Внимание — это все, что вам нужно

Ашиш Васвани*
Google Мозг
avaswani@google.com

Ноам Шазир*
Google Мозг
noam@google.com

Ники Пармар*
Исследования Google
nikip@google.com

Якоб Ушкорейт*
Исследования Google
usz@google.com

Ллион Джонс*
Исследования Google
llion@google.com

Эйдан Н. Гомес* †
Университет Торонто
aidan@cs.toronto.edu

Лукаш Кайзер*
Google Мозг
lukaszkaiser@google.com

Илья Полосухин* ‡
illia.polosukhin@gmail.com

Абстрактный

Доминирующие модели последовательной трансдукции основаны на сложных рекуррентных или сверточных нейронных сетях, которые включают кодер и декодер. Наиболее эффективные модели также соединяют кодер и декодер через механизм внимания. Мы предлагаем новую простую сетевую архитектуру, Transformer, основанную исключительно на механизмах внимания, полностью обходясь без рекуррентности и сверток. Эксперименты с двумя задачами машинного перевода показывают, что эти модели превосходят по качеству, будучи более параллелизуемыми и требуя значительно меньше времени на обучение. Наша модель достигает 28,4 BLEU в задаче перевода с английского на немецкий язык WMT 2014, улучшая существующие лучшие результаты, включая ансамбли, более чем на 2 BLEU. В задаче перевода с английского на французский язык WMT 2014 наша модель устанавливает новый современный показатель BLEU для одной модели в 41,8 после обучения в течение 3,5 дней на восьми графических процессорах, что составляет небольшую часть затрат на обучение лучших моделей из литературы. Мы показываем, что Transformer хорошо обобщается и на другие задачи, успешно применяя его для анализа английской электоральной аудитории как с большими, так и с ограниченными обучающими данными.

*Равный вклад. Порядок перечисления случайный. Якоб предложил заменить RNN на самовнимание и начал работу по оценке этой идеи. Ашиш с Ильей спроектировали и реализовали первые модели Transformer и принимали решающее участие во всех аспектах этой работы. Ноам предложил масштабированное внимание скалярного произведения, внимание с несколькими головами и представление позиции без параметров и стал другим человеком, вовлеченным почти в каждую деталь. Ники спроектировал, реализовал, настроил и оценил бесчисленное количество вариантов моделей в нашей исходной кодовой базе и tensor2tensor. Ллион также экспериментировал с новыми вариантами моделей, отвечал за нашу исходную кодовую базу и эффективный вывод и визуализацию. Лукаш и Эйдан провели бесчисленное количество долгих дней, проектируя различные части и реализуя tensor2tensor, заменив нашу более раннюю кодовую базу, значительно улучшив результаты и значительно ускорив наши исследования.

†Работа, выполненная во время работы в Google Brain.

‡Работа выполнялась во время работы в Google Research.

31-я конференция по нейронным системам обработки информации (NIPS 2017), Лонг-Бич, Калифорния, США.

1 Введение

Рекуррентные нейронные сети, в частности, долговременная кратковременная память [13] и управляемые рекуррентные [7] нейронные сети, прочно утвердились как современные подходы в моделировании последовательностей и проблемах трансдукции, таких как моделирование языка и машинный перевод [35, 2, 5]. С тех пор многочисленные усилия продолжают расширять границы рекуррентных языковых моделей и архитектур кодера-декодера [38, 24, 15].

Рекуррентные модели обычно факторизуют вычисления по позициям символов входных и выходных последовательностей. Выравнивая позиции по шагам во времени вычислений, они генерируют последовательность скрытых состояний $час_т$, как функция предыдущего скрытого состояния $час_{т-1}$ и вход для позиции $т$. Эта изначально последовательная природа исключает параллелизацию в обучающих примерах, что становится критическим при более длинных последовательностях, поскольку ограничения памяти ограничивают пакетирование между примерами. Недавние работы достигли значительного улучшения вычислительной эффективности с помощью трюков факторизации [21] и условных вычислений [32], а также улучшили производительность модели в случае последнего. Однако фундаментальное ограничение последовательных вычислений остается.

Механизмы внимания стали неотъемлемой частью убедительного моделирования последовательностей и моделей трансдукции в различных задачах, позволяя моделировать зависимости без учета их расстояния во входных или выходных последовательностях [2, 19]. Однако во всех случаях, за исключением нескольких [27], такие механизмы внимания используются в сочетании с рекуррентной сетью.

В этой работе мы предлагаем Transformer, модель архитектуры, избегающую повторения и вместо этого полностью полагающуюся на механизм внимания для рисования глобальных зависимостей между входом и выходом. Transformer допускает значительно больше распараллеливания и может достичь нового уровня искусства в качестве перевода после обучения всего лишь в течение двенадцати часов на восьми графических процессорах P100.

2 Предыстория

Цель сокращения последовательных вычислений также лежит в основе Extended Neural GPU [16], ByteNet [18] и ConvS2S [9], все из которых используют сверточные нейронные сети в качестве базового строительного блока, вычисляя скрытые представления параллельно для всех входных и выходных позиций. В этих моделях количество операций, необходимых для связывания сигналов из двух произвольных входных или выходных позиций, растет с расстоянием между позициями, линейно для ConvS2S и логарифмически для ByteNet. Это затрудняет изучение зависимостей между удаленными позициями [12]. В Transformer это сводится к постоянному количеству операций, хотя и ценой снижения эффективного разрешения из-за усреднения позиций, взвешенных по вниманию, эффект, которому мы противодействуем с помощью Multi-Head Attention, как описано в разделе 3.2.

Внимательность, иногда называемая интра-вниманием, — это механизм внимания, связывающий различные позиции одной последовательности для вычисления представления последовательности. Внимательность успешно использовалась в различных задачах, включая понимание прочитанного, абстрактное обобщение, текстовое выведение и изучение независимых от задачи представлений предложений [4, 27, 28, 22].

Сквозные сети памяти основаны на механизме повторяющегося внимания, а не на последовательно-выровненной рекуррентности, и, как было показано, хорошо работают при ответах на простые языковые вопросы и задачах моделирования языка [34].

Однако, насколько нам известно, Transformer — это первая модель трансдукции, которая полностью полагается на внутреннее внимание для вычисления представлений своего входа и выхода без использования последовательно-выровненных RNN или свертки. В следующих разделах мы опишем Transformer, мотивируем внутреннее внимание и обсудим его преимущества по сравнению с такими моделями, как [17, 18] и [9].

3 Модель Архитектуры

Большинство конкурентоспособных моделей нейронной последовательности имеют структуру кодер-декодер [5, 2, 35]. Здесь кодер отображает входную последовательность представлений символов $(х_1, ..., х_н)$к последовательности непрерывных представленийz = $(з_1, ..., з_н)$. Данныйз,Затем декодер генерирует выходную последовательность $(у_1, ..., г_м)$ символов по одному элементу за раз. На каждом шаге модель авторегрессивна [10], потребляя ранее сгенерированные символы в качестве дополнительных входных данных при генерации следующих.

