Внимание - это все, что вам нужно

Ашиш Васвани* Google Brain avaswani@google.com

Ноам Шазир* Google Brain noam@google.com

Ники Пармар* Исследования Google nikip@google.com

Якоб Ушкорейт* Исследования Google usz@google.com

Ллион Джонс* Google Research llion@google.com

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

Лукаш Кайзер* Мозг Google lukaszkaiser@google.com

Илья Полосухин* ‡ illia.polosukhin@gmail.com

Аннотация

Доминирующие модели передачи последовательности основаны на сложных рекуррентных или конволюционных нейронных сетях, включающих кодер и декодер. Самые эффективные модели также соединяют кодер и декодер через механизм внимания. Мы предлагаем новую простую сетевую архитектуру "Трансформатор", основанную исключительно на механизмах внимания и полностью отказывающуюся от рекуррентных и конволюционных механизмов. Эксперименты на двух задачах машинного перевода показали, что эти модели превосходят по качеству, при этом они лучше распараллеливаются и требуют значительно меньше времени на обучение. Наша модель достигает 28,4 BLEU в задаче англо-немецкого перевода WMT 2014, что превосходит лучшие результаты, полученные с помощью ансамблей, более чем на 2 BLEU. На задаче перевода с английского на французский в рамках WMT 2014 наша модель показывает новый рекордный результат BLEU для одной модели - 41,8 балла после обучения в течение 3,5 дней на восьми графических процессорах, что составляет лишь малую часть затрат на обучение лучших моделей из литературы. Мы показываем, что трансформатор хорошо обобщается на другие задачи, успешно применяя его для синтаксического разбора английского языка как на больших, так и на ограниченных обучающих данных.

*Равный вклад. Порядок перечисления случайный. Якоб предложил заменить RNN на самовнимание и начал работу по оценке этой идеи. Ашиш вместе с Иллией разработал и реализовал первые модели трансформеров и принимал самое активное участие во всех аспектах этой работы. Ноам предложил масштабируемый точечный продукт внимания, многоголовое внимание и представление позиции без параметров и стал вторым человеком, вовлеченным почти во все детали. Ники разработал, реализовал, настроил и оценил бесчисленные варианты моделей в нашей оригинальной кодовой базе и tensor2tensor. Ллион также экспериментировал с новыми вариантами моделей, отвечал

за нашу исходную кодовую базу, эффективные выводы и визуализации. Лукаш и Эйдан провели бесчисленное количество долгих дней, разрабатывая различные части и реализуя tensor2tensor, заменив нашу раннюю кодовую базу, значительно улучшив результаты и значительно ускорив наши исследования.

†Работа, выполненная во время работы в Google Brain.

‡Работа выполнена 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 нейронные сети с долговременной кратковременной памятью [13] и рекуррентные нейронные сети с воротами [7], прочно утвердились в качестве современных подходов к моделированию последовательностей и решению задач трансдукции, таких как моделирование языка и машинный перевод [35, 2, 5]. С тех пор многочисленные усилия продолжали расширять границы рекуррентных языковых моделей и архитектур кодеров-декодеров [38, 24, 15].

Рекуррентные модели обычно распределяют вычисления по символьным позициям входных и выходных последовательностей. Приравнивая позиции к шагам во времени вычислений, они генерируют последовательность скрытых состояний h_t , как функцию предыдущего скрытого состояния h_{t1} и входа для позиции t. Эта последовательная природа исключает распараллеливание внутри обучающих примеров, что становится критичным при больших длинах последовательностей, поскольку ограничения памяти ограничивают пакетную обработку примеров. В последних работах удалось добиться значительного повышения эффективности вычислений с помощью трюков факторизации [21] и условных вычислений [32], а также улучшить производительность модели в случае последних. Однако фундаментальное ограничение последовательных вычислений остается.

Механизмы внимания стали неотъемлемой частью моделей моделирования последовательности и преобразования в различных задачах, позволяя моделировать зависимости без учета их удаленности во входной или выходной последовательности [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 - архитектуру модели, которая отказывается от рекуррентности и полностью полагается на механизм внимания для выявления глобальных зависимостей между входом и выходом. Transformer позволяет значительно увеличить количество распараллеливаний и достичь нового уровня качества перевода после обучения в течение всего двенадцати часов на восьми графических процессорах P100.

2 Фон

Цель сокращения последовательных вычислений также лежит в основе моделей Extended Neural GPU [16], ByteNet [18] и ConvS2S [9], в которых в качестве базового строительного блока используются сверточные нейронные сети, параллельно вычисляющие скрытые представления для всех входных и выходных позиций. В этих моделях количество операций, необходимых для связи сигналов с двух произвольных входных или выходных позиций, растет с увеличением расстояния между позициями, линейно для ConvS2S и логарифмически для ByteNet. Это усложняет процесс обучения зависимостям между удаленными позициями [12]. В трансформере это число операций сокращается до постоянного, хотя и ценой снижения эффективного разрешения и з - з а усреднения позиций, взвешенных вниманием, - эффект, который мы нейтрализуем с помощью многоголового внимания, как описано в разделе 3.2.

Самовнимание, иногда называемое интравниманием, - это механизм внимания, связывающий различные позиции одной последовательности для вычисления представления этой последовательности. Самовнимание успешно используется в различных задачах, включая понимание прочитанного, абстрактное обобщение, текстовое вменение и обучение независимым от задачи представлениям предложений [4, 27, 28, 22].

