6. Faster LLMs

Presenter: Jonathan

Faster LLMs

- Distillation
  - Works best in task specific setup
  - Text-editing when there overlap between the input and output
  - Small models, when there is less overlap
- Speeding up LLM inference
  - General purpose
  - Requires large amount of compute

Case study: EdiT5 vs T5

- Two GEC models:
  - EdiT5 base (12-layer-encoder, 1-layer-decoder)
  - T5 base (12-layer-encoder, 12-layer-decoder)
- Profiles obtained on GPU
  - Profiles obtained with Tensorflow Profiler
  - PyTorch has similar tools

GEC

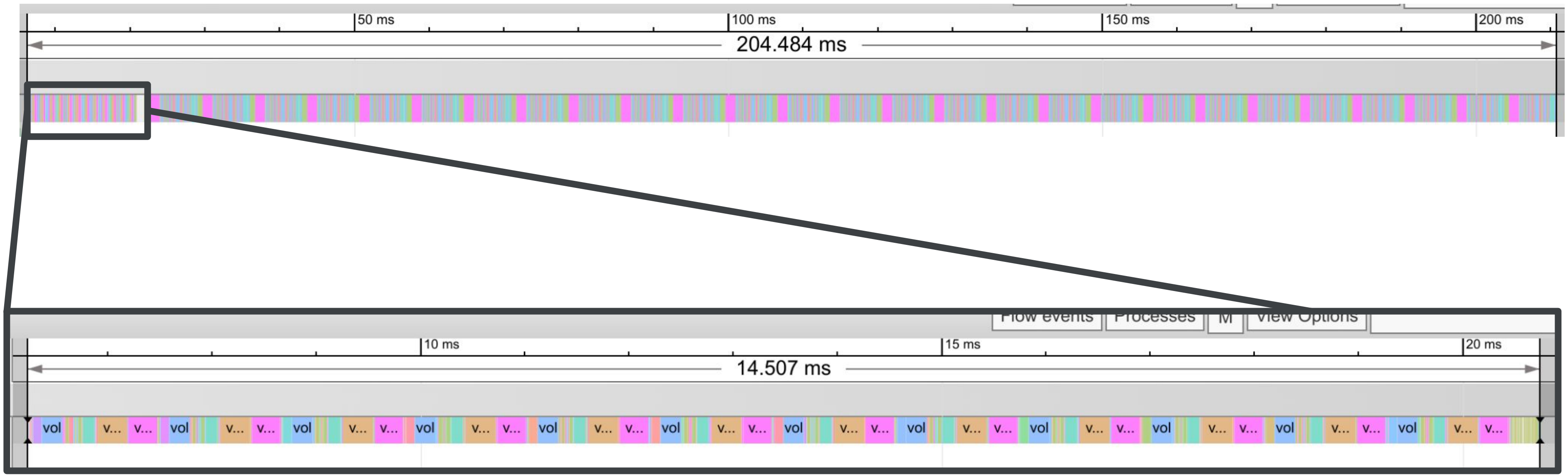
Input to correct (23 tokens):

i was walking through the park when struck by bicycle ... my arm hurts a little now .

Decoder output Seq2seq (27 tokens):

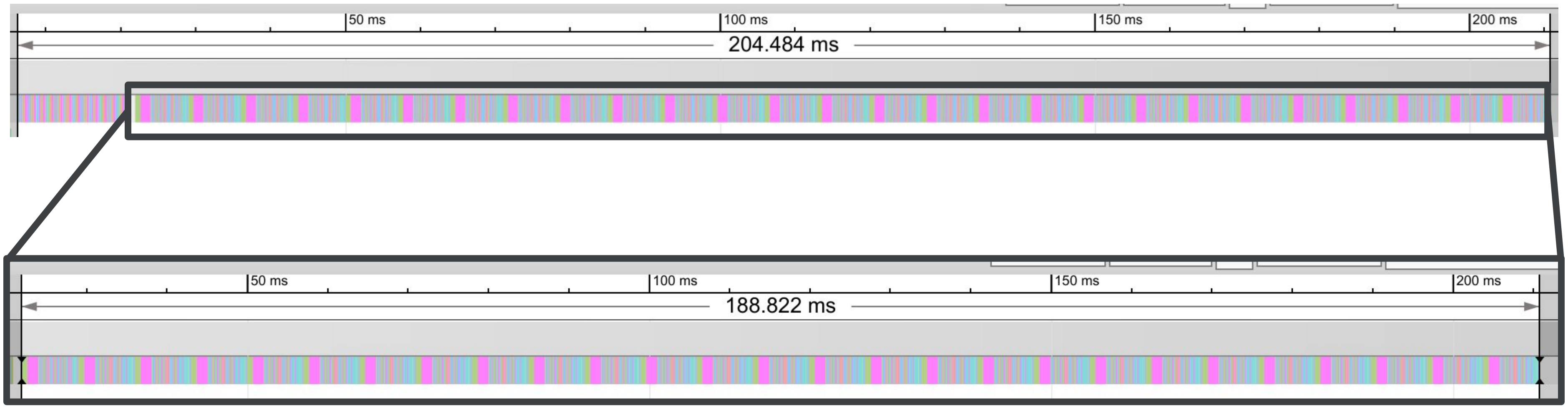
_I _was _walking _through _the _park _when **_I _was** _struck _by **_ a** _bicycle _ ... _my _arm _hurt s _ a _little _now _ . </s>

