Байесовская дистилляция моделей на базе трансформеров

15.12.2023

Аннотация

В данной работе исследовано несколько способов упрощения аппроксимирующих моделей с использованием методов дистилляции глубоких нейронных сетей. А именно исследована дистилляция моделей на базе трансформеров, а также модели RNN. В этом контексте вводятся понятия «учителя» и «ученика», при этом предполагается, что модель ученика имеет меньшее количество параметров, чем модель учителя. Для выбора модели ученика предлагается использовать байесовский подход, опираясь на апостериорное распределение параметров модели учителя. В статье предлагается механизм приведения пространства параметров модели учителя к пространству параметров модели ученика путем удаления слоя Attention из моделей. Также проводится теоретический анализ этого механизма дистилляции. В качестве базовых моделей для дистилляции взяты модель RNN с аддитивным вниманием и модель трансформера seq2seq для задачи перевода текстов с русского на английский. Получены априорные распределения для инициализации модели ученика в случае RNN и Трансформера. Ожидается лучшая сходимость, лучшее качество у дистиллированной модели по сравнению с моделью той же структуры, но с параметрами инициализации из произвольного распределения при одинаковых условиях обучения.

Байесовская дистилляция Трансформеры RNN

1 Введение

На данный момент многие модели глубокого обучения имеют избыточное количество параметров. А избыточное число параметров влечет не интерпретируемость модели и увеличение ее вычислительной сложности. Примером таких моделей являются трансформеры и RNN. Они используются во многих сложных задачах машинного обучения, в том числе, машинного перевода. Однако в какой-то момент эти модели становятся переусложненными для своих задач. Например, модели GPT или BERT. Потому возникает проблема уменьшения размерности этих моделей или дистилляции моделей. Эта проблема изучается в специальном классе задач по состязательным атакам. Дистилляцию моделей можно сформулировать как снижение сложности модели за счет выбора более простой модели с использованием информации о параметров и ответах более сложной фиксированной модели.

Идея дистилляции сформулирована Дж.Е. Хинтоном и В.Н. Вапником. Основной подход в их статьях заключался в использовании ответов моделей учителей для обучения модели ученика. По дистилляции проведено много успешных экспериментов. Так эксперимент по дистилляции на выборке MNIST [**?**] показал хорошую эффективность дистилляции сложной нейросетевой модели в модель меньшей сложности. Также был проведен эксперимент по дистилляции ансамблей моделей в одну модель в приложении к задаче распознания речи. В статье [**?**] был проведен эксперимент по обучению моделей на основе одной сложной модели с большим числом параметров с использованием метода дистилляции на ответах учителя. В статье [**?**] предложен метод передачи селективности нейронов (neuron selectivity transfer), основанный на минимизации специальной функции потерь. Данная функция потерь включает в себя softmax classification loss и функцию максимального среднего отклонения между выходами всех слоев модели учителя и ученика. Тем самым модель учителя передает знания ученику о том как формировать признаки по изображению. По данному методу были проведены успешные вычислительные эксперименты в статьях [**?**] и [**?**], где

в качестве обучающих выборок взяты датасеты CIFAR и ImageNet соответственно.

В статье [?] предлагается другой подход к дистилляции, основанный на байесовском выводе. Авторы предлагают использовать знания о весах модели учителя для инициализации весов модели ученика. Таким образом апостериорное распределение модели учителя присваивается априорному распределению модели ученика. Тем самым возникает задача приведения пространства параметров модели учителя к пространству параметров модели ученика. Авторы предлагают подход, основанный на последовательном приведении пространства параметров модели учителя, то есть последовательном изменении структуры этой модели. В качестве такой модели в данной статье авторы избирают полносвязную нейронную сеть, а структурные параметры, которые в последствии последовательно изменяются – это размер и число скрытых слоев. Также немаловажным является порядок изменения структуры. В данной статье порядок на параметрах задается случайный образом. Вычислительные эксперименты на синтетической выборке и на реальной выборке FashionMnist [?] показали, что дситиллированная модель обучается лучше, чем модель со случайной инициализацией параметров.

Подобно тому, как происходит дистилляция модели за счет удаления слоев в многослойной неросети, можно удалять слои внимания в трансформерах, использующих большое количество слоев внимания. В данной статье исследуется дистилляция моделей посредством удаления слоев внимания на основе байесовского вывода. Авторы предлагают метод последовательного приведения пространства параметров модели учителя посредством удаления блока attention. В качестве модели учителя используются модель RNN с вниманием и трансформер. В первом случае удаляется блок внимания и модель ученика представляет обычный RNN. Во втором случае удаляется слой энкодера трансформера (transformer encoder layer) и модель ученика имеет представляет из себя трансформер с меньшим числом энкодер-слоев. В случае модели трансформера последовательность изменений структуры включает в себя не только удаление слоя внимания, но и удаление слоев полносвязной нейронной сети (feed

forward).

2 Постановка задачи

Прежде чем поставить задачу для моделей трансформеров, сформулируем постановку задачи для дистилляции многослойных нейронных сетей. Данная задача исследована в статье [?].

2.1 Байесовская дистилляция

Пусть задана обучающая выборка :

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^n \quad y_i \in \mathbb{Y}, \tag{2.1}$$

где x_i – признаковые описания объектов, а y_i – целевые переменные. Обозначим $X = [x_1^T, .., x_n^T]$, $Y = [y_1, y_2., .., y_n]^T$. В случае классификации целевое множество $\mathbb{Y} = \{1, 2, 3.., m\}$,а в случае регрессии $\mathbb{Y} = \mathbb{R}$.