Сети памяти "конец-в-конец" основаны на механизме рекуррентного внимания, а не на рекуррентной последовательности, и, как было показано, демонстрируют хорошие результаты в задачах на ответы на вопросы на простом языке и моделировании языка [34].

Однако, насколько нам известно, Transformer - это первая модель трансдукции, полностью полагающаяся на самовнимание для вычисления представлений своих входных и выходных данных без использования RNN, выровненных по последовательности, или свертки. В следующих разделах мы опишем Трансформатор, мотивируем самовнимание и обсудим его преимущества перед такими моделями, как [17, 18] и [9].

3 Архитектура модели

Большинство конкурентных моделей нейронной трансдукции последовательности имеют структуру кодер-декодер [5, 2, 35]. Здесь кодер отображает входную последовательность символьных представлений $(x_1, ..., x_n)$ в последовательность непрерывных представлений $\mathbf{z} = (z_1, ..., z_n)$. Учитывая $\mathbf{z}$, декодер генерирует выходную последовательность $(y_1, ..., y_m)$ символов по одному элементу за раз. На каждом шаге модель работает в режиме авторегрессии [10], используя ранее сгенерированные символы в качестве дополнительного входа при генерации следующего.

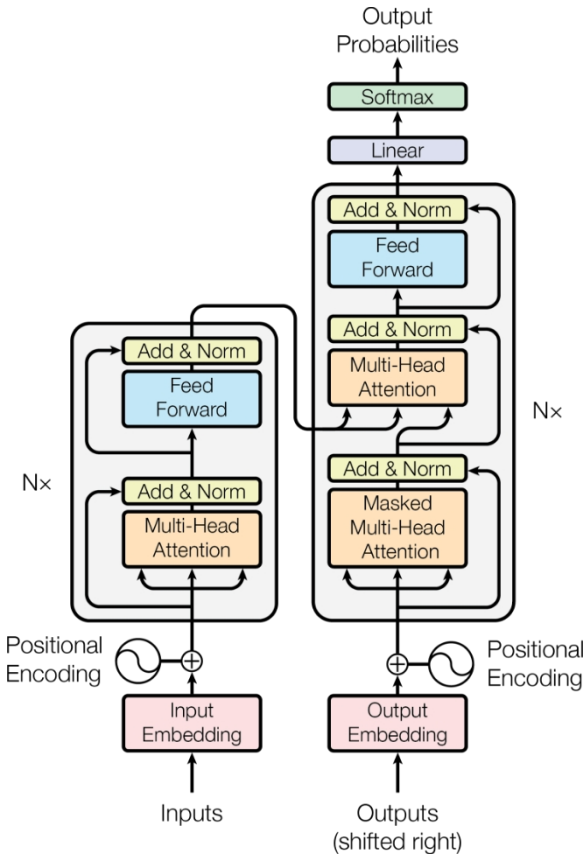


Рисунок 1: Трансформатор - модель архитектуры.

Трансформатор следует этой общей архитектуре, используя стековые самодостаточные и точечные, полностью связанные слои для кодера и декодера, показанные на левой и правой половинах рисунка 1 соответственно.

3.1 Стеки кодеров и декодеров

Кодировщик: Кодер состоит из стопки $N = 6$ одинаковых слоев. Каждый слой имеет два подуровня. Первый - это многоголовочный механизм самовнушения, а второй - простая позиционно-мудрая полносвязная сеть с прямой передачей. Мы используем остаточную связь [11] вокруг каждого из двух подслоев с последующей нормализацией слоев [1]. То есть выход каждого подслоя равен LayerNorm(x + Sublayer(x)), где Sublayer(x) - функция, реализуемая самим подслоем. Чтобы облегчить эти остаточные связи, все подуровни в модели, а также слои встраивания, производят выходы размерности $d_{\text{model}} = 512$.

Декодер: Декодер также состоит из стека $N = 6$ одинаковых слоев. В дополнение к двум подуровням в каждом слое кодера декодер вставляет третий подуровень, который выполняет многоголовое внимание над выходом стека кодеров. Как и в кодере, мы используем остаточные связи вокруг каждого из подуровней с последующей нормализацией слоев. Мы также модифицируем подуровень самовнимания в стеке декодера, чтобы не допустить внимания к последующим позициям. Эта маскировка в сочетании с тем фактом, что выходные вкрапления смещены на одну позицию, гарантирует, что предсказания для позиции i могут зависеть только от известных выходов в позициях меньше i.

3.2 Внимание

Функцию внимания можно описать как отображение запроса и набора пар ключ-значение на выход, где запрос, ключи, значения и выход - все векторы. Выходные данные

вычисляются как взвешенная сумма

Масштабное точечное произведение внимания Многоголовое внимание

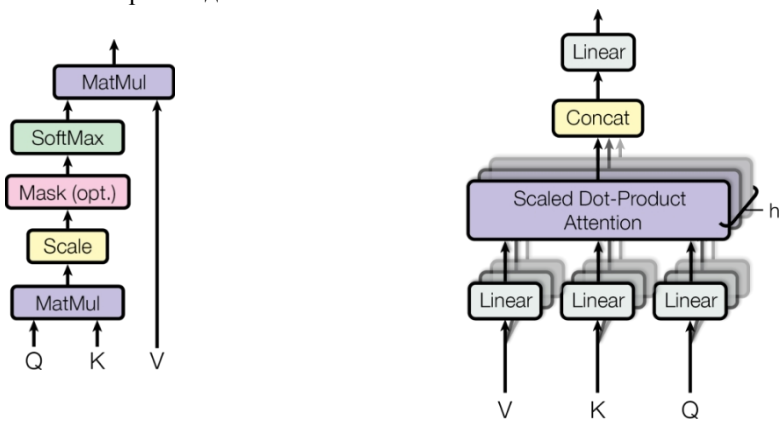


Рисунок 2: (слева) Масштабное точечное внимание. (справа) Многоголовое внимание состоит из нескольких параллельно работающих слоев внимания.