Seq2Seq, encoder



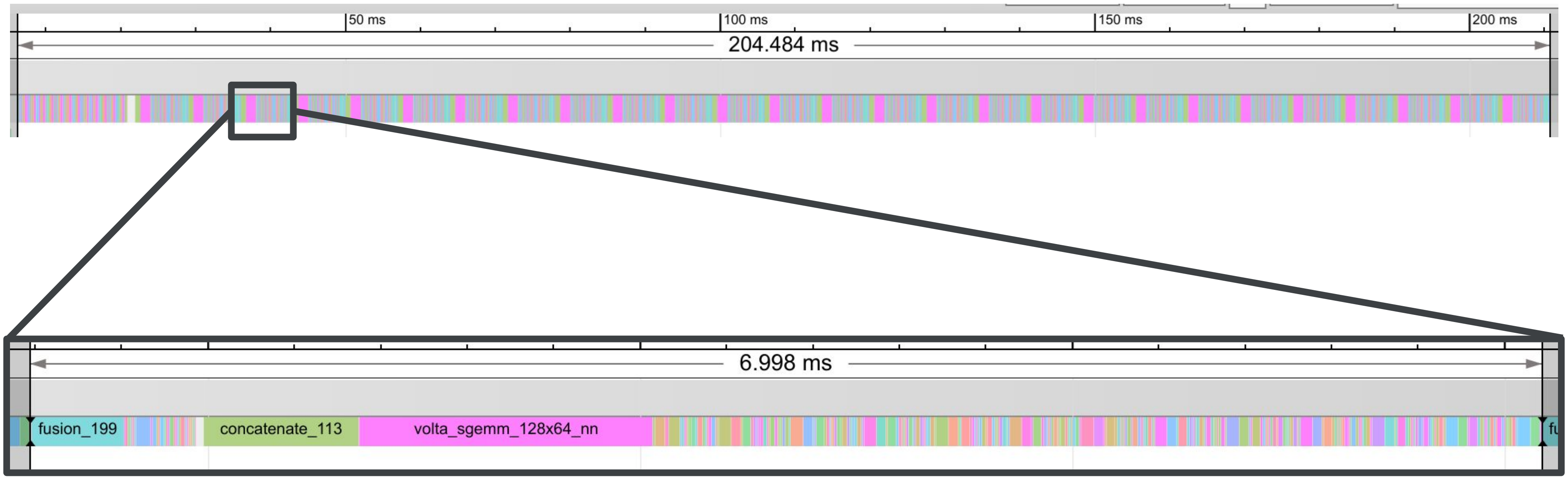
- Encoder takes 15ms

Seq2Seq, decoder



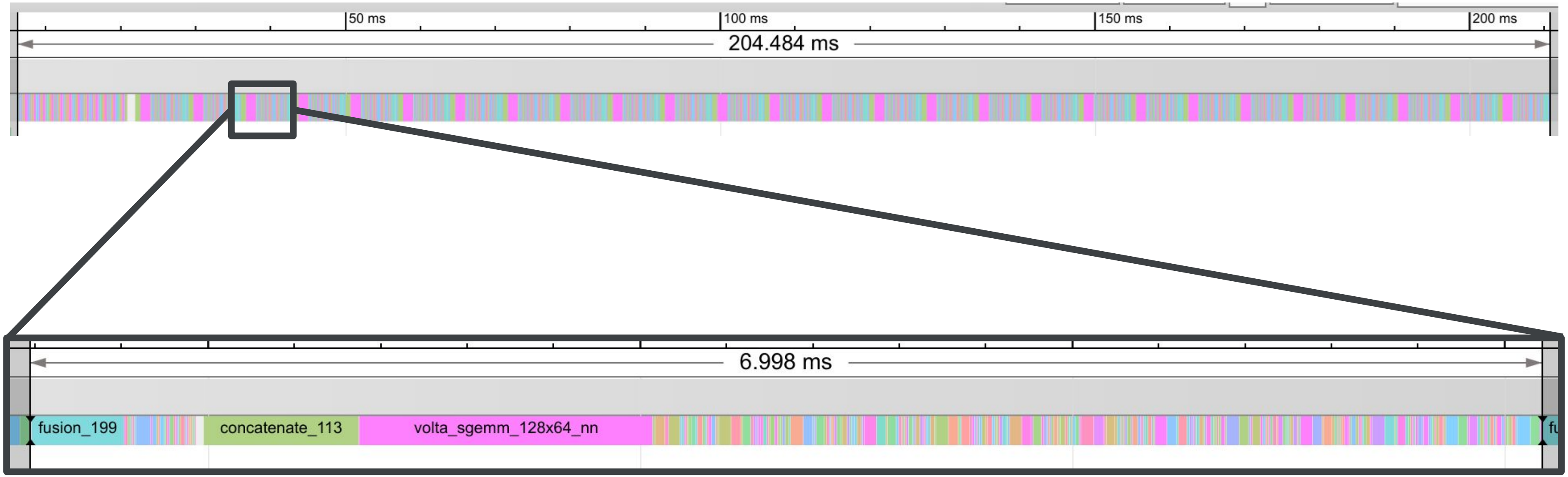
- Encoder takes 15ms
- Decoder takes 189ms

Seq2Seq, decoder step



- Encoder takes 15ms
- Decoder takes 189ms
- Single decoder step takes 7ms
  - 7 [ms/step] * 27 [steps] = 189ms

