ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет об исследовательском проекте

на тему: Инференс больших язвковых моделей типа декодер

(промежуточный, этап 1)

Выполнил:
Студент группы БПМИ223 _____ И.А.Бобошко
Подпись И.О.Фамилия

04.02.2025
Дата

Принял:
Руководитель проекта Потапов Иван Андреевич
Имя, Отчество, Фамилия

Senior Software Engineer
Должность, ученое звание

Zalando SE
Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки 04.02.2025 _____ _____
Оценка (по 10-ти бальной шкале) Подпись

Москва

Содержание

Аннотация

Цель данного проекта заключается в изучении устройства больших языковых моделей для генерации текста и современных методов оптимизации их инференса

1 Введение

Современные большие языковые модели стали неотъемлемой частью множества приложений, от автоматической генерации текста до интеллектуальных помощников и систем машинного перевода. Их развитие привело к значительному улучшению качества текстовой генерации, однако высокая вычислительная сложность и ресурсозатратность инференса (процесса генерации предсказаний) остаются серьезными вызовами.

В данной работе рассматриваются ключевые принципы устройства больших языковых моделей, их архитектурные особенности и современные методы оптимизации инференса. Анализируются подходы к уменьшению вычислительных затрат, включая квантование, сжатие моделей, а также алгоритмические усовершенствования.

Целью работы является исследование механизмов работы LLM и методов, позволяющих повысить их эффективность, сохраняя высокое качество генерации текста. Итогом станет обзор современных решений и перспективных направлений в области оптимизации инференса языковых моделей.

2 Языковая модель

Определение 1. Пусть $X = (x_1, \ldots, x_n)$ - последовательность токенов, Θ - параметры модели, тогда задача языкового моделирования заключается в предсказании следующего токена по текущему и предыдущим. Пользуясь методом максимального правдоподобия, можно свести задачу к классификации предсказанного токена на каждом шаге. [2]

$$p_\theta(X) = \prod_{t=1}^{n} p_\theta(x_t | x_{<t}) \longrightarrow \max_\Theta$$

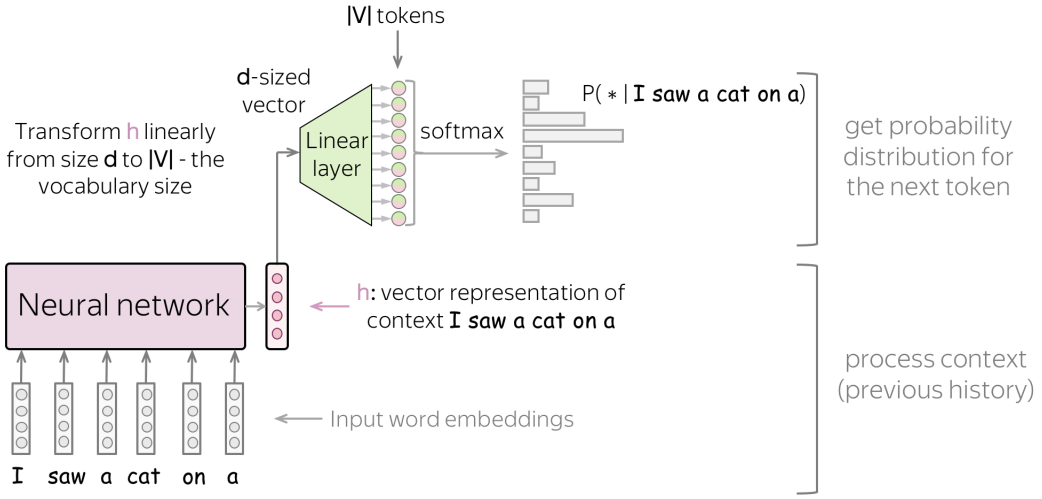
$$-\sum_{i=1}^{n}\log(p_\theta(x_t|x_{<t})) \longrightarrow \min_{\Theta}$$

Замечание 2. Данное определение описывает определенный тип языковых моделей, а именно left-to-right или регрессионная модель, в работе будут рассмотрены так же и другие варианты, например, masked (для обучения BERT)

Замечание 3. В контексте данной работы под токеном будут подразумеваться составляющие текстовых данных, например, символы, слова, предложения и другие лексемы.

