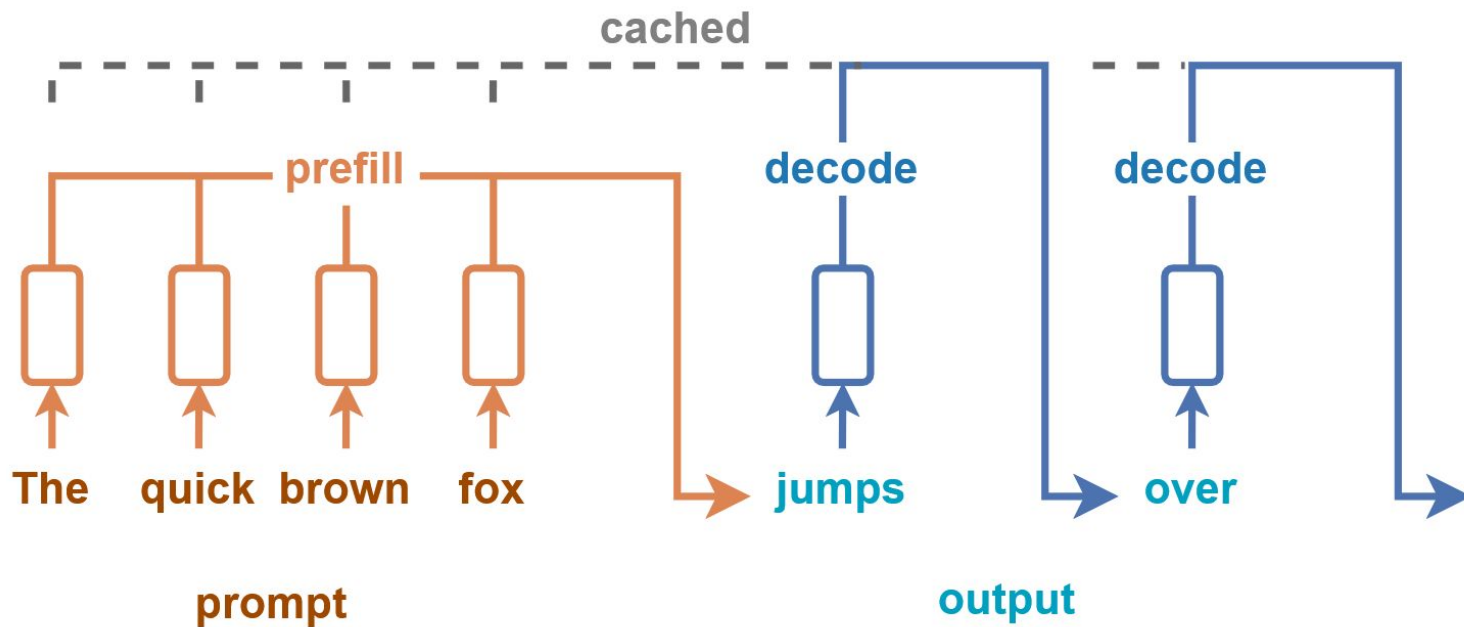


Спекулятивное декодирование: от теории к SOTA-архитектурам

Доклад выполнен для семинара
Лаборатории искусственного интеллекта, математики и
компьютерных наук им. А. А. Маркова

Георгий Ангени

Проблема инференса LLM



Основная концепция спекулятивного декодирования

- От последовательной генерации никуда не уйти*, но ради выигрыша в скорости генерации возьмем более простую *драфтовую* модель
- Периодически отдаем сгенерированные драфтовой моделью токены на вход *целевой* модели
- Последовательно определяем наибольший префикс, соответствующий распределению целевой модели и “принимаем” только его

```
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 9859
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in tokyo late
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in late morning trading . [END]
```

Fast Inference from Transformers via Speculative Decoding

<https://arxiv.org/pdf/2211.17192>

Accelerating Large Language Model Decoding with Speculative Sampling

<https://arxiv.org/pdf/2302.01318>

Как обеспечить совместную работу моделей?

- Распределение целевой модели – $p(x)$, распределение драфтовой – $q(x)$
- Необходимо сэмплировать $p(x)$ с помощью $q(x)$
- Знаем про rejection sampling!

1. Sample $x \sim q(x)$ and $r \sim U(0, 1)$.

2. If $r < \frac{p(x)}{Mq(x)}$ return x . $M = \max_x \frac{p(x)}{q(x)}$.

3. ~~Go to 1.~~ Sample $x \sim p(x)$.

- Оказывается, что принимать будем совсем мало токенов

$$\mathbb{E}_{q(x)} \frac{p(x)}{Mq(x)} = \sum_x p(x) \min_{x'} \frac{q(x')}{p(x')} = \min_{x'} \frac{q(x')}{p(x')}$$

Спекулятивное сэмплирование

- Алгоритм спекулятивного сэмплирования устроен следующим образом
 1. Sample $x \sim q(x)$.
 2. If $q(x) \leq p(x)$, return x .
 3. Sample $r \sim U(0, 1)$.
 4. If $r < \frac{p(x)}{q(x)}$, return x .
 5. Return $x' \sim \text{norm}(\max(0, p(x') - q(x')))$.
- Первые четыре шага можно упростить
 1. Sample $x \sim q(x)$ and $r \sim U(0, 1)$.
 2. If $r < \frac{p(x)}{q(x)}$, return x .