Seq2Seq, conclusions

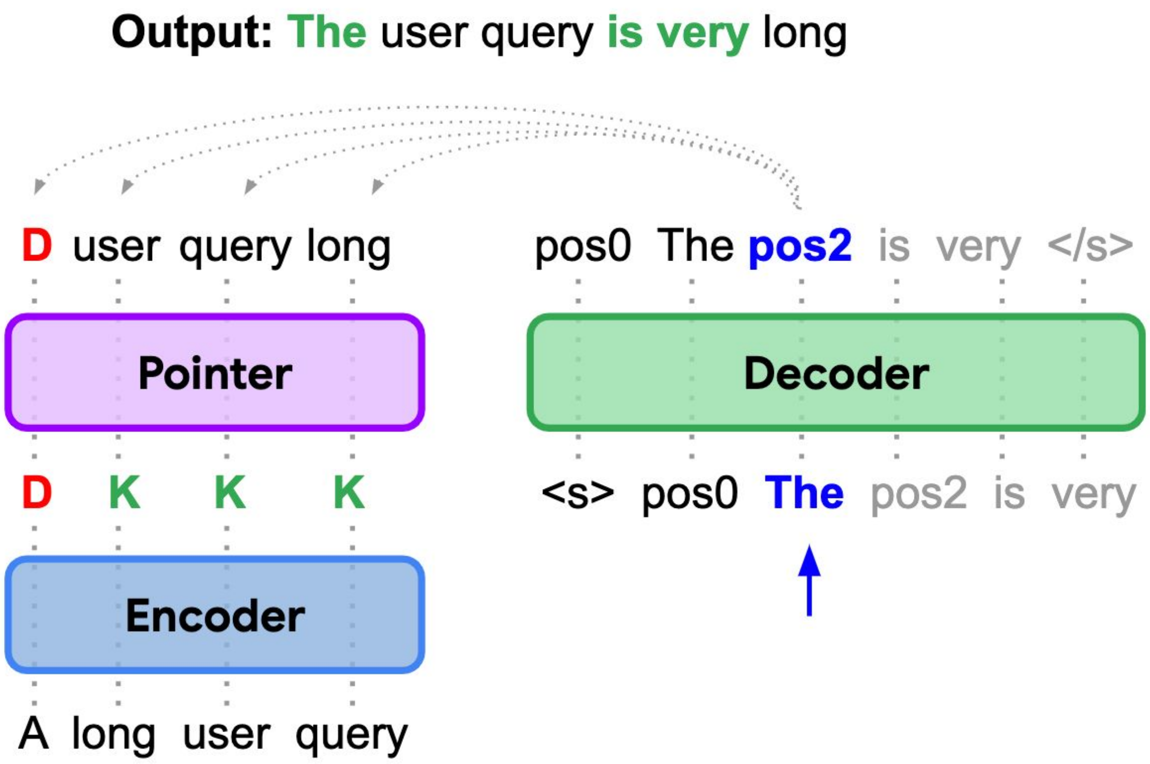


- Encoder takes 15ms
- Decoder takes 189ms
- Single decoder step takes 7ms
  - 7 [ms/step] * 27 [steps] = 189ms

If we want to reduce latency, target the decoder:

- Reduce the number of steps.
- Reduce the latency per step.