Модель учителя в случае полносвязной нейронной сети:

$$f(x) = \sigma \circ U_T \sigma \circ U_{T-1} \circ ...\sigma \circ U_1 x, \tag{2.2}$$

где σ – произвольная функция активации, U_i – матрица линейного преобразования, T – число слоев нейросети.

В выкладках удобно записывать слои модели в виде одного вектора :

$$u = vec(U_T, U_{T-1}, ..., U_1),$$

где vec – операция векторизации последовательности матриц. Индексация полученного вектора – $u_t^{k,l}$, где t – номер матрицы, $k \in \overline{1, ..., n_t}$, $l \in \overline{1, ..., n_{t-1}}$. Таким образом матрицы начиная с 1 по T-ую разворачиваются построчно. Пример представления в таком виде для модели $f(x) = \sigma \circ U_2 \sigma \circ U_1 x$:

$$u = [u_1^{1,1}, ..., u_1^{1,n_0}, ..., u_1^{n_1,1}, ..., u_1^{n_1,n_0}, u_2^{1,1}, ..., u_2^{1,n_1}, ..., u_2^{n_2,1}, ..., u_2^{n_2,n_1}] \tag{2.3}$$

Пусть $n_t \times n_{t-1}$ – размерность матрицы преобразования U_t. Тогда $n_0 = m$ – число признаков у объекта, $n_T = K$ – для классификации и $n_T = 1$ – для регресии. Следовательно число параметров модели:

$$N^{teacher} = \sum_{t=1}^{T} n_t n_{t-1}$$

Итак, пусть нам известно апостериорное распределение параметров u модели учителя f: $p(u|\mathcal{D})$. Тогда задача формулируется следующим образом: используя обучающую выборку $\mathcal{D}$ и апостериорное распределение учителя $p(u|\mathcal{D})$ выбрать модель ученика g из параметрического семейства функций, представляющих многослойную нейронную сеть с более простой структурой, чем у модели учителя:

$$g(x) = \sigma \circ W_L \sigma \circ W_{L-1} \circ ... \sigma \circ W_1 x \quad W_t \in \mathbb{R}^{n_t \times n_{t-1}},$$

где L – количество слоев модели ученика. Выбор параметров модели ученика эквивалентен оптимизации вектора параметров: $w \in \mathbb{R}^{N^{student}}$, где $N^{student}$ – число параметров модели ученика, которое считается аналогично случаю с учителем.

Параметры $w \in \mathbb{R}^{N^{student}}$ оптимизируются путем максимизация обоснованности модели:

$$\mathcal{L}(\mathcal{D}, A) = \log p(\mathcal{D}|A) = \log \int\limits_{w \in \mathbb{R}^{N^{student}}} p(\mathcal{D}|w) p(w|A) dw, \qquad (2.4)$$

где $p(w|A)$ – априорное распределение параметров модели ученика, A – гиперпараметры априорного распределения. Так как зачастую такой интеграл сложно решать аналитически, его решают с помощью вариационного вывода [**?**]. А именно, вводится вариационное распределение параметров ученика $q(w|\mu, \Theta)$, которое впоследствии аппроксимирует апостериорное распределение параметров ученика $p(w|A)$. Причем оптимальные $\bar{w}, \bar{\mu}, \bar{\Theta}$ – находятся из следующей оп-

тимизационной задачи:

$$\bar{w}, \bar{\mu}, \bar{\Theta} = arg \min_{w,\mu,\Theta} D_{KL}(q(w|\mu,\Theta)||p(w|A)) - \sum_{i=1}^{n} \log p(y_i|x_i, w), \tag{2.5}$$

где D_{KL} – расстояние Кульбака-Лейблера между вариационным распределением $q(w|\mu,\Theta)$ и априорным распределением $p(w|A)$. Второе слагаемое – логарифм правдоподобия выборки при текущих параметрах модели.

В статье [?] предлагается учитывать в данной формуле параметры учителя f. А именно автоы предлагают рассматривать априорное распределение параметров ученика $p(w|A)$ как функцию от апостериорного распределения параметров учителя $p(u|\mathcal{D})$. Предполагается нормальное распределение параметров модели учителя : $p(w|\mathcal{D}) \in \mathcal{N}(\mu,\Theta)$. На основе гиперпараметров μ, Θ авторы предлагают задать параметры A априорного распределения $p(w|A)$. В своей статье авторы рассматривают несколько случаев изменения структуры модели многослойной нейронной сети: удаление нейрона, удаление слоя и последовательные преобразования.

2.2 Дистилляция моделей с вниманием

Пусть задан датасет из элементов предложение-перевод:

$$\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}, \tag{2.6}$$

где $x_i = [x_i^1, x_i^2, ..., x_i^{k_i}]$ – последовательность токенов $x_i^k \in \overline{0, 1, 2, ..., V_{rus}}$ из словаря русских слов, $y_j = [y_i^1, y_i^2, ..., y_i^{l_j}]$ – перевод предложения x_i,последовательность токенов $y_i^k \in \overline{0, 1, 2, ..., V_{eng}}$ из словаря английский слов. V_{rus}, V_{eng} – размеры русского и английских словарей.