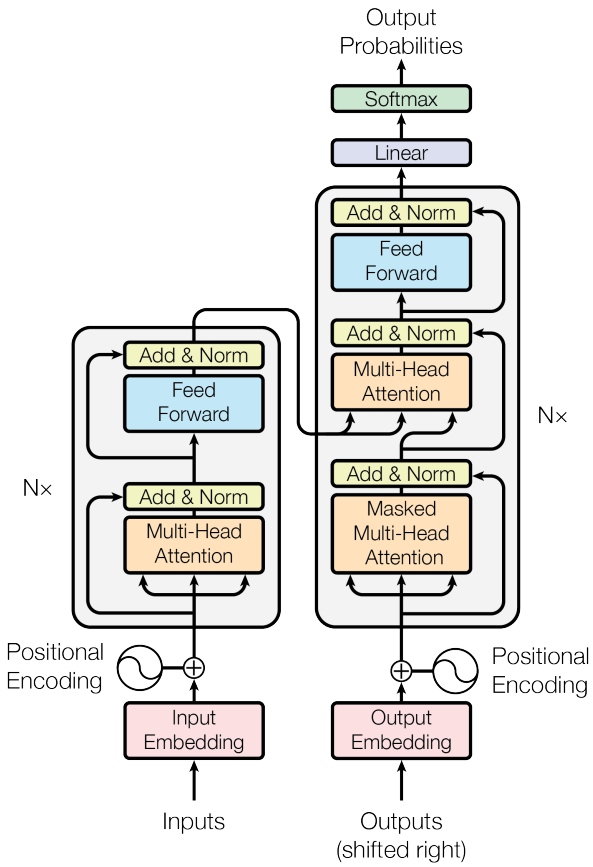


Рисунок 1: Архитектура модели Transformer.

Трансформер следует этой общей архитектуре, используя многоуровневое внутреннее внимание и точечные полностью связанные слои как для кодера, так и для декодера, показанные в левой и правой половинах рисунка 1 соответственно.

3.1 Стеки кодера и декодера

Кодер:Кодер состоит из стекаH=6идентичные слои. Каждый слой имеет два подслоя. Первый — это механизм самовнимания с несколькими головками, а второй — простая, позиционно полностью связанная сеть прямой связи. Мы используем остаточную связь [11] вокруг каждого из двух подслоев, за которой следует нормализация слоя [1]. То есть выход каждого подслоя — это LayerNorm(x+Подслой(x)),гдеПодслой(x)это функция, реализуемая самим подслоем. Для облегчения этих остаточных связей все подслои в модели, а также встроенные слои, производят выходные данные размерности$г_{\text{модель}}$=512.

Декодер:Декодер также состоит из стекаH=6идентичные слои. В дополнение к двум подслоям в каждом слое кодировщика, декодер вставляет третий подслой, который выполняет многоголовое внимание над выходом стека кодировщика. Подобно кодировщику, мы используем остаточные соединения вокруг каждого из подслоев, за которыми следует нормализация слоев. Мы также изменяем подслой собственного внимания в стеке декодера, чтобы предотвратить внимание позиций к последующим позициям. Эта маскировка, в сочетании с тем фактом, что выходные вложения смещены на одну позицию, гарантирует, что прогнозы для позиции$я$может зависеть только от известных выходов в позициях, меньших$я$.

3.2 Внимание

Функция внимания может быть описана как отображение запроса и набора пар ключ-значение на выход, где запрос, ключи, значения и выход являются векторами. Выход вычисляется как взвешенная сумма

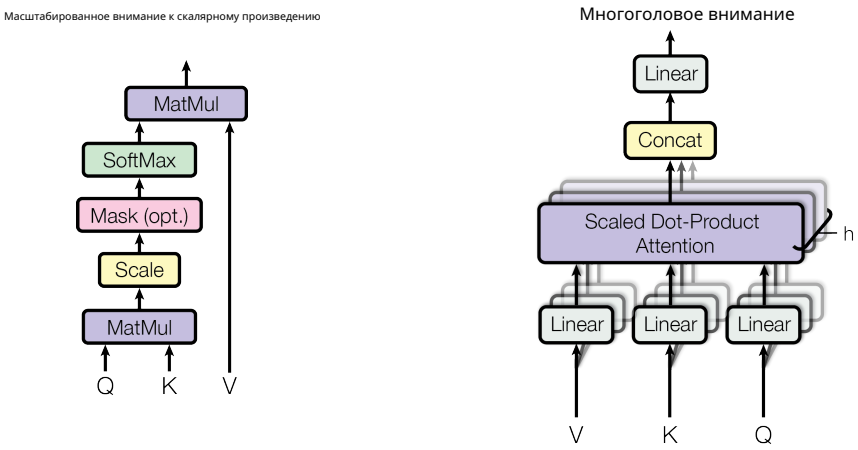


Рисунок 2: (слева) Масштабированное внимание по скалярному произведению. (справа) Многоголовое внимание состоит из нескольких параллельно работающих слоев внимания.

значений, где вес, присвоенный каждому значению, вычисляется функцией совместимости запроса с соответствующим ключом.

3.2.1 Масштабированное внимание к скалярному произведению

Мы называем наше особое внимание «Scaled Dot-Product Attention» (рисунок 2). Входные данные состоят из запросов и ключей размерности d_k, а $\sqrt{}$ nd значения размерности d_v. Мы вычисляем скалярные произведения запрос со всеми ключами, разделить каждый на d_k и применить функцию softmax для получения весов значений.

На практике мы вычисляем функцию внимания по набору запросов одновременно, упакованных в матрицу B. Ключи и значения также упакованы вместе в матрицы K и B. Мы вычисляем матрицу выходов следующим образом:

$$\text{Внимание}(K, K, B) = \text{софтмакс}(\frac{KK^T}{\sqrt{d_k}})B \quad (1)$$

Две наиболее часто используемые функции внимания — это аддитивное внимание [2] и внимание скалярного произведения (мультипликативное). Внимание скалярного произведения идентично нашему алгоритму, за исключением масштабирующего фактора $\frac{1}{\sqrt{d_k}}$. Аддитивное внимание вычисляет функцию совместимости с использованием сети прямой связи с один скрытый слой. Хотя эти два метода схожи по теоретической сложности, метод внимания к скалярному произведению на практике гораздо быстрее и эффективнее с точки зрения занимаемой памяти, поскольку его можно реализовать с использованием высокооптимизированного кода умножения матриц.

В то время как для малых значений d_k оба механизма работают одинаково, аддитивное внимание превосходит внимание, основанное на скалярном произведении, без масштабирования для больших значений d_k [3]. Мы подозреваем, что для больших значений d_k, скалярные произведения увеличиваются по величине, перемещая функцию softmax в области, где она имеет чрезвычайно малые градиенты[4]. Чтобы противодействовать этому эффекту, мы масштабируем скалярные произведения на $\frac{1}{\sqrt{d_k}}$.

