Модификации Transformer для обработки длинных последовательностей

Мурат Апишев (mel-lain@yandex.ru)

Июль, 2023

Обработка длинных последовательностей

- **Проблема**:
  - квадратичная по длине последовательности сложность self-attention
  - в ряде задач длина последовательности на входе и/или выходе может в десятки и сотни раз превышать ограничения модели
- **Пути решения**:
  - оптимизация подсчёта внимания
  - иерархическая обработка последовательности
  - добавление рекурентности
- **Примеры работ**:
  - Sparse Transformer
  - Longformer
  - Reformer
  - Linformer
  - BigBird
  - LongT5
  - Flash Attention
  - Unlimiformer

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Carnegie Mellon University, 2019

- Разбиение входа на сегменты, обрабатываемые рекурентно:
  - выходы очередного сегмента токенов кэшируются на всех слоях
  - при обработке $i-$го сегмента токенов есть возможность использовать выходы предыдущего
  - выходы двух сегментов конкатенируются, по ним считаются K и V
  - Q считаются только по токенам текущего сегмента, градиент тоже течёт только через эти токены
- Рекурентность не работает с абсолютными позиционными векторами
- Кодирование позиции переходит в репараметризованную формулу подсчёта self-attention с фиксированной синусоидальной матрицей относительных расстояний и парой общих обучаемых векторов

Generating Long Sequences with Sparse Transformers

OpenAI, 2019

- На обучении моделей локальный self-attention: при обработке токена внимание обращается не на всю последовательность, а на контекст
- Контекст определяется ядром:
  - сегмент до текущего токена (если вход двумерный – по горизонтали или вертикали)
  - фиксированные токены, доступные всем более поздним токенам (например, каждые 128 токенов брать 8 фиксированных, т.е. более поздние токены смотрят на большее число фиксированных)
- Подбором параметров сложность опускается с $O(n^2)$ до $O(n \cdot \sqrt{n})$
- Обучались модели GPT-2 с длиной последовательности до 12K токенов
- OpenAI в поздних работах активно используют этот подход

Reformer: The Efficient Transformer

Berkeley University, Google, 2020

- Self-attention считается не между всеми K и Q, а только между близкими
- Реализуется с помощью поиска соседей:
  - обучается общая для ключей и запросов весовая матрица
  - ключи собираются в LSH-индекс
  - для запроса ищется ближайший в индексе центроид
  - внимание для запроса считается только с соседями центроида
- Для экономии памяти за счёт замедления обучения используются RevNet (пересчёт выходов слоя по выходам следующего в backprop) и блочные вычисления на каждом слое
- Сложность можно снизить до $O(n \cdot \log n)$
- Обучались модели с длиной последовательности 64K токенов

Longformer: The Long-Document Transformer

Allen Institute for AI, 2020

- В кодировщике трансформера внимание разделяется на две компоненты: локальное и глобальное
- Глобальное: выбирается фиксированный набор токенов (могут быть важные, типа CLS или токенов вопроса для QA), на них смотрят все токены, они смотрят на все
- Локальное:
  - токен обращает внимание на N токенов в обе стороны
  - N растёт линейно с номером слоя
  - выбираются либо ближайших токенов (нижние слои), либо N с заданным шагом (допускается на верхних слоях)
- У глобального и локального self-attention свои весовые матрицы
- Тренируются модели с относительным позиционным кодированием, n при обучении наращивается от 2K до 23K, тестируется на 32K

Rethinking Attention with Performers

Google, Cambridge University, DeepMind, 2020

- Вместо упрощения подсчёта self-attention производится его аппроксимация без дополнительных предположений
- За счёт использования специальных ядер получаются новые матрицы K' и Q' с новой небольшой размерностью признаков $r < d$:

$$\mathrm{softmax}(Q_{n\times d}K^T_{n\times d})V_{n\times d} \approx Q'_{n\times r}(K'^T_{n\times r}V_{n\times d})$$

- Сложность падает с $O(n^2 \cdot d)$ до $O(n \cdot r \cdot d)$
- Модели обучались для разных с задач с длиной контекста от 8K до 12K