2.2.1 RNN

Модель RNN (recurrent neural network) [?] представляет из себя два блока – энкодер и декодер. Энкодер представляет из себя сумму ли-

нейных преобразований с функцией активации:

$$h_i = \sigma \circ (Ue_i + Vh_{i-1}) \quad \forall i \in \{1, ..., T\}, \tag{2.7}$$

где σ – функция активациии, $U \in \mathbb{R}^{r \times m}, V \in \mathbb{R}^{r \times r}$ – линейные преобразования, общие для всей последовательности преобразований, $h_i \in \mathbb{R}^r$ – скрытые состояния, $e_i \in \mathbb{R}^m$ – эмбеддинги токенов.

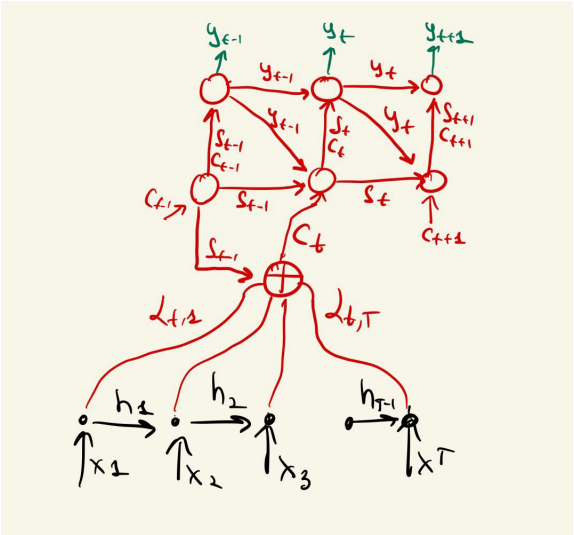

Рис. 1: Модель RNN с вниманием

На вход энкодера подаются эмбеддинги полученные с помощью обучаемых линейных преобразований, применяемых к one-hot представлениям токенов из словаря:

$$e_i = M_1 \overline{x_i} \quad M \in \mathbb{R}^{m_{enc} \times V_{rus}},$$

где M_1 – линейное преобразование, $\overline{x_i} = [0, 0, ..., 1, ..., 0, 0]^T \in \mathbb{R}^{V_{rus}}$ – one-hot вектор с единицей на позиции x_i, а m_{enc} – размерность скрытого слоя энкодера.

Декодер, в зависимости от вида RNN, представляет из себя последовательность линейных и нелинейных преобразований скрытых векторов, полученных от энкодера и от предыдущего состояния декодера. В данной работе использовалась модель LSTM [?]. Блок декодера принимает скрытый вектор y_{t-1} и вектор состояния сети s_{t-1}

в момент $t-1$, а также принимает эмбеддинг c_t, сформированный посредством применения блока attention к выходам энкодера $\{h_i\}_{i=1}^{T}$:

$$f_t = \sigma \circ (W_f[y_{t-1}, c_t] + b_f) \tag{2.8}$$
$$i_t = \sigma \circ (W_i[y_{t-1}, c_t] + b_i) \tag{2.9}$$
$$o_t = \sigma \circ (W_o[y_{t-1}, c_t] + b_o) \tag{2.10}$$
$$\bar{s_t} = \tanh(W_s[y_{t-1}, c_t] + b_s) \tag{2.11}$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \bar{s_t} \tag{2.12}$$
$$y_t = o_t \odot \tanh(s_t), \tag{2.13}$$

где f_t – фильтр забывания , i_t – фильтр обновления , o_t – фильтр выходных данных , $\bar{s_t}$ – кандидат нового состояния, s_t – смесь старого состояния и кандидата нового состояния с учетом фильтров, y_t – выходной сигнал, $\odot$ – покомпонентное умножение векторов, $W_f, W_i, W_o, W_s \in \mathbb{R}^{m_{dec} \times (m_{dec}+m_{enc})}$ – обучаемые линейные преобразования, $b_f, b_i, b_o, b_s \in \mathbb{R}^{m_{dec}}$ – обучаемые смещения, m_{dec} – размерность скрытого состояния декодера.

Механизм attention [?] по выходным эмбедингам энкодера $\{h_t\}_{t=1}^{T}$ формирует результирующий вектор c_t. В данной статье используется bahdanau attention [?], использующий в качестве alignment функции $a(h, s) = U_0 \tanh(U_1 h + U_2 s)$:

$$a_{t,j} = a(h_j, s_{t-1}) = U_0 \tanh(U_1 h_j + U_2 s_{t-1}) \tag{2.14}$$
$$\alpha_{t,j} = soft \max_{t \in 1,2,\ldots,T} a_{t,j} \tag{2.15}$$
$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j, \tag{2.16}$$

где $U_0 \in \mathbb{R}^{1 \times l}, U_1 \in \mathbb{R}^{l \times m_{enc}}, U_2 \in \mathbb{R}^{l \times m_{dec}}$ – линейные преобразования. l – размерность скрытого слоя alignment, $\alpha_{t,j} \in \mathbb{R}$

В случае модели без блока внимания результирующий эмбеддинг c_t от энкодера представляет из себя арифметическое среднее от всех

