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 работе используются понятия «учителя» и «ученика» и исследуются методы для получения априорного распределения параметров для модели ученика, имеющего меньшее число параметров, чем у модели учителя, 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 за собой 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 [1] или BERT [2]. Потому возникает проблема уменьшения размерности этих моделей или дистилляции моделей. Дистилляцию моделей можно сформулировать как снижение сложности модели ученика за счет использования информации о параметрах более сложной модели и ее ответах на обучающей выборке.

Впервые была предложена идея дистилляции Дж.Е. Хинтоном и В.Н. Вапником. В их статьях модель ученика использовала информацию об ответах модели учителя в качестве обучающей выборки. Таким образом, в статье [3] были проведены успешные эксперименты по обучению моделей с использованием метода дистилляции на ответах учителя. На выборке MNIST [4] была показана хорошая эффективность дистилляции сложной нейросетевой модели в модель меньшей сложности.

В статье [5] предложен метод дистилляции – метод передачи нейронной селективности (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 [6] и [7], где в качестве обучающих выборок были взяты датасеты CIFAR и ImageNet соответственно.

В статье [8] предлагается другой подход к дистилляции, основанный на байесовском выводе. Авторы предлагают использовать зна-

ния 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 позже 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 [9] показали, что дистиллированная модель обучается лучше, чем модель со случайной инициализацией параметров.

Подобно тому, как происходит дистилляция модели за счет удаления слоев в многослойной неронной сети, можно удалять слои энкодеров и декодеров в трансформерах, содержащих в себе модули 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 моделей с вниманием рассматриваются RNN с блоком внимания и модель трансформера. В первом случае удаляется блок внимания и модель ученика представляет обычный RNN. Во втором случае удаляется слой энкодера либо декодера трансформера и модель ученика представляет из себя трансформер с меньшим числом соответствующих 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), которые можно удалять методами, описанными в статье [8].

2 Постановка задачи

Прежде чем поставить задачу для моделей трансформеров, сформулируем постановку задачи для дистилляции многослойных нейронных сетей. Данная задача исследована в статье [8].

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 регрессии. 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 [10]. 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 [8] предлагается учитывать в данной формуле параметры учителя f. А именно авторы 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.

В постановке задачи рассмотрим частный случай удаления слоя декодера.

Модель учителя представима в виде: $\mathrm{f}\left(\mathbf{x}, \mathbf{y}\right) = \mathbf{G} \circ \mathbf{D}_T \circ \cdots \circ \mathbf{D}_2 \circ \mathbf{D}_1 \circ \mathbf{E}[\mathbf{x}, \mathbf{y}]$.

Модель ученика:

$$\mathbf{g}\big(\mathbf{x}, \mathbf{y}\big) = \mathbf{G} \circ \mathbf{D}_T \circ \cdots \circ \mathbf{D}_{k+1} \circ \mathbf{D}_{k-1} \circ \cdots \circ \mathbf{D}_2 \circ \mathbf{D}_1 \circ \mathbf{E}[\mathbf{x}, \mathbf{y}]. \quad (2.7)$$

Последовательность слоев энкодера и применение токен-эмбеддингов представим в виде единого модуля $\mathbf{E}$:

$$\mathbf{E} : \underbrace{\mathbb{R}^{L \times d_{model}}}_{\text{токены x}} \times \underbrace{\mathbb{R}^{M \times v_{eng}}}_{\text{токены y}} \to \underbrace{\mathbb{R}^{L \times d_{model}}}_{\text{память}} \times \underbrace{\mathbb{R}^{M \times d_{model}}}_{\text{токен эмбеддинги y}} . \quad (2.8)$$

Слой декодера представим в виде:

$$\mathbf{D}_k : \underbrace{\mathbb{R}^{L \times d_{model}}}_{\text{память}} \times \underbrace{\mathbb{R}^{M \times d_{model}}}_{\text{эмбеддинги декодера}} \to \underbrace{\mathbb{R}^{L \times d_{model}}}_{\text{память}} \times \underbrace{\mathbb{R}^{M \times d_{model}}}_{\text{новые эмбеддинги декодера}} . \quad (2.9)$$

Генератор:

$$\mathbf{G} : \underbrace{\mathbb{R}^{L \times d_{model}}}_{\text{память}} \times \underbrace{\mathbb{R}^{M \times d_{model}}}_{\text{эмбеддинги декодера}} \to \underbrace{\mathbb{R}^{M \times d_{model} \times v_d}}_{\text{логиты}}. \quad (2.10)$$

Заметим, что модули действуют на пару $\mathbf{x}, \mathbf{y}$, но в случае энкодера к $\mathbf{y}$ применяются только токен-эмбеддинги, а в случае декодера – память $\mathbf{x}$ не изменяется, но используется в multi-head блоке дслоя декодера. Это сделано для удобства записи в виде суперпозиции преобразований.

