ecom.tech

Эволюция Transformer: как меняется самая успешная архитектура в DL

Мурат Апишев, ecom.tech

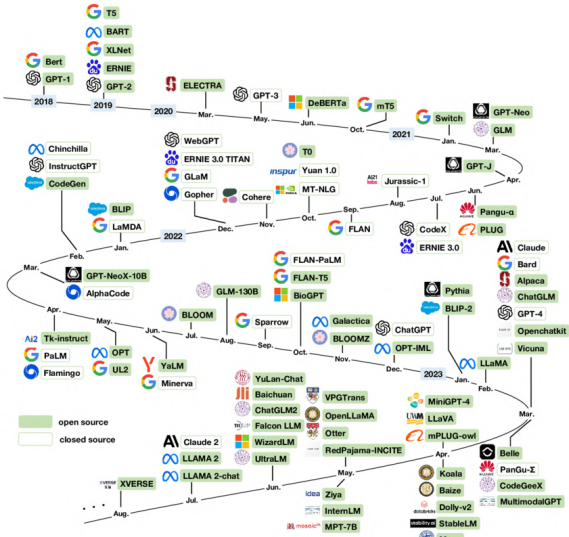
AiConf 2024

О чём будет рассказ

- LLM – основа современного NLP и AI

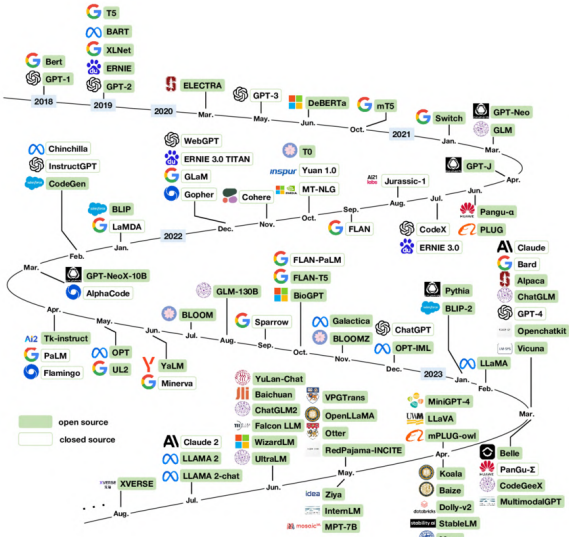
О чём будет рассказ

- Доминирующая архитектура — Transformer
- Много архитектурных модификаций и внедрений с 2017

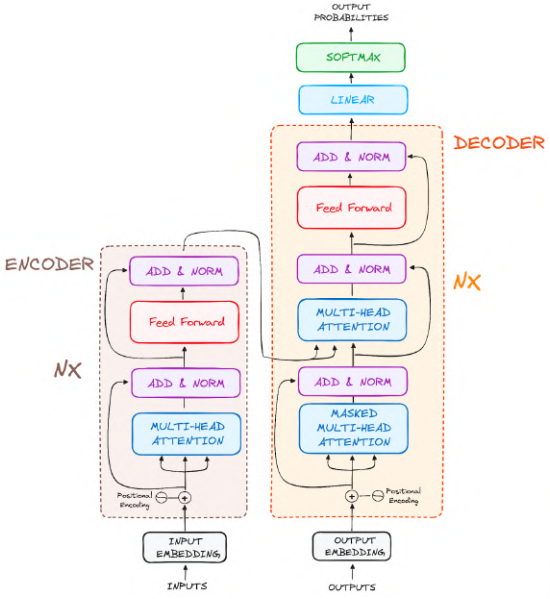


О чём будет рассказ

- **Цель доклада**: рассмотреть наиболее значимые и интересные из идей

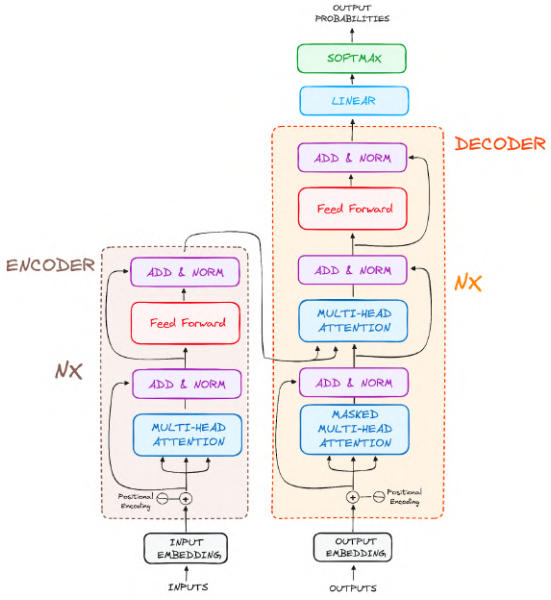


О чём будет рассказ



О чём будет рассказ

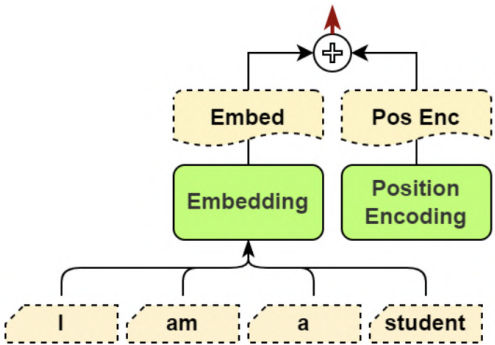
- Позиционное кодирование
- Нормализация
- Self-Attention
- Полносвязные слои



Позиционное кодирование

Позиционное кодирование: абсолютный подход

- Без информации о позициях токенов любой Transformer работает плохо
- Абсолютное кодирование
  - один вектор на одну позицию
  - вектор добавляется в вектору эмбеддинга на входе сети



Позиционное кодирование: абсолютный подход

- Без информации о позициях токенов любой Transformer работает плохо
- Абсолютное кодирование – источник:
  - обучаемые векторы
  - синусоидальные векторы

Позиционное кодирование: абсолютный подход

- Без информации о позициях токенов любой Transformer работает плохо
- Абсолютное кодирование – просто, но есть проблемы:
  - низкое качество кодирования $\Rightarrow$ хуже результаты
  - низкое качество экстраполяции $\Rightarrow$ нет обобщения на больший контекст

Позиционное кодирование: относительный подход

- Кодируется не позиция токена, а расстояние между парой
- Вместо эмбеддинга на входе – модификация self-attention

Позиционное кодирование: относительный подход

- Кодируется не позиция токена, а расстояние между парой
- Вместо эмбеддинга на входе – модификация self-attention
- Вектор расстояния прибавляется
  - при подсчёте логита веса внимания к вектору ключа
  - при подсчёте выхода внимания к вектору значения
- Векторы общие для голов и обучаются на каждое возможное расстояние

Позиционное кодирование: относительный подход

- Кодируется не позиция токена, а расстояние между парой
- Вместо эмбеддинга на входе – модификация self-attention
- Можно переписать формулу подсчёта внимания, репараметризовать и убрать зависимость от абсолютной позиции
- Разные части формулы и репараметризации дают кодирования Transformer-XL и DeBERTA

