ЗАДАНИЕ 2.

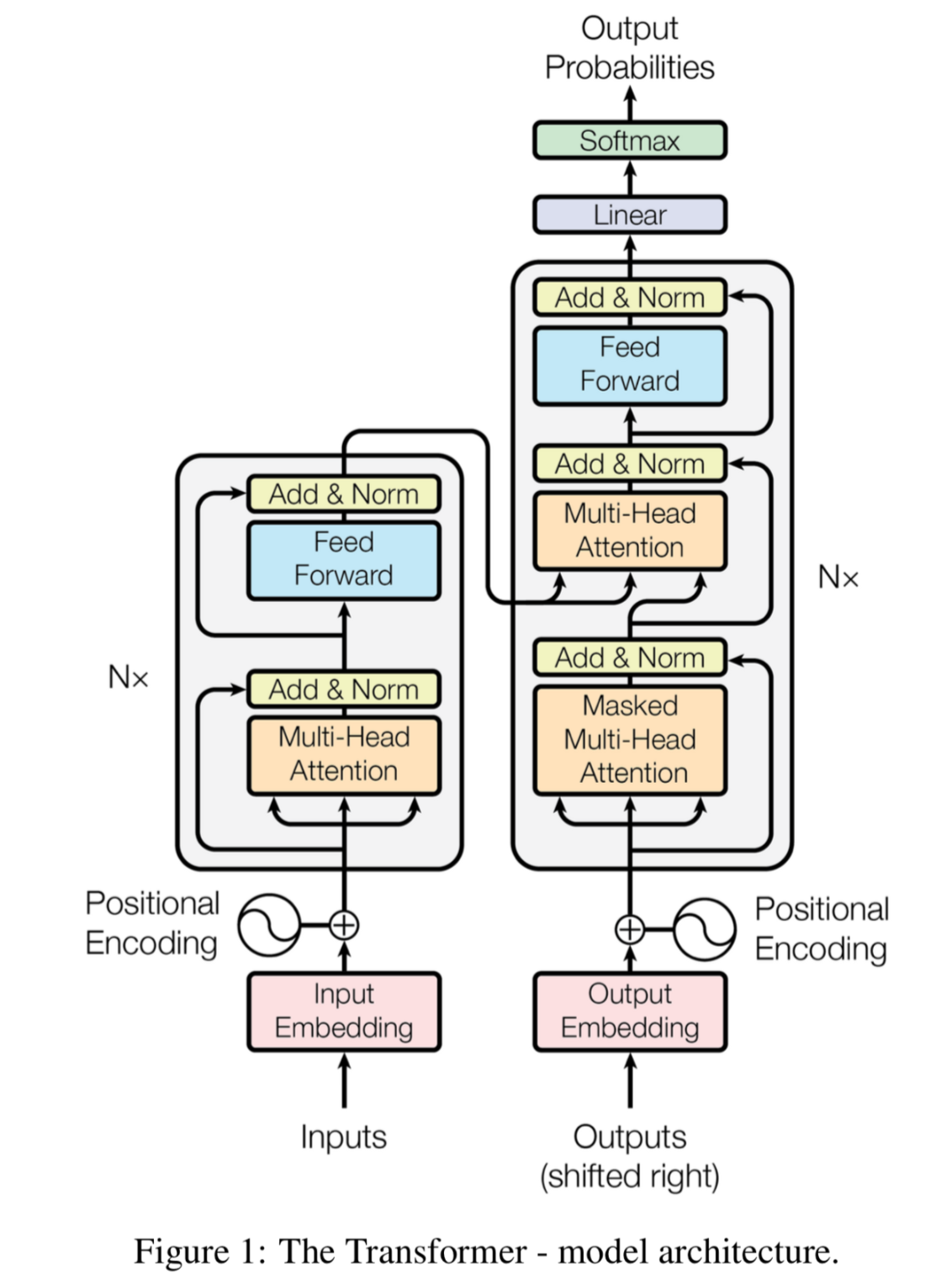
Сделайте краткий обзор какой-нибудь научной работы посвященной тому или иному алгоритму нейронных сетей, который не рассматривался на курсе. Проведите анализ: Чем отличается выбранная вами на рассмотрение архитектура нейронной сети от других архитектур? В чем плюсы и минусы данной архитектуры? Какие могут возникнуть трудности при применении данной архитектуры на практике?