скрытых векторов энкодера $\{h_t\}_{t=1}^{T}$:

$$c_t = \frac{1}{T}\sum_{t=1}^{T} h_t \tag{2.17}$$

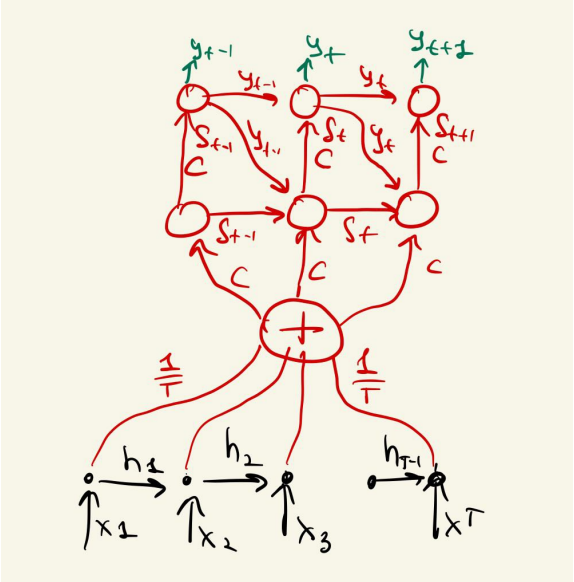

Рис. 2: Модель RNN без блока внимания

К выходным векторам декодера $\{y_t\}_{t=1}^{L}$ применяются обучаемые эмбеддинги, отображающие эти вектора в пространство размерности длины словаря V_{eng}. Получаемые вектора интерпретируются, как вероятности для следующего токена.

$$p_t = M_2 y_t \quad M_2 \in \mathbb{R}^{V_{eng} \times m_{dec}}, \tag{2.18}$$

где M_2 – линейное преобразование, $p_t = [p_t^1, p_t^2, ..., p_t^{V_{eng}}] \in \mathbb{R}^{V_{eng}}$ – вероятности токена на позиции t.

Предсказание токенов может происходить в разных режимах: greedy и sampling. На момент инференса, уже обученная модель в первом режиме итоговый токен предсказывает, как $y_t = arg \max_{j \in \{1,2,...,V_{eng}\}} p_t$, во втором режиме модель максимизирует правдоподобие сразу нескольких следующих токенов. Например, в случае глубины равной двум

модель перебирает возможные предсказания текущего токена и предсказывает по ним вероятности последующих токенов, а по полученным вероятностям считает правдоподобие последовательности и выбирает максимальное.

В режиме обучения модель предсказывает на один токен вперед, используя в качестве входа смесь ранее предсказанных токенов по тому же принципу и токенов из истинного перевода в заранее заданной пропорции. Таким образом ускоряется обучение модели.

Батч из переводов $\{out^i_{t,b}\}^{L\times batch\times V_{eng}}_{t,b,i}$ и истинные переводы $\{trg_{t,b}\}^{L\times batch}_{t,b}$,где $out_{t,b}$ – вектор разменности $\mathbb{R}^{V_{eng}}$, интерпретируемый как вектор вероятностей t-го токена, $batch$ – размер батча, а L – максимальная длина предложения в батче, сворачиваются в списки из токенов размерностей $\vec{x} \in \mathbb{R}^{(L\cdot batch)\times V_{eng}}$ и $\vec{y} \in \mathbb{R}^{L\cdot batch}$ соответственно, для подсчета кроссэнтропии. В качестве функции ошибки используется кроссэнтропия между списками из предсказаний и истинных переводов предложений. Кроссэнтропия считается следующим образом:

$$\mathcal{L}_{crossentropy} = \frac{1}{N}\sum_{n=1}^{N} l_n \tag{2.19}$$

$$l_n = -\log \frac{\exp(x_{n,y_n})}{\sum\limits_{c=1}^{V_{eng}} \exp(x_{n,c})}, \tag{2.20}$$

где $N = L\cdot batch$, x_{n,y_n} – интерпретируется как предсказание моделью вероятности токена y_n из истинного перевода находиться на позиции n. Чем больше эта верятность относительно вероятностей $x_{n,c}$ других токенов c , тем меньше слагаемое l_n.

Итак, теперь перейдем к постановке задачи. Модель учителя имеет блок внимания, модель ученика имеет такую же структуру, как модель учителя, но в нем отстутствует этот блок. Необходимо инициализировать веса модели ученика так, что в процессе обучения, описанного ранее, он давал значительно лучшие результаты в сравнении с моделью этой же структуры, но произвольной инициализации параметров. В качестве критерия качества результатов используется

метрика BLEU, которая считается на отложенной выборке по предсказаниям переводов.

2.2.2 Transformer

Модель трансформера [?] аналогично RNN включает в себя энкодер и декодер. Но они включают в себя более сложные модули. Основная идея моделей трансформеров в использовании self-attention блока. В данной работе в качестве self-attention использовался модуль multi-head-attention (MHA).

При подсчете attention используется подход key-query-value: key – интерпретируется как ключевой термин или некоторая смысловая сущность, query – интерпретируется как запрос или обращение внимания к этому термину, value – величина, которую необходимо вернуть при запросе к данному ключу.