Вывод вероятности принятия I

- Введем $\beta = \beta_{<x_t}$ – вероятность принять токен $x \sim q(x_t|x_{<x_t})$, а также $\alpha = \mathbb{E}\beta$ и найдем их
- Введем $D(p, q) = \sum_x |p(x) - M(x)| = \sum_x |q(x) - M(x)|$, где $M(x) = \frac{p(x)+q(x)}{2}$

$$\begin{aligned} D(p, q) &= \sum_x |p(x) - M(x)| = \sum_x \frac{|p(x)-q(x)|}{2} = \\ &= 1 - \sum_x \frac{p(x)+q(x)-|p(x)-q(x)|}{2} = 1 - \sum_x \min(p(x), q(x)) \end{aligned}$$

- Видно, что $0 \leq D(p, q) \leq 1$, при этом $D(p, q) = 0 \Leftrightarrow p = q$ и $D(p, q) = 1 \Leftrightarrow \text{supp } p \cap \text{supp } q = \emptyset$

Вывод вероятности принятия II

- Теперь выразим вероятности через введенную дивергенцию

$$\beta = \mathbb{E}_{x \sim q(x)} \min(1, \frac{p(x)}{q(x)}) = \sum_x \min(p(x), q(x)) = 1 - D(p, q)$$

$$\alpha = 1 - \mathbb{E} D(p, q) = \mathbb{E} \min(p, q)$$

- Сравним вероятность принятия с rejection sampling

$$\begin{aligned} \alpha' &= \mathbb{E}_{q(x)} \frac{p(x)}{Mq(x)} = \sum_x p(x) \min_{x'} \frac{q(x')}{p(x')} \leq \sum_x p(x) \min(1, \frac{q(x)}{p(x)}) = \\ &= \sum_x \min(p(x), q(x)) = \alpha \end{aligned}$$

Корректность спекулятивного сэмплирования

- Заметим, что $p'(x) = \text{norm}(\max(0, p(x) - q(x))) = \frac{p(x) - \min(q(x), p(x))}{\sum_{x'} p(x') - \min(q(x'), p(x'))} = \frac{p(x) - \min(q(x), p(x))}{1 - \beta}$

- Теперь распишем вероятность $P(x = x')$

$$P(\text{accepted}, x = x') = \min\left(1, \frac{p(x')}{q(x')}\right) \cdot q(x')$$

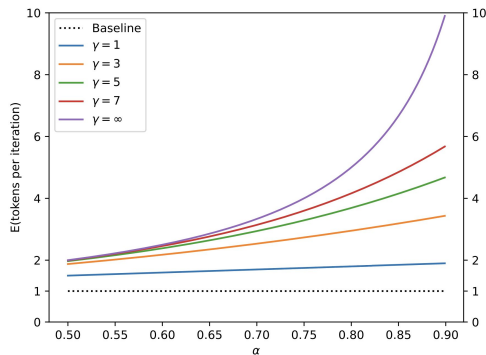
$$P(\text{rejected}, x = x') = (1 - \beta) \cdot p'(x') = p(x') - \min(q(x'), p(x'))$$

$$P(x = x') = P(\text{accepted}, x = x') + P(\text{rejected}, x = x') = p(x')$$

Теоретические оценки исходной реализации

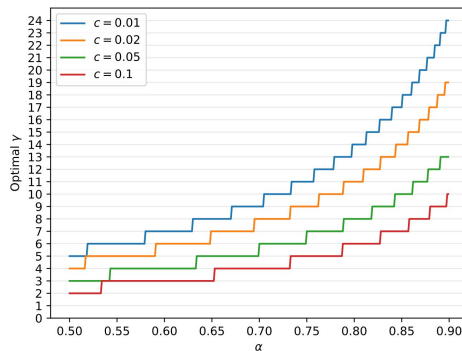
- В исходной работе делается большой упор на теоретические основы
- Выводится оптимальное значение γ – длины драфта в зависимости от acceptance rate α и cost coefficient c (и ops coefficient $\hat{c}$) и ожидаемые теоретические и практические приросты

$$E(\# \text{ generated tokens}) = \frac{1 - \alpha^{\gamma+1}}{1 - \alpha}$$



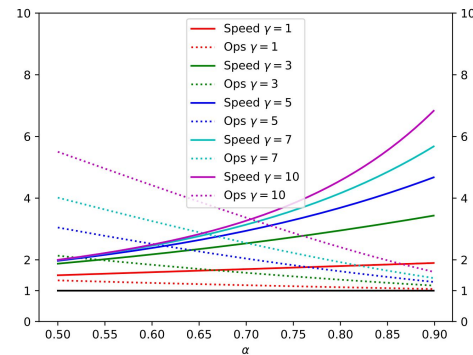
Ожидаемое число токенов за одну итерацию target model

Theorem 3.8. The expected improvement factor in total walltime by Algorithm 1 is $\frac{1 - \alpha^{\gamma+1}}{(1 - \alpha)(\gamma + 1)}$.



Наилучший γ как $f(\alpha)$ для разных c

Theorem 3.11. The expected factor of increase in the number of total operations of Algorithm 1 is $\frac{(1 - \alpha)(\gamma + 1)}{1 - \alpha^{\gamma+1}}$.



Прирост скорости и числа операций как $f(\alpha)$ для разных γ

Эксперименты с исходной реализацией

- Несмотря на обилие теоретических выкладок, в статье эксперименты проведены на уровне proof-of-concept
- Взяты модели с готовыми чекпоинтами
- Измеряется ускорение с единичным batch size на TPU-v4
- *Также была попытка использовать N-граммные модели, но у них оказался слишком низкий **acceptance rate***

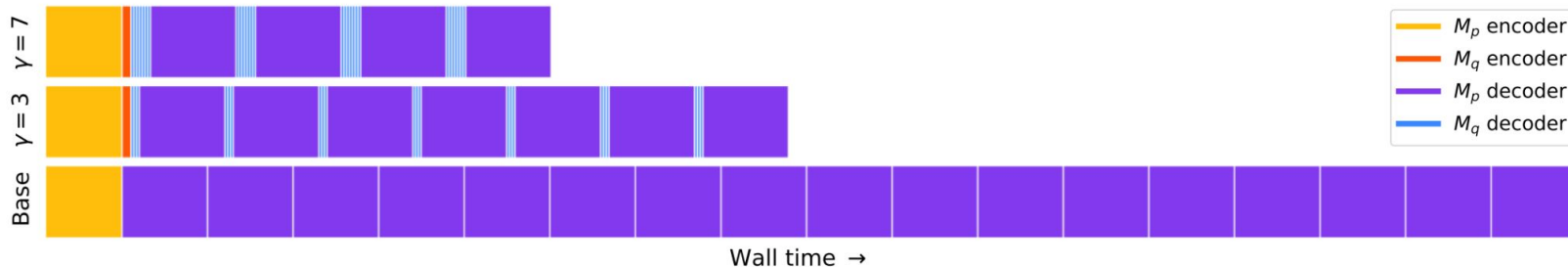
Table 2. Empirical results for speeding up inference from a T5-XXL 11B model.

