Бейзлайном было это. В целом ничего необычного, создают трансфомер и тренируют.

Мои модификации:

- Сделал батч меньшего размера (64), однако веса обновляются не каждый батч а реже (раз в 2 батча), чем искусственно увеличиваю размер батча но не замедляю работу
- Сделал предсказание на валидации (функция greedy_decode) с помощью батчей, ускорив работу. Начал считать bleu на валидации во время тренировки
- Сделал label smoothing с константой 0.1
- Выбрал оптимизатор adamw вместо adam, выбор обоснован моим успехом в другом задании по текстам
- вернул стандартные параметры (кроме lr) для adamw
- сделал шедулер примерно похожий на тот с лекции (сначала увеличиваем lr несколько эпох, потом уменьшаем все оставшееся время)
    - 2 эпохи обучаю с хорошим lr, потом увеличиваю lr и обучаю оставшиеся 30 эпох с уменьшением lr по валидации (BLEU в ReduceLROnPlateu). Параметры в ReduceLROnPlateu подобраны эмпирически. threshold поставил 0.2 BLEU чтобы не переобучалось.
- Изменил словари
    - Для входов использовал только токены из тестовой выборки
    - Уменьшил размеры словарей, для этого посчитал по полным словарям статистику по процентам и оставил примерно по 8000 токенов, получилось 95% от словаря для выходов и 100% словаря для входов
- Сделал обучаемые positional encoding, инициализировал тригонометрическими функциями
- Немного увеличил размер модели (чтобы не докопались, прикрепляю здесь значения)

```
EMB_SIZE = 768
NHEAD = 12
FFN_HID_DIM = 768
NUM_ENCODER_LAYERS = 5
NUM_DECODER_LAYERS = 5
```

- Beam search, лучшим параметром было k=3
    - Smart beam search, про него в конце

Я сохранил результаты нескольких экспериментов на тесте в списке ниже в хронологическом порядке.
Я либо пишу изменение, либо результат эксперимента:

- label smoothing, early stopping по BLEU, уменьшил размеры словарей до 8000
- 28.84 - последняя модель(без early stopping), 3-beam search
- 27.95 - последняя модель, greedy
- 27.44 - early stopping, greedy
- 28.65 - early stopping, 3-beam search
- 29.40 - последняя модель, smart beam search
- Сделал словарь test only для входов
- 27.97 - последняя модель, greedy
- 29.57 - последняя модель, smart beam search
- AdamW
- 28.34 - early stopping, greedy
- 29.58 - early stopping, smart beam search
- Learning Positional Encoding
- 28.70 - последняя модель, greedy
- 28.74 - early stopping, greedy
- 30.19 - early stopping, smart bs

Теперь про smart beam search. На другом курсе по текстам я узнал про вариации beam search, в частности top-k sampling, nuclear generation и генерация с температурой. Применил их все вместе, подбирая константы. Все замеры проводились на одной конкретной модели на 10% подвыборке валидации. Неважно какая модель, главное какая-то нормальная(ее бейзлайн был 29.38), поскольку перебирать параметры каждый раз при изменении модели невозможно (ОЧЕНЬ долго).

Описание функции beam_search

- В стандартном варианте beam search максимизируется сумма логарифмов условной вероятности (log sum)
- k - параметр beam search (сколько оставлять кандидатов, сколько выбирать кандидатов из каждого кандидата)
- В BLEU есть штраф за краткость, поэтому я постарался приравнять длинные выражения к коротким поделив log sum на длину. (i+1) - длина
- top-k generation это когда рассматриваем k кандидатов с наибольшей вероятностью на каждом шаге (используется мною не только в стандартном beam search)
- size - сколько кандидатов храним для smart beam search на каждом шаге
    - k=3 означает что на самом деле top-k = k и size = k, так как k используется только в стандартном beam search
- nuclear generation - оставляем минимальное число самых вероятных кандидатов, чтобы их сумма вероятностей была хотя бы bd. После чего перевзвешиваем вероятности, поделив каждое на их сумму.
- температурная генерация - делим значения до softmax на температуру, чтобы сильнее выразить/усреднить вероятности
- stats - посчитаем среднюю длину ответа в зависимости от длины запроса
- Длину ответа я ограничил как 1.1 * src_size + 5

Таблица с экспериментами:

- 29.38 - log sum - k=1
- 32.17 - log sum - k=3
- 32.06 - log sum / (i+1)^1.5 - k=3
- 32.3 - log sum / (i+1) - k=3
- 32.37 - log sum / sqrt(i+1) - k=3
- 32.41 - log sum / (i+1)^0.3 - k=3
- 31.33 - log sum / (i+1)^0.3 - k=5
- 32.58 - log sum / (i+1)^0.3, top-k generation - 5, size 10
- 32.35 - log sum / (i+1)^0.3, top-k nuclear k-20, size-5, bd-0.6
- 32.71 - log sum / (i+1)^0.3, top-k nuclear k-5, size-5, bd-0.6
- 32.58 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.5
- 33.06 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.7
- 32.94 - log sum / (i+1)^0.3, top-k nuclear k-10, size-10, bd-0.6
- 32.91 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.6, softmax / 0.9
- 32.62 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.7, softmax / 0.9
- 31.96 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.6, softmax / 0.7
- 32.94 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.7, softmax / 1.2
- 32.67 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.5, softmax / 1.2
- 32.59 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.6, softmax / 1.1
- 33.28 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.6, softmax / 1.2
- 32.6 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.6, softmax / 1.3
- 33.36 - log sum / (i+1)^0.3, top-k nuclear k-10, size-5, bd-0.6
- 32.57 - log sum / (i+1)^0.3, top-k nuclear k-10, size-6, bd-0.6
- 32.95 - log sum / (i+1)^0.3, top-k nuclear k-10, size-4, bd-0.6
- 33.15 - log sum / (i+1)^0.3, top-k nuclear k-7, size-5, bd-0.6
- 32.42 - log sum / (i+1)^0.3, top-k nuclear k-13, size-5, bd-0.6
- 33.17 - log sum / (i+1)^0.3, top-k nuclear k-9, size-5, bd-0.6
- 32.98 - log sum / (i+1)^0.3, top-k nuclear k-11, size-5, bd-0.6
- 33.12 - log sum / (i+1)^0.2, top-k nuclear k-10, size-5, bd-0.6
- 33.44 - log sum / (i+1)^0.4, top-k nuclear k-10, size-5, bd-0.6
- 33.47 - log sum / (i+1)^0.5, top-k nuclear k-10, size-5, bd-0.6
- 33.49 - log sum / (i+1)^0.6, top-k nuclear k-10, size-5, bd-0.6
- 33.53 - log sum / (i+1)^0.8, top-k nuclear k-10, size-5, bd-0.6
- 33.58 - log sum / (i+1)^1, top-k nuclear k-10, size-5, bd-0.6
- 32.98 - log sum / (i+1)^0, top-k nuclear k-10, size-5, bd-0.6
- перешел на минимизацию
- 33.32 - minus log sum * max(1, src/i), top-k nuclear k-10, size-5, bd-0.6
- 33.24 - minus log sum * max(1, src/i)^0.5, top-k nuclear k-10, size-5, bd-0.6
- 33.4 - minus log sum * max(1, src/i)^1.3, top-k nuclear k-10, size-5, bd-0.6
- Здесь я вспомнил что BLEU = prob * min(1, out_len/gold_len) => log(BLUE) = log(prob) + log(out_len/gold_len), поэтому я стал добавлять слагаемое log(out_len/src_len). В итоге не помогло
- 31.32 - minus log sum / (i+1) - log(i/src), top-k nuclear k-10, size-5, bd-0.6
- 32.78 - minus log sum / (i+1) - log(min(1, i/src)), top-k nuclear k-10, size-5, bd-0.6
- 31.68 - minus log sum / (i+1) - log(min(1, i/(src*1.1))), top-k nuclear k-10, size-5, bd-0.6

- 32.24 - minus log sum / (i+1) - log(min(1, i/(src*0.9))), top-k nuclear k-10, size-5, bd-0.6
- 32.81 - minus log sum / (i+1) - log(min(1, i/((src-2)*stats[src-2]))), top-k nuclear k-10, size-5, bd-0.6
- 32.86 - minus log sum / (i+1) - 0.5*log(min(1, i/((src-2)*stats[src-2]))), top-k nuclear k-10, size-5, bd-0.6
- 32.94 - minus log sum / (i+1) + 0.5*log(min(1, i/((src-2)*stats[src-2]))), top-k nuclear k-10, size-5, bd-0.6
- 32.67 - minus log sum / (i+1) - 2*log(min(1, i/((src-2)*stats[src-2]))), top-k nuclear k-10, size-5, bd-0.6
- 31.24 - minus log sum * log(max(1, (src-2)*.../i)), top-k nuclear k-10, size-5, bd-0.6
- 33.32 - minus log sum * max(1, (src-2)*.../i), top-k nuclear k-10, size-5, bd-0.6
- 33.21 - log sum / (i+1)^1.5, top-k nuclear k-10, size-5, bd-0.6
- тут я сдался - log sum / (i+1)^1.2, top-k nuclear k-10, size-5, bd-0.6

В итоге smart beam search на этой выборке улучшил бейзлайн 29.38 до 33.58, что стало большим успехом. Как я уже сказал, параметры наверняка можно улучшить под тестовую выборку, но я делать этого не стал.