Входные последовательности эмбедингов формируются как в RNN, но дополнительно к ним применятся позиционное кодирование (positional-encoding), который вносит в эмбеддинг информацию о взаимном расположении между токенами в одном предложении. Это необходимо в силу симметричности последующих операций, применяющихся в энкодере и декодере к эмбеддингам. Забегая вперед, скажем, что при отсутствии позиционного кодирования выходные векторы энкодера неизменны при перестановке токенов во входном предложении. Это означает, что модель не сможет выносить информацию заложенную в конкретных расстановках токенов, что критично в машинном переводе. Имеется ввиду, что если модель одинаково переводит предложения: «Автомобиль пропустил самокат» и «Самокат пропустил автомобиль»,– это будет означать, что модель не поняла, кто кого пропустил. По взаимному расположению слов и контексту можно определить часть речи данного слова. Кратко говоря, для модели не будет существовать такого понятия, как часть речи.

В данной работе реализован positional-encoding из статьи [?]. По-

зиционный энкодинг одного токена выглядит следующим образом:

$$PE(pos, 2i) = sin(\frac{pos}{10000^{2i/d_{model}}})$$
$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{2i/d_{model}}}),$$

где pos – позиция токена в предложении, а $i \in \{0, 1, ..., d_{model} - 1\}$ – индексация полученного эмбеддинга.

По входной последовательности эмбедингов $\{h_t\}_{t=1}^{T} \quad h_t \in \mathbb{R}^{d_{model}}$, полученной после позиционного кодирования, формируются ключи, запросы и величины с помощью обучаемых линейных преобразований $\{W_r^q, W_r^k, W_r^v\}_{r=1}^{N_{heads}}$:

$$Q_i = W_r^q \cdot h_i \tag{2.21}$$
$$K_i = W_r^k \cdot h_i \tag{2.22}$$
$$V_i = W_r^v \cdot h_i, \tag{2.23}$$

где $W_r^q, W_r^k \in \mathbb{R}^{d_{model} \times d_k}, W_r^v \in \mathbb{R}^{d_{model} \times d_v}$,N_{head} – количество отдельных блоков attention.

К полученным ключам, запросам и величинам применяется attention следующего вида:

$$head_i = Att(Q, K, V) \tag{2.24}$$
$$Att(Q, K, V) = soft\max(\{\frac{QK^T}{\sqrt{d_k}}\})V, \tag{2.25}$$

где $Q, K \in \mathbb{R}^{T \times d_k}, V \in \mathbb{R}^{T \times d_v}, head_i \in \mathbb{R}^{T \times d_v}$, а softmax применяется по размерности T – аналогично подсчету attention в RNN.

Далее к полученным головам применяется конкатенация по размерности d_v и дополнительное линейное преобразование W_o:

$$H = concat(head_1, head_2, ..., head_{N_{heads}}) \cdot W_o, \tag{2.26}$$

где $W_o \in \mathbb{R}^{N_{heads} \cdot d_v \times d_{model}}$, а $H \in \mathbb{R}^{T \times d_{model}}$. То есть на выходе получаются T преобразованных эмбеддингов.

Следующим шагом является применение еще нескольких линейных преобразований к полученным эмбеддингам – так называемые feed forward networks:

$$H'_t = \sigma \circ (H_t \cdot W_1 + b_1)W_2 + b_2, \tag{2.27}$$

где σ – функция активации ReLU, $H_t \in \mathbb{R}^{d_{model}}, W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}, W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}, b_1 \in \mathbb{R}^{d_{ff}}, b_2 \in \mathbb{R}^{d_{model}}$, d_{ff} – внутренняя размерность, в результате получается итоговая последовательность эмбеддингов $H' \in \mathbb{R}^{T \times d_{model}}$.

К выходам self-attention, а также выходам feed-forward применяется dropout и batch-normalization, а также residual connection – то есть к выходным векторам прибавляются входные. В совокупности данная последовательность преобразований представляет из себя один слой энкодера трансформера $E_i : \mathbb{R}^{T \times d_{model}} \to \mathbb{R}^{T \times d_{model}}$ и реализована в torch.nn [**?**]. А сам энкодер – это последовательность нескольких преобразований $\{E_i\}_{i=1}^{N_{enc}}$:

$$H'' = Enc(H) \quad H \in \mathbb{R}^{T \times d_{model}} \tag{2.28}$$

$$Enc = E_1 \circ E_2 \circ, ..., E_{N_{enc}} \tag{2.29}$$

$$E_i = I + norm \circ FFN \circ (I + norm \circ SA) \tag{2.30}$$

$$SA = dropout \circ MHA \tag{2.31}$$

$$FFN(H_t) = dropout \circ (\sigma \circ (H_t \cdot W_1 + b_1)W_2 + b_2), \tag{2.32}$$

где I – тождественное преобразование ,N_{enc} – число слоев энкодера, $norm$ – батч нормализация,SA – self-attention модуль, E_i – слой энкодера, FFN – модуль feed-forward, MHA – модуль multi-head-attention,$H'' \in \mathbb{R}^{T \times d_{model}}$ – результат применения энкодера к входной последовательности эмбеддингов.

Декодер трансформера имеет похожую структуру, но имеет следующие отличия: в multi-head-attention ключи и величины формируются из выходных векторов энкодера, а запросы – это преобразованные посредством multi-head-attention примененного к целевым эмбеддингам, полученным из токенов истинного перевода.

По похожей же причине, что и в энкодер, в декодер важно передавать эмбеддинги после позиционного кодирования. Без позиционно-

го кодирования, модель будет считать одинаково правдоподобными разные переводы с различием только в перестановках токенов.