	TASK	M_q	TEMP	γ	α	SPEED
english to german translation	ENDE	T5-SMALL ★	0	7	0.75	3.4X
	ENDE	T5-BASE	0	7	0.8	2.8X
	ENDE	T5-LARGE	0	7	0.82	1.7X
	ENDE	T5-SMALL ★	1	7	0.62	2.6X
	ENDE	T5-BASE	1	5	0.68	2.4X
	ENDE	T5-LARGE	1	3	0.71	1.4X
text summarization	CNNDM	T5-SMALL ★	0	5	0.65	3.1X
	CNNDM	T5-BASE	0	5	0.73	3.0X
	CNNDM	T5-LARGE	0	3	0.74	2.2X
	CNNDM	T5-SMALL ★	1	5	0.53	2.3X
	CNNDM	T5-BASE	1	3	0.55	2.2X
	CNNDM	T5-LARGE	1	3	0.56	1.7X

Архитектуры спекулятивного декодирования

Основная инженерная проблема

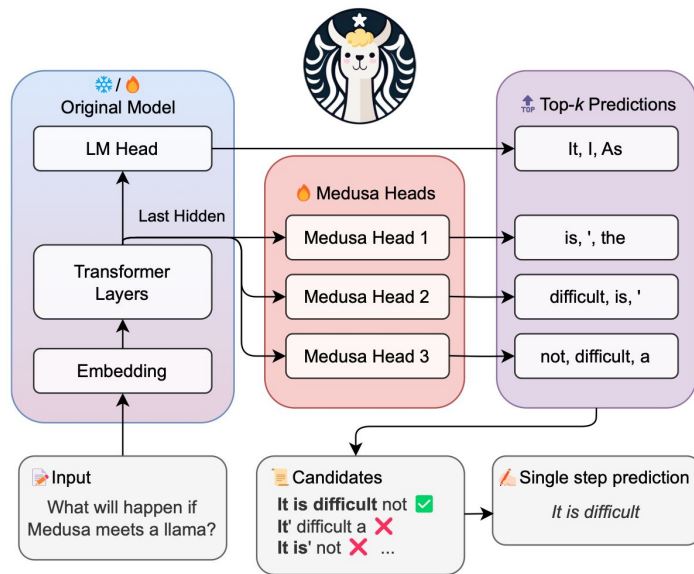
- При применении исходной реализации спекулятивного декодирования в системе поддерживается два эквивалентных контекста



- Имеет смысл разработать метод, использующий данные из целевой модели напрямую – это сэкономит память (и немного времени)

Medusa

- ДRAFT-овая модель – decoder-only, **Y** **ГОЛОВ** – i -ая под i -ый токен после токена, предсказанного целевой моделью
- Каждая голова представляет собой однослойный MLP + LM Head
- В качестве входа используют последние скрытые состояния целевой модели
- При верификации целевой моделью применяется tree attention



Верхнеуровневая архитектура Medusa

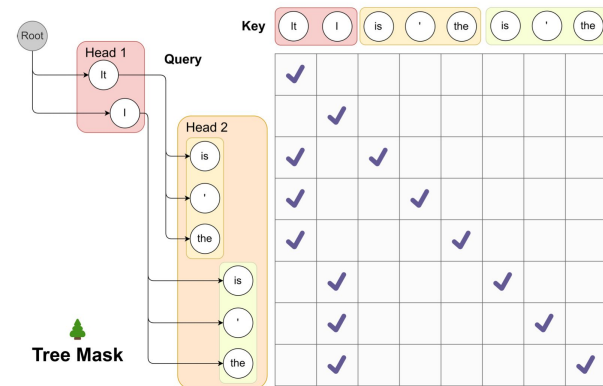
MEDUSA: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads

<https://arxiv.org/pdf/2401.10774>

Medusa – tree attention

- Dense tree attention

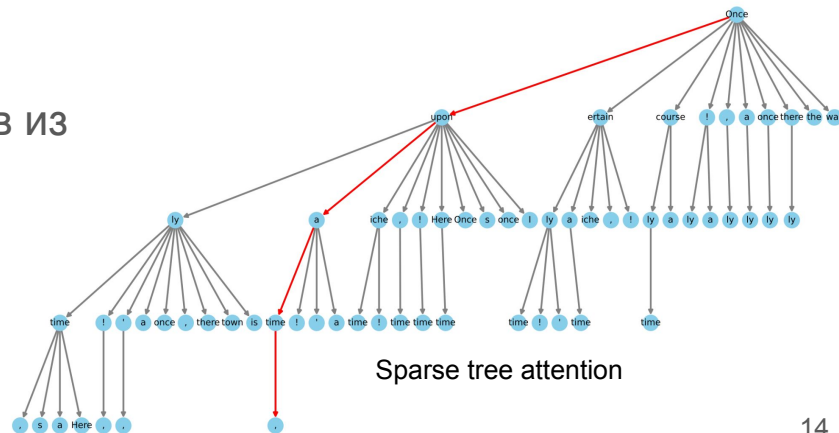
- В батч для верификации берем декартово произведение top-k токенов на всех позициях для наборов позиций от $\{1\}$ до $\{1, \dots, \gamma\}$
- Размер батча растет экспоненциально с увеличением γ (при $k > 1$)



Dense tree attention

- Sparse tree attention

- Жадно собираем поддереву вариантов из дерева для sparse tree attention
- Максимизируем значение $\sum_{[i_1, i_2, \dots, i_k] \in I} \prod_{j=1}^k a_j^{(i_j)}$, где $a_j^{(i_j)}$ равна вероятности принятия i -го токена j -ой головы



