Методы кодирования позиционной информации в Transformer

Мурат Апишев (mel-lain@yandex.ru)

Декабрь, 2023

Позиционное кодирование

- В общем случае Transformer обрабатывает векторы последовательности одновременно, нужно встраивать информацию о позиции каждого токена
- Важные аспекты:
  - уникальность представления для каждой позиции
  - независимость расстояний между парами токенов от длины входа
  - детерминированность представления
  - адаптация метода к расширению контекста модели
- Существует много подходов, рассмотрим нижеследующие:
  - Sinusoidal / Trainable Absolute
  - Relative
  - Transformer-XL
  - T5
  - DeBERTa
  - RoPE
  - ALiBi
  - xPos
  - NTK-Aware Scaled RoPE
  - Positional Interpolation RoPE
  - NoPE
  - YaRN

Основные обозначения и формулы

- $x = (x_1, \ldots, x_n)$ – векторы токенов входной последовательности
- $z = (z_1, \ldots, z_n)$ – векторы выходов головы self-attention
- d_x, d_z – размерности векторов x и z соответственно
- Q, K, V – векторы запросов, ключей и значений для слоя-головы
- W^Q, W^K, W^V – весовые матрицы для получения векторов запросов, ключей и значений для слоя-головы
- Формулы подсчёта self-attention:

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}}, \qquad \alpha_{ij} = \mathrm{softmax}(e)_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$z_i = \sum_{j=1}^{n} \alpha_{ij}\left(x_{ij} W^V\right)$$

Sinusoidal и Trainable Absolute [1]

- Первый, самый простой, но достаточно эффективный подход:
  - каждая абсолютная позиция кодируется вектором $\hat{p}_t \in \mathbb{R}^{d_x}$
  - элементы $\hat{p}_t^i$ вектора определяются формулой:

$$\hat{p}_t^i = \begin{cases} \sin(w_k t), & i = 2k \\ \cos(w_k t), & i = 2k+1 \end{cases}, \qquad w_k = \frac{1}{10000^{2k/d_x}} \tag{1}$$

  - вектор позиции прибавляется к вектору токена на входе модели
- Синусоидальные векторы можно заменить на случайные векторы, обучаемые с нуля вместе с моделью
- Это увеличивает число параметров, но лишает возможности экстраполяции при росте длины контекста

[1] Attention Is All You Need, 2017

Relative [2]

- Большинство популярных методов работы с позициями – относительные
- Кодируется не одиночный индекс, а пара позиций на разных расстояниях друг от друга
- Обычно относительное кодирование производится не путём сложения векторов на входе, а через модификацию подсчёта внимания
- В этой работе (одной из первых)
  - для каждого расстояния $i - j$ обучаются два вектора $a_{ij}^K, a_{ij}^V \in \mathbb{R}^{d_z}$
  - изменение формул self-attention:

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K + a_{ij}^K\right)^T}{\sqrt{d_z}}, \qquad z_i = \sum_{j=1}^{n} \alpha_{ij}\left(x_{ij} W^V + a_{ij}^V\right)$$

- Вычисления можно ускорить, раскрыв скобки перед подсчётом
- Обученные векторы общие для всех голов внимания и слоёв

[2] Self-Attention with Relative Position Representations, 2018

Transformer-XL [3]

- Вход модели делится на сегменты, обрабатываемые последовательно
- При обработке i-го сегмента используются выходы для $i+1$-го
- Абсолютное позиционное кодирование работать не будет: у обоих сегментов оно одинаковое
- Предлагается относительная схема:
  - как и в Relative, позиционная информация переходит в self-attention
  - расчёт e_{ij} в абсолютном кодировании ($x_i = x_i^{\mathbf{e}mb} + x_i^{\mathbf{p}os}$):

$$e_{ij} \propto \left((x_i^e + x_i^p)W^Q\right)\left((x_j^e + x_j^p)W^K\right)^T =$$
$$= x_i^e W^Q \left(W^K\right)^T \left(x_j^e\right)^T + x_i^e W^Q \left(W^K\right)^T \left(x_j^p\right)^T +$$
$$+ x_i^p W^Q \left(W^K\right)^T \left(x_j^e\right)^T + x_i^p W^Q \left(W^K\right)^T \left(x_j^p\right)^T$$