Теперь параметризуем параметры моделей. Образ действия оператора модели учителя на текущий перевод до t-го токена $\mathbf{y}_{0:t}$ будет выглядеть как:

$$\log p(\mathbf{y}_{1:t+1}) = \mathbf{f}\big(\mathbf{x}, \mathbf{y}_{0:t}\big) = \mathbf{G}(\mathbf{U}_{T+1}) \circ \mathbf{D}_T(\mathbf{U}_T) \circ \cdots \circ \mathbf{D}_1(\mathbf{U}_1) \circ \mathbf{E}(\mathbf{U}_0)[\mathbf{x}, \mathbf{y}_{0:t}], \quad (2.11)$$

где $\mathbf{D}$ слои декодера, $\mathbf{E}$ энкодер, $\mathbf{G}$ генератор, возвращающий логиты следующих токенов. Параметры учителя фиксированы

$$\mathbf{u} = \mathrm{vec}\big([\mathbf{U}_{T+1}, \mathbf{U}_T, \mathbf{U}_{T-1}, \cdots \mathbf{U}_1, \mathbf{U}_0]\big), \quad (2.12)$$

где $\mathbf{U}_k$ векторизированные параметры соответствующих модулей.

На основе выборки $\{\mathbf{x}_i, \mathbf{y_i}\}_{i=1}^m$ и значений учителя $\mathbf{f}(\hat{\mathbf{u}}, \mathbf{x}, \mathbf{y})$ требуется выбрать модель ученика:

$$\mathbf{g}\big(\mathbf{x}, \mathbf{y}_{0:t}\big) = \mathbf{G}(\mathbf{W}_{L+1}) \circ \mathbf{D}_L(\mathbf{W}_L) \circ \cdots \circ \mathbf{D}_1(\mathbf{W}_1) \circ \mathbf{E}(\mathbf{W}_0)[\mathbf{x}, \mathbf{y}_{0:t}], \quad L \leq T, \tag{2.13}$$

где $\mathbf{G}, \mathbf{D}, \mathbf{E}$ вводятся как и отображения учителя. Задача выбора модели $\mathbf{g}$ состоит в оптимизации вектора $\mathbf{w} = \mathrm{vec}\big([\mathbf{W}_{L+1}, \mathbf{W}_L, \mathbf{W}_{L-1}, \cdots, \mathbf{W}_1, \mathbf{W}_0]\big)$. Решается вариационным выводом

$$\hat{\mathbf{w}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}} = \arg\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w}} \mathrm{D}_{\mathrm{KL}}\big(q(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})||p(\mathbf{w}|\mathbf{A})\big) - \mathsf{E}_{\mathbf{w}\sim q} \sum_{i=1}^{m} \log p\big(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}\big).$$

Априорное распределение $p\big(\mathbf{w}|\mathbf{A}\big)$ задается как функция от апостериорного распределения параметров учителя $p\big(\mathbf{u}|\mathbf{X}, \mathbf{Y}\big)$. Оно задано:

$$p\big(\mathbf{u}|\mathbf{X}, \mathbf{Y}\big) = \mathcal{N}\big(\mathbf{m}, \boldsymbol{\Sigma}\big).$$

Так как размерности $\mathbf{u}$ и $\mathbf{w}$ не совпадают, то применяется выравнивающее преобразование – приведение параметров моделей в одно общее пространство:

$$p\big(\mathbf{w}|\mathbf{A}\big) = p\big(\upsilon|\mathbf{X}, \mathbf{y}\big), \quad \boldsymbol{\upsilon} = \psi\big(\mathbf{u}\big), \quad \psi : \mathbb{R}^{\mathrm{p_{tr}}} \to \mathbb{R}^{\mathrm{p_{st}}}.$$

В данной работе основное внимание уделено получению таких выравнивающих преобразований.