3.2.2 Многоголовое внимание

Вместо того, чтобы выполнять одну функцию внимания с $d_{\text{модель}}$-мерные ключи, значения и запросы, мы обнаружили, что полезно линейно проецировать запросы, ключи и значения h раз с различными, изученными линейными проекциями d_k, d_k и d_v измерения, соответственно. На каждой из этих спроецированных версий запросов, ключей и значений мы затем выполняем функцию внимания параллельно, получая d_v-мерный

[4] Чтобы проиллюстрировать, почему скалярные произведения становятся большими, предположим, что компоненты q и k являются независимыми случайными переменные со средним значением 0 и дисперсия 1. Затем их скалярное произведение, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, имеет среднее значение 0 и дисперсия d_k.

выходные значения. Они объединяются и снова проецируются, что приводит к конечным значениям, как показано на рисунке 2.

Многоголовое внимание позволяет модели совместно уделять внимание информации из разных подпространств представления в разных позициях. При наличии одной головы внимания усреднение препятствует этому.

$$\text{Многоголовочный}(K, K, B) = \text{Concat}(\text{голова}_1, \ldots, \text{голова}_{\text{час}})W^O$$

$$\text{где голова}_i = \text{Внимание}(KW^B_i, KW^K_i, KW^B_i)$$

Где проекции являются матрицами параметров $W^Q_i \in \mathbb{R}^{d_{\text{модель}} \times d_k}$, $W^K_i \in \mathbb{R}^{d_{\text{модель}} \times d_k}$, $W^B_i \in \mathbb{R}^{d_{\text{модель}} \times d_v}$ и $W^O \in \mathbb{R}^{hd_v \times d_{\text{модель}}}$.

В этой работе мы используем $h = 8$ параллельные слои внимания, или головы. Для каждого из них мы используем $d_k = d_v = d_{\text{модель}}/h = 64$. Из-за уменьшенной размерности каждой головы общие вычислительные затраты аналогичны затратам на внимание одной головы с полной размерностью.

3.2.3 Применение внимания в нашей модели

Трансформер использует многоголовое внимание тремя различными способами:

- В слоях "внимание кодировщика-декодировщика" запросы поступают из предыдущего слоя декодера, а ключи и значения памяти поступают из выходных данных кодировщика. Это позволяет каждой позиции в декодере обслуживать все позиции во входной последовательности. Это имитирует типичные механизмы внимания кодировщика-декодировщика в моделях последовательность-последовательность, таких как [38, 2, 9].

- Кодировщик содержит слои самовнимания. В слое самовнимания все ключи, значения и запросы приходят из одного и того же места, в данном случае из вывода предыдущего слоя кодировщика. Каждая позиция в кодировщике может обслуживать все позиции в предыдущем слое кодировщика.

- Аналогично, слои самовнимания в декодере позволяют каждой позиции в декодере уделять внимание всем позициям в декодере вплоть до этой позиции включительно. Нам нужно предотвратить левый поток информации в декодере, чтобы сохранить свойство авторегрессии. Мы реализуем это внутри масштабированного внимания скалярного произведения, маскируя (устанавливая значение $-\infty$) все значения на входе softmax, которые соответствуют недопустимым соединениям. См. рисунок 2.

3.3 Сети прямой связи по положению

В дополнение к подслоям внимания, каждый из слоев в нашем кодере и декодере содержит полностью связанную сеть прямой связи, которая применяется к каждой позиции отдельно и идентично. Это состоит из двух линейных преобразований с активацией ReLU между ними.

$$\text{ФФН}(x) = \text{макс}(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

Хотя линейные преобразования одинаковы в разных позициях, они используют разные параметры от слоя к слою. Другой способ описания этого — две свертки с размером ядра 1. Размерность входных и выходных данных равна $d_{\text{модель}} = 512$, и внутренний слой имеет размерность $d_{ff} = 2048$.

3.4 Встраивание и Softmax

Подобно другим моделям преобразования последовательностей, мы используем изученные вложения для преобразования входных токенов и выходных токенов в векторы размерности $d_{\text{модель}}$. Мы также используем обычное обученное линейное преобразование и функцию softmax для преобразования выходных данных декодера в предсказанные вероятности следующего токена. В нашей модели мы разделяем одну и ту же весовую матрицу между двумя слоями внедрения и пред-$\sqrt{}$ софтмакс Линейное преобразование, аналогичное [30]. В слоях внедрения мы умножаем эти веса на $\sqrt{d_{\text{модель}}}$.

Таблица 1: Максимальная длина пути, сложность на слой и минимальное количество последовательных операций для различных типов слоев. n длина последовательности, r это измерение представления, k размер ядра сверток и r размер района в ограниченном самовнимании.

Тип слоя	Сложность на слой	Последовательный Операции	Максимальная длина пути
Само-Внимание	$O(n^2 \cdot r)$	$O(1)$	$O(1)$
Рецидивирующий	$O(n \cdot r^2)$	$O(n)$	$O(n)$
Сверточный	$O(k \cdot n \cdot r^2)$ $O($	$O(1)$	$O(\text{бревно}_k(n))$
Само-Внимание (ограничено)	$r \cdot n \cdot r)$	$O(1)$	$O(n/p)$

3.5 Позиционное кодирование

Поскольку наша модель не содержит повторений и свёрток, для того, чтобы модель могла использовать порядок последовательности, мы должны ввести некоторую информацию об относительном или абсолютном положении токенов в последовательности. Для этого мы добавляем "позиционные кодировки" к входным вложениям в нижней части стеков кодера и декодера. Позиционные кодировки имеют ту же размерность $r_{\text{модель}}$ как вложения, так что эти два могут быть суммированы. Существует много вариантов позиционных кодировок, изученных и фиксированных [9].

В данной работе мы используем функции синуса и косинуса различной частоты:

$$\text{ЧП}_{(\text{поз},2я)} = \text{грех}(\text{поз} / 10000^{2\text{идентификатор}_{\text{модель}}}) \quad \text{ЧП}_{(\text{поз},}$$

$$^{2я+1)} = \text{потому что}(\text{поз} / 10000^{2\text{идентификатор}_{\text{модель}}})$$

где поз это положение и $я$ является измерением. То есть, каждое измерение позиционного кодирования соответствует синусоиде. Длины волн образуют геометрическую прогрессию от 2π к $10000 \cdot 2\pi$. Мы выбрали эту функцию, поскольку предположили, что она позволит модели легко научиться отслеживать относительные положения, поскольку для любого фиксированного смещения k, $\text{ЧП}_{\text{поз}+k}$ может быть представлена как линейная функция $\text{ЧП}_{\text{поз}}$.

Мы также экспериментировали с использованием обученных позиционных вложений [9] и обнаружили, что две версии дали почти идентичные результаты (см. строку (E) Таблицы 3). Мы выбрали синусоидальную версию, поскольку она может позволить модели экстраполировать на длины последовательностей, превышающие те, которые встречались во время обучения.