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

3.2.1 Масштабное точечное произведение Внимание

Мы называем наше особое внимание "Scaled Dot-Product Attention" (рис. 2). Входные данные состоят из
запросы и ключи размерности d_k, а также значения размерности d_v. Мы вычисляем точечные произведения
запрос со всеми ключами, d_k, и применяем функцию softmax для получения весов на разделите каждый из них на
значения.

На практике мы вычисляем функцию внимания одновременно на множестве запросов, объединенных в матрицу Q. Ключи и значения также объединены в матрицы K и V. Матрицу выходов мы вычисляем следующим образом:

$$\text{Внимание}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Две наиболее часто используемые функции внимания - это аддитивное внимание [2] и точечно-производное (мультипликативное) внимание. Точечно-продуктовое внимание идентично нашему алгоритму, за исключением коэффициента масштабирования $\frac{1}{\sqrt{d_k}}$. Аддитивное внимание вычисляет функцию совместимости с помощью сети с прямой связью с
один скрытый слой. Хотя теоретическая сложность этих двух методов схожа, на практике внимание к точечным продуктам гораздо быстрее и занимает меньше места, поскольку может быть реализовано с помощью оптимизированного кода умножения матриц.

Если при малых значениях d_k эти два механизма работают одинаково, то при больших значениях d_k аддитивное внимание превосходит внимание по точечным продуктам без масштабирования [3]. Мы подозреваем, что при больших значениях d_k точечные произведения возрастают по величине, выталкивая функцию softmax в области, где она имеет крайне малые градиенты [4]. Чтобы нейтрализовать этот эффект, мы масштабируем точечные произведения на $\frac{1}{\sqrt{d_k}}$.

3.2.2 Многоголовое внимание

Вместо того чтобы выполнять одну функцию внимания с d_{model}-мерными ключами,

значениями и запросами, мы сочли полезным линейно проецировать запросы, ключи и значения h раз с помощью различных, обученных линейных проекций на d_k , d_k и d_v измерения, соответственно. На каждой из этих спроецированных версий запросов, ключей и значений мы затем параллельно выполняем функцию внимания, получая d_v -мерных

[4]Чтобы проиллюстрировать, почему точечные произведения получаются большими, предположим, что компоненты q и k являются независимыми случайными величинами.

переменные со средним 0 и дисперсией 1. Тогда их точечное произведение, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, имеет среднее 0 и дисперсию d_k .

выходные значения. Они объединяются и снова проецируются, в результате чего получаются итоговые значения, как показано на рис. 2.

Многоголовое внимание позволяет модели совместно воспринимать информацию из разных подпространств репрезентации в разных позициях. При использовании одной головы внимания усреднение препятствует этому.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$
$$\text{где head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Где проекции - матрицы параметров $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

В данной работе мы используем $h = 8$ параллельных слоев внимания, или голов. Для каждого из них мы используем $d_k = d_v = d_{\text{model}}/h = 64$. Благодаря 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 кодера-декодера" запросы поступают с предыдущего слоя декодера, а ключи и значения памяти - с выхода кодера. Это позволяет каждой позиции в декодере обслуживать все позиции во входной последовательности. Это имитирует типичные механизмы внимания кодера-декодера в моделях "последовательность-последовательность", таких как [38, 2, 9].
- Кодировщик содержит слои самовнушения. В слое самовнимания все ключи, значения и запросы поступают из одного и того же места, в данном случае с выхода предыдущего слоя кодера. Каждая позиция в шифраторе может обслуживать все позиции в предыдущем слое шифратора.
- Аналогично, слои самовнимания в декодере позволяют каждой позиции в декодере обслуживать все позиции в декодере до этой позиции включительно. Чтобы сохранить свойство авторегрессии, нам нужно предотвратить левый поток информации в декодере. Мы реализуем это внутри масштабированного внимания точечного произведения, маскируя (устанавливая в) все значения на входе софтмакса, которые соответствуют незаконным связям. См. рисунок 2.

3.3 Позиционные сети с прямой передачей данных

Помимо подуровней внимания, каждый из слоев нашего кодера и декодера содержит полностью связанную сеть с прямой передачей, которая применяется к каждой позиции отдельно и идентично. Она состоит из двух линейных преобразований с активацией ReLU между ними.

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

Хотя линейные преобразования одинаковы для разных позиций, они используют разные параметры от слоя к слою. По-другому это можно описать как две свертки с размером ядра 1. Размерность входа и выхода составляет $d_{\text{model}} = 512$, а внутренний слой имеет размерность $d_{ff} = 2048$.

3.4 Вкрапления и Softmax

Как и в других моделях преобразования последовательности, мы используем выученные вкрапления для преобразования входных и выходных лексем в векторы размерности d_{model}.

Мы также используем обычную выученную линейную трансфор-мацию и функцию softmax для преобразования выхода декодера в предсказанные вероятности следующих лексем. В В нашей модели мы используем одну и ту же весовую матрицу между двумя слоями встраивания и предварительным слоем $\sqrt{\text{softmax}}$ линейное преобразование, аналогичное [30]. В слоях встраивания мы умножаем эти веса на d_{model}.