2.2.1 RNN

Модель RNN (recurrent neural network) [11] представляет из себя два блока – энкодер и декодер. Энкодер представляет из себя сумму линейных преобразований с функцией активации:

$$h_i = \sigma \circ (Ue_i + Vh_{i-1}) \quad \forall i \in \{1, ..., T\}, \tag{2.14}$$

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

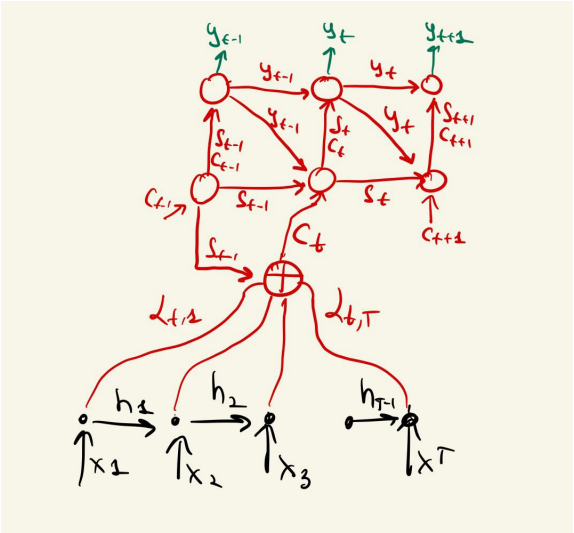


Рис. 1: Модель RNN с вниманием

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 [11]. 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 $\{h_i\}_{i=1}^{T}$:

$$f_t = \sigma \circ (W_f[y_{t-1}, c_t] + b_f) \tag{2.15}$$
$$i_t = \sigma \circ (W_i[y_{t-1}, c_t] + b_i) \tag{2.16}$$
$$o_t = \sigma \circ (W_o[y_{t-1}, c_t] + b_o) \tag{2.17}$$
$$\bar{s_t} = \tanh(W_s[y_{t-1}, c_t] + b_s) \tag{2.18}$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \bar{s_t} \tag{2.19}$$
$$y_t = o_t \odot \tanh(s_t), \tag{2.20}$$

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 [12] по выходным эмбеддингам энкодера $\{h_t\}_{t=1}^{T}$ формирует результирующий вектор c_t. В данной статье используется bahdanau attention [13], использующий в качестве alignment функции $a(h, s) = U_0 \tanh(U_1 h + U_2 s)$:

$$a_{t,j} = a(h_j, s_{t-1}) = U_0 \tanh(U_1 h_j + U_2 s_{t-1}) \tag{2.21}$$
$$\alpha_{t,j} = soft \max_{t \in 1,2,\ldots,T} a_{t,j} \tag{2.22}$$
$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j, \tag{2.23}$$

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

$$c_t = \frac{1}{T} \sum_{t=1}^{T} h_t \tag{2.24}$$

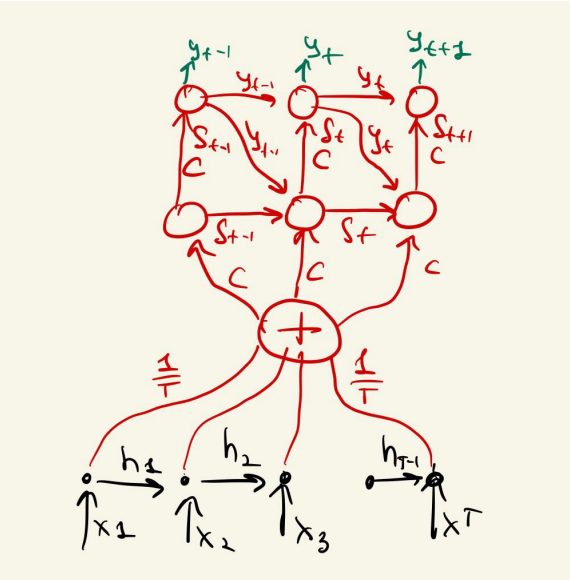


Рис. 2: Модель RNN без блока внимания

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

$$p_t = M_2 y_t \quad M_2 \in \mathbb{R}^{V_{eng} \times m_{dec}}, \tag{2.25}$$

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:

$$\mathcal{L}_{crossentropy} = \frac{1}{N}\sum_{n=1}^{N} l_n \tag{2.26}$$

$$l_n = -\log \frac{\exp(x_{n,y_n})}{\sum\limits_{c=1}^{V_{eng}} \exp(x_{n,c})}, \tag{2.27}$$