Sparse tree attention

Medusa – эксперименты и выводы

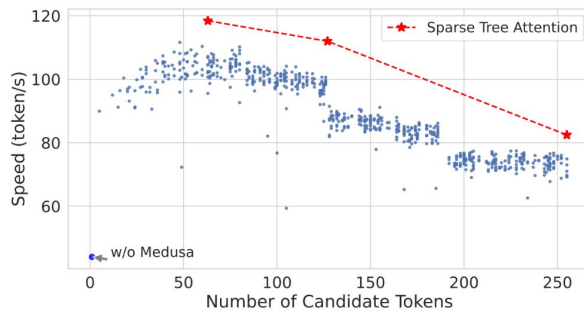
- Значительная доля прироста достигается благодаря использованию tree attention и дообучению целевой модели
- Фиксированная структура дерева подвергает метод деградации при сдвигах в данных
- Архитектура более эффективная относительно поддержки двух отдельных моделей, но “гадание” токенов со второй позиции заставляет задуматься об оптимальности подхода

Model Name	Vicuna-7B	Zephyr-7B	Vicuna-13B	Vicuna-33B
Acc. rate	3.47	3.14	3.51	3.01
Overhead	1.22	1.18	1.23	1.27
Quality	6.18 (+0.01)	7.25 (-0.07)	6.43 (-0.14)	7.18 (+0.05)
$S_{\text{SpecDecoding}}$	1.47	-	1.56	1.60
S_{MEDUSA}	2.83	2.66	2.83	2.35
$S_{\text{MEDUSA-1}}$	2.18		2.33	

метрики Medusa-2 (совместное дообучение) и Medusa-1 с использованием tree attention на MT-Bench

Technique	Speedup
Medusa-1 heads without tree attention	~1.5x
Adding tree attention	~1.9x
Using optimized tree configuration	~2.2x
Training heads with Medusa-2	~2.8x

Влияние каждой из техник на ускорение



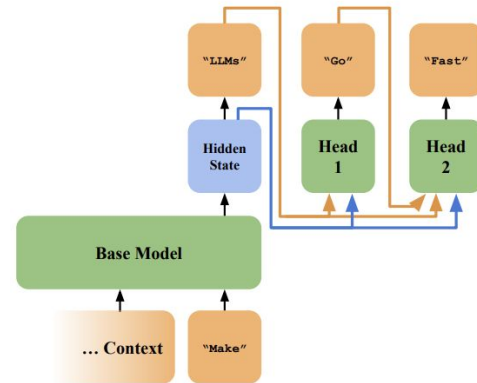
Dense vs. sparse tree attention

Hydra – stack more layers

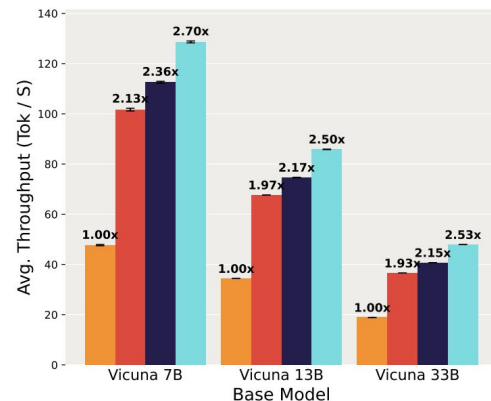
- Hydra – ускорение 1.1x относительно Medusa
 - Головы Medusa применяются последовательно
 - На вход каждой подается скрытое состояние (с предыдущего блока или с целевой модели) вместе с эмбедингами **всех токенов** драфта
- Hydra++ – ускорение 1.3x относительно Medusa
 - Каждая голова представляет собой четырехслойную MLP
 - Дополнительный transformer-like decoder-блок поверх последних скрытых слоев
 - Обучение на распределение целевой модели, а не на логарифм правдоподобия целевого токена
- В сообществе данный подход не закрепился

Hydra: Sequentially-Dependent Draft Heads for Medusa Decoding

<https://arxiv.org/pdf/2402.05109>



Схематичное изображение Hydra: синие стрелки – поток данных в Medusa, оранжевые – новые потоки в Hydra



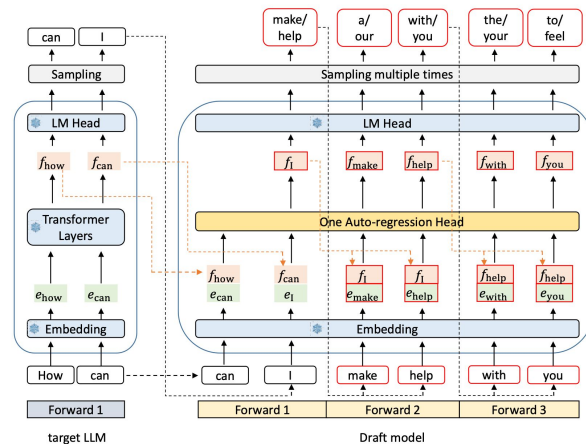
Пропускная способность базового декодирования, Medusa, Hydra и Hydra++¹⁶

EAGLE

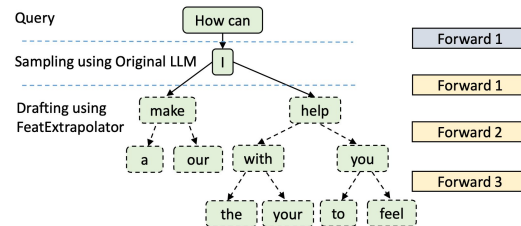
- Обучаем одну голову и применяем ее γ раз
- Голова состоит из слоя эмбединга, MLP для понижения размерности с transformer-like decoder-блоком и LM Head
- Эмбединги и LM Head – из целевой модели
- На вход подается последнее скрытое состояние и эмбединг **следующего** токена
- Голова предсказывает не токен, а следующие последнее скрытое состояние
- Возможно использование tree attention

EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty

<https://arxiv.org/pdf/2401.15077>



Верхнеуровневое представление
применения драфтовой модели EAGLE



Дерево драфтов EAGLE

EAGLE – обучение

- Взвешенная сумма между функцией потерь на факторах и на токенах
$$L = L_{reg} + w_{cls} L_{cls}$$
- На факторах используется функция Smooth-L1 между предсказанными факторами и факторами целевой модели на следующем токене

$$L_{reg} = \text{Smooth L1}(f_{i+1}, \text{Draft_Model}(T_{2:i+1}, F_{1:i}))$$

- На токенах используется кросс-энтропия между распределением целевой модели и предсказанным распределением

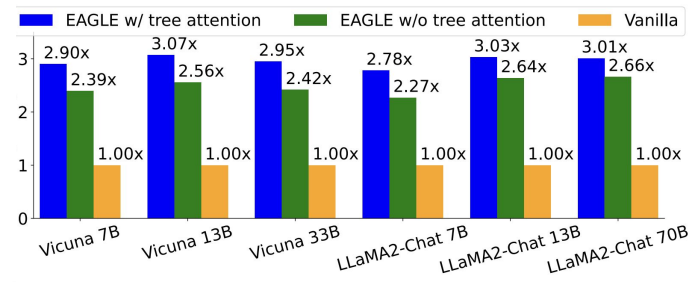
$$p_{i+2} = \text{Softmax}(\text{LM_Head}(f_{i+1}))$$

$$\hat{p}_{i+2} = \text{Softmax}(\text{LM_Head}(\hat{f}_{i+1}))$$

$$L_{cls} = \text{Cross_Entropy}(p_{i+2}, \hat{p}_{i+2})$$

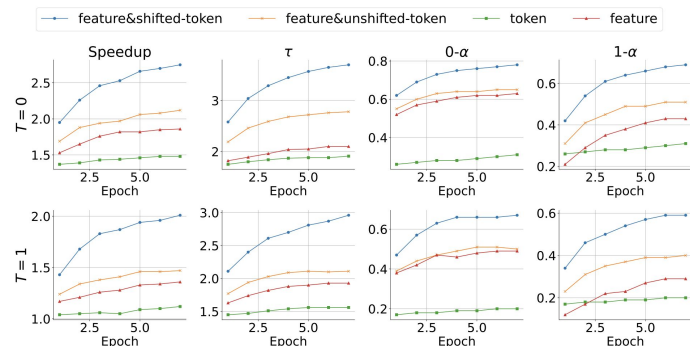
EAGLE – эксперименты и выводы

- Менее затратный и более универсальный метод – без tree attention и совместного дообучения ускорение сравнимо с Medusa
- Добавление следующего токена на вход драфтовой модели играет крайне важную роль в ускорении
- Структура tree attention все еще фиксированная, в работе ничего не говорится о процессе ее подбора



Vicuna			LLaMA2-Chat		
Size	Chain	Tree	Size	Chain	Tree
7B	3.20	3.94 (+0.74)	7B	3.00	3.62 (+0.62)
13B	3.23	3.98 (+0.75)	13B	3.18	3.90 (+0.68)
33B	2.97	3.68 (+0.71)	70B	3.12	3.81 (+0.69)

Ускорение и **ожидаемое число токенов** для разных моделей

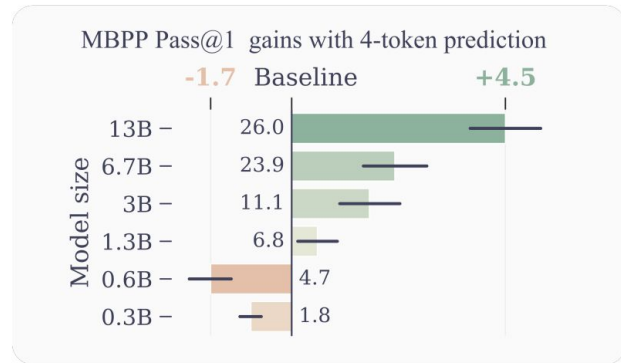
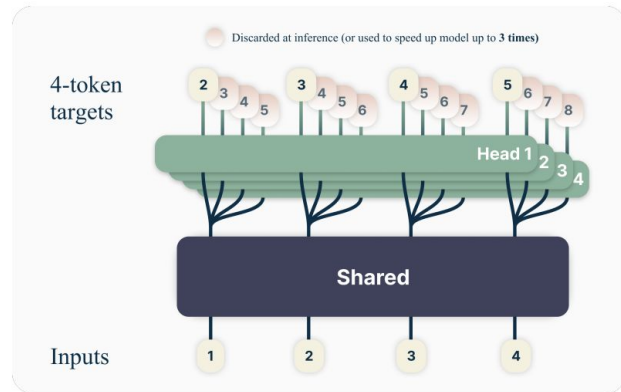


Ускорение, **ожидаемое число токенов** и **acceptance rate** первых двух токенов в зависимости от входа драфтовой модели

Коротко о Multi-Token Prediction

- Через несколько месяцев вышла работа об улучшении качества ответов LLM с помощью обучения вместе с целевой моделью γ независимых decoder-блоков, предсказывающих токены, идущие за токеном, который сэмплируется из LLM
- Архитектурных отличий от Medusa нет
- EAGLE в работе не цитируется

Better & Faster Large Language Models via Multi-token Prediction
<https://arxiv.org/pdf/2404.19737>