Таблица 1: Максимальная длина пути, сложность каждого слоя и минимальное количество последовательных операций для различных типов слоев. n - длина последовательности, d - размерность представления, k - размер ядра сверток и r - размер окрестности при ограниченном самовнимании.

Тип уровня	Сложность на уровне	Последовательный Операции	Максимальная длина пути
Самоуспокоение	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Повторяющийся	$O(n \cdot d)^2$	$O(n)$	$O(n)$
Конволюционный	$O(k \cdot n \cdot d)^2$	$O(1)$	$O(log_k(n))$
Самонаблюдение (ограничено)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Позиционное кодирование

Поскольку наша модель не содержит рекурсии и свертки, для того чтобы модель могла использовать порядок последовательности, мы должны ввести некоторую информацию об относительном или абсолютном положении лексем в последовательности. Для этого мы добавляем "позиционные кодировки" к входным вкраплениям в нижней части стеков кодера и декодера. Позиционные кодировки имеют ту же размерность d_{model} , что и эмбеддинги, так что их можно суммировать. Существует множество вариантов позиционных кодировок, как обучаемых, так и фиксированных [9].

В данной работе мы используем синусоидальные и косинусоидальные функции разных частот:

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

где pos - позиция, а i - размерность. То есть каждому измерению позиционного кодирования соответствует синусоида. Длины волн образуют геометрическую прогрессию от 2π до $10000 \cdot 2\pi$. Мы выбрали эту функцию, поскольку предположили, что она позволит модели легко научиться ориентироваться по относительным позициям, так как для любого фиксированного смещения k, PE_{pos+k} может быть представлена как линейная функция PE_{pos} .

Мы также экспериментировали с использованием вместо этого выученных позиционных вкраплений [9] и обнаружили, что эти две версии дают практически одинаковые результаты (см. таблицу 3, строка (E)). Мы выбрали синусоидальную версию, потому что она может позволить модели экстраполировать последовательности большей длины, чем те, которые встречались во время обучения.

4 Зачем нужно самовнушение

В этом разделе мы сравним различные аспекты слоев самовнимания с рекуррентными и сверточными слоями, обычно используемыми для отображения одной последовательности представлений символов переменной длины $(x_1, ..., x_n)$ на другую последовательность такой же длины $(z_1, ..., z_n)$, с $x_i, z_i \in \mathbb{R}^d$, такими как скрытый слой в типичном кодере или декодере последовательности. Мотивируя наше использование самовнимания, мы рассмотрим три желательных условия.

Один из них - это общая вычислительная сложность для каждого слоя. Другой - объем вычислений, который может быть распараллелен, измеряемый минимальным количеством необходимых последовательных операций.

Третье - длина пути между дальними зависимостями в сети. Изучение дальних зависимостей является ключевой проблемой во многих задачах преобразования последовательностей. Одним из ключевых факторов, влияющих на способность к обучению таким зависимостям, является длина путей, которые прямые и обратные сигналы должны преодолеть в сети. Чем короче эти пути между любой комбинацией позиций во входной и

выходной последовательностях, тем легче выучить дальние зависимости [12]. Поэтому мы также сравниваем максимальную длину пути между любыми двумя позициями на входе и выходе в сетях, состоящих из различных типов слоев.

Как показано в табл. 1, слой самонаблюдения соединяет все позиции с постоянным числом последовательно выполняемых операций, в то время как рекуррентный слой требует $O(n)$ последовательных операций. С точки зрения вычислительной сложности, самовтягивающиеся слои быстрее рекуррентных, когда последовательность

длина n меньше, чем размерность представления d, что чаще всего происходит с представлениями предложений, используемыми современными моделями машинного перевода, такими как слово-часть
[38] и представления байт-пар [31]. Для повышения производительности вычислений в задачах с очень длинными последовательностями можно ограничиться рассмотрением только окрестности размером r во входной последовательности с центром в соответствующей выходной позиции. Это увеличит максимальную длину пути до $O(n/r)$. Мы планируем исследовать этот подход в дальнейшей работе.

Один конволюционный слой с шириной ядра $k < n$ не соединяет все пары входных и выходных позиций. Для этого требуется стек из $O(n/k)$ конволюционных слоев в случае смежных ядер или $O(log_k (n))$ в случае расширенных конволюций [18], что увеличивает длину самых длинных путей между любыми двумя позициями в сети. Конволюционные слои обычно дороже рекуррентных в k раз. Разделяемые конволюции [6], однако, значительно снижают сложность, до $O(k\ n\ d + n\ d^2)$. Однако даже при $k = n$ сложность сепарабельной свертки равна комбинации слоя самовнушения и слоя точечной обратной связи - подходу, который мы используем в нашей модели.

В качестве побочного преимущества, самовнимание может дать более интерпретируемые модели. Мы проверили распределения внимания в наших моделях и представили и обсудили примеры в приложении. Мало того, что отдельные объекты внимания явно учатся выполнять разные задачи, многие из них демонстрируют поведение, связанное с синтаксической и семантической структурой предложений.

5 Обучение

В этом разделе описывается режим обучения наших моделей.

5.1 Обучающие данные и пакетная обработка

Мы обучались на стандартном англо-немецком наборе данных WMT 2014, состоящем из около 4,5 миллионов пар предложений. Предложения были закодированы с помощью кодировки байт-пар [3], в которой общий словарь источника и цели составляет около 37000 лексем. Для англо-французского языка мы использовали значительно больший англо-французский набор данных WMT 2014, состоящий из 36 миллионов предложений, и разделили лексемы на словарь из 32000 слов-фрагментов [38]. Пары предложений объединялись в группы по приблизительной длине последовательности. Каждая обучающая партия содержала набор пар предложений, содержащих примерно 25000 исходных лексем и 25000 целевых лексем.