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 вероятность 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 отсутствует 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 [12] 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 эмбеддингов формируются как в RNN, но дополнительно к ним применится позиционное кодирование (positional-encoding), которое 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 [12]. Позиционный 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 эмбеддингов $\{h_t\}_{t=1}^{T} \quad h_t \in \mathbb{R}^{d_{model}}$, полученной после позиционного кодирования, формируются ключи, запросы и величины с помощью обучаемых линейных преобразований $\{W_r^q, W_r^k, W_r^v\}_{r=1}^{N_{heads}}$:

$$Q_i = W_r^q \cdot h_i \tag{2.28}$$

$$K_i = W_r^k \cdot h_i \tag{2.29}$$

$$V_i = W_r^v \cdot h_i, \tag{2.30}$$

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:

$$head_i = Att(Q, K, V) \tag{2.31}$$

$$Att(Q, K, V) = soft\max(\{\frac{QK^T}{\sqrt{d_k}}\})V, \tag{2.32}$$

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:

$$H = concat(head_1, head_2, ..., head_{N_{heads}}) \cdot W_o, \tag{2.33}$$

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:

$$H'_t = \sigma \circ (H_t \cdot W_1 + b_1)W_2 + b_2, \tag{2.34}$$

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 [14]. А сам энкодер – это последовательность нескольких преобразований $\{E_i\}_{i=1}^{N_{enc}}$:

$$H'' = Enc(H) \quad H \in \mathbb{R}^{T \times d_{model}} \tag{2.35}$$

$$Enc = E_1 \circ E_2 \circ, ..., E_{N_{enc}} \tag{2.36}$$

$$E_i = I + norm \circ FFN \circ (I + norm \circ SA) \tag{2.37}$$

$$SA = dropout \circ MHA \tag{2.38}$$

$$FFN(H_t) = dropout \circ (\sigma \circ (H_t \cdot W_1 + b_1)W_2 + b_2), \tag{2.39}$$

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 позиционного 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 в виде верхнетреугольной матрицы, состоящую из 0 и $-\infty$, которая прибавляется к текущей матрице внимания, тем самым после применения softmax к итоговой матрице внимания размерности $T \times T$ коэффициенты становятся нулевыми. Таким образом i-ый

эмбеддинг выходного тензора декодера 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 целевой перевод предложения.

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 $H''_{enc} \in \mathbb{R}^{T \times d_{model}}$:

$$Q = H_{dec} \cdot W_r^q \tag{2.40}$$

$$K = H''_{enc} \cdot W_r^k \tag{2.41}$$

$$V = H''_{enc} \cdot W_r^v, \tag{2.42}$$

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:

$$H'_{dec} = concat(\{head_i\}_{i=1}^{N_{heads}}) \cdot W_o \in \mathbb{R}^{N_{heads} \cdot d_v \times d_{model}}, \tag{2.43}$$

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 эмбеддингам декодера H''_{dec}, которые уже можно интерпретировать, как перевод текста длины L в представлении эмбеддингов:

$$logits = H''_{dec} \cdot W_{generator}, \tag{2.44}$$

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, обнуляя 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 кросс энтропийный лосс и на этом заканчивается пайплайн трансформера.

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 декодера 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 модель 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:

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

где μ, Σ – гиперпараметры этого распределения. Используя эти гиперпараметры необходимо задать гиперпараметры A априорного распределения 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 выполнении 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:

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

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

Далее будут сформулированы теоремы о выравнивающих преобразованиях модели учителя RNN с вниманием. Они будут полагаться на факт, что при зануленении параметров U_0, либо пары параметров U_1, U_2, модель учителя становится эквивалентной модели ученика с той же подструктурой и параметрами.

Прежде, чем перейти к теоремам, 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

$$p(\upsilon|D) = \mathcal{N}(\mu_\upsilon + \Sigma_{\upsilon,\xi}\Sigma_\xi^{-1}(0 - \mu_\upsilon), \Sigma_\upsilon - \Sigma_{\upsilon,\xi}\Sigma_\xi^{-1}\Sigma_{\xi,\upsilon}), \tag{3.4}$$

где $\xi = [U_0, U_1, U_2]$ *– векторизованные параметры bahdanau attention,* υ *– 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 $\xi = [U_0, U_1, U_2] = 0$ 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, сформулированную ранее, можно получить искомое распределение подвектора υ вектора $u = [\upsilon, \xi] = [\upsilon, U_0, U_1, U_2]$. □

Также можно заметить, что для эквивалентности модели учителя и модели ученика достаточно обнулить 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*

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

$$R = \Sigma_z - \Sigma_{z,U_0}\Sigma_{U_0}^{-1}\Sigma_{U_0,z} \tag{3.6}$$

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

где $z = [\upsilon, U_1, U_2]$ *– оставшиеся* μ *и удаляемые параметры внимания* U_1, U_2*,* R *– ковариационная матрица между векторами* z, U_0

, U_0 –обнуляемые параметры внимания, $m_\upsilon, R_{\upsilon,\upsilon}$ – подвектор m и подматрица R, соответствующие подвектору υ в векторе z.