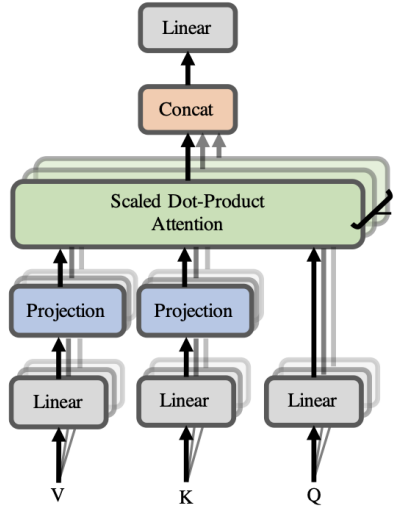
Linformer: Self-Attention with Linear Complexity

Facebook, 2020

- Аппроксимация подсчёта self-attention за счёт понижения ранга матриц K и V
- Понижается размерность длины последовательности с n до k:

$$\mathrm{softmax}(Q_{n\times d}K_{n\times d}^T)V_{n\times d} \approx \mathrm{softmax}(Q_{n\times d}K_{k\times d}^T)V_{k\times d}$$

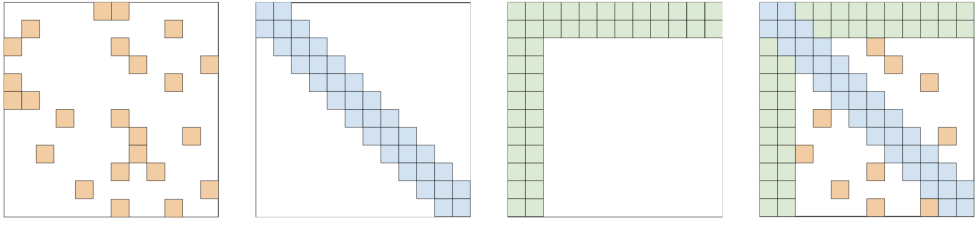
- Сложность падает с $O(n^2 \cdot d)$ до $O(n \cdot k \cdot d)$
- Модели обучались для разных с задач с длиной контекста от 8K до 12K



Big Bird: Transformers for Longer Sequences

Google, 2020

- Self-attention для токена разбивается на три части:
  - локальное внимание в пределах контекста токена
  - внимание на случайный разреженный набор токенов по всей последовательности
  - глобальное внимание (как в Longformer)
- Обучалась модель типа RoBERTa, но с n=4K вместо 512

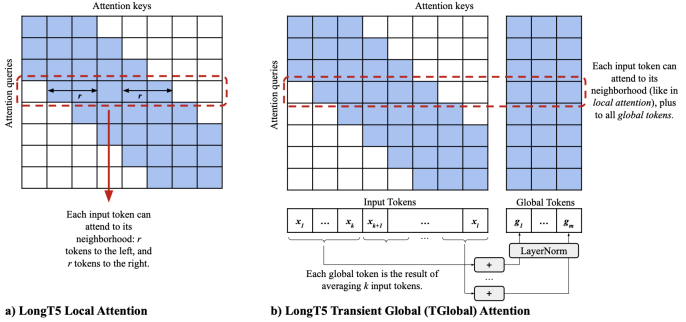


(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

LongT5: Efficient Text-To-Text Transformer for Long Sequences

Google, 2021

- Seq2seq модель с двумя компонентами внимания на архитектуре T5:
- Локальное: в пределах контекста токена (127 в каждую сторону)
- Глобальное:
  - вход делится на блоки по 16 токенов
  - глобальный токен блока равен сумме токенов блока
  - все токены смотрят на все глобальные токены