[3] Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, 2019

Transformer-XL

- Предлагается относительная схема:
  - репараметризация в относительном кодировании:

$$e_{ij} \propto \underbrace{x_i^e W^Q \left(W^{K,E}\right)^T \left(x_j^e\right)^T}_{(1)} + \underbrace{x_i^e W^Q \left(W^{K,R}\right)^T \left(R_{i-j}\right)^T}_{(2)} +$$
$$+ \underbrace{u \left(W^{K,E}\right)^T \left(x_j^e\right)^T}_{(3)} + \underbrace{v \left(W^{K,R}\right)^T \left(R_{i-j}\right)^T}_{(4)}$$

  - абсолютные позиционные эмбеддинги заменяются на фиксированную синусоидальную матрицу относительных R
  - векторы запросов для позиционной предлагается брать одинаковыми и моделировать обучаемыми векторами u и v
  - выделяются две весовые матрицы для получения векторов ключей эмбеддингов и позиций соответственно

Transformer-XL

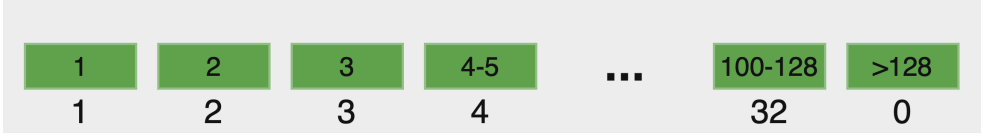
- Предлагается относительная схема:
  - репараметризация в относительном кодировании:

$$e_{ij} \propto \underbrace{x_i^e W^Q \left(W^{K,E}\right)^T \left(x_j^e\right)^T}_{(1)} + \underbrace{x_i^e W^Q \left(W^{K,R}\right)^T \left(R_{i-j}\right)^T}_{(2)} +$$
$$+ \underbrace{u \left(W^{K,E}\right)^T \left(x_j^e\right)^T}_{(3)} + \underbrace{v \left(W^{K,R}\right)^T \left(R_{i-j}\right)^T}_{(4)}$$

  - Интуитивный смысл каждого слагаемого:
    1. содержательная смысловая связь
    2. связь смысла и позиционной информации
    3. глобальное смысловое смещение
    4. глобальное позиционное смещение
- В Relative есть только (1) и (2), смещения отбрасываются
- Также $\left(W^{K,R}\right)^T \left(R_{i-j}\right)^T$ заменено одной обучаемой матрицей $\Rightarrow$ отказ от синусоидальных векторов и ухудшение обобщения на длинный контекст

T5 [4]

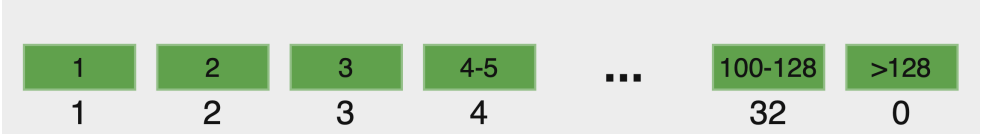
- Позиционная информация кодируется скаляром, который прибавляется к e_{ij} перед softmax
- Скаляры соответствуют различным расстояниям-отступам $(i - j)$
- Всего в модели 32 скаляра, которые в логарифмической шкале покрывают 128 отступов
- Близкие к 0 соседние отступы кодируются разными скалярами, а далёкие могут кодироваться одним



[4] Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2020

T5

- Всем расстояниям, большим 128, соответствует один и тот же скаляр
- Общий скаляр для больших расстояний способствует обобщению модели на длинный контекст



- Позиционные скаляры обучаемые и настраиваются вместе с моделью
- Каждый следующий слой расширяет окно улавливаемой позиционной информации (идейно схоже с CNN или окном внимания в Longformer[5])
- Обученные скаляры свои для каждой головы self-attention, но общие для всех слоёв модели

[5] Longformer: The Long-Document Transformer, 2020

DeBERTa [6]

- Формула подсчёта логитов в self-attention раскладывается (как в Transformer-XL), предлагается использовать первые 3 слагаемых:

$$e_{ij} \propto \underbrace{x_i^e W^Q \left(W^K\right)^T \left(x_j^e\right)^T}_{(1)} + \underbrace{x_i^e W^Q \left(W^K\right)^T \left(x_j^p\right)^T}_{(2)} +$$
$$+ \underbrace{x_i^p W^Q \left(W^K\right)^T \left(x_j^e\right)^T}_{(3)} + \underbrace{x_i^p W^Q \left(W^K\right)^T \left(x_j^p\right)^T}_{(4)}$$

- Мотивация в работе:
  - (1) и (2) важны, они используются и в Relative
  - (3) тоже важно для более полного моделирования информации об отступе
  - (4) – position-to-position term – для относительного кодирования несёт мало дополнительной информации