Доказательство. Исходя из леммы запишем апостериорное распределение для вектора $z = [\upsilon, U_1, U_2]$ 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 υ:

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

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

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

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

□

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 зануленении параметров U_1, U_2 – апостериорное распределение параметров описывается нормальным распределением

$$m = \mu_z + \Sigma_{z,\upsilon}\Sigma_\upsilon^{-1}(0 - \mu_z) \tag{3.10}$$

$$R = \Sigma_z - \Sigma_{z,\upsilon}\Sigma_\upsilon^{-1}\Sigma_{\upsilon,z} \tag{3.11}$$

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

где $z = [\upsilon, U_0]$ *– оставшиеся* μ *и удаляемые параметры внимания* U_0*,* R *– ковариационная матрица между векторами* z, υ *,* $\upsilon = [U_1, U_2]$ *– обнуляемые параметры внимания,* $m_\upsilon, R_{\upsilon,\upsilon}$ *– подвектор* m *и подматрица* R*, соответствующие подвектору* υ *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 декодера число голов меньше на единицу (если голов изначально больше 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 обнуляемые:

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

где v_1 – удаляемые параметры, v_2 – обнуляемые параметры, υ – 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, а также занулить параметры в модулях multi-head блока (в случае декодера) и блока self-attention, 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.

Рассмотрим случай удаления декодера подробно. Построим выравнивающее преобразование ψ из пространства параметров модели учителя в пространство параметров модели ученика:

$$\psi : \mathbb{R}^{\mathrm{p_{tr}}} \to \mathbb{R}^{\mathrm{p_{tr}} - 4 \cdot d_{model} \times d_k - 2 d_v \times d_{model} - 2 \cdot N_{heads} \cdot d_v \times d_{model} - 2 d_{model} \times d_{ff} - d_{ff} - d_{model}}, \tag{3.14}$$

где p_{tr} – число параметров учителя, остальные слагаемые – размерности пространств удаляемых параметров.

Распишем поподробней преобразования в модели учителя :

$$\mathbf{f}\big(\mathbf{x}, \mathbf{y}\big) = \mathbf{G}(\mathbf{U}_{T+1}) \circ \mathbf{D}_T(\mathbf{U}_T) \circ \cdots \circ \mathbf{D}_2(\mathbf{U}_2) \circ \mathbf{D}_1(\mathbf{U}_1) \circ \mathbf{E}(\mathbf{U}_0)[\mathbf{x}, \mathbf{y}]$$

$$\mathbf{D}_k\big(\mathbf{x}, \mathbf{y}\big) \equiv [\mathbf{x}, \mathbf{D}_k^0\big(\mathbf{x}, \mathbf{y}\big)]$$

$$\mathbf{D}_k^0(\mathbf{x}, \mathbf{y}) = \underbrace{(\mathbf{I} + \mathbf{FFN})}_{\text{feed-forward}} \circ \underbrace{(\mathbf{I} + \mathbf{MHA})[\mathbf{x}]}_{\text{multi-head attention}} \circ \underbrace{(\mathbf{I} + \mathbf{SA})}_{\text{self attention}}(\mathbf{y})$$

$$\mathbf{FFN}(\mathbf{y}) = (\mathbf{b}_2 + \sigma \circ \mathbf{U}_2^F \circ (\mathbf{b}_1 + \mathbf{y} \cdot \mathbf{U}_1^F))$$

$$\mathbf{MHA}(\mathbf{x}, \mathbf{y}) = \mathrm{heads}[\mathbf{U}_M^h](\mathbf{x}, \mathbf{y}) \cdot \mathbf{U}^M \qquad \mathbf{SA}(\mathbf{y}) = \mathrm{heads}[\mathbf{U}_S^h](\mathbf{y}) \cdot \mathbf{U}^S,$$

где σ функция активации, параметры модуля U_k: $\mathbf{U}_1^F \in \mathbb{R}^{d_{model} \times d_{ff}}$, $\mathbf{U}_2^F \in \mathbb{R}^{d_{ff} \times d_{model}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{ff}}$, $\mathbf{b}_2 \in \mathbb{R}^{d_{model}}$, $\mathbf{U}^M, \mathbf{U}^S \in \mathbb{R}^{N_{heads} \cdot d_v \times d_{model}}$, $\mathbf{U}_M^h, \mathbf{U}_S^h \in \mathbb{R}^{2 \cdot d_k \times d_{model} + \cdot d_v \times d_{model}}$.