Refresher on EdiT5



Source: EdiT5 paper (Mallinson et al. 2022)

How does EdiT5 reduce latency?

- Use 1-layer decoder
  - Isn’t limited to text-editing models
- It moves work into the encoder
  - Tagging, Reordering
- Limit use of autoregressive decoder

GEC

Input to correct (21 tokens):

i was walking through the park when struck by bicycle... my arm hurts a little now.

Decoder output Seq2seq (27 tokens):

`_I _was _walking _through _the _park _when _I _was _struck _by _ a _bicycle _ ... _my _arm _hurt s _ a _little _now _ . </s>`

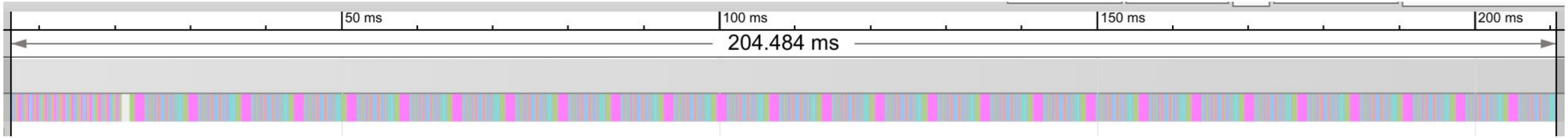
Decoder output EdiT5 (10 tokens)

`<extra_id_1> _I _was <extra_id_6> _I _was <extra_id_8> _ a </s>`

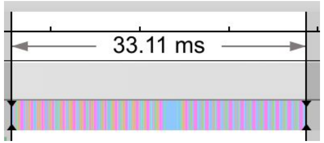
Note: extra ids are used to represent insertion positions.

EdiT5 vs Seq2Seq

Seq2seq model:



EdiT5 model:



How does EdiT5 reduce latency?

- Decoder step takes 1.3ms compared to 7ms
  - **5.4x** reduction
- There are 10 decoder steps, compared to 27
  - Another **2.7x** reduction

In summary: **14.5x** reduction in decoder latency compared to Seq2Seq, in exchange for **5ms** of overhead.

6-1. Faster Decoding

Paper

Fast Inference from Transformers via Speculative Decoding

Yaniv Leviathan * [1] Matan Kalman * [1] Yossi Matias [1]