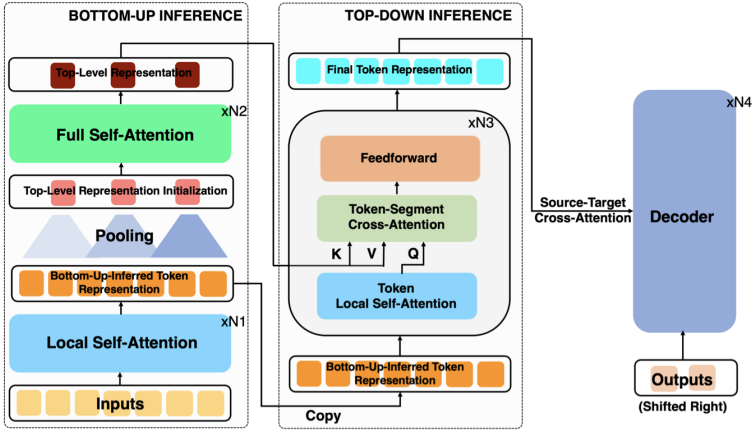
- На претрейне n=4K, на FT в разных задачах n=4-47K

Long Document Summarization With Top-Down and Bottom-Up Inference

Salesforce Research, 2022

- Двухуровневая обработка входа для seq2seq модели суммаризации
- **Шаг 1**:
  - локальный self-attention на последовательности из n токенов (w соседей)
  - пулинг поверх выходов для получения m глобальных токенов
  - полный self-attention на глобальных токенах
- **Шаг 2**:
  - выходы локального self-attention дают Q
  - выходы полного self-attention дают K и V
  - self-attention с этими Q, K и V даёт итоговый выход кодировщика
- Выходы кодировщика идут как обычно в cross-attention в декодировщик
- Пулинг может быть усреднением или более сложным и обучаемым
- Сложность: $O(Nw + M^2 + NM)$

Long Document Summarization With Top-Down and Bottom-Up Inference



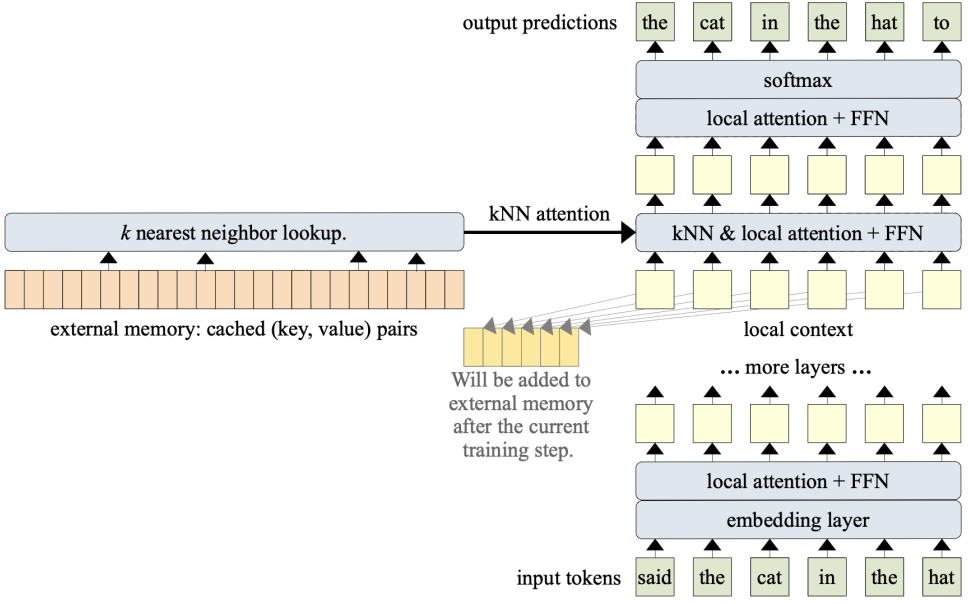
- Модель < 500M, учится суммаризируют книги (данные BookSum)
- Обучение последовательное – сперва на главах, потом на книгах как наборах суммаризаций глав (главы большие для обычных моделей)

Memorizing Transformers

Google, 2022

- Базовая архитектура Transformer-XL с сегментами по 512 токенов
- В последнем блоке добавляется слой kNN-augmented attention:
  - для слоя заводится блок памяти на M пар векторов ключей и значений
  - текущие K и V self-attention слоя добавляются в конец памяти
  - на ключах памяти запускается kNN, свой для запроса из текущего Q
  - self-attention для запроса считается только с ближайшими соседями
- Результаты обычного и kNN-augmented внимания складываются с обучаемым весом
- Память своя для каждой головы self-attention
- Если в памяти не хватает места, вытесняются наиболее старые пары
- Кодирование позиций внутри сегмента относительное из T5
- kNN приближённый, ScaNN или Faiss