Модель ученика имеет такую же структуру, но у нее нет одного слоя декодера $\mathbf{D}_k$:

$$\mathbf{g}\big(\mathbf{x}, \mathbf{y}\big) = \mathbf{G}(\mathbf{W}_{T+1}) \circ \cdots \circ \mathbf{D}_{k+1}(\mathbf{W}_{k+1}) \circ \mathbf{D}_{k-1}(\mathbf{W}_{k-1}) \circ \cdots \circ \mathbf{D}_1(\mathbf{W}_1) \circ \mathbf{E}(\mathbf{W}_0)[\mathbf{x}, \mathbf{y}$$

Модели эквивалентны, при условии что остальные параметры их модулей совпадают:

$$\mathbf{f} \mid_{\mathbf{U}_2^F, \mathbf{b}_2, \mathbf{U}^M, \mathbf{U}^S = 0} \equiv \mathbf{g}. \tag{3.15}$$

Таким образом параметры $\mathbf{u}$ модели $\mathbf{f}$ делятся на:

1. удаляемые $\boldsymbol{\xi}_2 = vec([b_1, U_1^F, U_M^h, U_S^h])$,
2. зануляемые $\boldsymbol{\xi}_1 = \text{vec}([b_2, U_2^F, U^S, U^M])$,
3. оставшиеся $\boldsymbol{\upsilon} = \text{vec}\big([\mathbf{U}_{T+1}, \cdots, \mathbf{U}_{k+1}, \mathbf{U}_{k-1}, \cdots, \mathbf{U}_0]\big)$.

При таком выборе параметров можно построить выравнивающее преобразование: $\boldsymbol{\upsilon} = \psi_{transformer}(\mathbf{u}, k)$, а далее применить следующую теорему:

Теорема 5. *Пусть выполняются следующие условия :*

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

Тогда при удалении параметров υ_1 *и занулении вектора параметров* υ_2 *– апостериорное распределение параметров описывается нормальным распределением*

$$
\begin{aligned}
m &= \mu_z + \Sigma_{z,\upsilon_2}\Sigma_{\upsilon_2}^{-1}(0 - \mu_z) \\
R &= \Sigma_z - \Sigma_{z,\upsilon_2}\Sigma_{\upsilon_2}^{-1}\Sigma_{\upsilon_2,z} \\
p(\upsilon|\mathcal{D}) &= \mathcal{N}(m_\upsilon, R_{\upsilon,\upsilon}),
\end{aligned}
$$

где $z = [\upsilon, \upsilon_1]$ *– оставшиеся* μ *и удаляемые параметры* υ_1, R *– ковариационная матрица между векторами* z, υ_2 , υ_2 *– обнуляемые параметры,* $m_\upsilon, R_{\upsilon,\upsilon}$ *– подвектор* m *и подматрица* R*, соответствующие подвектору* υ *в векторе* z*.*

Доказательство. Такое же, как у теорем про RNN □

Помимо перечисленных методов, удалить слой энкодера/декодера можно последовательно:

1. Удалить блок feed forward методами, описанными в статье [8]
2. Удалить блок multi head attention, используя теорему 5, в которой нет удаляемых параметров: $u_1 = \{\emptyset\}$

3.3 Случай некореллированных параметров

Рассмотрим частный случай, когда ковариационные матрицы нулевые $\mathbf{\Sigma}_{\bar{\boldsymbol{\xi}}_1,\boldsymbol{\xi}_1} = 0$, то есть зануляемыые параметры $\boldsymbol{\xi}_1$ и остальные параметры $\bar{\boldsymbol{\xi}}_1$ – некореллированы. Тогда распределение $p\big(\bar{\boldsymbol{\xi}}_1|\mathfrak{D}, \boldsymbol{\xi}_1 = \mathbf{0}\big) = \mathcal{N}\big(\boldsymbol{\mu}, \boldsymbol{\Xi}\big)$ имеет параметры $\boldsymbol{\mu}, \boldsymbol{\Xi}$:

$$\begin{aligned}\boldsymbol{\mu} &= \mathbf{m}_{\bar{\boldsymbol{\xi}}_1},\\ \boldsymbol{\Xi} &= \mathbf{\Sigma}_{\bar{\boldsymbol{\xi}}_1,\bar{\boldsymbol{\xi}}_1}.\end{aligned}$$

Следовательно апостериорное распределение параметров модели учителя υ:

$$p\big(\boldsymbol{\upsilon}|\mathfrak{D}\big) = \mathcal{N}\big(\boldsymbol{m}_{\boldsymbol{\upsilon}}, \mathbf{\Sigma}_{\boldsymbol{\upsilon},\boldsymbol{\upsilon}}\big).$$

Априорное распределение модели ученика :

$$p\big(\boldsymbol{w}|\mathbf{A}\big) = p\big(\boldsymbol{\upsilon}|\mathfrak{D}\big).$$

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 [15].