Accelerating Large Language Model Decoding with Speculative Sampling

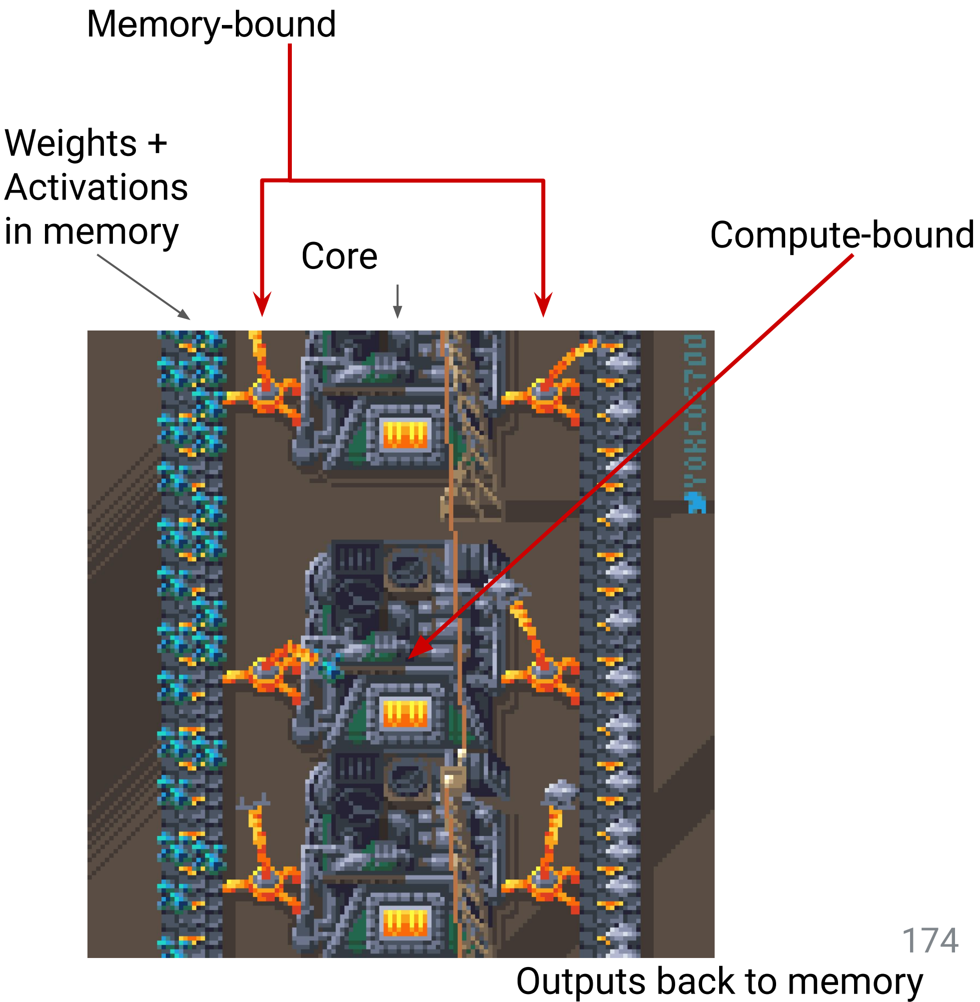
Charlie Chen[1], Sebastian Borgeaud[1], Geoffrey Irving[1], Jean-Baptiste Lespiau[1], Laurent Sifre[1] and John Jumper[1]

Other papers

- Sun et al. (2021) arxiv.org/abs/2106.04970
- Ge et al. (2022) arxiv.org/abs/2205.10350
- Leviathan et al. (2022) arxiv.org/abs/2211.17192
- Chen et al. (2023) arxiv.org/abs/2302.01318
- Kim et al. (2023) arxiv.org/abs/2302.07863

Why are decoders slow

- Transformer inference is slow
    - Largely memory-bound

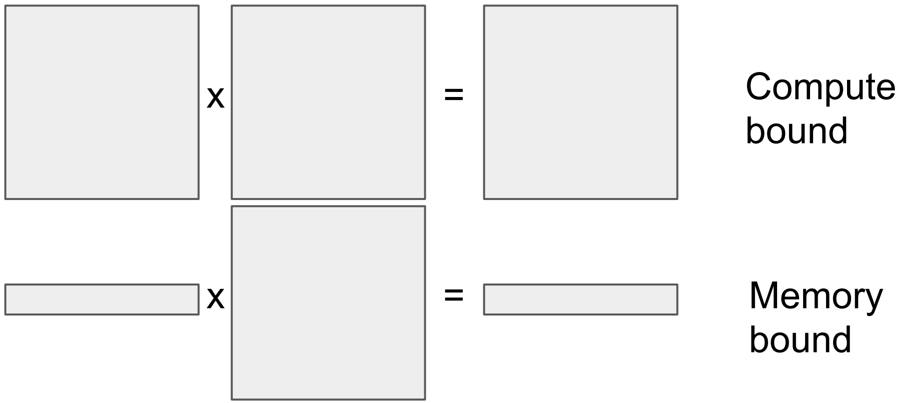


See [Making Deep Learning go Brrrr From First Principles]
(and also [got that iron smelting under control | pixel art : r/factorio])