Позиционное кодирование: сдвиги

- Ещё проще и эффективнее: прибавить скаляр к логиту веса внимания

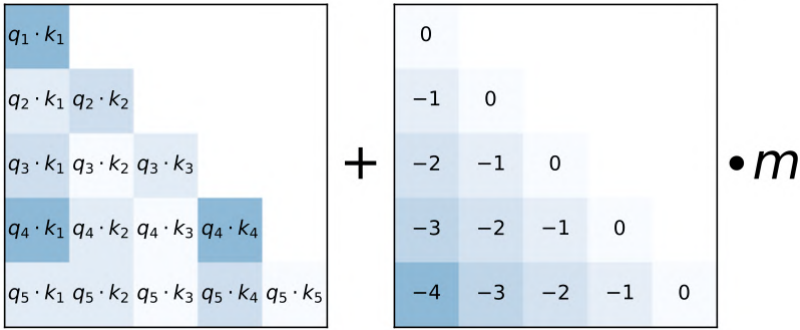
Позиционное кодирование: сдвиги

- Ещё проще и эффективнее: прибавить скаляр к логиту веса внимания
- В T5 сдвиги – 32 скаляра, распределённых по 128 расстояниям $i - j$

- Скаляры свои у каждой головы внимания, но общие для всех слоёв

Позиционное кодирование: сдвиги

- Ещё проще и эффективнее: прибавить скаляр к логиту веса внимания
- В AliBi скаляры для $i - j$ не обучаются, а считаются как $m(i - j)$
- m – заданное число, своё для каждой головы и общее для слоёв:



Позиционное кодирование: вращения

- RoPE – самый популярный метод позиционного кодирования
- Почти все современные LLM обучаются с ним или его модификациями

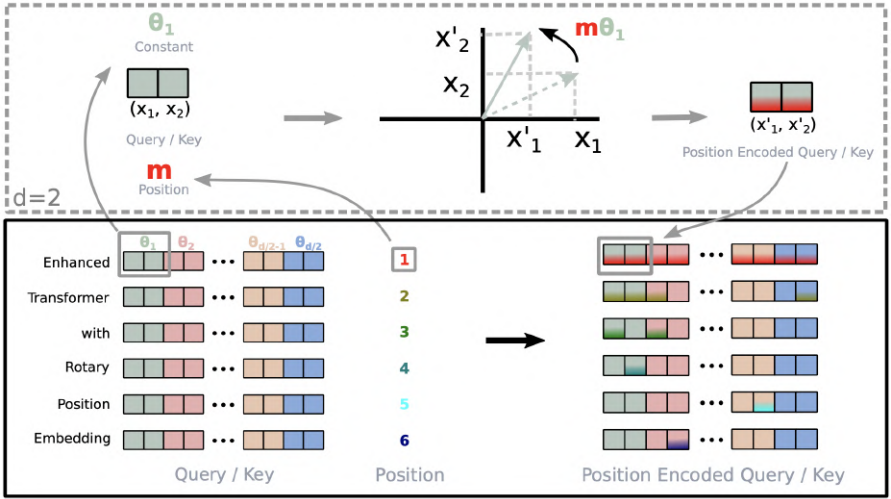
Позиционное кодирование: вращения

- RoPE – самый популярный метод позиционного кодирования

- Применение RoPE = поворот вектора запроса/ключа на некоторый угол
- Значение угла зависит от индекса его позиции
- Поворот обоих векторов на один угол не изменит вес внимания
- Это просто смещение позиций без изменения расстояния

Позиционное кодирование: вращения

- Применение RoPE = поворот вектора запроса/ключа на некоторый угол

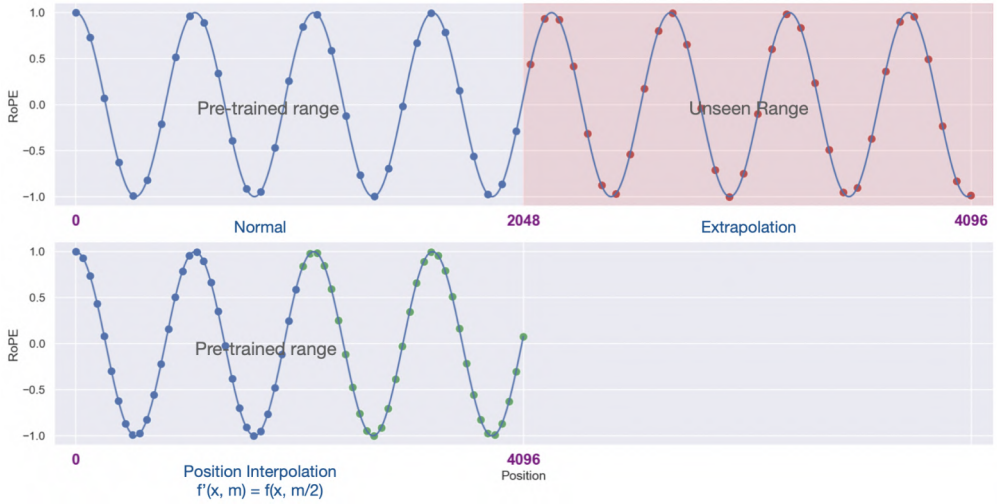


Позиционное кодирование: вращения с интерполяциями

- RoPE тоже слабо адаптируется к росту длины контекста
- **Идея:**
  - обучить модель с RoPE
  - вложить увеличенный контекст в тот же диапазон
  - опционально сделать небольшое дообучение
- Одновременно предложено в PI RoPE и SuperHOT RoPE

Позиционное кодирование: вращения с интерполяциями

- **Идея:** вложить увеличенный контекст в тот же диапазон



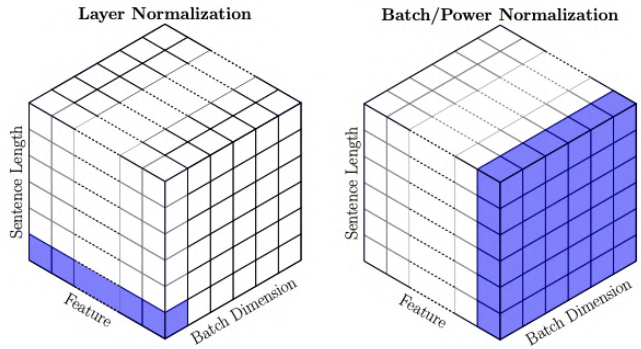
Позиционное кодирование: вращения с интерполяциями

- Есть разные подходы к расширению диапазона расстояний
  - точная настройка сжатия для каждой пары признаков в RoPE
  - сохранение качества на исходном диапазоне
  - повышение качества на расширенном диапазоне
  - работа в zero-shot