4. Почему самовнимание

В этом разделе мы сравниваем различные аспекты слоев внутреннего внимания с рекуррентными и сверточными слоями, обычно используемыми для отображения одной последовательности символов переменной длины. $(x_1, ..., x_n)$ к другой последовательности такой же длины $(z_1, ..., z_n)$, с $x_я, z_я \in \mathbb{P}^r$, например, скрытый слой в типичном кодере или декодере последовательной трансдукции. Мотивируя наше использование самовнимания, мы рассматриваем три желательных условия.

Один из них — общая вычислительная сложность на слой. Другой — объем вычислений, которые можно распараллелить, измеряемый минимальным количеством требуемых последовательных операций.

Третий — длина пути между долгосрочными зависимостями в сети. Изучение долгосрочных зависимостей является ключевой проблемой во многих задачах преобразования последовательностей. Одним из ключевых факторов, влияющих на способность изучать такие зависимости, является длина путей, которые должны пройти прямые и обратные сигналы в сети. Чем короче эти пути между любой комбинацией позиций во входных и выходных последовательностях, тем легче изучать долгосрочные зависимости [12]. Поэтому мы также сравниваем максимальную длину пути между любыми двумя входными и выходными позициями в сетях, состоящих из разных типов слоев.

Как отмечено в Таблице 1, слой внутреннего внимания связывает все позиции с постоянным числом последовательно выполняемых операций, тогда как рекуррентный слой требует $O(n)$ последовательные операции. С точки зрения вычислительной сложности слои внутреннего внимания быстрее, чем рекуррентные слои, когда последовательность

длина n меньше размерности представления d, что чаще всего происходит с представлениями предложений, используемыми современными моделями в машинном переводе, такими как представления частей слов [38] и пар байтов [31]. Для улучшения вычислительной производительности для задач, включающих очень длинные последовательности, внутреннее внимание может быть ограничено рассмотрением только окрестности размером r в последовательности ввода, центрированной вокруг соответствующей позиции вывода. Это увеличит максимальную длину пути до $O(n/r)$. Мы планируем более подробно изучить этот подход в будущей работе.

Один сверточный слой с шириной ядра $k < n$ не соединяет все пары входных и выходных позиций. Для этого требуется стек $O(n/k)$ сверточные слои в случае смежных ядер, или $O(\log_k(n))$ в случае расширенных сверток [18], увеличивая длину самых длинных путей между любыми двумя позициями в сети. Сверточные слои, как правило, дороже, чем рекуррентные слои, в раз k. Отделимые свертки [6], однако, значительно уменьшают сложность, $O(k \cdot n \cdot d + n \cdot d^2)$. Даже с $k = n$ Однако сложность разделимой свертки равна комбинации слоя внутреннего внимания и слоя точечной прямой связи, подход, который мы используем в нашей модели.

В качестве побочного эффекта, самовнимание может дать более интерпретируемые модели. Мы проверяем распределение внимания из наших моделей и представляем и обсуждаем примеры в приложении. Мало того, что отдельные головы внимания явно учатся выполнять различные задачи, многие, по-видимому, демонстрируют поведение, связанное с синтаксической и семантической структурой предложений.

5 Обучение

В этом разделе описывается режим обучения наших моделей.

5.1 Обучающие данные и пакетирование

Мы обучались на стандартном наборе данных WMT 2014 English-German, состоящем из примерно 4,5 миллионов пар предложений. Предложения были закодированы с использованием кодирования пар байтов [3], которое имеет общий исходно-целевой словарь из примерно 37000 токенов. Для английского-французского мы использовали значительно больший набор данных WMT 2014 English-French, состоящий из 36M предложений и разделенных токенов на словарь из 32000 словосочетаний [38]. Пары предложений были объединены вместе по приблизительной длине последовательности. Каждая обучающая партия содержала набор пар предложений, содержащих примерно 25000 исходных токенов и 25000 целевых токенов.

5.2 Оборудование и график

Мы обучили наши модели на одной машине с 8 графическими процессорами NVIDIA P100. Для наших базовых моделей, использующих гиперпараметры, описанные в статье, каждый шаг обучения занимал около 0,4 секунды. Мы обучили базовые модели в общей сложности 100 000 шагов или 12 часов. Для наших больших моделей (описанных в нижней строке таблицы 3) время шага составило 1,0 секунды. Большие модели обучались в течение 300 000 шагов (3,5 дня).

5.3 Оптимизатор

Мы использовали оптимизатор Adam [20] с $\beta_1 = 0.9$, $\beta_2 = 0.98$ и $\epsilon = 10^{-9}$. Мы варьировали скорость обучения в ходе обучения по формуле:

$$lrate = d_{\text{модель}}^{-0.5} \cdot \min(\text{шаг\_число}^{-0.5}, \text{шаг\_число} \cdot \text{разогревать\_шаги}^{-1.5}) \quad (3)$$

Это соответствует линейному увеличению скорости обучения для первого $\textit{разогревать\_шаги}$ шаги обучения, и уменьшая его затем пропорционально обратному квадратному корню из номера шага. Мы использовали $\textit{разогревать\_шаги} = 4000$.

5.4 Регуляризация

В ходе обучения мы применяем три типа регуляризации:

Таблица 2: Transformer показывает лучшие результаты BLEU, чем предыдущие современные модели, на тестах newstest2014 с английского на немецкий и с английского на французский языки при значительно меньших затратах на обучение.

Модель	БЛЮ		Стоимость обучения (FLOP)	
	EN-DE	EN-FR	EN-DE	EN-FR
БайтНет [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
МО [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Ансамбль Deep-Att + PosUnk [39]		40.4		$8.0 \cdot 10^{20}$
Ансамбль GNMT + RL [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
Ансамбль ConvS2S [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Трансформер (базовая модель)	27,3	38,1	$3.3 \cdot 10^{18}$	
Трансформер (большой)	28,4	41,8	$2.3 \cdot 10^{19}$	

Остаточный отсевМы применяем dropout [33] к выходу каждого подслоя, прежде чем он будет добавлен к входу подслоя и нормализован. Кроме того, мы применяем dropout к суммам вложений и позиционных кодировок в стеках кодера и декодера. Для базовой модели мы используем скорость $P_{\text{уронить}}$=0.1.

СглаживаниеэтикетокВ ходе обучения мы использовали сглаживание меток стоимостиϵ_{ls}=0.1 [36] Это снижает растерянность, поскольку модель учится быть более неуверенной, но повышает точность и оценку BLEU.

6 результатов

6.1 Машинный перевод

В задаче перевода с английского на немецкий язык в рамках теста WMT 2014 модель большого трансформатора (Transformer (big) в таблице 2) превосходит лучшие ранее представленные модели (включая ансамбли) более чем на2.0 BLEU, устанавливающий новую современную оценку BLEU28.4.Конфигурация этой модели указана в нижней строке Таблицы 3. Обучение заняло3.5дней на8Графические процессоры P100. Даже наша базовая модель превосходит все ранее опубликованные модели и ансамбли, при этом затраты на обучение составляют лишь малую часть затрат на обучение любой из конкурирующих моделей.