[6] DeBERTa: Decoding-enhanced BERT with Disentangled Attention, 2021

DeBERTa

- Репараметризация очень похожа на Transformer-XL:

$$e_{ij} \propto \underbrace{x_i^e W^Q \left(W^K\right)^T \left(x_j^e\right)^T}_{(1)} + \underbrace{x_i^e W^Q \left(W^{K,R}\right)^T \left(R_{i-j}\right)^T}_{(2)} + \underbrace{R_{i-j} W^{Q,R} \left(W^K\right)^T \left(x_j^e\right)^T}_{(3)}$$

- Новые весовые матрицы свои у каждой пары голова-слой
- Матрица расстояний R_{i-j} обучаемая и общая для всех слоёв и своя у каждой головы
- Для всех отступов, не попадающих в размер R_{i-j}, берутся векторы ближайшего с соответствующего конца отступа
- Перед последним слоем в модели используются обучаемые абсолютные позиционные эмбеддинги

RoPE [7]

- Если исключить позиционные эмбеддинги, то в обобщенном варианте

$$e_{ij} \propto \langle f_q(x_i, i), f_k(x_j, j) \rangle$$

  т.е. зависит от функций от эмбеддингов и абсолютных позиций

- Предлагается подобрать функции g, f_q, f_k, такие, что

$$\langle f_q(x_i, i), f_k(x_j, j) = g(x_i, x_j, i - j) \rangle$$

- Можно доказать, что для случая $d_z = 2$ подойдёт преобразование

$$f_q(x_i, i) = \begin{pmatrix} \cos i\theta & -\sin i\theta \\ \sin i\theta & \cos i\theta \end{pmatrix} \begin{pmatrix} W_{11}^Q & W_{12}^Q \\ W_{21}^Q & W_{22}^Q \end{pmatrix} \begin{pmatrix} x_i^1 \\ x_i^2 \end{pmatrix}$$

  где $\theta \in \mathbb{R}, \theta \neq 0$ – заданная константа, формула для f_k аналогична

[7] RoFormer: Enhanced Transformer with Rotary Position Embedding, 2021

RoPE

- Можно доказать, что для случая $d_z = 2$ подойдёт преобразование

$$f_q(x_i, i) = \begin{pmatrix} \cos i\theta & -\sin i\theta \\ \sin i\theta & \cos i\theta \end{pmatrix} \begin{pmatrix} W_{11}^Q & W_{12}^Q \\ W_{21}^Q & W_{22}^Q \end{pmatrix} \begin{pmatrix} x_i^1 \\ x_i^2 \end{pmatrix}$$

- Подходящая функция g при этом определяется так:

$$g(x_i, x_j, i-j) = \mathrm{Re}[\left(W^Q x_i\right)\left(W^K x_j\right)^* e^{\mathbf{i}(i-j)\theta}]$$

где

  - $\mathrm{Re}[\mathrm{X}]$ – вещественная часть $X \in \mathbb{C}$
  - A^* – сопряжённая матрица к A
  - $\mathbf{i}$ – мнимая единица

- Применение RoPE заключается в повороте вектора запроса/ключа на угол, зависящий от индекса его позиции
- Поворот обоих векторов на один угол (т.е. смещение позиций без изменения расстояния) сохранит значение скалярного произведения