Encoding v Decoding

Running a transformer decoder step with K tokens **scales sublinearly** with K

- Throughput: Batching
- Latency: Speculative Decoding

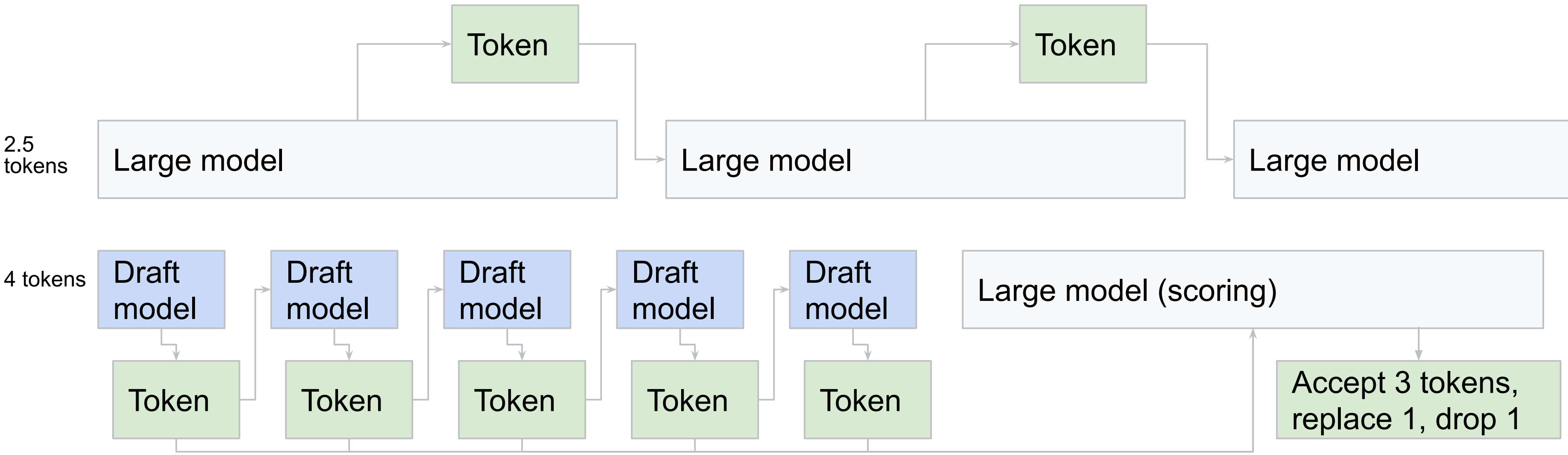


- Encoders are generally computer bound, we parallelize the encode
- whereas decoders are memory bound

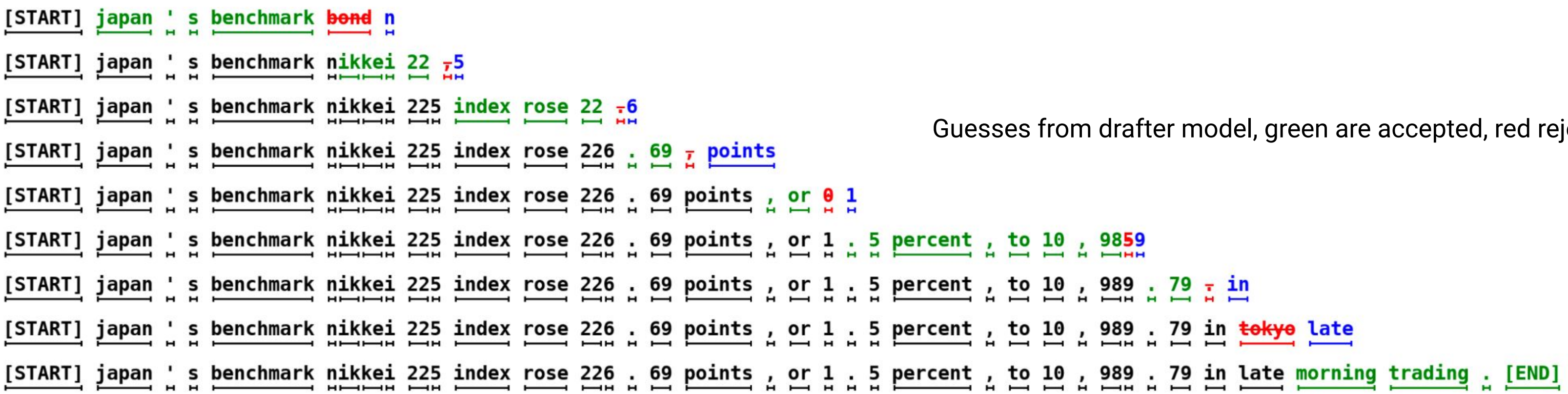
Solution: Batching, for latency!

- Have a drafter model, much smaller than the original model
- Decode (**AR**) many tokens from the drafter (span of gamma tokens)
- Use the large model to compute probabilities for all tokens **in parallel**
- Accept a prefix of the span

Batching, for latency!



Example



Guesses from drafter model, green are accepted, red rejected.