Самый первый self-attention в слое декодера включает в себя целевую маску, которая перемножается с ключом и величиной, тем самым зануляя веса в итоговой матрице внимания размерности $T \times T$. Так как далее следует softmax, то коэффициенты становятся ненулевыми, однако – НЕПОНЯТНО ПОЧЕМУ ОНИ НЕНУЛЕВЫЕ , ТОГДА ВЕДЬ БУДЕТ ЗАГЛЯДЫВАНИЕ В БУДУЩЕЕ. ЗДЕСЬ НУЖНО ОБЪЯСНИТЬ ПОЧЕМУ В ВЫХОДНОМ ТЕНЗОРЕ ДЕКОДЕРА i ЫЙ ЭМБЕДДИНГ НЕ ЗАВИСИТ ОТ i + 1 ЭМБЕДДИНГА ВХОДНОГО ТЕНЗОРА, БЛАГОДАРЯ ЧЕМУ ВЫХОДНОЙ ТЕНЗОР ИНТЕРПРЕТИРУЕМЫЙ КАК ТЕКСТ ПЕРЕВОДА В ПРЕДСТАВЛЕНИИ ЭМБЕДДИНГОВ СГЕНЕРИРОВАН КОРРЕКТНО, ТО ЕСТЬ НЕ БЫЛО ЗАГЛЯДЫВАНИЯ В ИСТИННЫЙ ПЕРЕВОД TARGET.

К перобразованным эмбеддингам $H_{dec} \in \mathbb{R}^{L \times d_{model}}$, как и в энкодере, применяются residual connection и нормализация, после чего они попадают в блок multi-head-attention, но только в качестве запросов. В качестве ключей и величин подаются выходы энкодера $H''_{enc} \in \mathbb{R}^{T \times d_{model}}$:

$$Q = H_{dec} \cdot W_r^q \tag{2.33}$$

$$K = H''_{enc} \cdot W_r^k \tag{2.34}$$

$$V = H''_{enc} \cdot W_r^v, \tag{2.35}$$

где $W_r^q, W_r^k \in \mathbb{R}^{d_{model} \times d_k}, W_r^v \in \mathbb{R}^{d_{model} \times d_v}, K \in \mathbb{R}^{T \times d_k}, Q \in \mathbb{R}^{L \times d_k}, V \in \mathbb{R}^{T \times d_v}$, T – длина последовательности на русском языке, L – длина текущей последовательности перевода на английском языке.

Тогда результат применения attention:

$$QK^T \in \mathbb{R}^{L \times T}$$

$$head_i = softmax(\{\frac{QK^T}{\sqrt{d_k}}\})V$$

$$head_i \in \mathbb{R}^{L \times d_v},$$

где $V \in \mathbb{R}^{T \times d_v}$.

Полученные выходные эмбеддинги:

$$H'_{dec} = concat(\{head_i\}_{i=1}^{N_{heads}}) \cdot W_o \in \mathbb{R}^{N_{heads} \cdot d_v \times d_{model}}, \tag{2.36}$$

где N_{heads} - - гиперпараметр MHA в декодере. Далее они попадают в стандартный feed forward, где преобразуются в $H''_{dec} \in \mathbb{R}^{L \times d_{model}}$.

Итак, рассказанные выше модули собираются в единый слой декодера. В трансформерах обычно используют несколько таких слоев. Удобно оказывается то, что входные и выходные тензоры имеют одинаковые размерности. Это и позволяет использовать много слоев в энкодере и декодере.

Заключительной частью трансформера является линейный слой, применяемый к выходным эмбедингам декодера H''_{dec}, которые уже можно интерпретировать, как перевод текста длины L в представлении эмбеддингов:

$$logits = H''_{dec} \cdot W_{generator}, \tag{2.37}$$

где $W_{generator} \in \mathbb{R}^{d_{model} \times V_{eng}}$ – линейное преобразование в пространство токенов английского языка, $logits \in \mathbb{R}^{L \times V_{eng}}$ – вектора, интерпретируемые как распределения вероятностей токенов.

Операции в трансформерах хорошо параллелятся и можно подавать на вход декодера и энкодера батчи эмбеддингов предложений, получая на выходе батчи правдоподобий переводов предложений. Однако здесь тоже есть подводные камни – размеры предложений в одном батче могут быть разными. Поэтому приходится заполнять предложения паддинг токенами в конце, чтобы составить корректный тензор батча, который будет корректно преобразовываться в трансформере. Это порождает другую проблему – лишние паддинг токены будут замедлять обучение. решение этой проблемы – паддинг маски, которые подаются в энкодер и декодер. В нужный момент эти булевы маски покомпонентно умножаются на тензор, зануляя часть его компонент. Тогда все операции в трансформере с некоторым предложением из батча становятся эквивалентными операциям

с батчем единичной размерности, включающим это предложение без паддинг токенов.

Итак, результирующие логиты вместе с истинными переводами подаются в кроссэнтропийный лосс и на этом заканчивается пайплайн трансормера.

Таким образом формулируется несколько постановок задач дистилляции для трансформеров. Пусть модель учителя имеет стандартную структуру трансформера. описанной ранее, модель ученика имеет такую же структуру, но может отличаться от структуры учителя следующим образом:

1. Один выбранный слой энкодера или декодера отсутствует
2. В одном выбранном слое энкодера или денкодера число голов меньше на единицу
3. Совокупность предыдущих пунктов

Тогда, при заданной структуре ученика по правилам выше необходимо инициализировать веса модели ученика так, что в процессе обучения, он давал лучшие метрики качества, чем мо дель той же структуры, но с произвольной инициализацией весов и с тем же пайплайном обучения. В качестве критерия качества, как и в случае RNN, интересна метрика BLEU, которая считается на отложенной выборке по предсказаниям переводов.