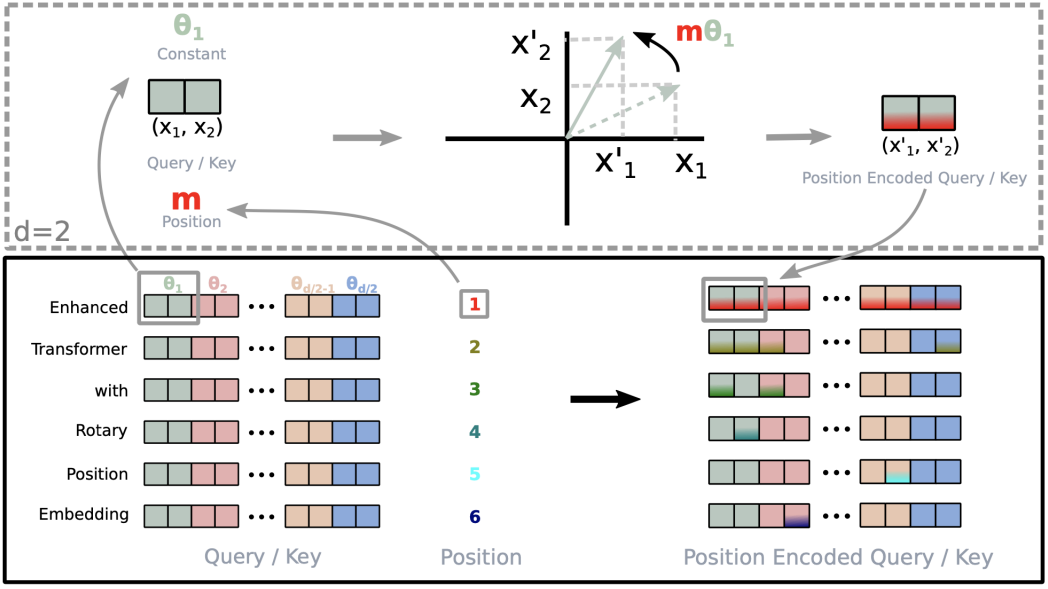
RoPE

- Полученное преобразование обобщается на любую чётную d_z
- Для этого d_z делится на $d_z/2$ двумерных подпространств, к каждому из которых применяется свой матрица поворота
- Итоговое выражение для f_q (f_k аналогично): $f_q(x_i, i) = R^{d_z}_{\Theta,i} W^Q x_i$, где

$$R^{d_x}_{\Theta,i} = \begin{pmatrix} \cos i\theta_1 & -\sin i\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin i\theta_1 & \cos i\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos i\theta_2 & -\sin i\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin i\theta_2 & \cos i\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cos i\theta_{d_z/2} & -\sin i\theta_{d_z/2} \\ 0 & 0 & \cdots & 0 & 0 & \sin i\theta_{d_z/2} & \cos i\theta_{d_z/2} \end{pmatrix},$$

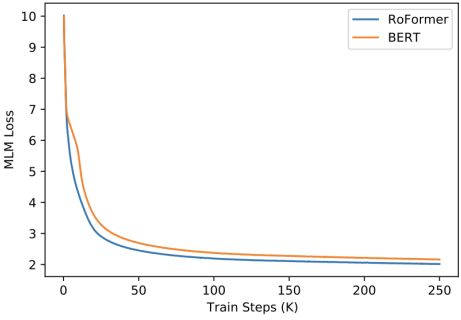
$$\Theta = \{\theta_k = 10000^{-2(k-1)/d_z}, k \in [1, \ldots, d_z/2]\}$$

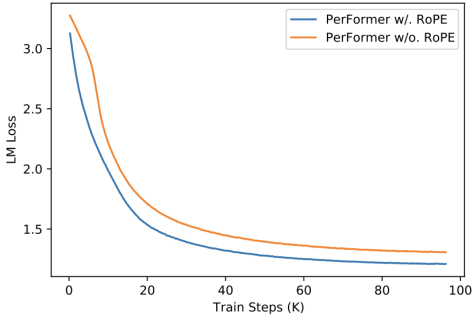
RoPE


θ1
Constant
(x1, x2)
Query / Key
m
Position
d=2
x'2
x2
mθ1
x'1
x1
(x'1, x'2)
Position Encoded Query / Key
θ1
θ2
θd/2-1
θd/2
Enhanced
Transformer
with
Rotary
Position
Embedding
1
2
3
4
5
6
Query / Key
Position
Position Encoded Query / Key

RoPE

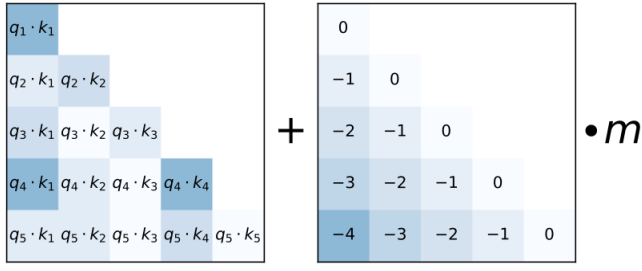
- Один из самых популярных подходов к позиционному кодированию, применяется в LLaMA, Qwen, Mistral
- В отличие от Absolute применяется не отдельным координатам, а к парам, и использует умножение на sin / cos вместо суммы
- В разных экспериментах показывает себя лучше, чем Absolute и Relative
- Теоретически должен помогать модели обобщаться на более длинный контекст, чем при обучении, но по факту это не работает
- Используется в качестве основы для более продвинутых методов





ALiBi [8]