Метрики архитектур с различным числом параметров на Mostly Basic Python Problems

Экскурс в DeepSeek – два понимания MTP

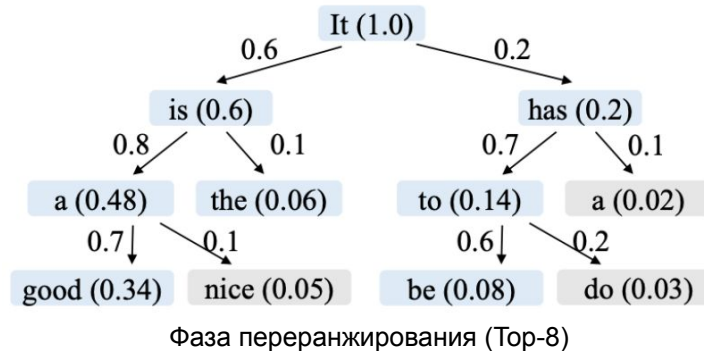
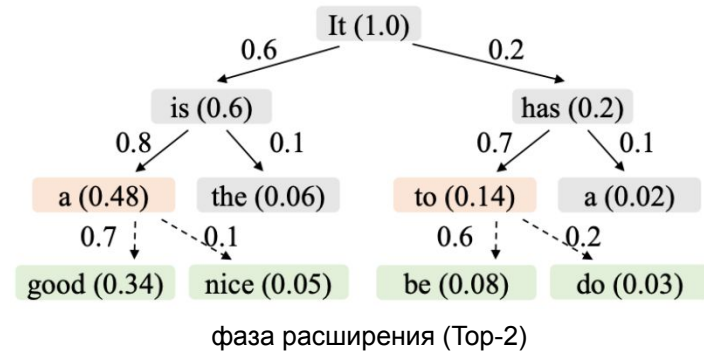
- Разработчики DeepSeek в своей работе про DeepSeek-V3 пишут, что при обучении модели используют multi-token prediction
- Вместо голов Medusa применяют **последовательный ряд** decoder-блоков
- Отмечается “схожесть” с EAGLE, но утверждается, что в первую очередь MTP нужен в качестве метода, повышающего качество ответов модели
- В сообществе закрепилось использование названия MTP при выпуске моделей со встроенным обученным спекулятивным модулем
- Вариация от DeepSeek со временем получила свое название **NextN**

EAGLE-2

- Вводится адаптивное дерево драфтов
- Используют наблюдение: сильная линейная корреляция между **acceptance rate** и уверенностью модели в токене
- Аналогично Medusa, максимизируем оценку $V_i = \prod_{t_j \in \text{Path}(\text{root}, t_i)} p_j \approx \prod_{t_j \in \text{Path}(\text{root}, t_i)} c_j$
- Архитектура драфтовой модели такая же
- Улучшение направлено исключительно на однопоточный сценарий

EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees

<https://arxiv.org/pdf/2406.16858>

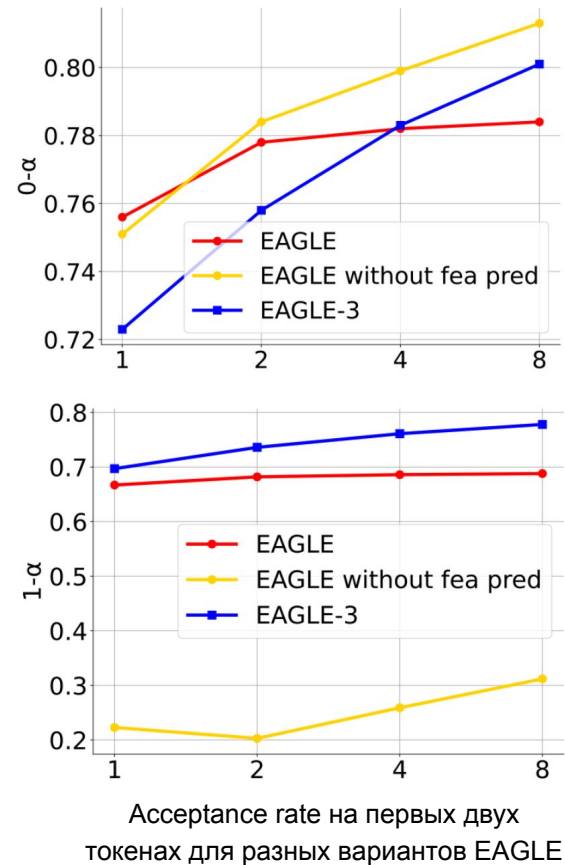


	MT-bench		GSM8K	
Method	Speedup	τ	Speedup	τ
w/o both	2.81x	3.92	2.85x	3.93
w/o value	3.21x	4.39	2.93x	3.96
w/o reranking	3.48x	4.86	3.50x	4.85
EAGLE-2	3.62x	4.98	3.63x	4.97

Ablation study для EAGLE-2 на модели Vicuna-7B²²

EAGLE-3 – постановка проблемы

- Обучение драфтовой модели на факторы вместо токенов позволяет ей делать качественные предсказания на несколько шагов вперед
- Как убрать обучение на факторы без потерь?
- L1-функцию потерь между факторами нельзя выбрасывать – приведет к переобучению
- Другая проблема – последнее скрытое состояние не позволяет учитывать более емкий сигнал от целевой модели

