIGN', 'y', 'x', '+', '1']
  ['ASSIGN', 'x', 'y', '+', 'i']
  ```

- **Словарь токенов** (52 элемента):
  - Ключевые слова (6): FOR, IF, THEN, ELSE, ASSIGN, PRINT
  - Переменные (4): x, y, i, j
  - Операторы (8): +, -, *, /, <, >, ==, !=
  - Константы (5): 0, 1, 2, 5, 10
  - Служебные (3): <PAD>, <EOS>, <UNK>

- **Кодирование** (раздел 0.2.5):

  $$\text{encode("FOR i 0")} \rightarrow [12, 7, 3, 1]$$

  где 1 - токен <EOS>

Выходные данные

Спецификация:

- Для каждой входной последовательности длиной N:
  - Вход модели: первые N-1 токенов

– Ожидаемый выход: N-й токен

- **Пример преобразования**:

```
Вход: ['FOR', 'i', '0'] → [12, 7, 3]
Выход: '10' → 5
```

- **Постобработка**:

1. Применение softmax к выходу модели
2. Фильтрация по синтаксическим правилам (раздел 0.8.2)
3. Декодирование:

$$\text{argmax}(output) \rightarrow \text{INV\_VOCAB}[5] \rightarrow' 10'$$

Построение модели

Архитектура LSTM (раздел 0.5.6):

1. **Слой эмбеддингов**:
   - Размерность: 128
   - Инициализация: Xavier Uniform
   - Преобразует входные индексы в плотные векторы:

$$E \in \mathbb{R}^{V \times 128}, \text{где } V - \text{размер словаря}$$

2. **LSTM слой**:
   - Количество нейронов: 256
   - Инициализация скрытого состояния: нулевая

- Уравнения:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
\tilde{C}_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}
$$

3. **Выходной слой**:

   - Полносвязный слой с softmax-активацией
   - Размерность выхода: размер словаря (52)
   - Функция преобразования:

$$p(y_t) = \text{softmax}(W_{hy}h_t + b_y)$$

Процесс обучения

Детали реализации:

- **Функция потерь** (раздел 0.5.1):

$$\mathcal{L} = -\sum_{i=1}^{V} y_i \log(p_i)$$

  где y_i - one-hot кодировка истинного токена

- **Оптимизация** (раздел 0.5.2):
  - Алгоритм: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
  - Learning rate: 0.001
  - Градиентный клиппинг: 1.0

- **Регуляризация**:
  - Dropout (p=0.2) между слоями
  - L2-регуляризация ($\lambda = 0.01$)

```
Пример вывода:
Epoch 1: Train Loss: 1.852 | Val Loss: 1.643
Epoch 2: Train Loss: 1.401 | Val Loss: 1.382
...
Epoch 10: Train Loss: 0.689 | Val Loss: 0.702
```

Пример работы

```
Вход: "for (int i = 0"
1. Токенизация: ['FOR', 'i', '0']
2. Кодирование: [12, 7, 3]
3. Прогноз модели:
   - Логиты: [1.2 для '10', 0.8 для '5', ...]
   - После softmax: p('10')=0.65, p('5')=0.25, ...
4. Результат:
   TOK: ['FOR', 'i', '0', '10']
   Код: for(int i=0; i<10; i++) {}
```

Тестирование модели

```
>>> for (int i = 0
TOK: ['FOR', 'i', '0']
for(int i=0;i<10;i++){
}

>>> x = y +
TOK: ['ASSIGN', 'x', 'y', '+', '1']
x = y + 1;
```

```
>>> x = y +
TOK: ['ASSIGN', 'x', 'y', '+', 'i']
x = y + i;

>>> if (x <
TOK: ['IF', 'x', '<', 'y']
if(x < y){
}
```