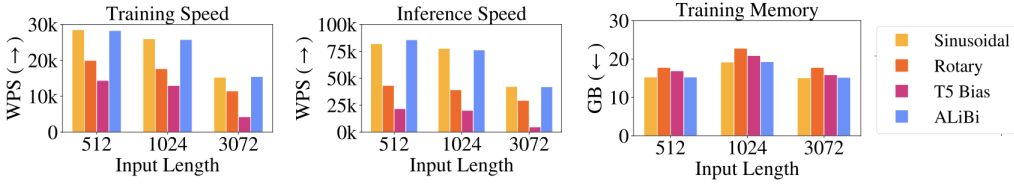
- Метод идейно схож с T5: вместо добавления позиционных эмбеддингов или репараметризации подсчёта внимания к e_{ij} добавляется скаляр
- В отличии от T5 скаляры не обучаемые и представляют собой произведение $m(i-j)$, где m – заданное на старте число, своё для каждой головы (и общее для слоёв)
- Например, для 8 голов это $\frac{1}{2^1}, \frac{1}{2^2}, \ldots, \frac{1}{2^8}$



[8] Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation, 2022

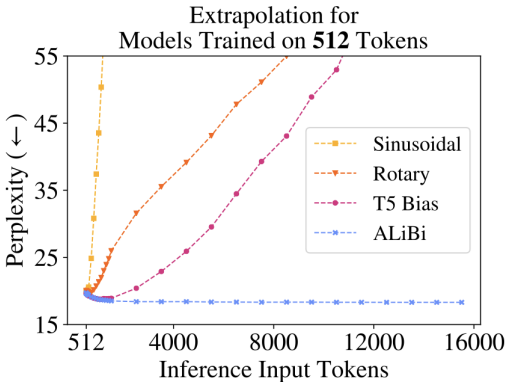
ALiBi

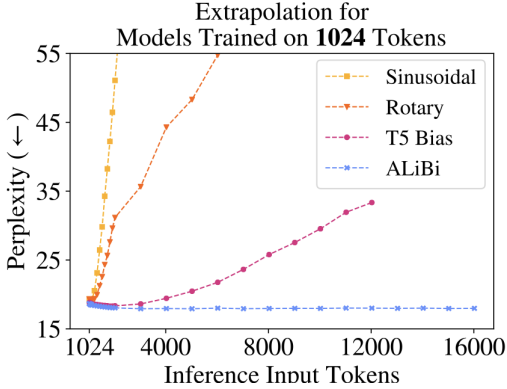
- В сравнении Absolute vs. RoPE vs T5 только T5 показал хорошую способность к обобщению на последовательностях с большей длиной, чем на обучении (до 600 токенов сверх исходных 512)
- Но его преимущество перекрывается возрастающими вычислительными затратами, проще обучить модель с Absolute с большим контекстом
- AliBi оказывается достаточно эффективным с т.з. скорости и памяти



ALiBi

- AliBi оказывается достаточно эффективным с т.з. скорости и памяти
- AliBi обеспечивает большую степень обобщения, позволяя модели работать с контекстом, в разы более длинным, чем на обучении (по крайней мере с т.з. перплексии)





xPos [9]

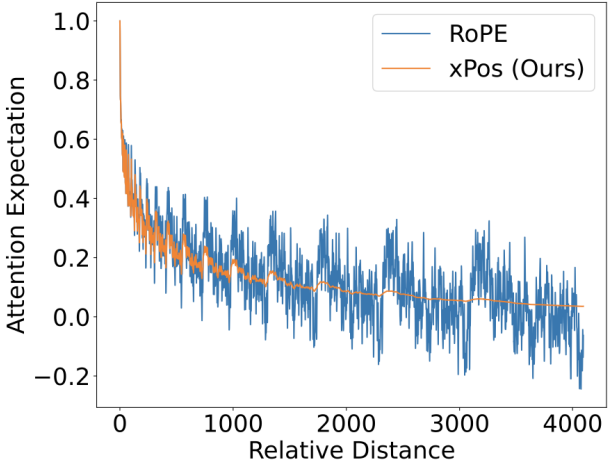
- Предлагается подход, основанный на доработке RoPE
- Вводится понятие «ожидаемого значения внимания» для пары токенов на заданном расстоянии
- Показывается, что в обычном RoPE при существенном росте расстояния эта величина начинает осциллировать, это портит качество модели
- Проблема объясняется тем, что значения косинуса не являются монотонными при угле поворота, большем π
- Решение: добавить задаваемые априорно дополнительные множители для каждой пары компонентов векторов Q и K
- Они масштабируют компоненты векторов и стабилизируют график ожидаемого значения внимания