В задании по переводу с английского на французский язык WMT 2014 наша большая модель набрала оценку BLEU 41.0, превзойдя все ранее опубликованные отдельные модели, менее чем за1/4стоимость обучения предыдущей современной модели. Модель Transformer (большая), обученная для английского-французского, использовала показатель отсева$P_{\text{уронить}}$=0.1,вместо0.3.

Для базовых моделей мы использовали одну модель, полученную путем усреднения последних 5 контрольных точек, которые были записаны с 10-минутными интервалами. Для больших моделей мы усреднили последние 20 контрольных точек. Мы использовали поиск пучка с размером пучка4и штраф за длинуα=0.6 [38]. Эти гиперпараметры были выбраны после экспериментов на наборе разработки. Мы установили максимальную длину выходных данных во время вывода на входную длину +50,но прекращайте раньше, если это возможно [38].

Таблица 2 суммирует наши результаты и сравнивает наше качество перевода и затраты на обучение с другими архитектурами моделей из литературы. Мы оцениваем количество операций с плавающей точкой, используемых для обучения модели, умножая время обучения, количество используемых графических процессоров и оценку поддерживаемой мощности одинарной точности с плавающей точкой каждого графического процессора[5].

6.2 Варианты моделей

Чтобы оценить важность различных компонентов Трансформера, мы варьировали нашу базовую модель различными способами, измеряя изменение производительности при переводе с английского на немецкий на

[5]Мы использовали значения 2,8, 3,7, 6,0 и 9,5 TFLOPS для K80, K40, M40 и P100 соответственно.

Таблица 3: Варианты архитектуры Transformer. Неперечисленные значения идентичны значениям базовой модели. Все метрики взяты из набора для разработки перевода с английского на немецкий, newstest2013. Перечисленные оплошности указаны на слово, согласно нашей кодировке пар байтов, и их не следует сравнивать с оплошностями на слово.

	N	$d_{\text{модель}}$	d_{ff}	h	d_k	d_v	$P_{\text{уронить}}$	ϵ_{ls}	тренироваться шаги	ЗПП (разраб)	БЛЮ (разраб)	параметры $\times 10^6$
база	6	512	2048	8	64	64	0.1	0.1	100K	4,92	25,8	65
(А)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4,91	25,8	
				32	16	16				5.01	25.4	
(Б)					16					5.16	25.1	58
					32					5.01	25.4	60
(С)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4,88	25,5	80
		256			32	32				5,75	24,5	28
		1024			128	128				4,66	26,0	168
			1024							5.12	25.4	53
			4096							4,75	26,2	90
(Д)							0.0			5,77	24,6	
							0.2			4,95	25,5	
								0.0		4,67	25,3	
								0.2		5,47	25,7	
(Э)	позиционное вложение вместо синусоид									4,92	25,7	
большой	6	1024	4096	16			0.3		300K	4.33	26.4	213

набор разработки, newstest2013. Мы использовали поиск луча, как описано в предыдущем разделе, но без усреднения контрольных точек. Мы представляем эти результаты в Таблице 3.

В строках Таблицы 3 (А) мы варьируем количество головок внимания и размеры ключа и значения внимания, сохраняя объем вычислений постоянным, как описано в Разделе 3.2.2. Хотя внимание с одной головкой на 0,9 BLEU хуже, чем наилучшая настройка, качество также падает при слишком большом количестве головок.

В строках таблицы 3 (В) мы видим, что уменьшение размера клавиши вниманияd_kухудшает качество модели. Это говорит о том, что определение совместимости — непростая задача, и что более сложная функция совместимости, чем скалярное произведение, может оказаться полезной. Далее в строках (С) и (D) мы наблюдаем, что, как и ожидалось, более крупные модели лучше, а исключение очень полезно для избежания переобучения. В строке (Е) мы заменяем наше синусоидальное позиционное кодирование на изученные позиционные вложения [9] и наблюдаем почти идентичные результаты для базовой модели.

6.3 Анализ английских избирательных округов

Чтобы оценить, может ли Transformer обобщаться на другие задачи, мы провели эксперименты по разбору английской выборки. Эта задача представляет определенные трудности: выходные данные подвержены сильным структурным ограничениям и значительно длиннее входных данных. Более того, модели RNN sequence-to-sequence не смогли достичь современных результатов в режимах с малыми данными [37].

Мы обучили 4-слойный трансформатор с$d_{\text{модель}}$=1024в разделе Wall Street Journal (WSJ) Penn Treebank [25], около 40 тыс. обучающих предложений. Мы также обучали его в полуконтролируемой обстановке, используя более крупные высокодостоверные и BerkleyParser корпуса из примерно 17 млн предложений [37]. Мы использовали словарь из 16 тыс. токенов для настройки только WSJ и словарь из 32 тыс. токенов для полуконтролируемой настройки.

Мы провели лишь небольшое количество экспериментов для выбора выпадающих значений, как внимания, так и остатка (раздел 5.4), скорости обучения и размера пучка на наборе разработки Раздела 22, все остальные параметры остались неизменными по сравнению с базовой моделью перевода с английского на немецкий. Во время вывода мы

Таблица 4: Transformer хорошо обобщает данные по английскому избирательным округам (результаты приведены в разделе 23 WSJ)

Парсер	Обучение	WSJ 23 F1
Виньялс и Кайзер эль др. (2014) [37]	Только WSJ, дискриминационный	88.3
Петров и др. (2006) [29]	Только WSJ, дискриминационный	90,4
Чжу и др. (2013) [40]	Только WSJ, дискриминационный	90,4
Дайер и др. (2016) [8]	Только WSJ, дискриминационный	91.7
Трансформатор (4 слоя)	Только WSJ, дискриминационный	91.3
Чжу и др. (2013) [40]	полуконтролируемый	91.3
Huang & Harper (2009) [14]	полуконтролируемый	91.3
McClosky et al. (2006) [26]	полуконтролируемый	92.1
Виньялс и Кайзер эль др. (2014) [37]	полуконтролируемый	92.1
Трансформатор (4 слоя)	полуконтролируемый	92.7
Луонг и др. (2015) [23]	многозадачность	93.0
Дайер и др. (2016) [8]	генеративный	93.3

увеличена максимальная длина выходных данных до входной длины +300.Мы использовали размер луча21иα=0. 3 как для WSJ, так и для полуконтролируемой среды.

Наши результаты в Таблице 4 показывают, что, несмотря на отсутствие настройки под конкретную задачу, наша модель работает на удивление хорошо, показывая лучшие результаты, чем все ранее представленные модели, за исключением грамматики рекуррентных нейронных сетей [8].

В отличие от моделей RNN «последовательность-в-последовательность» [37], Transformer превосходит Berkeley-Parser [29] даже при обучении только на обучающем наборе WSJ из 40 тыс. предложений.