2.1 Обучение нейросетевых языковых моделей

1. Подаем модели на вход эмбеддинги предыдущих токенов, то есть для i-ого токена, передаем $(x_0, \ldots, x_{i-1})$
2. Получаем векторное представление контекста как выход нейросети
3. Обучаем линейный классификатор на предсказание следующего токена с помощью закодированного контекста. Зачастую применяют это линейное отображение, потому что размер эмбеддинга не совпадает с количеством классов, а также это дает больше возможностей при обучении. [4]



2.2 Оценка качества

Определение 4. Cross-Entropy

$$CE(y_1, \ldots, y_N) = -\frac{1}{N}\sum_{i=1}^{N}\log p(x_i|x_{<i}) \tag{1}$$

Чем ниже значение Cross-Entropy, тем лучше модель предсказывает правильные токены, то есть тем лучше она отражает структуру языка.

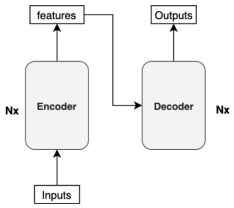
Определение 5. Perplexity

$$Perplexity(y_1, \ldots, y_N) = 2^{CE(y_1,\ldots,y_N)} \tag{2}$$

Чем меньше, тем лучше. Минимальное значение - 1, когда мы предсказали все токены правильно, то есть нулевой лосс. Самый худший сценарий $Perplexity(y_1, \ldots, y_N) = |Vocab|$, то есть равномерное распределение.

3 Encoder-Decoder архитектура

- Encoder принимает исходные входные данные и возвращает новое информативное представление для входов
- Decoder принимает представления, полученные из Encoder, и генерирует конечные выходы модели [2]



Замечание 6. Данный архитектурный подход зачастую используют при решении seq-to-seq задач. Например, рассмотрим как она работает для решения задачи машинного перевода. В Encoder поступает токены на source языке, мы пропускаем их через слой и получаем новое представление, дальше в Decoder на основе этого представления и предыдущей истории получаем токены уже на target языке

4 Attention

4.1 Self-Attention

Пусть $X \in \mathbb{R}^{l \times d_k}$ - входные эмбеддинги, где l - длина последовательности, d_k - размер эмбеддинга токена. Из входной матрицы получим три новых с помощью линейный преобразований, где $W \in \mathbb{R}^{d_k \times d_k}$:

$$X \longrightarrow XW_Q = Q \in \mathbb{R}^{l \times d_k}$$

$$X \longrightarrow XW_K = K \in \mathbb{R}^{l \times d_k}$$

$$X \longrightarrow XW_V = V \in \mathbb{R}^{l \times d_k}$$

Тогда слой attention можно задать следующим образом:

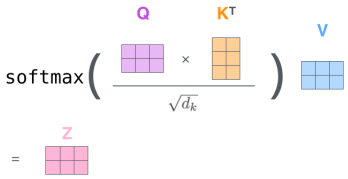
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

То есть сначала мы получаем три новых представления для исходной последовательности, которые будут выполнять разные функции:

- W_Q отвечает за запросы
- W_K отвечает за ключи
- W_V отвечает за представления, которые будут использоваться для получения выходов слоя

Далее мы считаем QK^T - скалярные произведения Q-эмбеддингов и K-эмбеддингов, то есть для каждого запроса q_i (строка матрицы Q) мы считаем скалярное произведение со всеми ключами, нормируем на d_k - необходимо для того, чтобы в softmax не поступали слишком большие числа, далее, применяя softmax, мы получаем "веса важности"каждого ключа для данного запроса.

В конце мы домножаем на V, тем самым мы берем каждое значение с весами полученными из прошлого шага.



4.2 Multihead-Attention

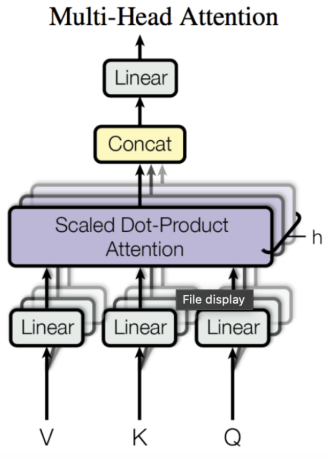
Вместо трех матриц W_Q, W_K, W_V у нас будет W_Q^h, W_K^h, W_V^h, где $h = 1, \ldots, H$, $W^h \in \mathbb{R}^{\frac{d_k}{H} \times \frac{d_k}{H}}$. Таким образов у нас паралельно будут считаться несколько attention score, что позволяет извлечь больше информации из последовательности:

$$\text{head}_h(Q_h, K_h, V_h) = \text{softmax}(\frac{Q_h K_h^T}{\sqrt{d_k}})V_h$$

Тогда итоговый результат будет конкатенацией результатов, полученных с помощью каждой головы, также применяют линейное преобразование для сконкатенированного выхода