Нормализация

Нормализация: LayerNorm

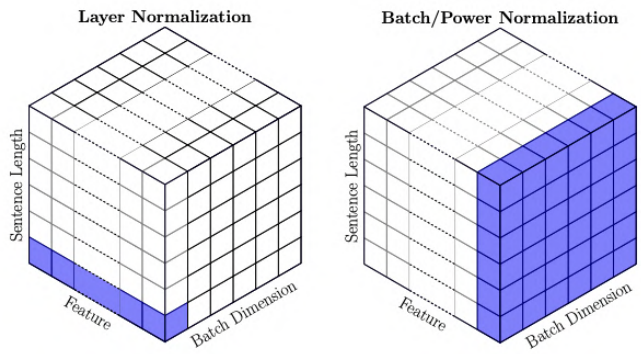
- Нормализация повышает скорость и стабильность обучения
- В моделях CV распространена более старая Batch-нормализация



Layer Normalization, 2016
PowerNorm: Rethinking Batch Normalization in Transformers, 2020

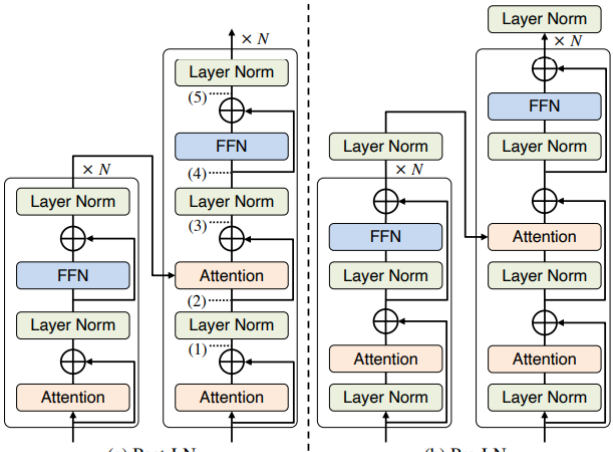
Нормализация: LayerNorm

- В RNN и Transformer использовать Batch-нормализацию неудобно:
  - нужны большие батчи для сбора статистики
  - в NLP у последовательностей разная длина $\rightarrow$ флуктуации статистик
- Альтернатива – Layer-нормализация вектора каждого токена



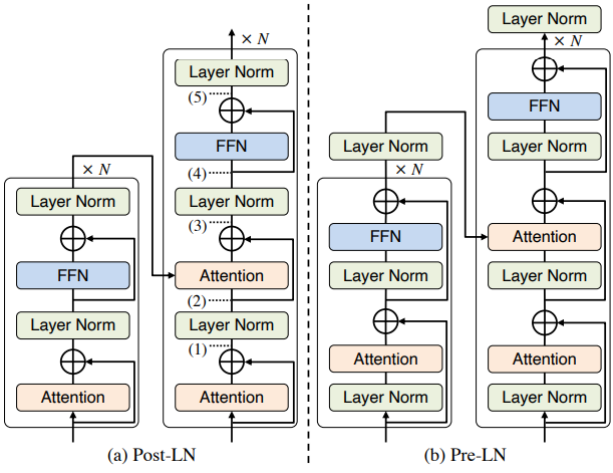
Нормализация: порядок встраивания

- В исходном Transformer нормализация после MHSA и FFN (Post-LN)
- Ниже стабильность и выше влияние LR и warmup



Нормализация: порядок встраивания

- В современных моделях доминирует Pre-LN
- Вариации типа Sub-LN в Magneto непопулярны



Нормализация: RMSNorm

- LayerNorm – центрирование и нормирование на отклонение активаций a
- Изначально считалось, что обе операции существенно важны

Нормализация: RMSNorm

- LayerNorm – центрирование и нормирование на отклонение активаций a
- Изначально считалось, что обе операции существенно важны
- RMSNorm – LayerNorm без центрирования:

$$\hat{a}_i = \frac{a_i - \cancel{\mu}}{\sigma}, \quad \cancel{\mu = \frac{1}{n}\sum_{i=1}^{n} a_i}, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \cancel{\mu})^2} = \mathrm{RMS}(a)$$

Нормализация: RMSNorm

- LayerNorm – центрирование и нормирование на отклонение активаций a
- Изначально считалось, что обе операции существенно важны
- RMSNorm – LayerNorm без центрирования:

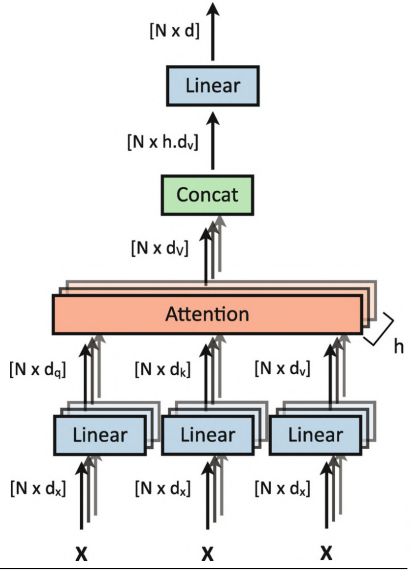
$$\hat{a}_i = \frac{a_i - \cancel{\mu}}{\sigma}, \quad \cancel{\mu = \frac{1}{n}\sum_{i=1}^{n} a_i}, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \cancel{\mu})^2} = \mathrm{RMS}(a)$$

- Качество то же, но быстрее
- RMSNorm – стандарт для современных LLM (LLaMA, Qwen, Mistral)

Model	Time
Baseline	315±6.30s
BatchNorm-Everywhere	348±10.5s
BatchNorm-LSTM	345±11.2s
LayerNorm	392±5.70s
RMSNorm	333±5.20s (15.1%)
pRMSNorm	330±5.50s (15.8%)

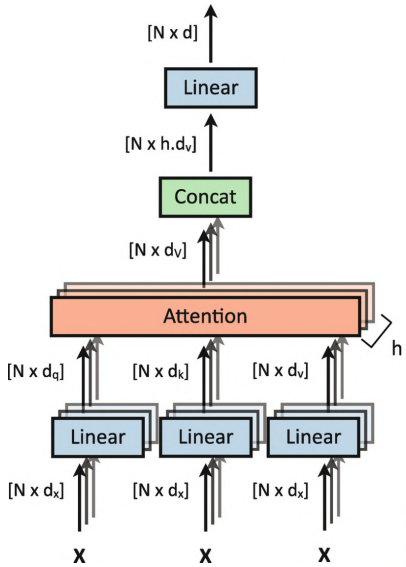
Механизм внимания

Механизм внимания: базовый MHSA



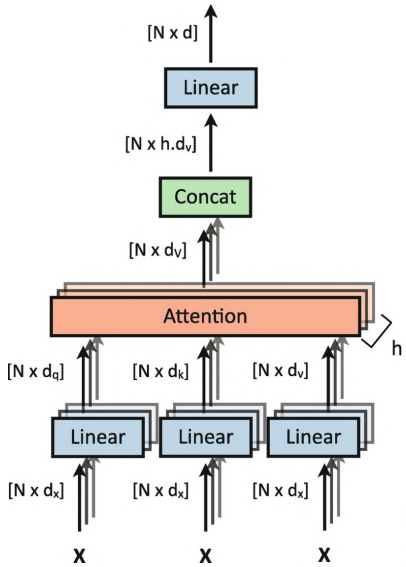
Механизм внимания: базовый MHSA

- Основные проблемы:
  - сложность подсчёта SA $O(n^2)$
  - потребление памяти при кэшировании векторов K и V
- Прямое увеличение длины контекста неэффективно



