Механизмы внимания и чем их едят.

Зададимся вопросом, что такое внимание? Обычно можно найти определение, что внимание - это сосредоточенность человека на определенной сущности, под сущностью можно понимать что угодно, действие, объект, мышление и тд. Сама по себе сущность внимания очень обсуждаемая, есть очень много видов внимания, от чего оно зависит и тд, однако из многочисленных видов внимания(произвольное непроизвольное и тд), мы рассмотрим пример механизма внимания направленную на одного человека, так же далее будут приведены аналогии с концентрацией, как одно из мер внимания, и математическим представлениями внимания в этом сфере машинного обучения.

Начнем с главной идеи механизма внимания в машинном обучении. Например у нас есть текст: 'Надо сдавать лабораторные до срока выполнения". Возьмем второе слово "сдавать", чтобы показать близость между словами учитывая контекст, мы можем токенизировать слова данного предложения, и каждому слову присвоить вектор, например в данном случае используем можно использовать тип кодировки one-hot, так как объем словаря маленький, далее мы считаем скалярное произведение с другими словами(включая себя же), получается 6-мерный вектор, состоящий из весов, вес - обозначает значимость слов в контексте, для удобства нормализуем эти вектора с использованием функции softmax.

$$w_i' = \mathrm{softmax}(w_1, \ldots, w_7) = e^{w_i}/(e^{w_1} + \ldots + e^{w_7}).$$

Таким образом, у нас получаются нормализированные вектора весов, которые пригодятся в следующей работе.

Далее рассмотрим простейшие структуры енкодера и декодера:
Енкодер - это рекуррентная сеть, которая принимает на вход текст, и после работы в финальной ячейке хранит вектор скрытого состояния, который накопил в себе обо всем source-предложении (context vector). Этот вектор поступает как первый вектор скрытого состояние в декодер.
Декодер - тоже реккуретная сеть, у который первый вектор скрытого состояние - вектор из енкодера. Затем на вход первой ячейки декодера подаётся служебный токен **(begin of sentence). На выходе этой ячейки сеть обучают выдавать слово-перевод "кот" (см. рисунок). Для этого выходы ячеек пропускают через линейный слой (fc) с числом нейронов равным числу слов в словаре. Затем softmax-функция (sm), выдаёт "вероятности" слов из которых выбирается номер максимальной (argmax). Вектор полученного слова "кот" передаётся на вход второй ячейки и т.д. пока не получится служебный токен <EOS> (end of sentence).**

Valpha - вектор скрытого состояние на ячейке альфа в енкодере, u - в декодере, следующий вектор скрытого состояние считается как его сумма со взвешенным

скрытым состоянием энкодера

$$\mathbf{u}'_\alpha = \sum_\beta w_{\alpha\beta}\,\mathbf{v}_\beta, \qquad \sum_\beta w_{\alpha\beta} = 1, \qquad w_{\alpha\beta} = f\bigl(\mathbf{u}_\alpha, \mathbf{v}_\beta\bigr).$$

Веса внимания к скрытым состояниям декодера определяются следующим образом

$$w_{\alpha\beta} = \mathrm{softmax}\bigl(\mu\,\mathbf{u}_\alpha \mathbf{v}_\beta\bigr) = \frac{e^{\mu\,\mathbf{u}_\alpha \mathbf{v}_\beta}}{\sum_\gamma e^{\mu\,\mathbf{u}_\alpha \mathbf{v}_\gamma}}.$$

Функция внимания.
Пусть есть три матрицы: матрица Q запросов **(query), матрица K** ключей **(key) и матрица V** значений **(value). Введенные матрицы являются аргументами** функции внимания

$$\mathbf{A} = \mathrm{Attn}(\mathbf{Q},\,\mathbf{K},\,\mathbf{V}) \;=\; \mathrm{softmax}\Bigl(\frac{\mathbf{Q}\cdot\mathbf{K}^\top}{\sqrt{E}}\Bigr)\,\mathbf{V},$$

Вернёмся теперь к механизму внимания, описанному в предыдущем разделе. Пусть U=uαi - векторы скрытых состояний декодера, а V=vαi - энкодера, где первый индекс - номер вектора, а второй - его компоненты (каждая строчка матриц U и V это α-тый вектор). Тогда, компоненты векторов u′α - добавок к скрытому состоянию декодера являются строками матрицы (μ=1/√E):

$$\mathbf{U}' = \mathrm{Attn}(\mathbf{U},\,\mathbf{V},\,\mathbf{V}).$$

Объяснение почему корень из размерности:принято делить на корень из размерности E векторов матриц Q и K. Мотивация для этого может быть следующей. Пусть компоненты двух векторов q и k являются независимыми случайными величинами с нулевым средним и единичной дисперсией. Тогда скалярное произведение q·k также имеет нулевое среднее и дисперсию равную размерности векторов E, т.е. типичные значения q·k находятся в интервале ±√E. Масштабирование переводит их к интервалу ±1**

$$\mathbf{A} \;=\; \mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathbf{h}_1,\ldots,\mathbf{h}_H)\,\mathbf{W}^O, \qquad \mathbf{h}_i = \mathrm{Attn}(\mathbf{Q}\,\mathbf{W}_i^Q,\;\mathbf{K}\,\mathbf{W}_i^K,\;\mathbf{V}\,\mathbf{W}_i^V).$$

Фокусы внимания - головы. H - количество голов внимания В исходной статье (2017), где были введены головы внимания, было положено Ei=Eh=E/H
По факту входящие матрицы Q,K,V сначала подвергаются линейному преобразованию, а *затем* разрезаются по вертикали на H голов и веса внимания

вычисляются уже независимым образом для каждой головы.