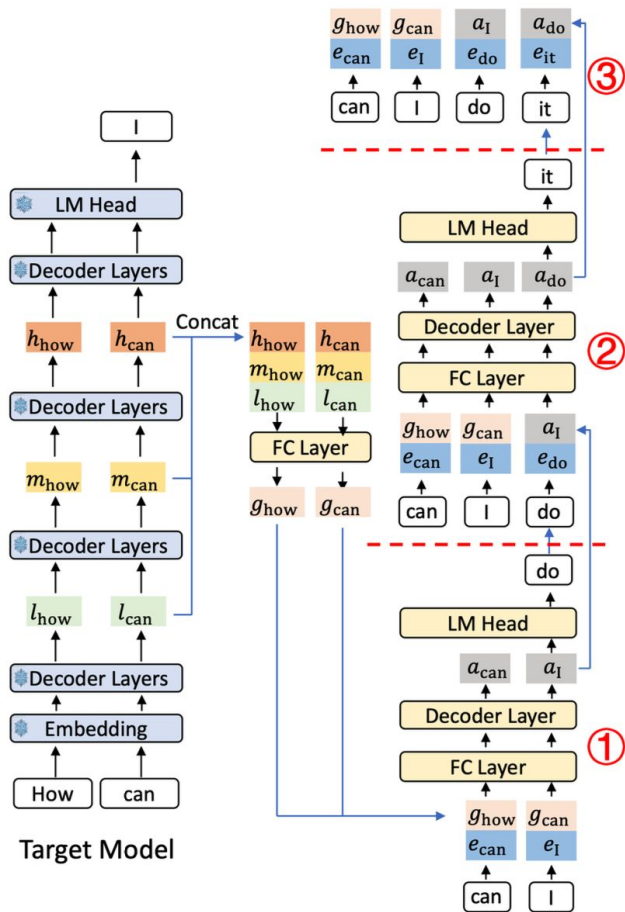


EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test

<https://arxiv.org/pdf/2406.16858>

EAGLE-3

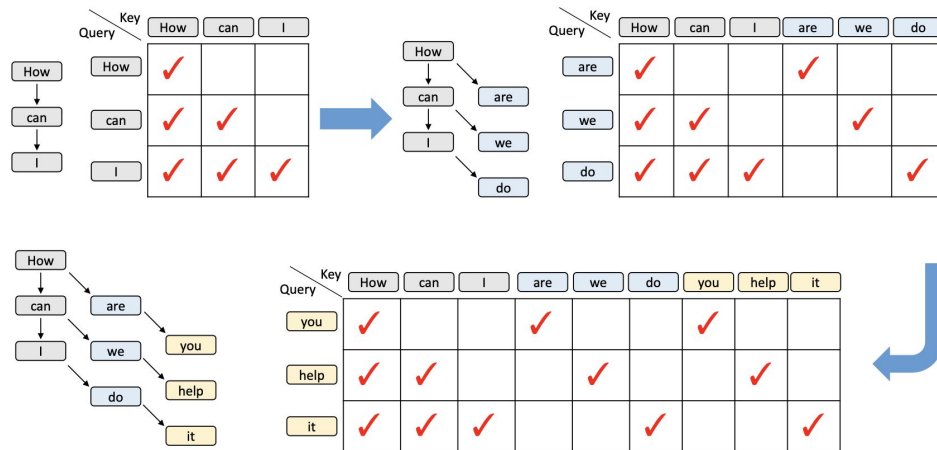
- Главная идея – training-time test – перед backpropagation делаем не один, а **Y** шагов, имитируя реальное применение драфтера
- После первого шага факторы целевой модели используются вместе с факторами, предсказанными драфтовой моделью
- Факторы на входе драфтовой модели получаются путем понижения размерности факторов с трех скрытых состояний целевой модели



Верхнеуровневое представление применения драфтовой модели EAGLE-3

EAGLE-3 – обучение

- На i -ом шаге берется средняя по всей последовательности кросс-энтропия между распределением драфтера и распределением целевой модели (со сдвигом на i токенов)
- Требуемая на обучение драфтовой модели память в γ раз больше



Пример увеличения маски внимания при обучении драфтера на три токена

EAGLE-3 – текущий state-of-the-art

- И training-time test без обучения на факторы, и использование факторов целевой модели с разных слоев дает прирост
- При этом обе модификации универсальны с точки зрения сценария применения
- Ускорение подтверждается тестами на движках для инференса

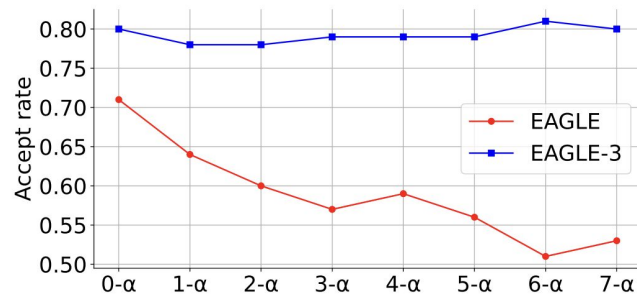
Method	MT-bench		GSM8K	
	Speedup	τ	Speedup	τ
EAGLE-2	3.16x	4.05	3.39x	4.24
+ remove fea con	3.82x	5.37	3.77x	5.22
+ fused features (ours)	4.40x	6.13	4.48x	6.23

Ablation study для EAGLE-3 на
Llama 3.1 8B на MT-Bench

Batch size	2	4	8	16	24	32	48	56	64
EAGLE	1.40x	1.38x	1.23x	1.02x	0.93x	0.94x	0.88x	0.99x	0.99x
EAGLE-3	1.81x	1.82x	1.62x	1.48x	1.39x	1.32x	1.38x	1.34x	1.38x

Batch size	2	4	8	16	24	32	48	56
EAGLE	1.30x	1.25x	1.21x	1.10x	1.03x	0.93x	0.82x	0.71x
EAGLE-3	1.75x	1.68x	1.58x	1.49x	1.42x	1.36x	1.21x	1.01x

Ускорение инференса на движках
SGLang (сверху) и vLLM (снизу)



Сравнение acceptance rate

Дальнейшие направления исследований

Блочная верификация

- Предлагается верифицировать совместно
- Совместная вероятность “слабого” токена с последующими все еще может быть высока
- Доказывается оптимальность с точки зрения **acceptance rate**
- Приносит небольшой, но консистентный прирост (ускорение на 5-8%)

$$p_{\text{res}}^{\text{block}}(x \mid \mathbf{c}, X^i) = \frac{\max\{p_i \cdot \mathcal{M}_b(x \mid \mathbf{c}, X^i) - \mathcal{M}_s(x \mid \mathbf{c}, X^i), 0\}}{\sum_{x' \in \mathcal{X}} \max\{p_i \cdot \mathcal{M}_b(x' \mid \mathbf{c}, X^i) - \mathcal{M}_s(x' \mid \mathbf{c}, X^i), 0\}}$$
$$h_i^{\text{block}} = \frac{\sum_{x' \in \mathcal{X}} \max\{p_i \cdot \mathcal{M}_b(x' \mid \mathbf{c}, X^i) - \mathcal{M}_s(x' \mid \mathbf{c}, X^i), 0\}}{\sum_{x' \in \mathcal{X}} \max\{p_i \cdot \mathcal{M}_b(x' \mid \mathbf{c}, X^i) - \mathcal{M}_s(x' \mid \mathbf{c}, X^i), 0\} + 1 - p_i}$$

Block Verification Accelerates Speculative Decoding

<https://arxiv.org/pdf/2403.10444>

Algorithm 2 Block Verification

Input: Draft block X^γ ; small model distributions $\forall i < \gamma, \mathcal{M}_s(\cdot \mid \mathbf{c}, X^i)$; large model distributions $\forall i \leq \gamma, \mathcal{M}_b(\cdot \mid \mathbf{c}, X^i)$.