5.2 Оборудование и расписание

Мы обучали 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. В общей сложности мы обучили базовые модели за 100 000 шагов или 12 часов. Для наших больших моделей (описаны 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 изменяли скорость обучения в процессе обучения по формуле:

$$lrate = d_{\text{модель}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps)^{-1.5} \quad (3)$$

Это соответствует линейному увеличению скорости обучения для первых шагов обучения $warmup\_steps$ и последующему ее уменьшению пропорционально обратному квадратному корню из номера шага. Мы использовали значение $warmup\_steps = 4000$.

5.4 Регуляризация

В процессе обучения мы используем три типа регуляризации:

Таблица 2: Трансформер достигает лучших показателей BLEU, чем предыдущие современные модели, в тестах newstest2014 "Английский - немецкий" и "Английский - французский" при меньших затратах на обучение.

Модель	BLEU		Стоимость обучения (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [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	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
Министерство энергетики [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Ансамбль Deep-Att + PosUnk [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
Ансамбль ConvS2S [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Трансформатор (базовая модель)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Трансформатор (большой)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Остаточное выпадение Мы применяем выпадение [33] к выходу каждого подуровня, прежде чем он будет добавлен к входу подуровня и нормализован. Кроме того, мы применяем отсев к суммам вкраплений и позиционных кодировок в стеках кодера и декодера. Для базовой модели мы используем коэффициент $P_{drop} = 0{,}1$.

Сглаживание меток Во время обучения мы использовали сглаживание меток со значением $\epsilon_{ls} = 0{,}1$ [36]. Это уменьшает недоумение, так как модель учится быть более неуверенной, но повышает точность и оценку BLEU.

6 Результаты

6.1 Машинный перевод

На задаче перевода с английского на немецкий язык WMT 2014 модель большого трансформатора (Transformer (big) в табл. 2) превосходит лучшие из ранее представленных моделей (включая ансамбли) более чем на 2,0 BLEU, устанавливая новый современный показатель BLEU, равный 28,4. Конфигурация этой модели приведена в нижней строке таблицы 3. Обучение заняло 3,5 дня на 8 графических процессорах P100. Даже наша базовая модель превосходит все ранее опубликованные модели и ансамбли, при этом затраты на обучение в разы меньше, чем у любой из конкурирующих моделей.

На задаче перевода с английского на французский в рамках WMT 2014 наша большая модель достигла оценки BLEU 41,0, превзойдя все ранее опубликованные одиночные модели, при этом затраты на обучение составили менее 1/4 от стоимости предыдущей современной модели. В модели Transformer (big), обученной для перевода с английского на французский, использовался коэффициент отсева $P_{drop} = 0{,}1$, а не 0,3.

Для базовых моделей мы использовали одну модель, полученную путем усреднения последних 5 контрольных точек, которые записывались с интервалом в 10 минут. Для больших моделей мы усредняли последние 20 контрольных точек. Мы использовали лучевой поиск с размером луча 4 и штрафом за длину $\alpha = 0{,}6$ [38]. Эти гиперпараметры были выбраны после экспериментов на наборе разработок. Мы установили максимальную длину выходного сигнала во время вывода на уровне длины входного сигнала + 50, но по возможности завершали работу раньше [38].

В таблице 2 приведены результаты и сравнение качества перевода и стоимости обучения с другими архитектурами моделей из литературы. Мы оцениваем количество операций с плавающей запятой, используемых для обучения модели, путем умножения времени обучения, количества используемых графических процессоров и оценки устойчивой производительности каждого графического процессора с плавающей запятой одинарной точности [5].

6.2 Разновидности 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: Вариации архитектуры Transformer. Неперечисленные значения идентичны значениям базовой модели. Все метрики получены на наборе разработок англо-немецкого перевода newstest2013. Перечисленные недоумения относятся к фрагментам слов в соответствии с нашей кодировкой байт-пар и не должны сравниваться с недоумениями в словах.

	N	dmodel	dff	h	d_k	d_v	$Pdrop$	ϵls	этапы	PPL (dev)	BLEU (dev)	params $\times 10^6$
база	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				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	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	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
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	позиционное вложение вместо синусоид									4.92	25.7	
больш	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

набор разработчиков, newstest2013. Мы использовали поиск по лучу, как описано в предыдущем разделе, но без усреднения по контрольным точкам. Эти результаты представлены в таблице 3.

В строках (A) таблицы 3 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 в лучшем случае, качество также падает с увеличением числа головок.

В строках (B) таблицы 3 мы видим, что уменьшение размера ключа внимания d_k ухудшает качество модели. Это говорит о том, что определить совместимость не так просто и что более сложная функция совместимости, чем точечное произведение, может быть полезной. Далее в строках (C) и (D) мы видим, что, как и ожидалось, большие модели лучше, а отсев очень помогает избежать чрезмерной подгонки. В строке (E) мы заменяем наше синусоидальное позиционное кодирование на выученные позиционные вкрапления [9] и получаем практически те же результаты, что и в базовой модели.

6.3 Разбор английского избирательного округа

Чтобы оценить возможность применения трансформатора для решения других задач, мы провели эксперименты по синтаксическому разбору английского языка. Эта задача представляет особые трудности: на выходной сигнал накладываются сильные структурные ограничения, и он значительно длиннее входного. Кроме того, модели последовательности RNN не смогли достичь лучших результатов в режиме малых данных [37].