[9] A Length-Extrapolatable Transformer, 2023

xPos

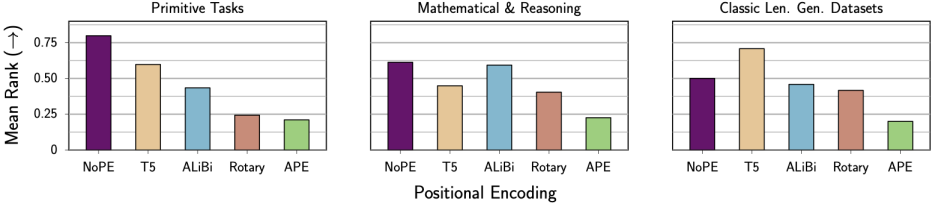
Algorithm 1: Attention with xPos

def $\mathrm{rot}(x)$:
  return $[-x_1, x_0, -x_3, x_2, ...]$
Initialization:
$\theta_i = 1/10000^{2i/d}, \theta \in \mathbb{R}^{d/2}$
$\hat{\zeta}_i = (i/(d/2) + \gamma)/(1+\gamma),\ \hat{\zeta} \in \mathbb{R}^{d/2}$
Input: $Q, K, V \in \mathbb{R}^{h \times l \times d}, M \in \mathbb{R}^{d \times d}$
$C_{mn} = \cos m\theta_n, C \in \mathbb{R}^{l \times d/2}$
$S_{mn} = \sin m\theta_n, S \in \mathbb{R}^{l \times d/2}$
$T_{mn} = \hat{\zeta}_n^m, T \in \mathbb{R}^{l \times d/2}$
$Q = (Q \times C + \mathrm{rot}(Q) \times S) \times T$
$K = (K \times C + \mathrm{rot}(K) \times S) \times T^{-1}$
$output = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}} \cdot M)V$
return *output*



NoPE [10]

- Кодировщик Transformer не может работать без позиционного кодирования – получится мешок слов
- Декодировщик теоретически может – если работает авторегрессионно
- Формулируются теоремы о том, что такая модель может выделять
  - на первом слое абсолютную позиционую информацию
  - на всех последующих – относительную
- Эксперимент: модель $100M$, контекст 20 (на тесте до 40), обучение на задачу (3 группы из 10 задач)



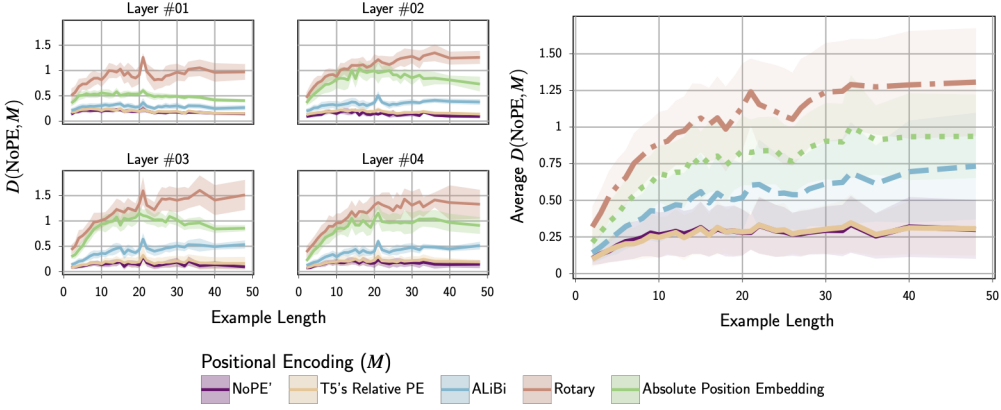
[10] The Impact of Positional Encoding on Length Generalization in Transformers, 2023

NoPE

- Близость между моделями A и B на слое ℓ можно определить как

$$D^{\ell}(A,B) = \min_{(P,Q)\in A_{\ell}\times B_{\ell}} D(P,Q), \quad D(P,Q) = \frac{1}{n}\sum_{i=1}^{n} D_{JSD}(P_i \| Q_i)$$