Making the distributions match

Drafter results: _my _favourite _pet _was _a _dog _named _rex

- Q distribution (drafter model) for each token
- P distribution (large model) for next token given prefix

Distributions can be: Sampling or greedy

Q - draft model

_dog: 0.5

_cat: 0.2

_the: 0.02

…

P - large model

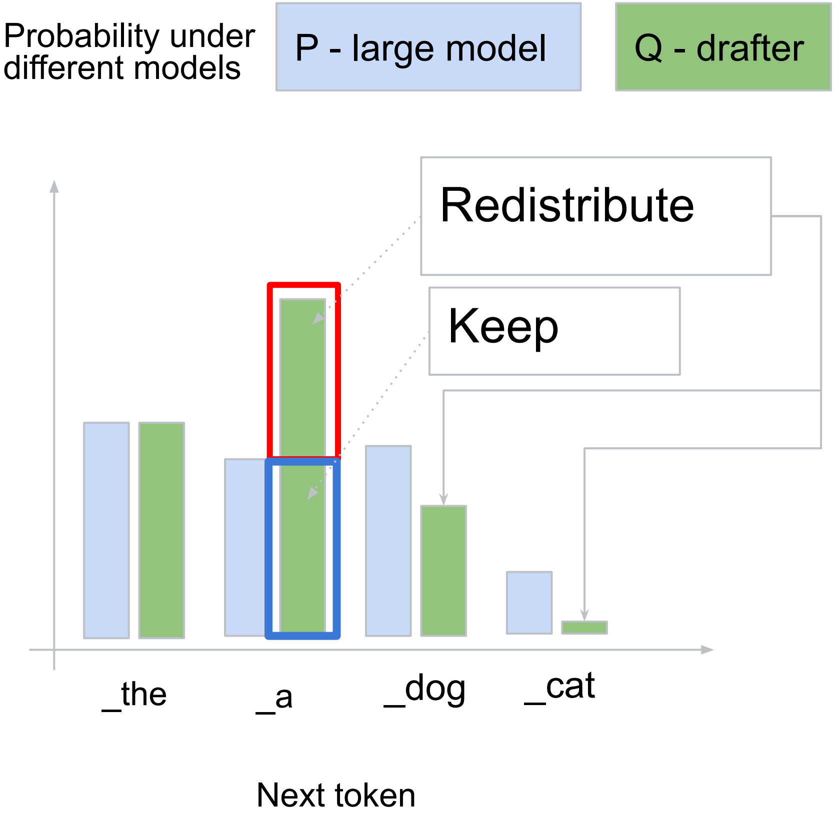
_dog: 0.4

_cat: 0.35

_the: 0.02

…

Making the distributions match



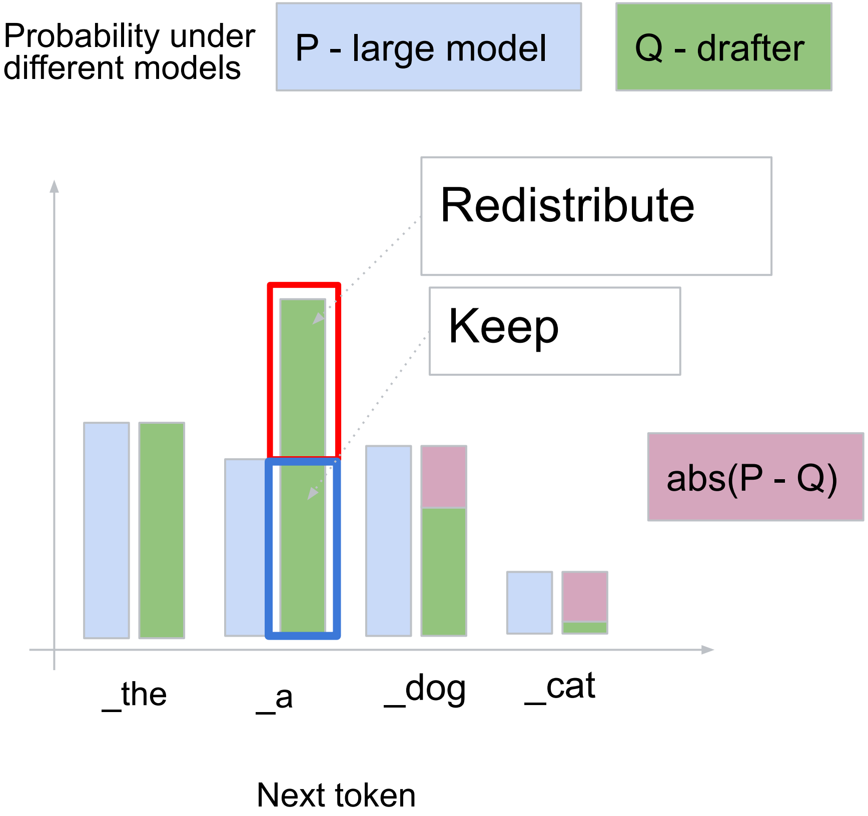
Case 1: Q(token) > P(token)

- Keep with probability P(token)/Q(token)
- Probability of sampling **and keeping** is now P(token).
- Reject: sample a new token from among those where Q(token) <= P(token), proportional to abs(P(token) - Q(token)).