В статье [13] приводятся графики зависимости качества моделей от длины тестовых предложений. Так, например, на длинах предложений равных 20 модель RNN с вниманием имеет величину BLEU на тестовых данных около 25, а модель без внимания около 20. Так как средняя длина предложений в нашем обучающем и тестовом датасетах порядка 20, можно предполагать, что модель RNN с блоком внимания будет давать лучшее значение BLEU примерно на 20 процентов, чем модель без внимания при одинаковых условиях обучения.

Распределение длин предложений в данном датасете выглядит следующим образом:

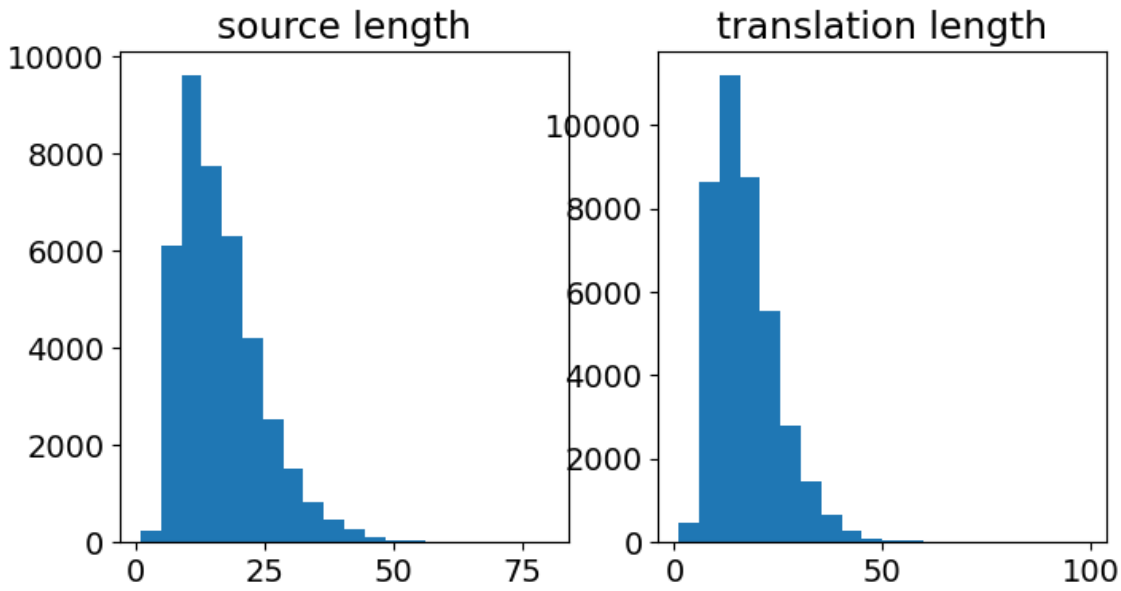


Рис. 3: Распределение длин текстов в обучающей выборке

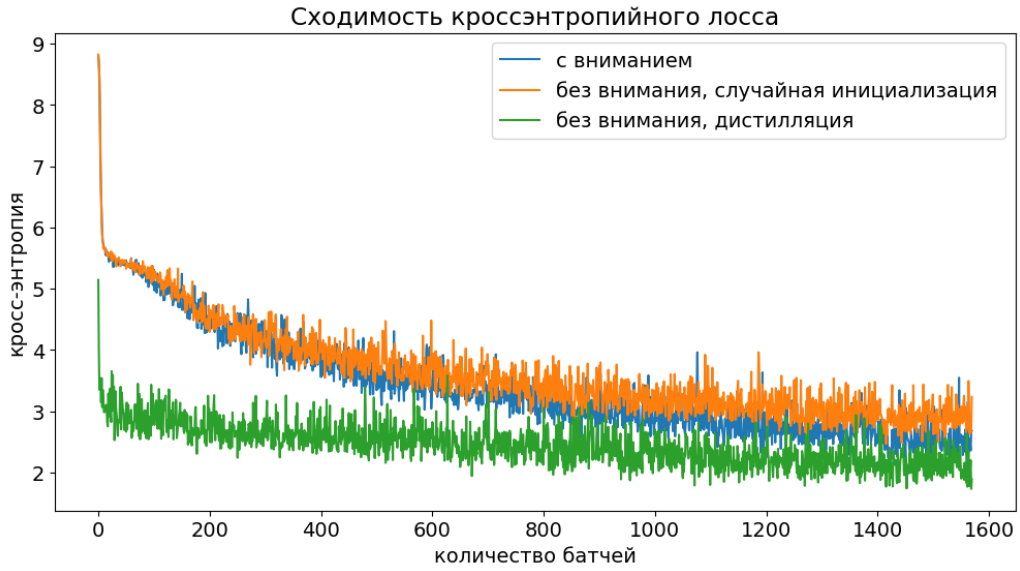
5 Вычислительный эксперимент

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

- Размер эмбеддингов в декодере и энкодере – 512
- Размерность скрытых слоев – 512
- Размерность линейных преобразований в модуле внимания – 512