3 Дистилляция

Параметры модели учителя будем описывать вектором u. Воспользуемся предположением о том, что апостериорное распределение параметров учителя является нормальным:

$$p(u|\mathcal{D}) = \mathcal{N}(\mu, \Theta), \tag{3.1}$$

где μ, Θ – гиперпараметры этого распределения. Используя эти гиперпараметры необходимо задать гиперпараметры A априорного рас-

пределения параметров модели ученика $p(w|A)$. Далее будет предполагаться, что априорное распределение параметров ученика – нормальное.

В качестве априорного распределения на параметры модели ученика, можно использовать апостериорное распределение на подмножество параметров модели учителя при выполнениии дополнительных условий на модель учителя, которые будут сформулированы позже. То есть, получая апостериорное распределение на подмножество параметров модели учителя, его можно использовать, как априорное распределение модели ученика.

3.1 RNN

Возьмем в качестве модели учителя RNN с вниманием, а в качестве модели ученика – RNN без внимания. Структуры данных моделей были рассмотрены ранее.

Пусть параметры u модели учителя собраны в вектор так, что параметры блока attention находятся в конце вектора:

$$u = [\nu, U_0, U_1, U_2], \tag{3.2}$$

где ν – вектор параметров модели учителя, сопадающий по размерности с вектором параметров модели ученика, U_0, U_1, U_2 – параметры bahdanau attention, развернутые в вектора.

Прежде, чем перейти к теореме, сформулируем следующую лемму – одно из свойств нормального распределения:

Лемма 1. *Пусть* $[\xi_1, \xi_2]^T = \mathcal{N}(m, R)$, *тогда*

$$p(\xi_1|\xi_2 = x) = \mathcal{N}(m_1 + R_{\xi_1,\xi_2}R_{\xi_2}^{-1}(x - m_1), R_{\xi_1} - R_{\xi_1,\xi_2}R_{\xi_2}^{-1}R_{\xi_2,\xi_1}), \tag{3.3}$$

где m_1, m_2 *– подвектора матожидания* m, *соответствующие подвекторам* ξ_1, ξ_2, R_{ξ_1,ξ_2} *– ковариация между векторами* ξ_1 *и* ξ_2, R_{ξ_1} *– ковариация вектора* ξ_1 *с самим собой.*

Теорема 2. *Пусть выполняются следующие условия :*

1. *Апостериорное распределение модели $p(u|\mathcal{D})$ учителя имеет нормальное распределение (3.1)*

Тогда при удалении блока attention – апостериорное распределение параметров описывается нормальным распределением

$$p(\nu|D) = \mathcal{N}(\mu_\nu + \Theta_{\nu,v}\Theta_v^{-1}(0 - \mu_\nu), \Theta_\nu - \Theta_{\nu,v}\Theta_v^{-1}\Theta_{v,\nu}), \tag{3.4}$$

где $v = [U_0, U_1, U_2]$ – векторизованные параметры bahdanau attention, ν – остальные векторизованные параметры.

Доказательство. Заметим, что при зануленении всех параметров модуля внимания $U_0, U_1, U_2 = 0$, получаемые коэффициенты будут одинаковы и равны $\frac{1}{T}$, так как $soft\max([0, 0, ..., 0]) = [\frac{1}{T}, \frac{1}{T}, ..., \frac{1}{T}]$. Но при таких $v = [U_0, U_1, U_2] = 0$ модель учителя станет эквивалентной модели ученика, ведь результирующий вектор будет получаться, как среднее от выходных векторов энкодера. При зафиксированных нулевых параметрах модели учителя. Используя лемму, сформулирвованную ранее, можно получить искомое распределение подвектора ν вектора $u = [\nu, v] = [\nu, U_0, U_1, U_2]$. □

Также можно заметить, что для эквивалентности модели учителя и модели ученика достаточно занулить только вектор U_0 или только пару векторов U_1, U_2. Таким образом можно сформулировать следующую теорему:

Теорема 3. *Пусть выполняются следующие условия :*

1. *Апостериорное распределение модели $p(u|\mathcal{D})$ учителя имеет нормальное распределение (3.1)*

Тогда при удалении параметров attention U_1, U_2 и зануленении параметров U_0 – апостериорное распределение параметров описывается

нормальным распределением

$$m = \mu_z + \Theta_{z,U_0}\Theta_{U_0}^{-1}(0 - \mu_z) \tag{3.5}$$

$$R = \Theta_z - \Theta_{z,U_0}\Theta_{U_0}^{-1}\Theta_{U_0,z} \tag{3.6}$$

$$p(\nu|\mathcal{D}) = \mathcal{N}(m_\nu, R_{\nu,\nu}), \tag{3.7}$$

где $z = [\nu, U_1, U_2]$ *– оставшиеся* μ *и удаляемые параметры внимания* U_1, U_2, R *– ковариационная матрица между векторами* z, U_0 , U_0 *– зануляемые параметры внимания,* $m_\nu, R_{\nu,\nu}$ *– подвектор* m *и подматрица* R*, соответствующие подвектору* ν *в векторе* z*.*

Доказательство. Исходя из леммы запишем апостериорное распределение для вектора $z = [\nu, U_1, U_2]$ при условии $U_0 = 0$:

$$p(z|\mathcal{D}, U_0 = 0) = \mathcal{N}(m, R), \tag{3.8}$$

где m, R – гиперпараметры, записанные в условии теоремы.