Механизм внимания: базовый MHSA

- Идеи модификаций MHSA:
  - разреженное и локальное внимание
  - рекуррентная обработка
  - понижение размерностей векторов
  - использование KNN-индексов
  - иерархическая обработка
  - уменьшение числа наборов K и V

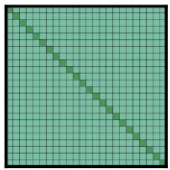


Механизм внимания: разреженность и локальность

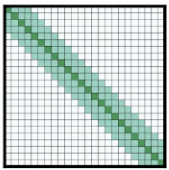
- Матрица весов внимания существенно разреженная
- Если вычислять только нужные подмножества, можно существенно уменьшить вычисления с небольшим ухудшением качества

Механизм внимания: разреженность и локальность

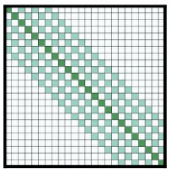
- Основные варианты подсчёта весов внимания для токена:
  - с непосредственными соседями (внутри групп или SWA)
  - с соседями с заданным Dilation Rate разреживанием (которое можно усиливать на верхних слоях)
  - со случайными токенами последовательности
  - с глобальными токенами (они со всеми, все с ними)



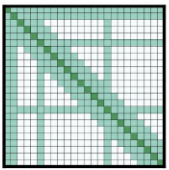
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



(d) Global+sliding window

Механизм внимания: разреженность и локальность

- Основные варианты подсчёта весов внимания для токена:
  - с непосредственными соседями (внутри групп или SWA)
  - с соседями с заданным Dilation Rate разреживанием (которое можно усиливать на верхних слоях)
  - со случайными токенами последовательности
  - с глобальными токенами (они со всеми, все с ними)
- Работы: Sparse Transformer, Longformer, Big Bird, LongT5, LongNet
- Те же идеи используют GPT-3, Qwen, Mistral, Qwen 2

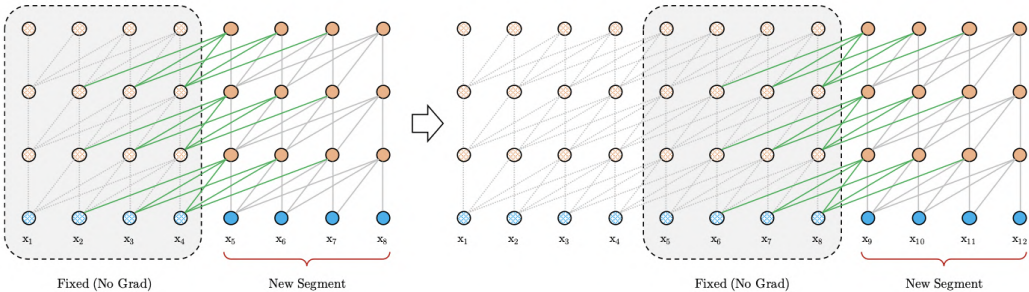
Generating Long Sequences with Sparse Transformers, 2019
Longformer: The Long-Document Transformer, 2020
Big Bird: Transformers for Longer Sequences, 2020
LongT5: Efficient Text-To-Text Transformer for Long Sequences, 2022
LongNet: Scaling Transformers to 1,000,000,000 Tokens, 2023
Language Models are Few-Shot Learners, 2020
Qwen Technical Report, 2023
Mistral 7B, 2023
Qwen2 Technical Report, 2024

Механизм внимания: рекурентность по сегментам

- **Идея:** разделить длинную последовательность на сегменты, обрабатывать один за другим и передавать вперёд информацию

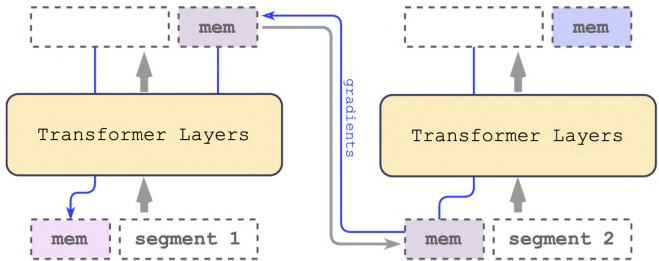
Механизм внимания: рекурентность по сегментам

- Transformer-XL:
  - выходы SA сегмента i идут в кэш и используются для $(i+1)$-го
  - выходы двух сегментов конкатенируются, по ним считаются K и V
  - Q – только по токенам текущего сегмента, градиенты тоже только по ним



Механизм внимания: рекурентность по сегментам

- RMT:
  - в начало каждого сегмента добавляются M токенов – «векторов памяти»
  - кодировщик – выходные векторы i-го сегмента для этих токенов идут первыми на вход для $i + 1$-го сегмента
  - декодировщик – векторы памяти есть ещё и в конце, первые несут информацию из $i - 1$-го сегмента, а векторы последних понесут в $i + 1$-й

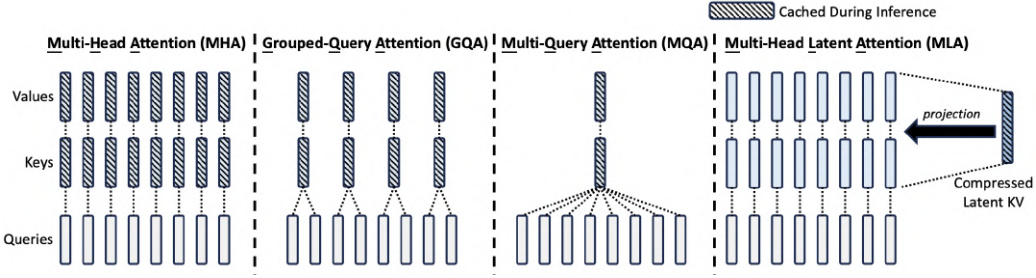


Механизм внимания: число ключей и значений

- Q должны быть свои для каждой головы внимания, K и V – нет, их можно разделять между головами для экономии вычислений и памяти

Механизм внимания: число ключей и значений

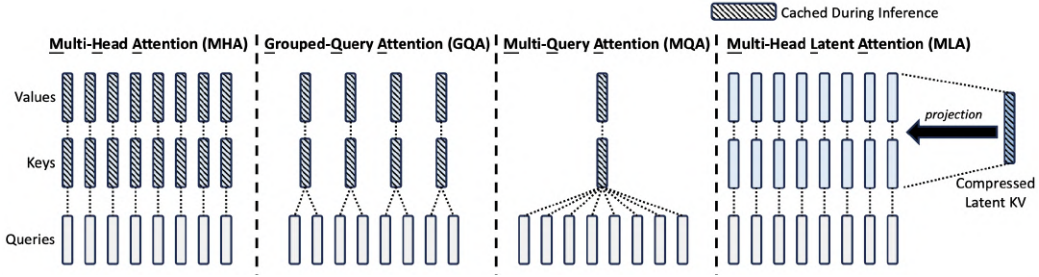
- MQA – один набор K и V на все головы



Fast Transformer Decoding: One Write-Head is All You Need, 2019
DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model, 2024