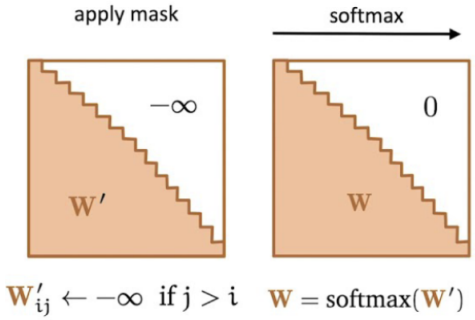
$$\text{MultiHead}(Q, K, V) = \text{concat}(\text{head}_1, \ldots, \text{head}_H)W_O$$

Данный прием позволяет извлечь больше различных аспектов связи между токенами



4.3 Masked Attention

Данная модификация нужна, чтобы не учитывать токены из будущего, достигается за счет применения маски, которая зануляет в softmax ненужные токены



5 Transformer

В оригиналой статье данная модель была предложена для решения задачи машинного перевода. [3]

5.1 Attention

В данной моделе используются разные виды attention в encoder и decoder

- Encoder-Encoder - обычный MultiHeadAttention
- Decoder-Decoder - masked MultiHeadAttention
- Encoder-Decoder - Q из decoder; K, V из encoder

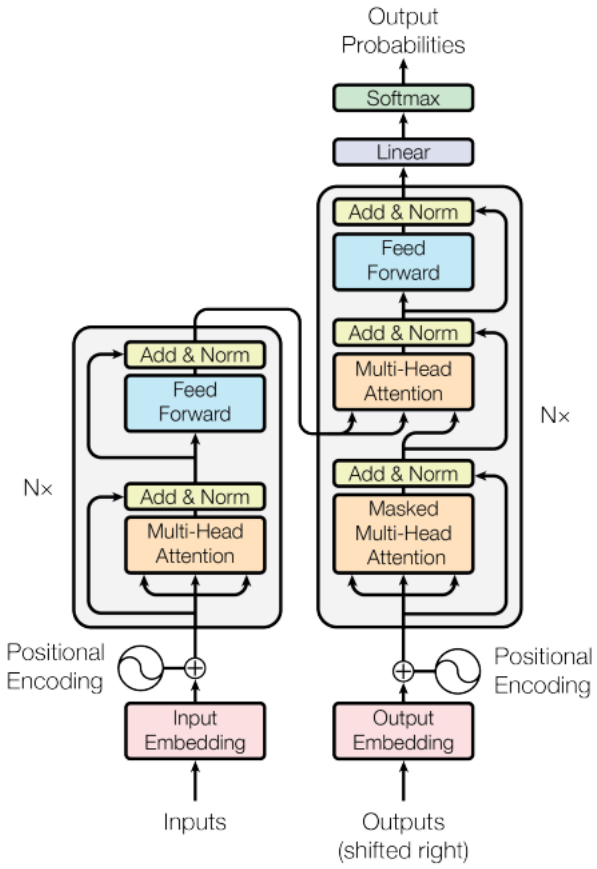


Figure 1: The Transformer - model architecture.

5.2 Positional Encoding

В трансформере мы никак не учитываем порядок токенов, поэтому авторы статьи добавляют еще один вид эмбеддинга, который решает эту проблему

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3}$$

где pos — позиция слова, i — индекс измерения, а d_{model} — размерность эмбеддинга. Эти функции позволяют представить позиции так, чтобы они могли быть легко интерпретируемы моделью, сохраняя различия между последовательностями различных длин.

5.3 Add and Norm

Механизм *Add and Norm* используется в архитектуре трансформеров для стабилизации обучения и улучшения сходимости модели. Он применяется после каждого self-attention и FFN сети. Этот механизм состоит из двух частей:

- Residual Connection : к выходу слоя добавляется его входное значение:

$$y = x + f(x) \tag{4}$$

  Это позволяет сохранять информацию из исходного входа, облегчая процесс обучения.

- Layer Normalization : нормализует выходные данные по их среднему и дисперсии:

$$\hat{y} = \frac{y - \mu}{\sigma} * \gamma + \beta \tag{5}$$

где μ — среднее значение, σ — стандартное отклонение, а γ и β — обучаемые параметры.

Этот процесс помогает модели быстрее сходиться, улучшает стабильность градиентов и делает обучение более эффективным.

5.4 Feed-forward block

После подсчета attentiona в каждом блоке в конце применяется полносвязная нейросеть:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{6}$$

Можно считать, что attention извлекает важные связи, а FFN обрабатывает их. Стоит отметить, что в данном блоке мы сначала увеличиваем ширину в 4 раза, а потом возвращаем в исходное состояния, поэтому данный блок можно считать один из самых затратных мест в архитектуре.