Мы обучили 4-слойный трансформатор с $d_{model} = 1024$ на части Wall Street Journal (WSJ) из Penn Treebank [25], около 40 тыс. обучающих предложений. Мы также обучали его в полусупервизорном режиме, используя более крупные корпорации High Confidence и BerkleyParser, содержащие около 17 млн предложений [37]. Мы использовали словарный запас из 16 тыс. лексем только для WSJ и 32 тыс. лексем для полусупервизорного обучения.

Мы провели лишь небольшое количество экспериментов по выбору отсева, внимания и остатка (раздел 5.4), скорости обучения и размера пучка на наборе разработки Section 22, все остальные параметры остались неизменными по сравнению с базовой моделью перевода с английского на немецкий. В процессе вывода мы

Таблица 4: Трансформатор хорошо подходит для синтаксического разбора английского языка (результаты приведены в разделе 23 WSJ)

Парсер	Обучение	WSJ 23 F1
Виньялс и Кайзер и др. (2014) [37]	Только WSJ, дискриминационный	88.3
Петров и др. (2006) [29]	Только WSJ, дискриминационный	90.4
Zhu et al. (2013) [40]	Только WSJ, дискриминационный	90.4
Дайер и др. (2016) [8]	Только WSJ, дискриминационный	91.7
Трансформер (4 слоя)	Только WSJ, дискриминационный	91.3
Zhu et al. (2013) [40]	полунаблюдаемый	91.3
Хуанг и Харпер (2009) [14]	полунаблюдаемый	91.3
Макклоски и др. (2006) [26]	полунаблюдаемый	92.1
Виньялс и Кайзер и др. (2014) [37]	полунаблюдаемый	92.1
Трансформер (4 слоя)	полунаблюдаемый	92.7
Луонг и др. (2015) [23]	многозадачный	93.0
Дайер и др. (2016) [8]	генеративный	93.3

увеличили максимальную выходную длину до входной длины + 300. Мы использовали размер луча 21 и $\alpha = 0{,}3$.
как для WSJ, так и для полунаблюдаемой системы.

Результаты, приведенные в таблице 4, показывают, что, несмотря на отсутствие настройки под конкретную задачу, наша модель работает очень хорошо, получая лучшие результаты, чем все ранее представленные модели, за исключением рекуррентной нейросетевой грамматики [8].

В отличие от моделей последовательности RNN [37], трансформатор превосходит парсер Беркли [29] даже при обучении только на учебном наборе WSJ из 40 тыс. предложений.

7 Заключение

В этой работе мы представили Transformer, первую модель преобразования последовательности, полностью основанную на внимании, заменив рекуррентные слои, наиболее часто используемые в архитектурах кодеров-декодеров, на многоголовое самовнимание.

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

Мы с нетерпением ждем будущего моделей, основанных на внимании, и планируем применить их в других задачах. Мы планируем расширить трансформатор на задачи с другими модальностями ввода и вывода, кроме текста, и изучить механизмы локального, ограниченного внимания для эффективной обработки больших вводов и выводов, таких как изображения, аудио и видео. Еще одна цель нашего исследования - сделать генерацию менее последовательной.

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

Мы благодар**ны** Налу Калчбреннеру и Стефану Гоувсу за их плодотворные комментарии, исправления и вдохновение.

Ссылки

[1] Джимми Лей Ба, Джейми Райан Кирос и Джеффри Е Хинтон. Нормализация слоев.

arXiv препринт arXiv:1607.06450, 2016.

[2] Дмитрий Бахданов, Кюнгхюн Чо и Йошуа Бенгио. Нейронный машинный перевод с совместным обучением выравниванию и переводу. *CoRR*, abs/1409.0473, 2014.

[3] Денни Бритц, Анна Голди, Минь-Тханг Лыонг и Куок В. Ле. Массовое исследование нейронных архитектур машинного перевода. *CoRR*, abs/1703.03906, 2017.

[4] Цзяньпенг Ченг, Ли Дун и Мирелла Лапата. Сети долговременной памяти для машинного чтения. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Кюнгхюн Чо, Барт ван Мерриенбоер, Чаглар Гюльчехре, Фети Бугарес, Хольгер Швенк и Йошуа Бенгио. Обучение представлению фраз с помощью кодера-декодера rnn для статистического машинного перевода. *CoRR*, abs/1406.1078, 2014.

[6] Франсуа Шолле. Xception: Глубокое обучение с глубинными сепарабельными свертками. *arXiv препринт arXiv:1610.02357*, 2016.

[7] Чжун Юн Чун, Чаглар Гюльчехре, Кюнгхюн Чо и Йошуа Бенгио. Эмпирическая оценка рекуррентных нейронных сетей с регулировкой на моделировании последовательности. *CoRR*, abs/1412.3555, 2014.

[8] Крис Дайер, Адхигуна Кункоро, Мигель Баллестерос и Ной А. Смит. Рекуррентные нейросетевые грамматики. В *сборнике трудов NAACL*, 2016.

[9] Йонас Геринг, Майкл Аули, Давид Гранжье, Денис Ярац и Янн Н. Дофин. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

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

[11] Кайминг Хэ, Сяньюй Чжан, Шаоцин Рен и Цзянь Сунь. Глубокое остаточное обучение для распознавания изображений. В *материалах конференции IEEE по компьютерному зрению и распознаванию образов*, страницы 770-778, 2016.