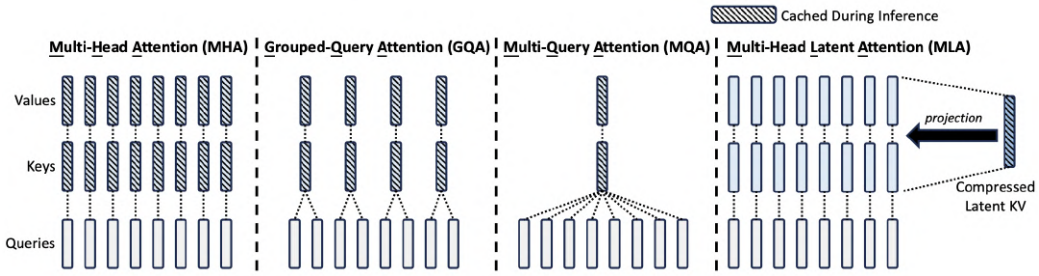
Механизм внимания: число ключей и значений

- MQA – один набор K и V на все головы
- GQA – один набор K и V на каждую группу голов



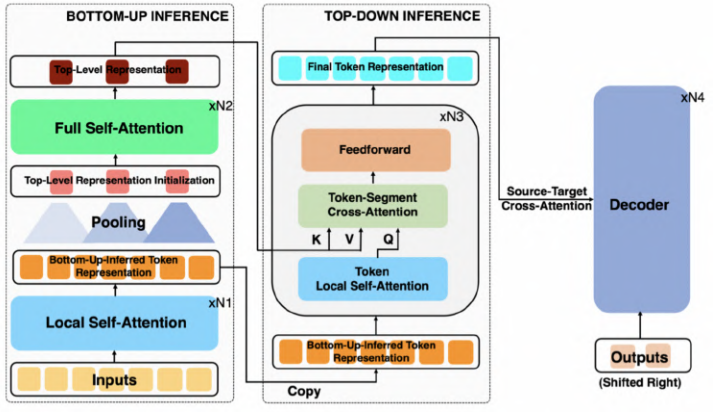
Механизм внимания: число ключей и значений

- MQA – один набор K и V на все головы
- GQA – один набор K и V на каждую группу голов
- MLA
  - на токен вычисляется и кэшируется один вектор большей размерности
  - он проектируется общей весовой матрицей в набор K/V для всех голов



Механизм внимания: иерархическая обработка

- Top Down Transformer: локальное внимание + полный Cross-SA



- SLED: локальное внимание с контекстом, Cross-SA в декодировщике

Механизм внимания: иерархическая обработка

- Кодировщик:
  - параллельный self-attention для всего входа
  - полный пересчёт для генерации токена

Механизм внимания: иерархическая обработка

- Кодировщик:
  - параллельный self-attention для всего входа
  - полный пересчёт для генерации токена
- Декодировщик:
  - эффективный self-attention для генерации токена
  - большие кэши и долгая обработка префикса

Механизм внимания: иерархическая обработка

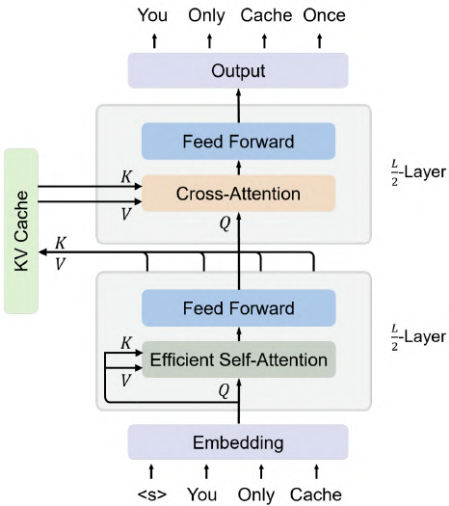
- Кодировщик:
  - параллельный self-attention для всего входа
  - полный пересчёт для генерации токена
- Декодировщик:
  - эффективный self-attention для генерации токена
  - большие кэши и долгая обработка префикса
- Полный Transformer:
  - быстрая обрабатка префикса
  - эффективный self-attention для генерации токена
  - большие кэши и долгая обработка префикса
  - кодировщик не используется при генерации

Механизм внимания: иерархическая обработка

- **Идея:** архитектура типа «кодировщик-декодировщик», но с
  - авторегрессионной обработкой префикса
  - использованием кодировщика и для генерируемых токенов
  - уменьшением размеров кэшей в декодировщике

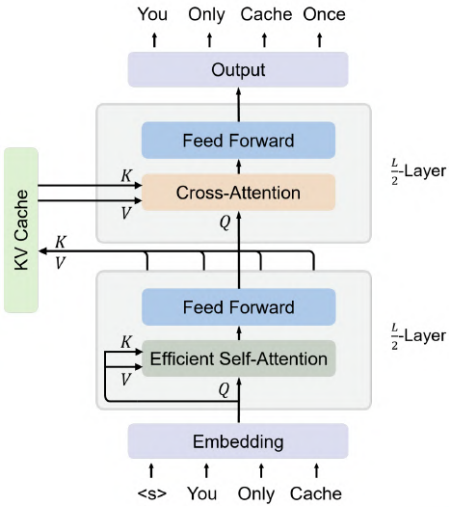
Механизм внимания: иерархическая обработка

- **YOCO**: декодировщик разбивается пополам по слоям
- Нижний – обычный декодировщик, но с SWA работает быстро и с кэшем $O(1)$



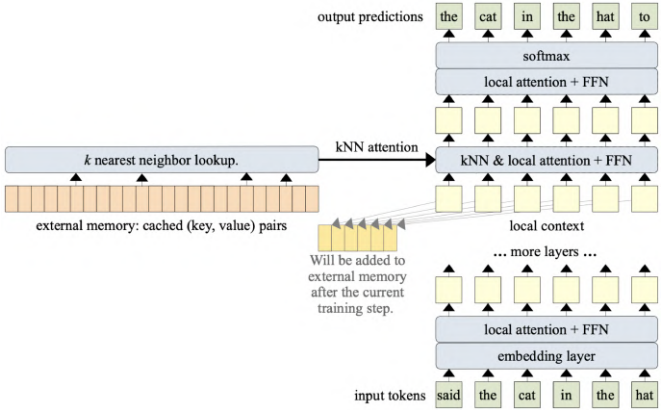
Механизм внимания: иерархическая обработка

- **YOCO**: декодировщик разбивается пополам по слоям
- Верхний – полный MHSA, но с Cross-SA, с общими на всех слоях K и V из кэша
- Префикс обрабатывается только нижним блоком – ускорение