Апостериорное распределение для оставшихся параметров ν:

$$p(\nu|\mathcal{D}) = \int\limits_{U_1,U_2} p([\nu, U_1, U_2]|\mathcal{D}, U_0 = 0)dU_1dU_2 \tag{3.9}$$

Заметим, что при зануленении U_0 предсказания модели не зависят от U_1, U_2, поэтому этими параметрами можно пренебречь.

Распределение $p(\nu|\mathcal{D})$ – находится с помощью маргинализации распределения (3.8) по параметрам U_1, U_2:

$$p(\nu|\mathcal{D}) = \mathcal{N}(m_\nu, R_{\nu,\nu})$$

□

Как следствие, сформулируем следующую теорему:

Теорема 4. *Пусть выполняются следующие условия :*

1. *Апостериорное распределение модели* $p(u|\mathcal{D})$ *учителя имеет нормальное распределение (3.1)*

Тогда при удалении параметров attention U_0 и занулении параметров U_1, U_2 – апостериорное распределение параметров описывается нормальным распределением

$$m = \mu_z + \Theta_{z,v}\Theta_v^{-1}(0 - \mu_z) \tag{3.10}$$

$$R = \Theta_z - \Theta_{z,v}\Theta_v^{-1}\Theta_{v,z} \tag{3.11}$$

$$p(\nu|\mathcal{D}) = \mathcal{N}(m_\nu, R_{\nu,\nu}), \tag{3.12}$$

где $z = [\nu, U_0]$ – оставшиеся μ и удаляемые параметры внимания U_0, R – ковариационная матрица между векторами z, v , $v = [U_1, U_2]$ – зануляемые параметры внимания, $m_\nu, R_{\nu,\nu}$ – подвектор m и подматрица R, соответствующие подвектору ν в векторе z.

3.2 Трансформер

Пусть модель учителя – стандартный трансформер, описанный в постановке задачи. Модель ученика – преобразованная модель учителя, по одному из следующих правил:

1. Один выбранный слой энкодера или декодера удаляется (если их изначально больше одного)

2. В одном выбранном слое энкодера или денкодера число голов меньше на единицу (если голов изначально болше одной)

3. Совокупность предыдущих пунктов

Пусть все параметры трансформера представлены в виде вектора u, в котором первый подвектор – остающиеся параметры, далее идет подвектор удаляемых параметров, а потом зануляемые:

$$u = [\nu, v_1, v_2], \tag{3.13}$$

где v_1 – удаляемые параметры, v_2 – зануляемые параметры, ν – оставшиеся параметры.

Во первых, заметим, что во всех модулях слоев энкодера и декодера используются residual connection. Это позволяет найти параметры, при которых модель ученика и модель учителя будут эквивалентны. Если занулить все веса в FFN одного из слоя энкодера или декодера, то весь слой будет делать тождественное преобразование, из чего будет следовать, что модель учителя будет эквивалентна модели ученика.

Также можно уменьшать число голов в multi-head-attention. Для начала нужно найти такие параметры у модели учителя, что она станет эквивалентна модели ученика. Во первых, можно занулять часть параметров линейного преобразования, применяемого к головам. Во вторых, можно занулять параметры одного из трех линейных преобразований, соответствующих key, query и value.

Сформулируем теорему в более общем виде, включающем все случаи, описанные выше:

Теорема 5. *Пусть выполняются следующие условия :*

1. *Апостериорное распределение модели учителя имеет нормальное распределение* $p(u|\mathcal{D}) = \mathcal{N}(\mu, \Theta)$

Тогда при удалении параметров v_1 *и занулении вектора параметров* v_2 *– апостериорное распределение параметров описывается нормальным распределением*

$$m = \mu_z + \Theta_{z,v_2}\Theta_{v_2}^{-1}(0 - \mu_z)$$
$$R = \Theta_z - \Theta_{z,v_2}\Theta_{v_2}^{-1}\Theta_{v_2,z}$$
$$p(\nu|\mathcal{D}) = \mathcal{N}(m_\nu, R_{\nu,\nu}),$$

где $z = [\nu, v_1]$ *– оставшиеся* μ *и удаляемые параметры* v_1, R *– ковариационная матрица между векторами* z, v_2 , v_2 *– зануляемые параметры,* $m_\nu, R_{\nu,\nu}$ *– подвектор* m *и подматрица* R, *соответствующие подвектору* ν *в векторе* z.

Доказательство. Такое же, как у теорем про RNN □

4 Сбор данных

В качестве датасета для переводов используется выборка из 10 тысяч пар предложений переводов с русского на английский язык, взятая из курсов по NLP Нейчева [?].

5 Вычислительный эксперимент

В качестве модели учителя взята модель RNN с вниманием со следующими гиперпараметрами:

- Размер эмбеддингов в декодере и энкодере – 128
- Размерность скрытых слоев – 128
- Размерность линейных преобразований в модуле внимания – 128

В качестве модели ученика – взята модель с теми же гиперпараметрами, но с усреднением по выходным слоям энкодера и с отстствующим блоком внимания.

6 Заключение