1: Sample $\eta_1, \dots, \eta_\gamma \sim U(0, 1)$.

2: Set $\tau = 0$, $p_0 = 1$.

3: **for** $i = 1, \dots, \gamma$ **do**

4: Set $p_i = \min\{p_{i-1} \frac{\mathcal{M}_b(X_i \mid \mathbf{c}, X^{i-1})}{\mathcal{M}_s(X_i \mid \mathbf{c}, X^{i-1})}, 1\}$.

5: Set h_i^{block} as in Equation (5).

6: **if** $\eta_i \leq h_i^{\text{block}}$ **then**

7: Set $\tau = i$.

8: **else**

9: **continue.**

10: **end if**

11: **end for**

12: **if** $\tau = \gamma$ **then**

13: Sample Y from $\mathcal{M}_b(\cdot \mid \mathbf{c}, X^\gamma)$.

14: **else**

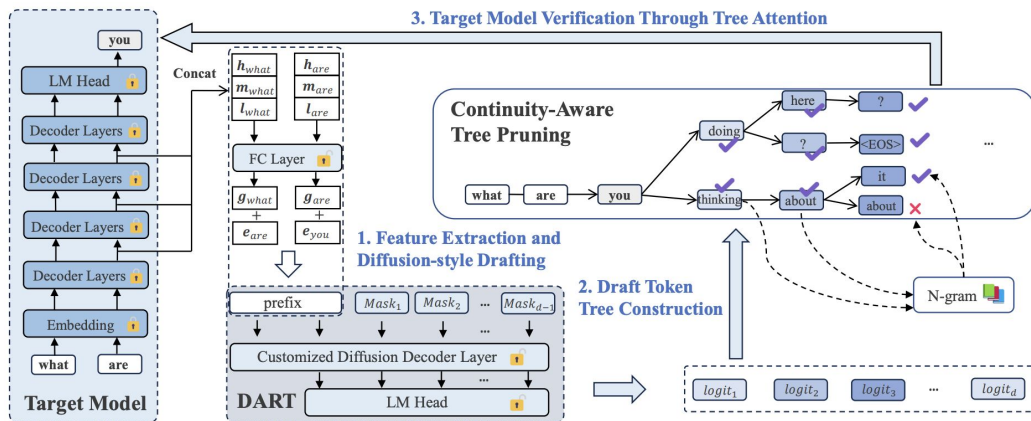
15: Sample Y from $p_{\text{res}}^{\text{block}}(\cdot \mid \mathbf{c}, X^\tau)$ as in Equation (4).

16: **end if**

17: **Return** X^τ, Y .

Параллельное построение драфтов

- Сделаем генерацию драфта за один шаг: маскируем d токенов после префикса и предсказываем за один проход decoder-блока
- Дерево драфтов переоценивается по n-граммной модели

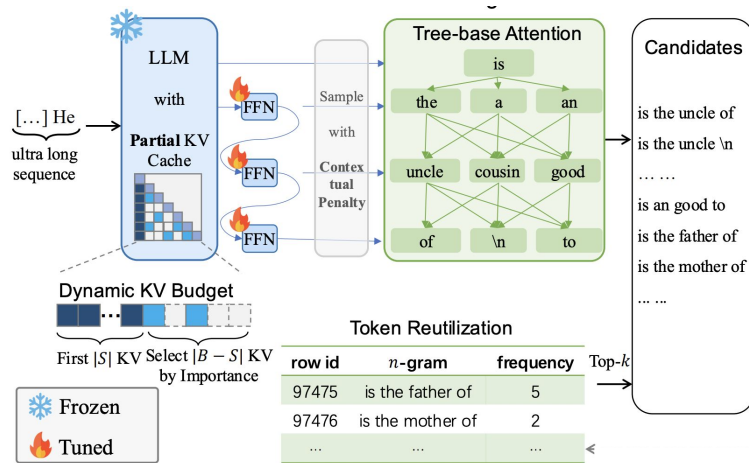


DART: Diffusion-Inspired Speculative Decoding for Fast LLM Inference

<https://arxiv.org/pdf/2406.16858>

SD для длинных контекстов

- На длинных контекстах драфтовая модель становится memory-bound
- Первый подход
 - Возьмем в качестве драфтовой модели целевую с частичным KV-кэшем
 - Поддерживаем n-граммный словарь, чтобы нивелировать низкий acceptance rate
- Второй подход
 - Используем RAG-based драфтер – входной контекст состоит из частей с низким косинусным расстоянием до запроса (но сдвигаем распределение в сторону драфтера)



TokenSwift: Lossless Acceleration of Ultra Long Sequence Generation

<https://arxiv.org/pdf/2502.18890>

RAPID: Long-Context Inference with Retrieval-Augmented Speculative Decoding

<https://arxiv.org/pdf/2502.20330>

Нужно ли сейчас обучать свой SD?

- Спекулятивное декодирование до сих пор остается одним из основных методов увеличения пропускной способности LLM
- Существуют фреймворки от разработчиков движков для обучения SD:
 - SpecForge – <https://github.com/sgl-project/SpecForge> (от разработчиков SGLang)
 - Speculators – <https://github.com/vllm-project/speculators> (от разработчиков vLLM)
 - ModelOpt – <https://github.com/NVIDIA/Model-Optimizer> (от разработчиков TensorRT-LLM)
- AI-компании публикуют свои SD для открытых моделей и модели с SD:
 - <https://huggingface.co/collections/nvidia/speculative-decoding-modules>
 - <https://huggingface.co/collections/lmsys/specbundle>
 - <https://huggingface.co/deepseek-ai/DeepSeek-V3.2>
 - <https://huggingface.co/zai-org/GLM-5>