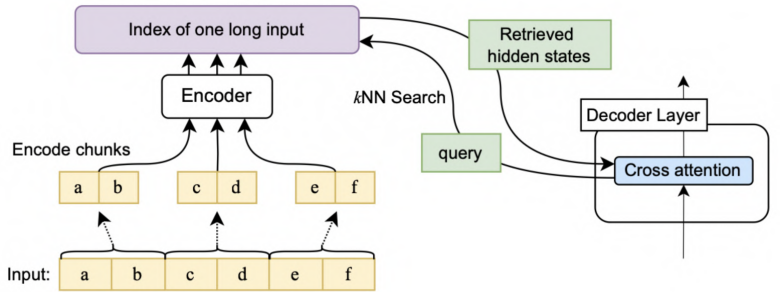
Механизм внимания: сегменты и kNN-индексы

- Memorizing Transformer: в последнем блоке kNN-Augmented Attention
  - блок памяти для K и V размера M, добавление в конец
  - складываются результаты обычного MHSA и с K и V из kNN



Механизм внимания: сегменты и kNN-индексы

- Unlimiformer: полный Transformer, локальное внимание в кодировщике, выходы h_e кэшируются
- kNN-Augmented Attention с Cross-SA в слоях декодировщика
- Хранение h_e (общих для всех слоёв) вместо K и V экономит память
- K и V легко получаются изменением порядка подсчёта QK^T



Механизм внимания: RetNet

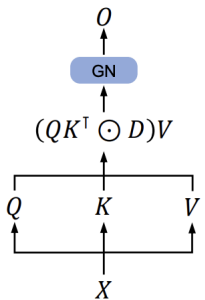
- RetNet похож на Transformer, вместо MHSA – MH Retention

Механизм внимания: RetNet

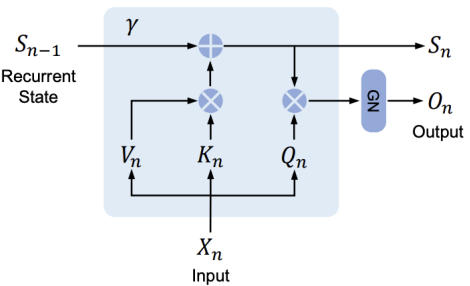
- RetNet похож на Transformer, вместо MHSA – MH Retention
- Retention – механизм преобразования последовательности
- При ограничениях на вид параметров может вычисляться
  - рекурентно (как RNN)
  - параллельно (как MHSA)

Механизм внимания: RetNet

- Retention – механизм преобразования последовательности
- При ограничениях на вид параметров может вычисляться
  - рекурентно (как RNN)
  - параллельно (как MHSA)

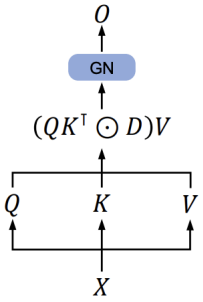


(a) Parallel representation.

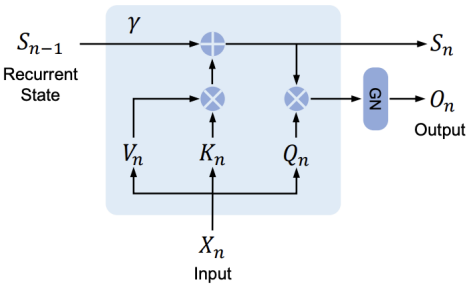


(b) Recurrent representation.

Механизм внимания: RetNet



(a) Parallel representation.



(b) Recurrent representation.

- Гибридное представление для обучения на длинном входе:
  - вход разбивается на части, каждая обрабатывается параллельно
  - между частями обработка рекуррентная (родственно Cross-SA)

Механизм внимания: State Space Models

- Transformer $\approx$ Self-Attention, вместо оптимизации можно заменить
- Transformer: качественно, но неэффективно (запоминается весь вход)
- RNN: некачественно, но эффективно (запоминается вектор состояния)
- SSM: качественно и эффективно

Механизм внимания: State Space Models

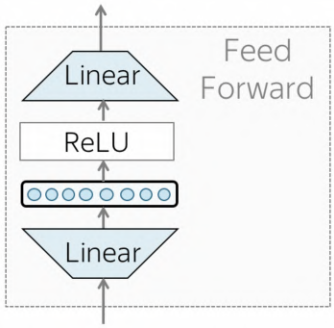
- Transformer $\approx$ Self-Attention, вместо оптимизации можно заменить
- Transformer: качественно, но неэффективно (запоминается весь вход)
- RNN: некачественно, но эффективно (запоминается вектор состояния)
- SSM: качественно и эффективно
- Модели, родственные Linear RNN и Linear Attention
- Разные представления позволяют эффективное обучение и инференс
- В обзоре Mamba-360 таксономия из более 20 моделей в 4 группах

Mamba-360: Survey of State Space Models as Transformer Alternative for Long Sequence Modelling: Methods, App..., 2024
Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention, 2020
Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality, 2024

Полносвязные слои

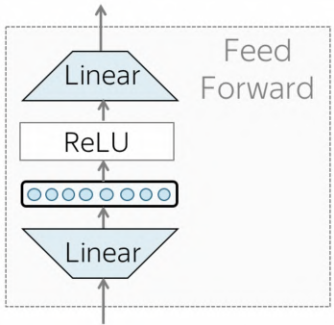
Полносвязные слои: базовый FFN

- Базовый вариант (без bias): $\max(xW_1, 0)W_2$
- FFN – до 2/3 от общего числа параметров сети



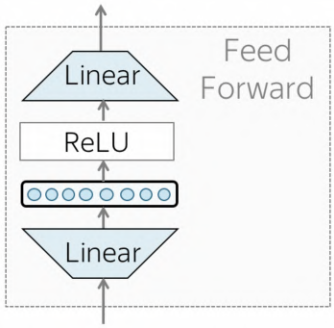
Полносвязные слои: базовый FFN

- Базовый вариант (без bias): $\max(xW_1, 0)W_2$
- FFN – до 2/3 от общего числа параметров сети
- Self-attention взаимная важность токенов
- FFN обрабатывает эту информацию и содержит основные «знания» модели, включая фактологию



Полносвязные слои: базовый FFN

- Базовый вариант (без bias): $\max(xW_1, 0)W_2$
- FFN – до 2/3 от общего числа параметров сети
- Self-attention взаимная важность токенов
- FFN обрабатывает эту информацию и содержит основные «знания» модели, включая фактологию



- FFN можно рассматривать как KV-хранилище, первый слой содержит ключи, второй – значения
- Пробуют редактировать факты изменением весов FFN обученной модели

Transformer Feed-Forward Layers Are Key-Value Memories, 2021
Mass-Editing Memory in a Transformer, 2023
Locating and Editing Factual Associations in GPT, 2023

Полносвязные слои: функции активации

- FFN с ReLU – не единственный возможный вариант блока с полносвязными слоями

Полносвязные слои: функции активации

- FFN с ReLU – не единственный возможный вариант блока с полносвязными слоями
- Пробуют различные активации:
  - $\mathrm{GeLU} = x\Phi(x)$, $\Phi(x)$ – функция распределения $\mathcal{N}(0, 1)$
  - $\mathrm{Swish} = x\sigma(\beta x)$, $\sigma(x)$ – сигмоида, β – гиперпараметр (при $\beta = 1$ – SiLU)
  - $\mathrm{Mish} = x \tanh(\mathrm{softplus}(x))$, $\mathrm{softplus}(x) = \ln(1 + x)$