7 Заключение

В этой работе мы представили Transformer — первую модель последовательной трансдукции, полностью основанную на внимании, в которой повторяющиеся слои, наиболее часто используемые в архитектурах кодера-декодера, заменены многоголовым внутренним вниманием.

Для задач перевода Transformer может обучаться значительно быстрее, чем архитектуры, основанные на рекуррентных или сверточных слоях. В задачах перевода с английского на немецкий WMT 2014 и с английского на французский WMT 2014 мы достигли нового уровня мастерства. В первой задаче наша лучшая модель превосходит даже все ранее представленные ансамбли.

Мы взволнованы будущим моделей, основанных на внимании, и планируем применять их к другим задачам. Мы планируем расширить Transformer на проблемы, связанные с модальностями ввода и вывода, отличными от текста, и исследовать локальные, ограниченные механизмы внимания для эффективной обработки больших объемов ввода и вывода, таких как изображения, аудио и видео. Сделать генерацию менее последовательной — еще одна наша исследовательская цель.

Код, который мы использовали для обучения и оценки наших моделей, доступен по адресуhttps://github.com/tensorflow/tensor2tensor.

БлагодарностиМы благодарны Нал Кальхбреннер и Стефану Гоувсу за их плодотворные комментарии, исправления и вдохновение.

Ссылки

[1] Джимми Лей Ба, Джейми Райан Кирос и Джеффри Э. Хинтон. Нормализация слоев.*Препринт arXiv arXiv:1607.06450*, 2016.

[2] Дмитрий Богданов, Кёнхён Чо и Йошуа Бенджио. Нейронный машинный перевод путем совместного обучения выравниванию и переводу.*CoRR*, абс/1409.0473, 2014.

[3] Денни Бритц, Анна Голди, Минь-Тханг Луонг и Куок В. Ле. Масштабное исследование архитектур нейронного машинного перевода.*CoRR*, абс/1703.03906, 2017.

[4] Цзяньпэн Чэн, Ли Донг и Мирелла Лапата. Долговременная кратковременная память — сети для машинного чтения.*Препринт arXiv arXiv:1601.06733*, 2016.

[5] Кёнхён Чо, Барт ван Мерриенбур, Чаглар Гульчехре, Фетхи Бугарес, Хольгер Швенк и Йошуа Бенджио. Изучение представлений фраз с использованием кодировщика-декодера rnn для статистического машинного перевода.*CoRR*, абс/1406.1078, 2014.

[6] Франсуа Шолле. Xception: Глубокое обучение с разделяемыми по глубине извилинами. *Препринт arXiv arXiv:1610.02357*, 2016.

[7] Джуньёнг Чунг, Чаглар Гюльчехре, Кёнхён Чо и Йошуа Бенджио. Эмпирическая оценка рекуррентных нейронных сетей с гейтом при моделировании последовательностей.*CoRR*, абс/1412.3555, 2014.

[8] Крис Дайер, Адхигуна Кункоро, Мигель Баллестерос и Ноа А. Смит. Рекуррентные нейронные сетевые грамматики. В*Труды NAACL*, 2016.

[9] Йонас Геринг, Майкл Аули, Дэвид Гранжье, Денис Яратс и Ян Н. Дофин. «Сверточная последовательность для обучения последовательностям».*препринт arXiv arXiv:1705.03122v2*, 2017.

[10] Алекс Грейвс. Генерация последовательностей с помощью рекуррентных нейронных сетей.*Препринт arXiv arXiv:1308.0850*, 2013.

[11] Каймин Хэ, Сянъюй Чжан, Шаоцин Жэнь и Цзянь Сунь. Глубокое остаточное обучение для распознавания изображений. В*Труды конференции IEEE по компьютерному зрению и распознаванию образов*, страницы 770–778, 2016.

[12] Зепп Хохрайтер, Йошуа Бенджио, Паоло Фраскони и Юрген Шмидхубер. Градиентный поток в рекуррентных сетях: сложность изучения долгосрочных зависимостей, 2001.

[13] Зепп Хохрайтер и Юрген Шмидхубер. Длинная кратковременная память.*Нейронные вычисления*, 9(8):1735–1780, 1997.

[14] Чжунцян Хуан и Мэри Харпер. Самообучающиеся грамматики PCFG со скрытыми аннотациями на разных языках. В*Труды конференции 2009 года по эмпирическим методам обработки естественного языка*, страницы 832–841. ACL, август 2009 г.

[15] Рафал Юзефович, Ориол Виньялс, Майк Шустер, Ноам Шазир и Йонгхуэй Ву. Исследование пределов моделирования языка.*Препринт arXiv arXiv:1602.02410*, 2016.

[16] Лукаш Кайзер и Сами Бенгио. Может ли активная память заменить внимание? В*Достижения в области нейронных систем обработки информации (NIPS)*, 2016.

[17] Лукаш Кайзер и Илья Суцкевер. Нейронные графические процессоры обучаются алгоритмам. В*Международная конференция по обучению репрезентациям (ICLR)*, 2016.

[18] Нал Кальчбреннер, Лассе Эспехольт, Карен Симонян, Аарон ван ден Оорд, Алекс Грейвс и Корай Кавукчуоглу. Нейронный машинный перевод в линейном времени.*препринт arXiv arXiv:1610.10099v2* , 2017.

[19] Юн Ким, Карл Дентон, Луонг Хоанг и Александр М. Раш. Структурированные сети внимания. В *Международная конференция по обучению репрезентациям*, 2017.

[20] Дидерик Кингма и Джимми Ба. Адам: Метод стохастической оптимизации. В*МЦЛР*, 2015.

[21] Алексей Кучаев и Борис Гинзбург. Факторизационные трюки для сетей LSTM.*Препринт arXiv arXiv:1703.10722*, 2017.

[22] Чжоухан Линь, Минвэй Фэн, Цицерон Ногейра душ Сантуш, Мо Ю, Бин Сян, Боуэн Чжоу и Йошуа Бенжио. Структурированное встраивание предложений, ориентированных на самообслуживание.*Препринт arXiv arXiv:1703.03130*, 2017.

[23] Минь-Тханг Луонг, Куок В. Ле, Илья Суцкевер, Ориол Виньялс и Лукаш Кайзер. Многозадачная последовательность для последовательного обучения.*Препринт arXiv arXiv:1511.06114*, 2015.

[24] Минь-Тханг Луонг, Хиеу Фам и Кристофер Д. Мэннинг. Эффективные подходы к нейронному машинному переводу на основе внимания.*Препринт arXiv arXiv:1508.04025*, 2015.

[25] Митчелл П. Маркус, Мэри Энн Марцинкевич и Беатрис Санторини. Создание большого аннотированного корпуса английского языка: Penn Treebank.*Компьютерная лингвистика*, 19(2):313–330, 1993.