Оригинал статьи: <https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04>

**What is a Transformer?**

(Введение в сети типа Transformer)

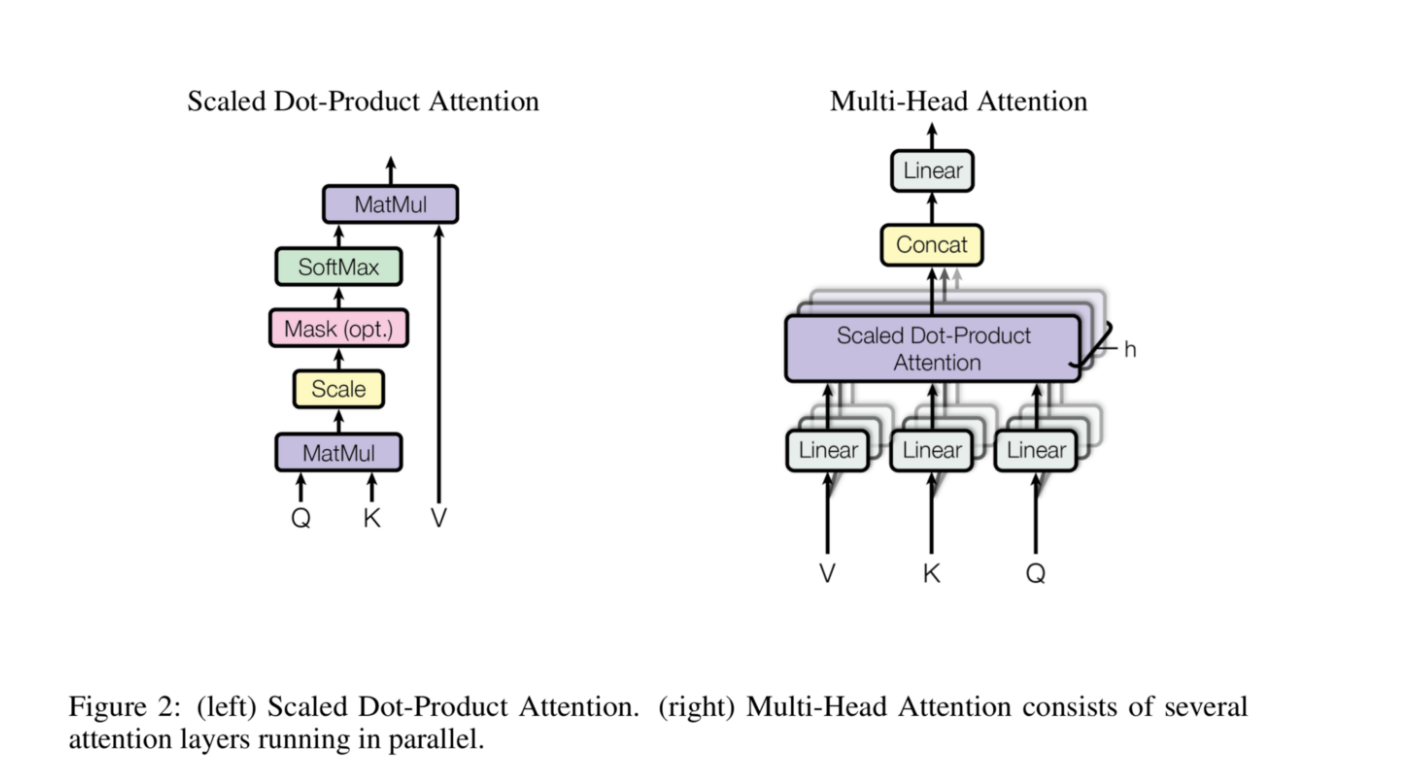
**Архитектура сети Transformer**



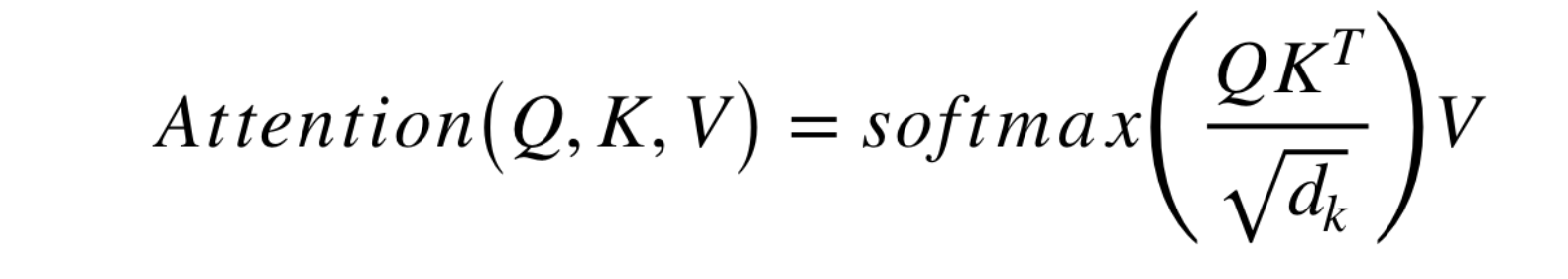
Энкодер находится слева, а декодер - справа. И энкодер, и декодер состоят из модулей, которые могут быть несколько раз состыкованы друг с другом, что описано Nx символом на рисунке. Мы видим, что модули состоят в основном из слоев Multi-Head Attention и Feed Forward. Входы и выходы (целевые предложения) сначала вставляются в n-мерное пространство(n-мерный вектор), поскольку мы не можем использовать строки напрямую.

Одна небольшая, но важная часть модели - это позиционное кодирование разных слов. Поскольку у нас нет рекуррентных сетей, которые могут запомнить, как последовательности вводятся в модель, нам нужно каким-то образом присвоить каждому слову / части в нашей последовательности относительное положение, поскольку последовательность зависит от порядка ее элементов. Эти позиции добавляются к встроенному представлению (n-мерному вектору) каждого слова.

Давайте подробнее рассмотрим эти блоки Multi-Head Attention в модели:

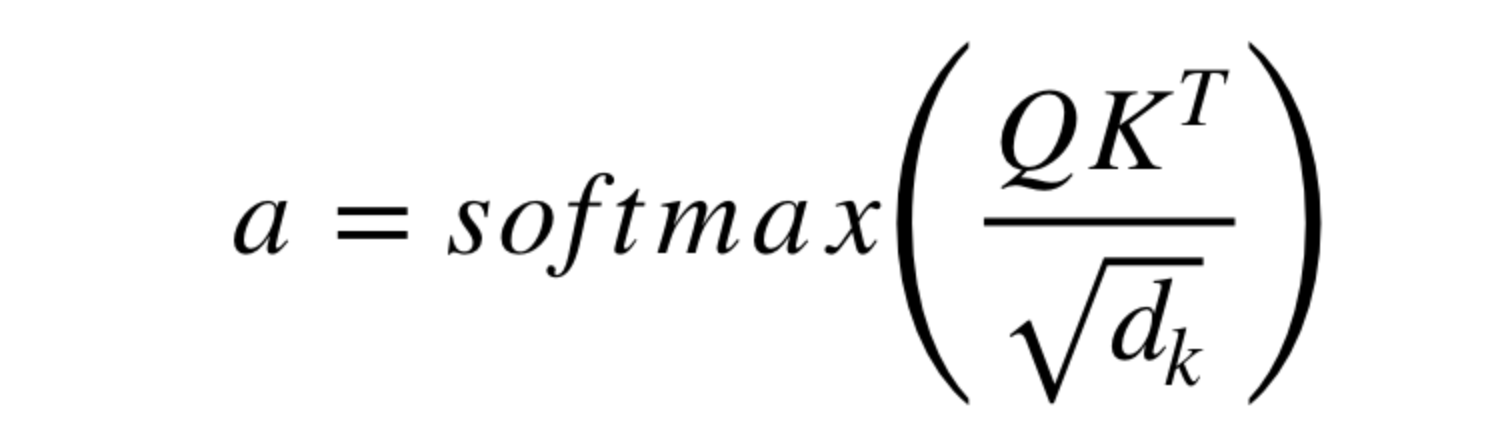


Начнем с описания механизма внимания слева. Это не очень сложно и описывается следующим уравнением:



Q - это матрица, содержащая запрос (векторное представление одного слова в последовательности), K - все ключи (векторные представления всех слов в последовательности), а V - значения, которые снова являются векторными представлениями всех слова в последовательности. Для энкодера и декодера, модулей внимания с несколькими заголовками, V состоит из той же последовательности слов, что и Q. Однако для модуля внимания, который принимает во внимание последовательности энкодера и декодера, V отличается от последовательности, представленной Q.

Чтобы немного упростить это, мы могли бы сказать, что значения в V умножаются и суммируются с некоторыми весами внимания a, где наши веса определяются следующим образом:



Это означает, что веса a определяются тем, как на каждое слово последовательности (представленной Q) влияют все другие слова в последовательности (представленные K). Кроме того, функция SoftMax применяется к весам a, чтобы иметь распределение между 0 и 1. Эти веса затем применяются ко всем словам в последовательности, которые вводятся в V (те же векторы, что и Q для энкодера и декодера, но разные для модуля, для которого имеются входы энкодера и декодера).

На рисунке справа показано, как этот механизм внимания можно распараллелить на несколько механизмов, которые можно использовать бок о бок. Механизм внимания повторяется несколько раз с линейными проекциями Q, K и V. Это позволяет системе учиться на различных представлениях Q, K и V, что полезно для модели. Эти линейные представления выполняются путем умножения Q, K и V на весовые матрицы W, полученные во время обучения.

Эти матрицы Q, K и V различны для каждой позиции модулей внимания в структуре в зависимости от того, находятся ли они в энкодере, декодере или промежуточном слое между энкодером и декодером. Причина в том, что мы хотим обработать либо всю входную последовательность энкодера, либо часть входной последовательности декодера. Модуль внимания с несколькими заголовками, который соединяет энкодер и декодер, будет следить за тем, чтобы входная последовательность энкодера учитывалась вместе с входной последовательностью декодера до заданной позиции.

После заголовков с множественным вниманием(multi-attention heads) в энкодере и декодере у нас есть точечный слой прямой связи. Эта небольшая сеть прямой связи имеет идентичные параметры для каждой позиции, которые можно описать как отдельное идентичное линейное преобразование каждого элемента из данной последовательности.

**Плюсы и минусы данной архитектуры Transformer**

**Плюсы**

* Она не делает никаких предположений о временных / пространственных отношениях между данными. Это идеально подходит для обработки набора объектов
* Выходы с каждого слоя можно вычислять параллельно, а не последовательно, как RNN.
* Отдаленные друг от друга элементы в предложении могут влиять на вывод друг друга, не проходя через множество RNN-шагов или слоев свертки
* Она может обучаться на зависимостях между данными на большом диапазоне. Это проблема для многих задач последовательности.

**Минусы**

* Для временного ряда выход (output) для каждого временного шага рассчитывается исходя из *всей истории,* а не только из входных данных и текущего скрытого состояния. Это *может* быть менее эффективно.
* Если вход(input) *действительно* имеет временное / пространственное соотношение, как текст, должно быть добавлено некоторое позиционное кодирование или модель будет видеть груду слов.

**Какие могут возникнуть трудности при применении данной архитектуры на практике**

При применении на практике чаще всего возникают повышенные требования к памяти компьютера и мощности процессора, т.к. обычно используется большой набор данных.