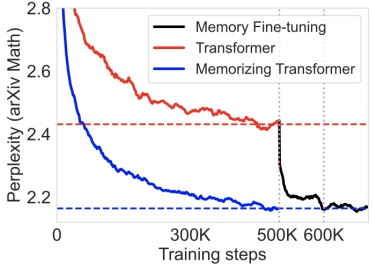
Memorizing Transformers



Memorizing Transformers

Context	Memory	XL cache	arXiv	PG19	C4(4K+)	GitHub	Isabelle
512	None	None	3.29	13.71	17.20	3.05	3.09
2048	None	None	2.69	12.37	14.81	2.22	2.39
512	None	512	2.67	12.34	15.38	2.26	2.46
2048	None	2048	2.42	11.88	14.03	2.10	2.16
512	1536	None	2.61	12.50	14.97	2.20	2.33
512	8192	None	2.49	12.29	14.42	2.09	2.19
512	8192	512	2.37	11.93	14.04	2.03	2.08
512	65K	512	2.31	11.62	14.04	1.87	2.06
2048	8192	2048	2.33	11.84	13.80	1.98	2.06
2048	65K	2048	**2.26**	**11.37**	**13.64**	**1.80**	**1.99**

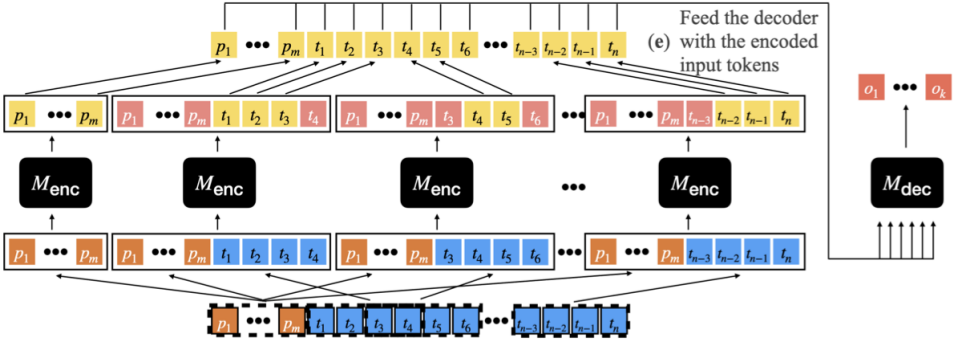
Table 4: Average token-level perplexities of each model when trained for 500k steps.



Efficient Long-Text Understanding with Short-Text Models

Tel-Aviv University, 2022

- Увеличение длины входа предобученных моделей seq2seq на FT:
  - вход (16K) нарезается на сегменты с контекстом с обеих сторон (256)
  - опционально к каждому сегменту добавляется префикс (промпт, вопрос)
  - каждый сегмент обрабатывается кодировщиком независимо
  - из результатов удаляются лишние токены, закодированная последовательность идёт в cross-attention в декодировщик