[26] Дэвид МакКлоски, Юджин Чарняк и Марк Джонсон. Эффективное самообучение синтаксическому анализу. В *Труды конференции по технологиям естественного языка NAACL, Основная конференция*, страницы 152–159. ACL, июнь 2006 г.

[27] Анкур Парих, Оскар Тэкстрем, Дипаньян Дас и Якоб Ушкорейт. Разложимая модель внимания. В *Эмпирические методы обработки естественного языка*, 2016.

[28] Ромен Паулюс, Кайминг Сюн и Ричард Сохер. Глубоко укрепленная модель для абстрактного резюмирования.*Препринт arXiv arXiv:1705.04304*, 2017.

[29] Слав Петров, Леон Барретт, Ромен Тибо и Дэн Кляйн. Изучение точной, компактной и интерпретируемой древовидной аннотации. В*Труды 21-й Международной конференции по компьютерной лингвистике и 44-го ежегодного собрания ACL*, страницы 433–440. ACL, июль 2006 г.

[30] Офир Пресс и Лиор Вольф. Использование выходного встраивания для улучшения языковых моделей. *Препринт arXiv arXiv:1608.05859*, 2016.

[31] Рико Сеннрих, Барри Хэддоу и Александра Бирч. Нейронный машинный перевод редких слов с подсловными единицами.*Препринт arXiv arXiv:1508.07909*, 2015.

[32] Ноам Шазир, Азалия Мирхосейни, Кшиштоф Мазиаж, Энди Дэвис, Куок Ле, Джеффри Хинтон и Джефф Дин. Возмутительно большие нейронные сети: слой редко-контролируемой смеси экспертов. *Препринт arXiv arXiv:1701.06538*, 2017.

[33] Нитиш Шривастава, Джеффри Э. Хинтон, Алекс Крижевский, Илья Суцкевер и Руслан Салахутдинов. Dropout: простой способ предотвратить переобучение нейронных сетей.*Журнал исследований машинного обучения*, 15(1):1929–1958, 2014.

[34] Сайнбаяр Сухэ-Батор, Артур Шлам, Джейсон Уэстон и Роб Фергюс. Сквозные сети памяти. В редакторах К. Кортеса, Н. Д. Лоуренса, Д. Д. Ли, М. Сугиямы и Р. Гарнетта: *Достижения в области нейронных систем обработки информации 28*, страницы 2440–2448. Curran Associates, Inc., 2015.

[35] Илья Суцкевер, Ориол Виньялс и Куок В. В. Ле. Последовательность к последовательности обучения с нейронными сетями. В*Достижения в области нейронных систем обработки информации*, страницы 3104–3112, 2014.

[36] Кристиан Сегеди, Винсент Ванхоук, Сергей Иоффе, Джонатан Шленс и Збигнев Война. Переосмысление начальной архитектуры для компьютерного зрения.*CoRR*, абс/1512.00567, 2015.

[37] Виньялс и Кайзер, Ку, Петров, Суцкевер и Хинтон. Грамматика как иностранный язык. В *Достижения в области нейронных систем обработки информации*, 2015.

[38] Юнхуэй Ву, Майк Шустер, Чжифэн Чен, Куок В. Ле, Мохаммад Норузи, Вольфганг Машери, Максим Крикун, Юань Цао, Цинь Гао, Клаус Машери и др. Нейронная система машинного перевода Google: преодоление разрыва между человеческим и машинным переводом.*Препринт arXiv arXiv:1609.08144*, 2016.

[39] Цзе Чжоу, Ин Цао, Сюгуан Ван, Пэн Ли и Вэй Сюй. Глубокие рекуррентные модели с быстрыми связями для нейронного машинного перевода.*CoRR*, абс/1606.04199, 2016.

[40] Мухуа Чжу, Юэ Чжан, Вэньлян Чен, Минь Чжан и Цзинбо Чжу. Быстрый и точный анализ компонентов сдвиг-сокращение. В*Труды 51-го ежегодного собрания ACL (Том 1: Длинные доклады)*, страницы 434–443. ACL, август 2013 г.