[12] Сепп Хохрайтер, Йошуа Бенгио, Паоло Фраскони и Юрген Шмидхубер. Градиентный поток в рекуррентных сетях: трудности обучения долгосрочным зависимостям, 2001.

[13] Сепп Хохрайтер и Юрген Шмидхубер. Длительная кратковременная память. *Нейронные вычисления*, 9(8):1735-1780, 1997.

[14] Чжунцян Хуан и Мэри Харпер. Самообучение грамматик PCFG с использованием латентных аннотаций на разных языках. В *материалах конференции 2009 года по эмпирическим методам обработки естественного языка*, страницы 832-841. ACL, август 2009 года.

[15] Рафал Юзефович, Ориол Виньялс, Майк Шустер, Ноам Шазир и Йонгхуи Ву. Исследование пределов языкового моделирования. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Лукаш Кайзер и Сами Бенгио. Может ли активная память заменить внимание? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Лукаш Кайзер и Илья Суцкевер. Нейронные графические процессоры учат алгоритмы. *Международная конференция по изучению представлений (ICLR)*, 2016.

[18] Наль Калчбреннер, Лассе Эспехольт, Карен Симонян, Аарон ван ден Оорд, Алекс Грейвс и Ко-Рей Кавуккуоглу. Нейронный машинный перевод за линейное время. *arXiv препринт arXiv:1610.10099v2*, 2017.

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

[20] Дидерик Кингма и Джимми Ба. Adam: метод стохастической оптимизации. В *сборнике ICLR*, 2015.

[21] Алексей Кучаев, Борис Гинзбург. Трюки факторизации для LSTM-сетей. *arXiv препринт arXiv:1703.10722*, 2017.

[22] Чжоухань Линь, Минвэй Фэн, Сисеро Ногейра дос Сантос, Мо Юй, Бин Сян, Боуэн Чжоу и Йошуа Бенгио. Структурированное самоаттентивное встраивание предложений. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Минь-Тханг Лыонг, Куок В. Ле, Илья Суцкевер, Ориол Виньялс и Лукаш Кайзер. Многозадачное обучение по принципу "последовательность к последовательности".

arXiv preprint arXiv:1511.06114, 2015.

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

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Создание большого аннотированного корпуса английского языка: The penn treebank. *Вычислительная лингвистика*, 19(2):313-330, 1993.

[26] Дэвид Макклоски, Юджин Чарниак и Марк Джонсон. Эффективное самообучение для синтаксического анализа. В *материалах конференции по технологиям человеческого языка NAACL, основная конференция*, страницы 152-159. ACL, июнь 2006 года.

[27] Анкур Парих, Оскар Тэкстрем, Дипанджан Дас и Якоб Ушкорейт. Декомпозируемая модель внимания. В сборнике *"Эмпирические методы обработки естественного языка"*, 2016.

[28] Ромен Паулюс, Кайминг Сьонг и Ричард Сошер. Глубокая усиленная модель для абстрактного обобщения. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Слав Петров, Леон Барретт, Ромен Тибо и Дэн Клейн. Обучение точным, компактным и интерпретируемым аннотациям деревьев. В *материалах 21-й Международной конференции по вычислительной лингвистике и 44-й ежегодной встречи ACL*, страницы 433-440. ACL, июль 2006 года.

[30] Офир Пресс и Лиор Вольф. Использование вкрапления вывода для улучшения языковых моделей. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Рико Сеннрих, Барри Хэддоу и Александра Бирч. Нейронный машинный перевод редких слов с помощью подсловных единиц. *arXiv препринт arXiv:1508.07909*, 2015.

[32] Ноам Шазир, Азалия Мирхосейни, Кшиштоф Мазиарж, Энди Дэвис, Квок Ле, Джеффри Хинтон и Джефф Дин. Возмутительно большие нейронные сети: Слой смеси экспертов с разреженным распределением. *arXiv preprint arXiv:1701.06538*, 2017.

[33] Нитиш Шривастава, Джеффри Хинтон, Алекс Крижевский, Илья Суцкевер и Руслан Салахутди-нов. Dropout: простой способ предотвратить перебор нейронных сетей. *Journal of Machine Learning Research*, 15(1):1929-1958, 2014.

[34] Сайнбаяр Сухбаатар, Артур Сзлам, Джейсон Уэстон и Роб Фергюс. Сети сквозной памяти. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440-2448. Curran Associates, Inc., 2015.

[35] Илья Суцкевер, Ориол Виньялс и Куок В.В. Ле. Обучение от последовательности к последовательности с помощью нейронных сетей. In *Advances in Neural Information Processing Systems*, pages 3104-3112, 2014.

[36] Кристиан Сегеди, Винсент Ванхуке, Сергей Иоффе, Джонатон Шленс и Збигнев Война. Переосмысление архитектуры концепции для компьютерного зрения. *CoRR*, abs/1512.00567, 2015.

[37] Виньялс и Кайзер, Ку, Петров, Суцкевер и Хинтон. Грамматика как иностранный язык. В *Advances in Neural Information Processing Systems*, 2015.

[38] Йонгуй Ву, Майк Шустер, Жифенг Чен, Куок В Ле, Мохаммад Норузи, Вольфганг Мачерей, Максим Крикун, Юань Цао, Цинь Гао, Клаус Мачерей и др. Нейронная система машинного перевода Google: Преодоление разрыва между человеческим и машинным переводом. *arXiv preprint arXiv:1609.08144*, 2016.