Efficient Long-Text Understanding with Short-Text Models

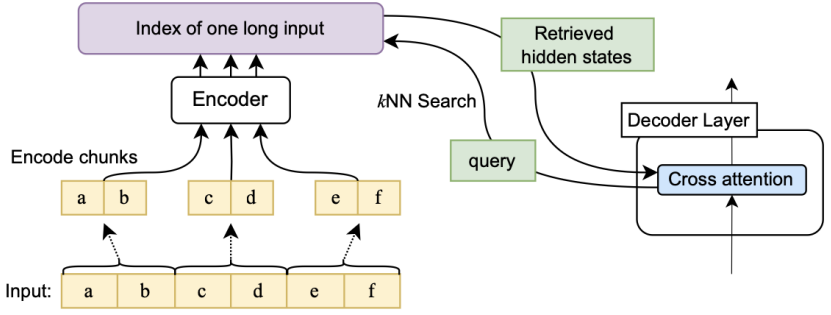
Model	(Chunk/Input)	#Params	Avg	GovRep ROUGE-1/2/L	SumScr ROUGE-1/2/L	QMSum ROUGE-1/2/L	Qspr F1	Nrtv F1	QALT EM-T/H	CNLI EM
Development Scores										
LED_{base}	$(256/16K)$	162M	-	**57.3/27.9/30.0**	30.7/6.3/17.9	32.5/9.0/21.1	30.4	20.2	30.9	82.3
T5_{base}	$(1K/1K)$	220M	-	32.8/11.7/20.2	22.2/3.7/15.3	26.1/6.6/19.8	13.2	14.9	35.1	76.8
T5_{base}-SLED	$(256/16K)$	220M	-	47.0/20.2/25.2	25.3/5.0/16.6	29.9/8.7/21.4	38.2	18.2	34.6	82.4
$\text{BART}_{\text{base}}$	$(1K/1K)$	139M	-	47.7/18.5/22.3	30.1/7.0/18.3	32.2/9.3/21.1	23.3	15.9	33.8	78.4
$\text{BART}_{\text{base}}$-SLED	$(256/16K)$	139M	-	55.7/24.8/25.8	33.6/8.5/19.2	34.4/11.5/22.7	35.8	21.3	33.7	**85.3**
$\text{BART}_{\text{large}}$	$(1K/1K)$	406M	-	50.6/19.8/23.5	32.1/7.4/18.7	33.3/9.4/21.6	24.5	17.9	36.1	79.3
$\text{BART}_{\text{large}}$-SLED	$(256/16K)$	406M	-	**57.4**/26.3/27.5	**35.3/8.8/19.5**	**36.3/12.2/23.3**	**42.5**	**23.6**	**37.2**	**85.3**
Test Scores										
LED_{base}	$(256/16K)$	162M	33.6	56.8/**27.3/29.2**	30.0/6.0/17.5	31.3/8.6/20.5	34.8	21.0	28.5/28.3	82.9
T5_{base}	$(1K/1K)$	220M	26.3	33.2/12.1/20.4	21.4/3.6/15.0	24.2/5.9/18.6	16.3	15.0	31.9/28.6	76.3
T5_{base}-SLED	$(256/16K)$	220M	33.3	46.6/20.1/25.1	24.5/4.6/16.5	28.4/8.7/20.5	43.0	18.9	31.2/29.4	81.4
$\text{BART}_{\text{base}}$	$(1K/1K)$	139M	30.6	48.0/19.1/22.7	30.1/6.6/18.1	31.2/9.1/20.3	27.6	16.0	32.5/31.6	77.1
$\text{BART}_{\text{base}}$-SLED	$(256/16K)$	139M	35.4	54.7/24.4/25.4	32.7/7.9/19.1	33.8/**11.7/22.6**	41.1	21.5	29.7/30.4	85.6
$\text{BART}_{\text{large}}$	$(1K/1K)$	406M	32.1	50.7/20.1/23.5	31.6/6.8/18.5	32.0/9.1/20.8	29.2	18.3	**34.8**/33.9	79.7
$\text{BART}_{\text{large}}$-SLED	$(256/16K)$	406M	**38.0**	**57.5**/26.3/27.4	**35.2/8.7/19.4**	**34.2**/11.0/22.0	**46.9**	**24.1**	**34.8/34.8**	**87.3**
$\text{LED}_{\text{base}}^{\text{SCROLLS}\dagger}$	$(1K/16K)$	162M	29.2	**56.2**/26.6/28.8	24.2/4.5/15.4	25.1/6.7/18.8	26.6	18.5	25.8/25.4	71.5
$\text{LongT5}_{\text{base}}^{\dagger}$	$(255/16K)$	220M	38.2	53.5/27.3/29.3	34.8/**9.6**/21.1	33.9/11.0/22.8	46.6	23.0	37.9/36.6	85.6
$\text{LongT5}_{\text{large}}^{\dagger}$	$(255/16K)$	770M	40.5	54.2/27.8/29.8	35.6/9.2/**21.2**	35.1/**12.0**/23.3	52.3	27.2	40.6/38.6	87.3
$\text{LongT5}_{\text{XL}}^{\dagger}$	$(255/16K)$	3B	**41.9**	54.7/**28.2/30.2**	**35.8/9.6**/21.1	**34.9**/11.8/**23.5**	**53.1**	**29.3**	**46.0/42.1**	88.2
$\text{UL2}^{\dagger}$	$(2K/2K)$	20B	37.9	53.6/26.1/28.8	32.9/7.8/19.4	31.1/8.5/20.4	37.6	24.2	45.8/40.7	**88.7**