Визуализации внимания

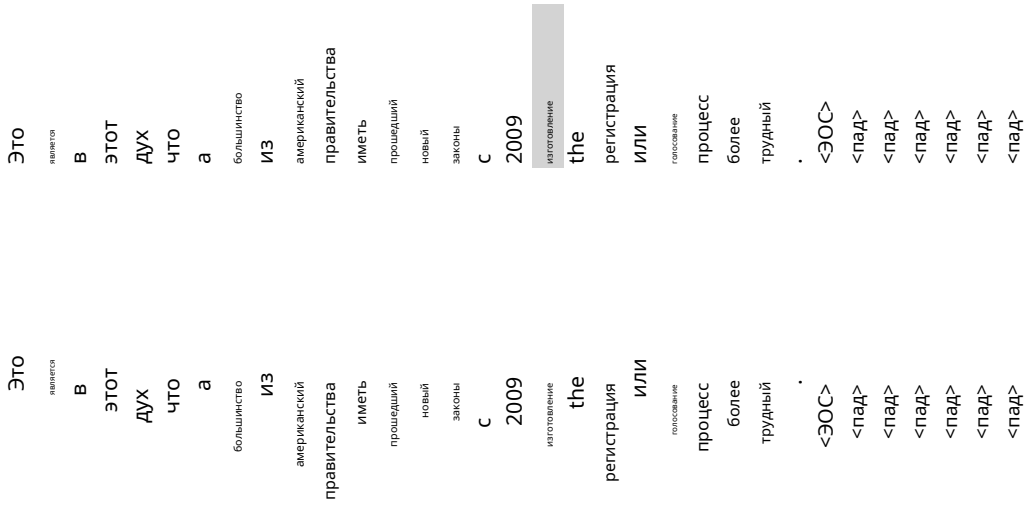
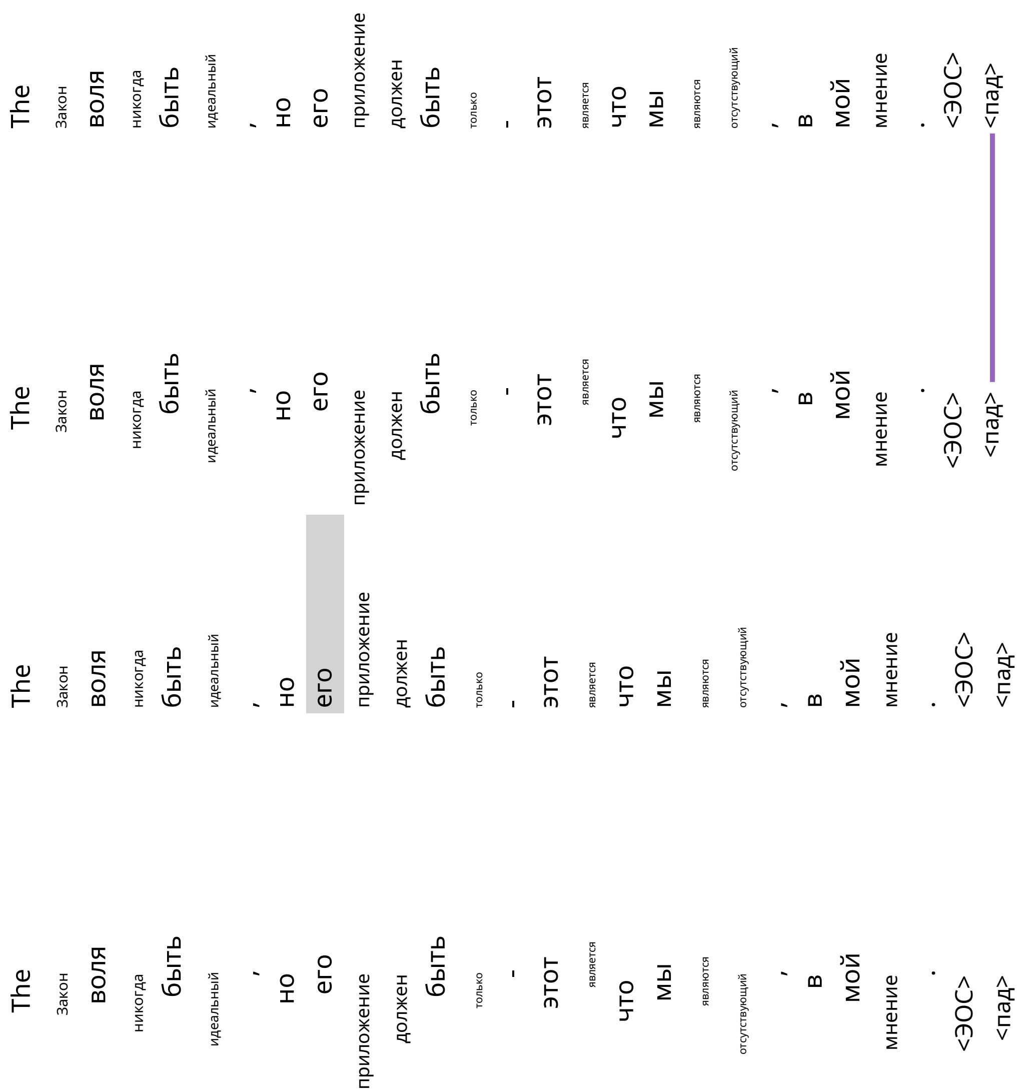
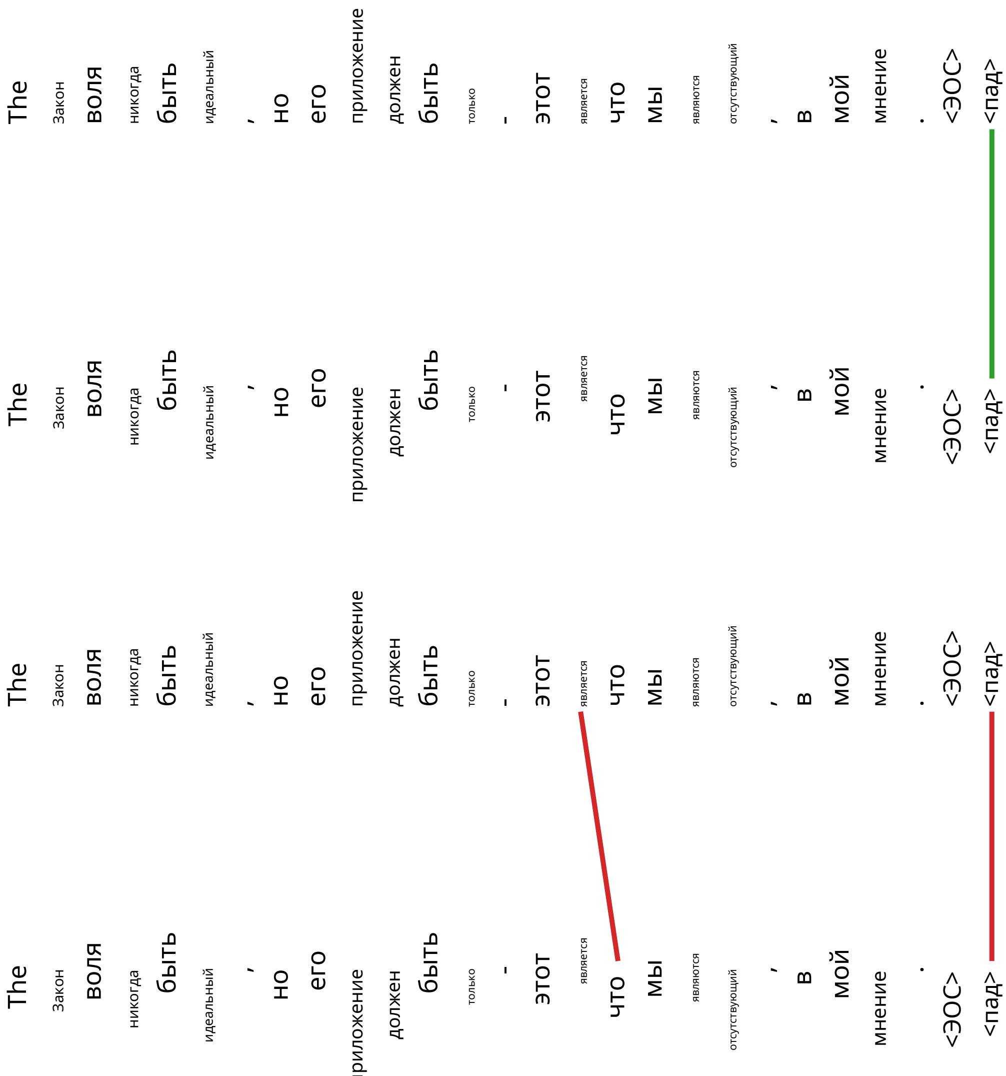


Рисунок 3: Пример механизма внимания, следующего за дальними зависимостями в самовнимании кодировщика в слое 5 из 6. Многие из головок внимания обращают внимание на дальнюю зависимость глагола «making», завершая фразу «making...more difficulty». Здесь показаны внимания только для слова «making». Разные цвета представляют разные головы. Лучше всего просматривать в цвете.



Фигу повторно 4: Твоэ ттнтиаон с alsoв лаэ-5 объявлено о ф, а, по-видимому, участвует в разрешении анафоры. Вверху: Фull в тuтиеоон для 5. Боттом: объявление Является олаТед в десять тио its от просто слова «its» для привлечения внимания возглавляет 5 s ай 6. Нотета ттхе ттнтиа сравеных-э с арп фи лтхи слово.



Фигуре 5: Многие из головок внимания демонстрируют поведение, которое, по-видимому, связано со структурой сентенции. Вэй ветвосусчахи рЭсаб ве, фрм т две разные головки от самовосприятия кодировщика атлавыт 5 из 6. Чте чеас слеарыл арйе к перфг для разных задач.