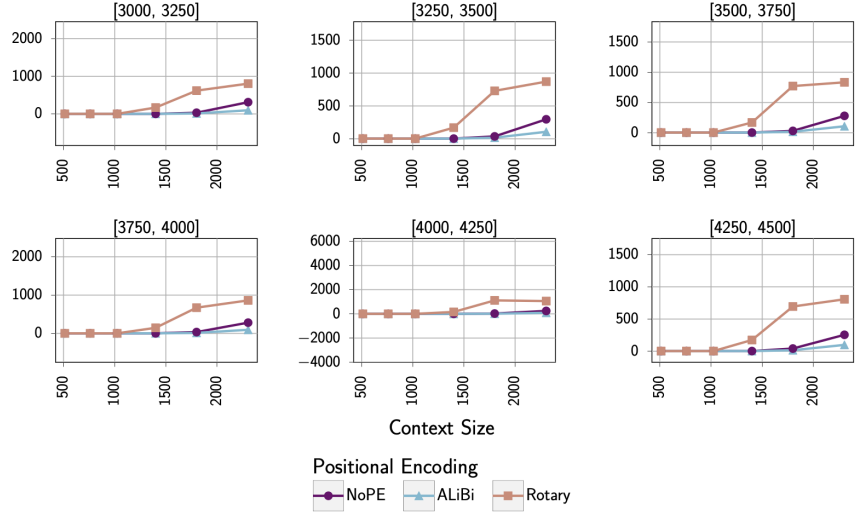
где D_{JSD} – дивергенция Йенсена-Шеннона по выходам двух голов



- Ближе всего NoPE к позиционному кодированию из T5

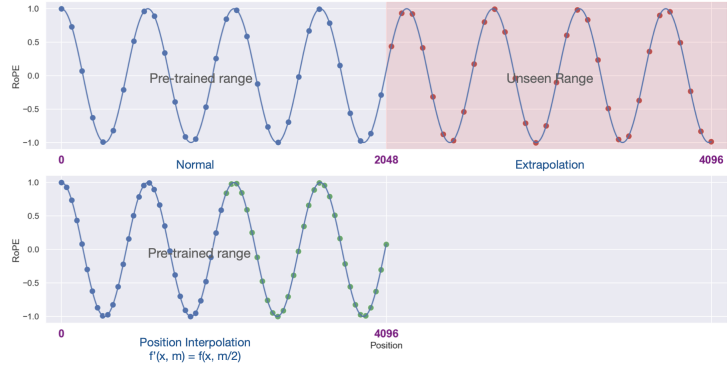
NoPE

- Эксперимент: модель $1.3B$, контекст 1024 (на тесте до 2560), обучение авторегрессионное на данных из StarCoder
- Перплексия на 200 последних токенах (сверху – длина исходных текстов)



Positional Interpolation RoPE[11] (SuperHOT RoPE)[12]

- Все описанные подходы делают экстраполяцию: обучение на одном диапазоне, работа на другом
- Вместо этого можно вложить увеличенный контекст в тот же диапазон с минимальным дообучением (интерполяция)



[11] Extending Context Window of Large Language Models via Positional Interpolation, 2023

[12] https://kaiokendev.github.io/til#extending-context-to-8k, 2023

Positional Interpolation RoPE

- Вспомним функцию g для получения e_{ij} из RoPE:

$$g(x_i, x_j, i-j) = \mathrm{Re}[(W^Q x_i)(W^K x_j)^* e^{\mathbf{i}(i-j)\theta}]$$

- Можно определить вместо неё новую функцию g':

$$g'(x_i, x_j, i-j) = g\left(x_i, x_j, \frac{(i-j)L}{L'}\right)$$

  где L и L' – исходная и увеличенная длины контекста

- Показано, что для адаптации к новому контексту достаточно дообучения на $\sim 10^4 - 10^5$ последовательностях

- Замеры перплексии и качества на части бенчмарков LLaMA и суммаризации показывают преимущество 1K шагов дообучения с PI над 10K шагами обычного FT

NTK-Aware Scaled RoPE [13]

- В PI RoPE по сути предлагается делать линейную интерполяцию, что не является оптимальным вариантом
- Альтернатива: вместо масштаба изменять основание (которое 10000), и, как следствие, «скорость вращения» векторов:

$$b_{new} = b \cdot \left(\frac{L'}{L}\right)^{\frac{d_z}{d_z-2}}$$

где b и b_{new} – основания, $\frac{L'}{L}$ – фактор масштаба

- Может работать адекватно даже без дообучения
- Оба интерполяционных метода реализованы в transformers (формулы немного отличаются от оригинальных, обсудим далее), классы:
  - `LlamaLinearScalingRotaryEmbedding`
  - `LlamaDynamicNTKScalingRotaryEmbedding`

[13] https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have, 2023

YaRN [15]