Бенчмарк SCROLLS

Unlimiformer: Long-Range Transformers with Unlimited Length Input

Carnegie Mellon University, 2023

- Увеличение длины входа предобученных моделей seq2seq без дообучения
- Вход кодировщика разбивается на сегменты с пересечением, обрабатываемые независимо, в конце контекст удаляется (как и выше)
- Результаты всего сохраняются и используются в cross-attention декодировщика с kNN-индесом (как в Memorizing Transformers)



Unlimiformer: Long-Range Transformers with Unlimited Length Input

- **Проблема**: на каждом слое и в каждой голове нужен свой kNN-индекс, который нужно перестраивать для каждого набора пар ключ-значение
- Это затратно по памяти и времени, поэтому в Memorizing Transformers модифицированный слой добавляется только в последнем блоке
- **Решение:**
  - переписывается формула подсчёта весов внимания (h_e и h_d – выходы кодировщика и декодировщика, W_q и W_k – веса внимания):

$$QK^T = (h_d W_q)(h_e W_k)^T = (h_d W_q) W_k^T h_e^T = (h_d W_q W_k^T) h_e^T$$

  - теперь в индексе лежат не ключи, а общие для всех выходы h_e
  - запросы для kNN-индекса формируются на каждом слое-голове
  - для извлечённых h_e легко посчитать значения V

Unlimiformer: Long-Range Transformers with Unlimited Length Input

- Варианты FT без явного обучения с Unlimiformer:
  - FT на задачу с ограниченной длиной сэмплов + Unlimiformer на тесте
  - то же самое, но с Unlimiformer на валидации для возможности раннего останова и выбора более хорошего чекпойнта
  - то же самое, но сэмплы не обрезаются, а нарезаются на сегменты максимальной длины и подаются как отдльные сэмплы

Method name	Training input	total # tokens in example seen at training time	Validation input (early stopping)	Test input
Baseline	1024	1024	1024	1024
+test Unlimiformer	1024	1024	1024	unlimited
+early stop w/ Unlimiformer	1024	1024	unlimited	unlimited
Train chunked +test Unlimiformer	1024	all	unlimited	unlimited

Unlimiformer: Long-Range Transformers with Unlimited Length Input

- Варианты FT с Unlimiformer:
  - FT с Unlimiformer входом 8-16K + с неограниченный контекст на тесте
  - то же самое, но при обучении вместо kNN выбираются случайные токены
  - чередование двух подходов: первый учит, второй регуляризует для обращения внимания не только на top-k ключей

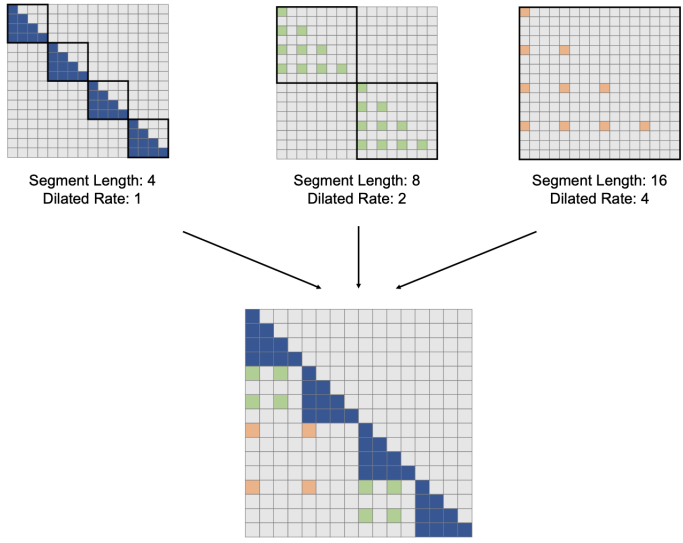
Method name	Training input	total # tokens in example seen at training time	Validation input (early stopping)	Test input
SLED (Ivgi et al., 2022)	16k	16k	16k	16k
Longformer (Beltagy et al., 2020)	16k	16k	16k	16k
Random-encoded training	8-16k	8-16k	unlimited	unlimited
Retrieval training	8-16k	8-16k	unlimited	unlimited
Alternating training	8-16k	8-16k	unlimited	unlimited