Сначала для векторов запроса, ключей и значений делают линейное преобразование при помощи трёх матриц $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$. При этом для каждой (i-той) головы $\mathbf{h}_i$ набор матриц свой (свой "фокус внимания"). Если:

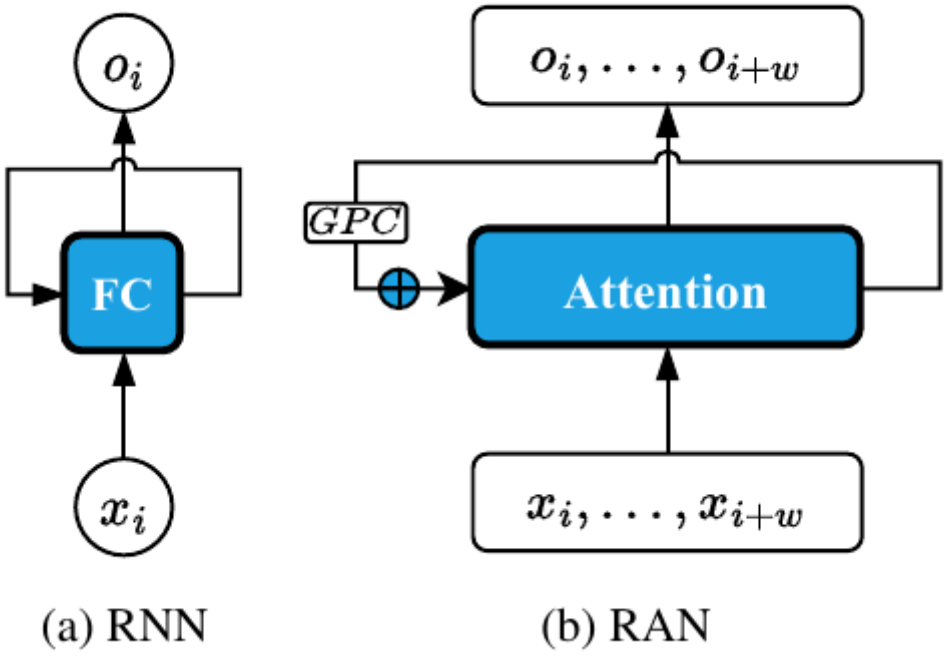
$$\mathbf{Q}: (N, E), \quad \mathbf{K}: (M, E_k), \quad \mathbf{V}: (M, E_v), \quad \mathbf{A}: (N, E_a),$$

то, в общем случае, матрицы могут иметь следующие формы:

$$\mathbf{W}_i^Q: (E, E_i), \quad \mathbf{W}_i^K: (E_k, E_i), \quad \mathbf{W}_i^V: (E_v, E_h), \quad \mathbf{h}_i: (N, E_h), \quad \mathbf{W}^O: (E_h H, E_a).$$

После вычисления функции внимания `Attn` для каждой головы, получаются матрицы формы (N, E_h). Их объединяют в одну (конкатенируют) по последнему индексу, что даёт матрицу $(N, E_h H)$. Её свёртка с $\mathbf{W}^O$ приводит к финальной матрице $\mathbf{A}$ формы (N, E_a).

Рассмотрим рекуррентное внимание для длинных текстов на примере RAN(Reccurent attention network) (Краткий рассказ)



Не будем глубоко углубляться в математику, если захотите прочесть эту статью, оставлю на неё ссылочку, статья была написана в июле. Самая идея заключается в том, что в рекуррентной сети внимания в отличие от обычных и популярных рекуррентных сетей внимания таких как LSTM и GRU, которые были созданы для избавления от проблемы затухания градиента в ванильной РНН, так как проблема затухания градиента не позволяла запоминать информацию на длительной дистанции, обычный слой заменен на слой внимания, который работает очень интересно.
Ran проходит через последовательности без повторения на окна, она обращается к pMHSA(позиционные множественным головам внимания) в оконной области для выделения локальных зависимостей. Для дальнейшего распространения RAN создает GPC(Global perception cell) vector из репрезентации слоя самовнимания текущего окна. GPC вектор далее конкатенируется с токенами следущего окна в который входит слой само-внимания. Новый GPC вектор будет идти к последующим окнам с остаточным

смыслом, для уменьшения градиентного затухания и для обновления в той же манере.
Сама по себе функция GPC вектора двойственна:
1)Представление уровненой контекстной информации
2)Он сохраняет очень хорошо

```
Функцию внимания можно описать как сопоставление запроса (query) и набора пар ключ-значение (key-value) с выходными данными, где запрос, ключи, значения и выходные данные являются векторами. Результат вычисляется как взвешенная сумма значений, где вес, присвоенный каждому значению, вычисляется функцией совместимости запроса и соответствующего ключа.
```