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

Сходимость кросс-энтропийного лосса для модели с вниманием, модели без внимания с инициализацией из равномерного распределения, дистиллированной модели без внимания, посредством копирования весов модели с вниманием:



	модель учителя	**модель ученика**	**модель ученика (дистиллированная)**
BLEU	23.70	13.35	17.73

Дистиллированная модель сходится лучше, а также показывает лучшую метрику BLEU на отложенной выборке, чем модель той же структуры, но с произвольной инициализацией.

В качестве приближенных значений ковариационной матрицы апостериорного распределения между векторами параметров можно использовать:

1. Выборочную ковариацию на последних K итерациях обучения модели.

2. Выборочную ковариацию на итоговых значениях параметров после K сэмплированных процессов обучения с одинаковыми условиями.

В качестве матожидания апостериорного распределения модели учителя можно использовать окончательные значения весов модели после обучения.

В предположении, что удаляемые параметры и оставшиеся параметры некореллированы, апостериорное распределение параметров

модели учителя после удаления блока внимания в соответствии с теоремами, предложенными ранее, будет такое же, что у распределения подвектора параметров апостериорного распределения изначальной модели учителя. То есть для дистилляции в таком предположении достаточно скопировать параметры изначальной модели учителя, чтобы получить параметры для модели ученика.

6 Заключение

В данной работе исследовалась задача дистилляции моделей на базе трансформеров. Рассмотрена задача машинного перевода. Предложены методы дистилляции модели RNN с аддитивным модулем внимания: полное или частичное зануление весов модуля с удалением оставшейся его части. Предложены методы дистилляции модели трансформера посредством удаления слоя энкодера/декодера либо уменьшения числа голов в multi-head блоке внимания. Был проведен теоретический анализ предложенных методов и получены апостериорные распределения. Также проведен эксперимент для частного случая некореллированных параметров, в котором дистиллированная модель показывает лучшие метрики и сходимость по сравнению с моделью той же структуры, но с произвольной инициализацией.

Список литературы

[1] *Brown T., et al.* GPT3: Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems. 2020. V. 33. P. 1877–1901.

[2] *Devlin J., Chang M., Lee K., Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minnesota. 2019. V. 1. P. 4171–4186.

[3] *Hinton G., Dean J., Vinyals O.* Distilling the Knowledge in a Neural Network // NIPS Deep Learning and Representation Learning Workshop. 2015.

[4] *Burges C., Cortes C., LeCun Y.* MNIST dataset of handwritten digits. 1998. http://yann.lecun.com/exdb/mnist/index.html.

[5] *Huang Z., Naiyan W.* Like What You Like: Knowledge Distill via Neuron Selectivity Transfer // arXiv:1707.01219. 2017.

[6] *Hinton G., Krizhevsky A., Nair V.* CIFAR-10 (Canadian Institute for Advanced Research) // http://www.cs.toronto.edu/ kriz/cifar.html

[7] *Deng J., et al.* Imagenet: A Large-scale Hierarchical Image Database // Proc. IEEE Conf. on Computer Vision and Pattern Recognition. Miami. 2009. P. 248–255.

[8] *Grabovoy A.V., Strijov V.V.* Байесовская дистилляция моделей глубокго обучения // Automation and Remote Control. 2021. Т. 82. № 11. С. 1846-1856.

[9] *Rasul K., Vollgraf R., Xiao H.* Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms // arXiv preprint arXiv:1708.07747. 2017.

[10] *Graves A.* Practical Variational Inference for Neural Networks // Advances in Neural Information Processing Systems. 2011. V. 24. P. 2348–2356.

[11] *Hochreiter S., Schmidhuber J.* Long Short-Term Memory // 1997. Neural Computation, 9(8), 1735-1780.

[12] *Vaswani A. et al.* Attention is All you Need // 2017. Neural Information Processing Systems.

[13] *Bahdanau D., Cho K., Bengio Y.* Neural machine translation by jointly learning to align and translate // 2015. ICLR.

[14] *Paszke, Adam et al.* PyTorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703. Retrieved from https://pytorch.org

[15] *Neychev R.* https://github.com/neychev/made_nlp_course