Полносвязные слои: функции активации

- FFN с ReLU – не единственный возможный вариант блока с полносвязными слоями
- Пробуют различные активации:
  - $\mathrm{GeLU} = x\Phi(x)$, $\Phi(x)$ – функция распределения $\mathcal{N}(0, 1)$
  - $\mathrm{Swish} = x\sigma(\beta x)$, $\sigma(x)$ – сигмоида, β – гиперпараметр (при $\beta = 1$ – SiLU)
  - $\mathrm{Mish} = x\tanh(\mathrm{softplus}(x))$, $\mathrm{softplus}(x) = \ln(1 + x)$
- Все функции – гладкие модификации ReLU
  - нет проблемы большого числа нулевых бесполезных нейронов
  - нет резкого перехода в нуле
- Нет однозначного мнения о том, какая из них лучше
- GeLU используется в FFN блоках BERT и GPT-2

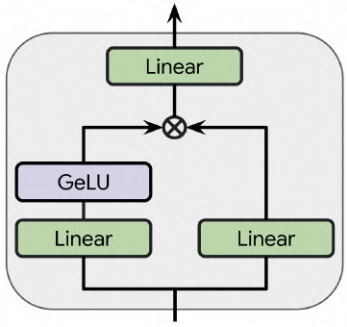
Полносвязные слои: GLU

- Gated Linear Units:

$$\mathrm{GLU}(x) = \sigma(xW_1) \otimes xW_2,$$

$\otimes$ – покомпонентное умножение

- В GLU тоже можно заменять активации, популярные варианты
GeGLU (GeLU) и SwiGLU (Swish / SiLU)



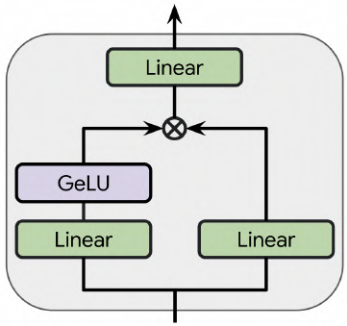
Полносвязные слои: GLU

- Gated Linear Units:

  $$\mathrm{GLU}(x) = \sigma(xW_1) \otimes xW_2,$$

  $\otimes$ – покомпонентное умножение

- В GLU тоже можно заменять активации, популярные варианты
  GeGLU (GeLU) и SwiGLU (Swish / SiLU)



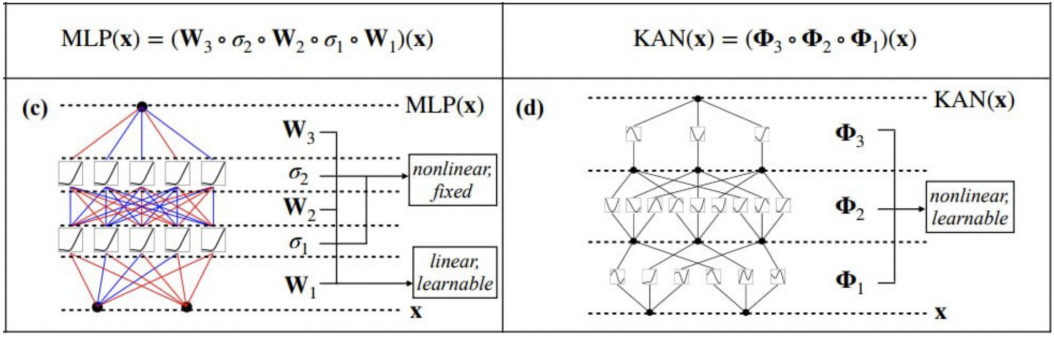
- В FFN первый слой и активации заменяются на GLU
- Три матрицы весов, для сохранения общего числа параметров внутренняя размерность блока уменьшается на треть
- Модификация активно используется (LLaMA, Qwen, Mistral, Griffin)

Полносвязные слои: KAN

- Основная задача нейронной сети – приблизить на основе данных некоторую функцию в многомерном пространстве аргументов
- FFN работают благодаря универсальной теореме аппроксимации
- Альтернативный подход – теорема Колмогорова-Арнольда:

Любая непрерывная функция многих переменных может быть представлена в виде суперпозиции непрерывных функций одной переменной с использованием сложения

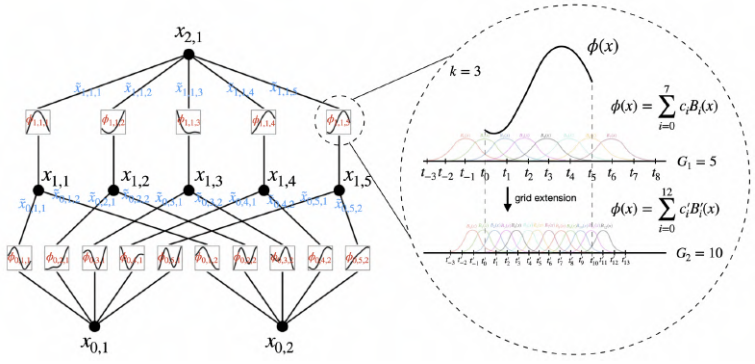
Полносвязные слои: KAN



- KAN:
  - рёбра сети – обучаемые активации
  - узлы сети – операция сложения

Полносвязные слои: KAN

- Функция активации – взвешенная сумма двух слагаемых:
  - residual connection с активацией
  - линейная комбинация сплайнов



- Веса слагаемых и линейной комбинации обучаемые

Полносвязные слои: KAN

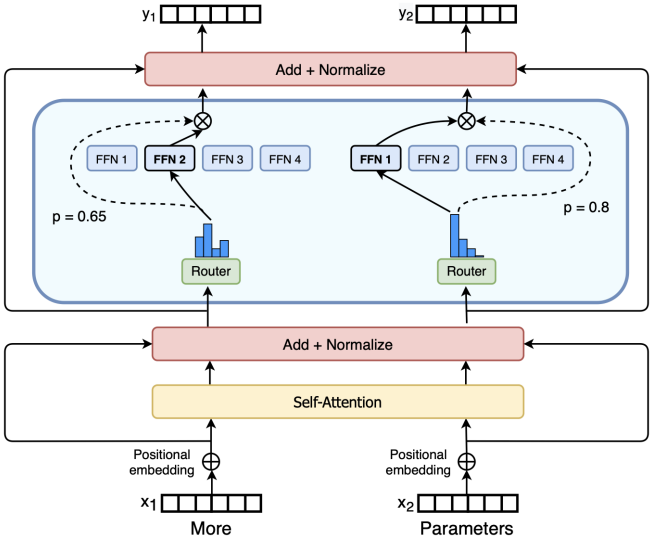
- KAN выглядит перспективно и активно развивается
- Модификации сети: полиномы Чебышева, вейвлеты или коэффициенты Фурье вместо сплайнов
- Повышение производительности: efficient-kan
- Но экспериментальных доказательств превосходства над FFN в общем случае недостаточно, часть результатов противоречивы