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

[40] Мухуа Чжу, Юэ Чжан, Вэньлян Чэнь, Минь Чжан и Цзинбо Чжу. Быстрый и точный сдвигово-редуктивный синтаксический разбор. В *материалах 51-й ежегодной встречи ACL (том 1: длинные доклады)*, страницы 434-443. ACL, август 2013 года.

Визуализации внимания

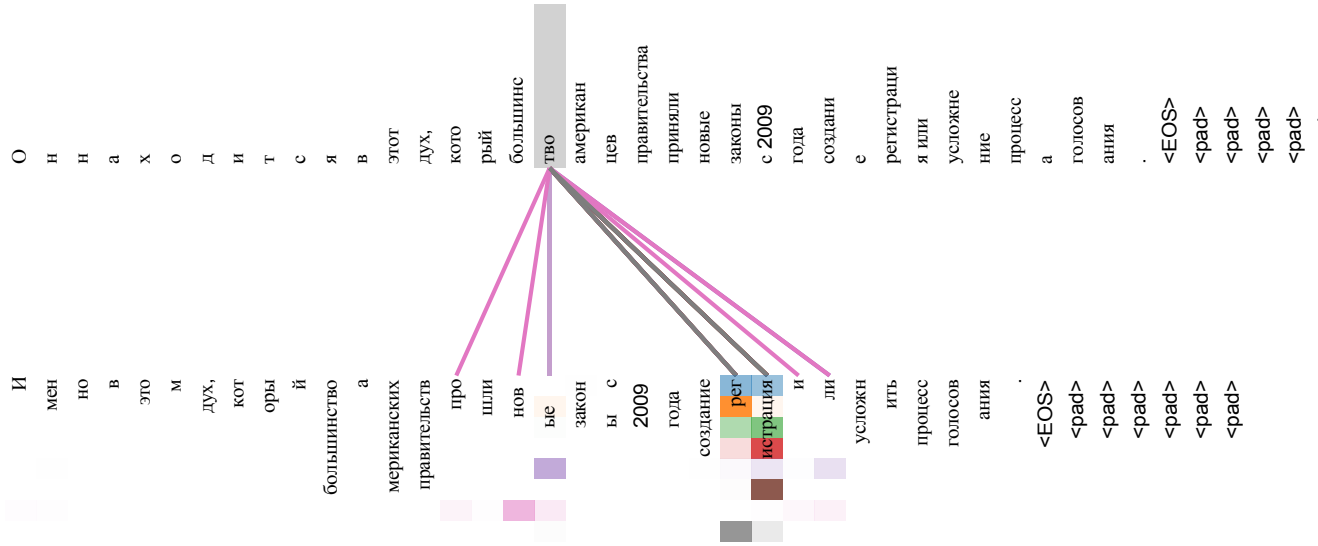


Рисунок 3. Пример механизма внимания, следующего за дальними зависимостями в самовнимании кодера в слое 5 из 6. Многие головки внимания посещают дальнюю зависимость от глагола 'making', завершая фразу 'making...more difficult'. Здесь показано внимание только к слову 'making'. Разные цвета обозначают разные головы. Лучше всего смотреть в цвете.

Рисунок 4: Две головы внимания, также в слое 5 из 6, очевидно, вовлеченные в разрешение анафоры. Вверху: полное внимание для головы 5. Внизу: Изолированные аттенции только слова 'its' для головок внимания 5 и 6. Обратите внимание, что аттенции очень резкие для этого слова.

Закон никогда не быть идеальным , но его применение следует быть справедливым - Вот кто мы такие отсутствует , на мой взгляд . <EOS> <pad>

Закон никогда не будет совершенным, но его применение должно быть справедливым - Вот кто мы такие отсутствует, в моём мнение. <EOS> <pad>

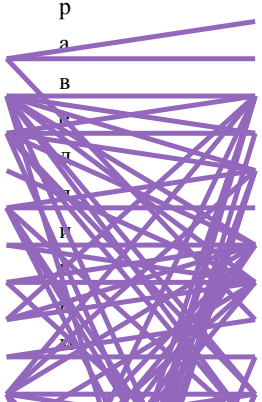
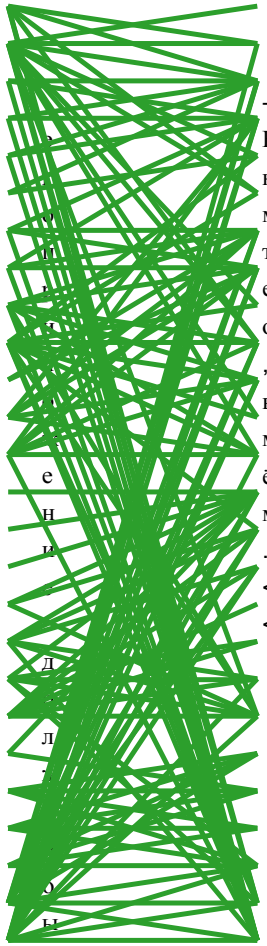


Рисунок 5: Многие из головок внимания демонстрируют поведение, которое, по-видимому, связано со структурой предложения. Мы приводим два таких примера из двух разных головок самовнимания кодера на уровне 5 из 6. Эти головки явно научились выполнять разные задачи.

Закон никогда не быть идеальным , но его применение следует быть справедливым - Вот кто мы такие отсутствует , на мой взгляд .

<EOS>

<pad>



Закон никогда не будет совершенным, но применение должно быть справедливым - Вот кто мы такие отсутствует, в моём мнение.

<EOS>

<pad>