- Развитие идеи позиционной интерполяции в RoPE, замечания авторов на основе теории Neural Tangent Kernel (NTK)[14]:
  - для DL-моделей задача выучить многомерный комплексный вектор для кодирования одномерной позиционной информации является проблемной
  - RoPE похож на специальный одномерный вид Fourier Features из NTK
  - равномерное растяжение векторов RoPE (как в PI) приводит к потере высокочастотных деталей, необходимых для различения очень похожих и близких в тексте токенов
  - т.е. минимальный поворот, отличающий позиции, не должен быть слишком маленьким
  - это может быть причиной некоторого падения качества PI RoPE на не-длинных сэмплах после дообучения (чего быть не должно)

[14] Fourier features let networks learn high frequency functions in low dimensional domains, 2020
[15] YaRN: Efficient Context Window Extension of Large Language Models, 2023

YaRN

- Замечания авторов на основе теории Neural Tangent Kernel (NTK):
  - один из способов борьбы с проблемой (простой, но не единственный) – уже описанный выше NTK-Aware Scaled RoPE
  - при сравнении без дообучения этот метод показывает себя лучше PI RoPE
  - но получается не совсем интерполяция, некоторые измерения экстраполируются слишком большими значениями
  - как следствие, при дообучении NTK-Aware уступает обычному PI RoPE
- Введем определение *длины волны* – числа токенов, необходимого для полного прохода RoPE круга 2π в измерении d:

$$\lambda_d = \frac{2\pi}{\theta_d}$$

- RoPE PI и NTK-Aware не учитывают длину волны и считают все измерения RoPE одинаково важными для модели

YaRN

- Наблюдения показывают, что у части измерений $\lambda_d > L$, у части наоборот, дисбаланс может быть сильным
- С каждым измерением можно работать по-своему в заивимости от его длины волны
- В методе NTK-by-parts RoPE[16] предлагается:
  - если $\lambda_d \ll L$ – не интерполировать
  - если $\lambda_d \geqslant L$ – интерполировать без экстраполяции (как в PI RoPE)
  - для прочих делать обычную NTK-Aware интерполяцию
- Работает лучше PI RoPE и NTK-Aware и с дообучением, и без

[16] https://github.com/jquesnelle/scaled-rope/pull/1, 2023

YaRN

- Следующий шаг – Dynamic NTK[17]:
  - при обучении используются последовательности разной длины
  - можно фиксировать масштаб L'/L для всех сэмплов
  - а можно опираться на длину сэмпла ℓ': $\max(1, \ell'/L)$
  - этот подход повышает устойчивость модели к изменению контекста в обе стороны
- Утверждается, что добавление температуры t в знаменатель формулы для e_{ij} хорошо влияет на перплексию при расширении контекста
- YaRN = NTK-by-parts RoPE + температура при подсчёте логитов (совместим с оптимизациями внимания типа Flash Attention)
- Для моделей LLaMA и LLaMA 2 рекомендуется брать t по формуле:

$$\sqrt{\frac{1}{t}} = 0.1 \ln\left(\frac{L'}{L}\right) + 1$$

[17] https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases, 2023

YaRN

Extension Method	Trained Tokens	Context Window	Evaluation Context Window Size				
			2048	4096	6144	8192	10240
PI ($s = 2$)	1B	8k	3.92	3.51	**3.51**	**3.34**	8.07
NTK ($\theta = 20$k)	1B	8k	4.20	3.75	3.74	3.59	6.24
YaRN ($s = 2$)	400M	8k	**3.91**	**3.50**	**3.51**	3.35	**6.04**

Table 1: Sliding window perplexity ($S = 256$) of ten 128k Proof-pile documents over Llama-2 extended via PI, NTK and YaRN

Model Size	Model Name	Context Window	Extension Method	ARC-c	Hellaswag	MMLU	TruthfulQA
7B	Llama 2	4k	None	**53.1**	77.8	**43.8**	39.0
7B	Together	32k	PI	47.6	76.1	43.3	**39.2**
7B	Code Llama	100k	NTK	39.9	60.8	31.1	37.8
7B	YaRN ($s = 16$)	64k	YaRN	52.3	**78.8**	42.5	38.2
7B	YaRN ($s = 32$)	128k	YaRN	52.1	78.4	41.7	37.3
13B	Llama 2	4k	None	**59.4**	82.1	**55.8**	37.4
13B	Code Llama	100k	NTK	40.9	63.4	32.8	**43.8**
13B	YaRN ($s = 16$)	64k	YaRN	58.1	**82.3**	52.8	37.8
13B	YaRN ($s = 32$)	128k	YaRN	58.0	82.2	51.9	37.3

Table 3: Performance of context window extensions methods on the Hugging Face Open LLM benchmark suite compared with original Llama 2 baselines

Спасибо за внимание!