Chebyshev Polynomial-Based Kolmogorov-Arnold Networks, 2024
https://github.com/GistNoesis/FourierKAN
Wav-KAN: Wavelet Kolmogorov-Arnold Networks, 2024
https://github.com/Blealtan/efficient-kan

Полносвязные слои: Mixture-of-Experts

- Если учить оптимально, то больше весов $\Rightarrow$ выше качество
- Но больше весов $\Rightarrow$ медленнее инференс
- Решение – Mixture-of-Experts, «экспертами» становятся FFN-слои

Полносвязные слои: Mixture-of-Experts



Полносвязные слои: Mixture-of-Experts

- Параметров сильно больше, но при обработке каждого токена активируется небольшая часть
- Эксперт учится решать свои типы задач, имеет свои доменные знания

Полносвязные слои: проблемы MoE

- Нагрузка распределяется по экспертам неравномерно, сеть вырождается, качество падает

Полносвязные слои: проблемы MoE

- Вместо доменной информации эксперты улавливают родственные токены: имена, артикли, спецсимволы, определённые части речи

Punctuation	Layer 2 Layer 6	, , , , , , , , , - , , , , ,).) , , , , , : . : , & , & & ? & - , , ? , , , . <extra_id_27>
Conjunctions and articles	Layer 3 Layer 6	The the the the the the the the the The the the the the the The the the the a and and and and and and and or and a and . the the if ? a designed does been is not
Verbs	Layer 1	died falling identified fell closed left posted lost felt left said read miss place struggling falling signed died falling designed based disagree submitted develop

Полносвязные слои: развитие MoE

- Регуляризация loss для Router для балансировки распределения и добавление ограничений на ёмкость эксперта

Полносвязные слои: развитие MoE, Mixtral

- Популярная открытая сравнительно небольшая модель – Mixtral
  - 8 экспертов – блоков FFN со SwiGLU, 2 активных на токен
  - 47B параметров, 13B активных на токен
  - эффективная реализация за счёт разреженного матричного умножения

Полносвязные слои: развитие MoE, Mixtral

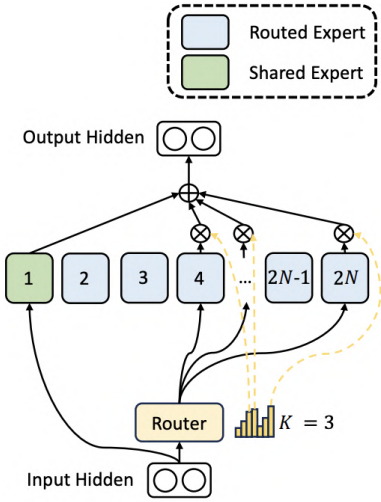
- Популярная открытая сравнительно небольшая модель – Mixtral
  - 8 экспертов – блоков FFN со SwiGLU, 2 активных на токен
  - 47B параметров, 13B активных на токен
  - эффективная реализация за счёт разреженного матричного умножения

```
 Question: Solve -42*r + 27*c = -1167 and 130*r
Answer: 4

Question: Calculate -841880142.544 + 411127.
Answer: -841469015.544

Question: Let x(g) = 9*g + 1. Let q(c) = 2*c +
Answer: 54*a - 30
```

```
 A model airplane flies slower when flying into th
wind and faster with wind at its back. When launch
right angles to the wind, a cross wind, its ground
compared with flying in still air is
(A) the same (B) greater (C) less (D) either grea
or less depending on wind speed
```

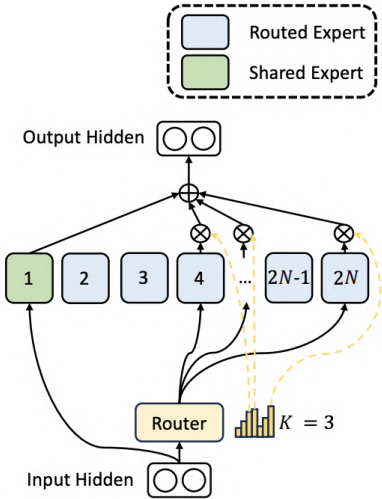
Полносвязные слои: развитие MoE, DeepSeekMoE

- Вектор токена разделяется на части
- Каждая часть идёт к своему эксперту
- Больше экспертов, меньше размерность
- Общие эксперты для всех токенов



Полносвязные слои: развитие MoE, DeepSeekMoE

- Тот же объём вычислений и число параметров – выше качество и скорость инференса, чем у обычного MoE
- Эксперты более доменные и важные – удаление 2-3 ощутимо роняет перплексию
- В версии 2 добавляется Multi-Head Latent Attention для уменьшения кэшей

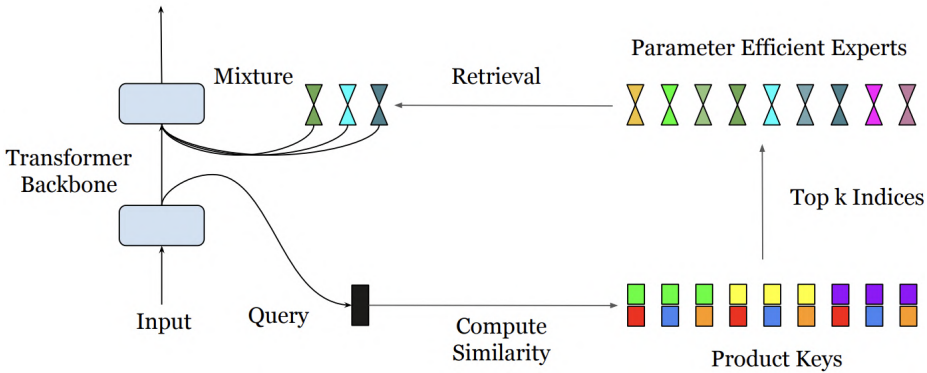


Полносвязные слои: развитие MoE, PEER

- Много небольших узких экспертов лучше, чем мало больших и общих
- Гипотезы:
  - выше качество
  - лучше интерпретируемость
  - возможность обучения на потоках данных без забывания
- PEER – MoE с большим числом экспертов из одного нейрона

Полносвязные слои: развитие MoE, PEER

- Вектор токена x проектируется в запрос (как в MHSA)
- У каждого эксперта есть вектор ключа, отбор топ-K экспертов
- Ответ – $\sigma(u^T x)v$, u, v – параметры эксперта
- Векторы выбранных экспертов складываются с весами из softmax



Полносвязные слои: развитие MoE, PEER

- Если $K = 1$, то h голов такого MoE равны по размеру одному эксперту обычного MoE с h нейронами
- Но вместо фиксированной матрицы – разные комбинации строк для разных токенов и запросных весов
- Слой может заменять FFN или встраиваться между обученными блоками
- Направление выглядит перспективно, но тестов ещё недостаточно

ecom.tech

https://shorturl.at/hgaYK

Текущий рецепт LLM:

- Byte-level BPE
- RoPE и его вариации
- pre-LayerNorm, RMSNorm
- SwiGLU
- Sparse Attention, SWA, GQA
- MoE

Мурат Апишев, ecom.tech
Технический руководитель
поиска для «Мегамаркета»
mel-lain@yandex.ru

AiConf 2024