6 Quantization

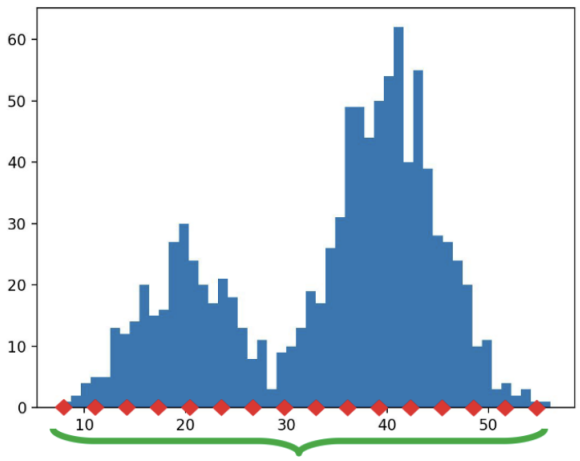
Определение 7. Квантизация - это способ построения отображения из более мощного ("непрерывного") множества в менее мощное (дискретное) множество.

Плюсы квантизации:

- Модель меньше весит
- Лучше пропускная способность
- Операции, как GEMM и GEMV, быстрее работают для целочисленных данных

Замечание 8. В контексе трансформеров больше всего места занимают веса FFN слоев, поэтому в основном квантизуют именно их.

6.1 Линейная квантизация



На картинке изображено распределение весов, а красных точки из квантизованного множества.

Encode: $q = (w/s + z).clip(uint)$

Decode: $w \approx sq - z$

$s = (max(w) - min(w))/2^k$

$z = -min(w)/s$

7 Стратегии генерации текста

Обученная языковая модель выдает нам распределение следующего токена. Базовый метод заключается в семплинге следующего токена из полученного распределия. Для более качественной генерации можно выдавать:

$$x^* = \underset{x}{argmax} \prod_{t=1}^{n} p_\theta(x_t|x_{<t}) \tag{7}$$

В связи с тем, что поиск такого x требует полного перебора в этом разделе будут рассмотрены некоторые эвристики, которые дают хорошое качество.

7.1 Жадный алгоритм

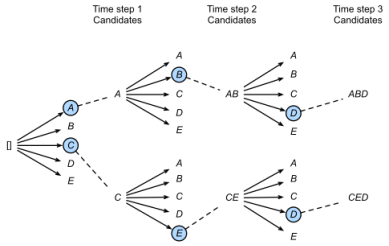
На каждом шаге выбираем наиболее вероятный токен:

$$x^* = \prod_{t=1}^{n} \underset{x_t}{argmax}(p_\theta(x_t|x_{<t})) \tag{8}$$

Данный метод быстро работает, но при этом уступает в качестве. Также стоит отметить, что он зачастую зацикливает генерацию.

7.2 Beam Search

В отличие от greedy в данном методе мы рассматривает несколько вариантов параллельно. На каждом шаге мы оставляем только топ-k из этих веток.



7.3 Temrature sampling

Данный метод заключается в шкалировании логитов модели:

$$softmax(x) = \frac{exp(x)}{\sum_i exp(x)} \longrightarrow \frac{exp(x/T)}{\sum_i exp(x/T)} \tag{9}$$

Чем больше гиперпараметр T, тем ближе будет итоговое распределение к равномерному и наоборот, чем меньше T, тем ближе к вырожденному (то есть к жадной генерации).

7.4 Top-k

В данном подходе в качестве ответа выбирается класс из топ-k классов. У данного метода есть проблема при фиксированных k, например, бывает, что у нас будет равномерное распредение выходов, тогда мы потеряем часть информации, а также когда у нас есть несколько наиболее вероятных классов, тогда мы можем выдать в качестве ответа класс с около нулевой вероятностью

7.5 Top-p

Выбирает слова из динамически изменяемого набора, включающего суммарно p вероятности [1]:

$$P(x) \in \{x_i | \sum P(x_i) \leq p\} \tag{10}$$

Плюсы: баланс между случайностью и вероятностью. Минусы: возможные несвязные тексты при высоких значениях p.

Список литературы

[1] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL: https://arxiv.org/abs/1904.09751, arXiv:1904.09751.

[2] I. Sadrtdinov. Deep learning 1. 2023. URL: https://github.com/isadrtdinov/intro-to-dl-hse/tree/2023-2024/lecture-notes.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL: https://arxiv.org/abs/1706.03762, arXiv:1706.03762.

[4] Elena Voita. NLP Course For You, Sep 2020. URL: https://lena-voita.github.io/nlp_course.html.