- В разных задачах конкурентов побеждают разные подходы
- Репозиторий: https://github.com/abertsch72/unlimiformer

LongNet: Scaling Transformers to 1,000,000,000 Tokens

Microsoft, 2023

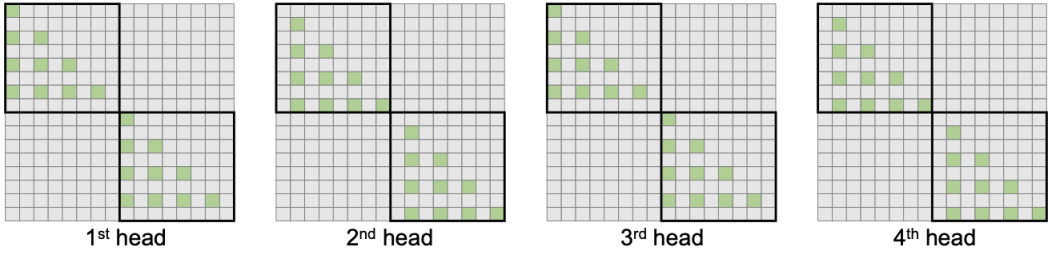
- Последовательность разбивается на сегменты длиной w, обрабатываемые независимо
- Внутри сегмента self-attention считается разреженно, участвуют только $1/r$ токенов с индексом $i : i \mod r = 0$



LongNet: Scaling Transformers to 1,000,000,000 Tokens

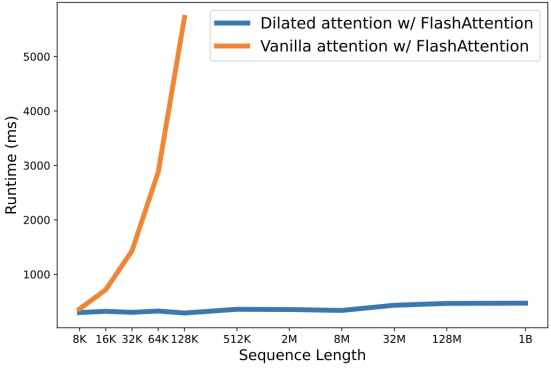
Microsoft, 2023

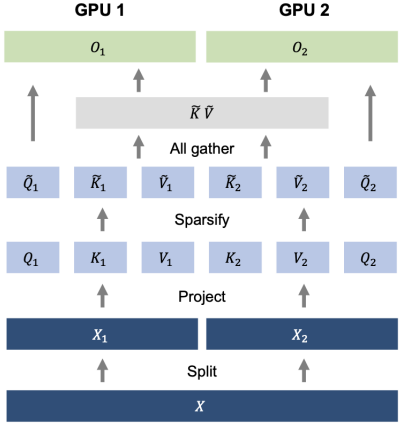
- Выходные результаты конкатенируются
- В одной голове несколько подсчётов с разными (w, r), результаты суммируются с весами, пропорциональными их знаменателям softmax
- В разных головах для одних и тех же (w, r) делаются сдвиги



LongNet: Scaling Transformers to 1,000,000,000 Tokens

- Длинная последовательность распределяется по GPU
- Малые сегменты на одну GPU, большие – на разные, для подсчёта их self-attention агрегируются разреженные K и V





Спасибо за внимание!