Case 2: Q(token) <= P(token)

- Just accept.

Making the distributions match



Case 1: Q(token) > P(token)

- Keep with probability P(token)/Q(token)
- Probability of sampling **and keeping** is now P(token).
- Reject: sample a new token from among those where Q(token) <= P(token), proportional to abs(P(token) - Q(token)).

Case 2: Q(token) <= P(token)

- Just accept.

Tradeoffs

Constants

- Alpha: Per-token acceptance probability
- Gamma - Number of tokens we decode from the draft model for each token from the large model.

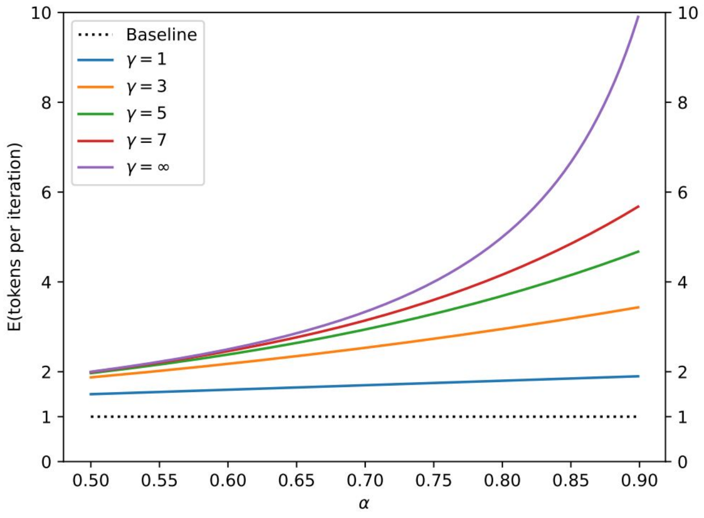


Figure 2. The expected number of tokens generated by Algorithm 1 as a function of α for various values of γ.

The drafter

- Small models
  - lower alpha but faster drafter inference
- Text-editing models
  - Need to support accepted tokens from the language model
- Statistical language models
  - Limited power
- Textual overlap with the input
  - Does this work for all cases

Results

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

TASK	M_q	TEMP	γ	α	SPEED
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
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

- Greedy easier than sampling
- Works even with extremely cheap drafters

Table 3. Empirical α values for various models M_p, approximation models M_q, and sampling settings. T=0 and T=1 denote argmax and standard sampling respectively[6].

M_p	M_q	SMPL	α
GPT-LIKE (97M)	UNIGRAM	T=0	0.03
GPT-LIKE (97M)	BIGRAM	T=0	0.05
GPT-LIKE (97M)	GPT-LIKE (6M)	T=0	0.88
GPT-LIKE (97M)	UNIGRAM	T=1	0.03
GPT-LIKE (97M)	BIGRAM	T=1	0.05
GPT-LIKE (97M)	GPT-LIKE (6M)	T=1	0.89
T5-XXL (ENDE)	UNIGRAM	T=0	0.08
T5-XXL (ENDE)	BIGRAM	T=0	0.20
T5-XXL (ENDE)	T5-SMALL	T=0	0.75
T5-XXL (ENDE)	T5-BASE	T=0	0.80
T5-XXL (ENDE)	T5-LARGE	T=0	0.82
T5-XXL (ENDE)	UNIGRAM	T=1	0.07
T5-XXL (ENDE)	BIGRAM	T=1	0.19
T5-XXL (ENDE)	T5-SMALL	T=1	0.62
T5-XXL (ENDE)	T5-BASE	T=1	0.68
T5-XXL (ENDE)	T5-LARGE	T=1	0.71
T5-XXL (CNNDM)	UNIGRAM	T=0	0.13
T5-XXL (CNNDM)	BIGRAM	T=0	0.23
T5-XXL (CNNDM)	T5-SMALL	T=0	0.65
T5-XXL (CNNDM)	T5-BASE	T=0	0.73
T5-XXL (CNNDM)	T5-LARGE	T=0	0.74
T5-XXL (CNNDM)	UNIGRAM	T=1	0.08
T5-XXL (CNNDM)	BIGRAM	T=1	0.16
T5-XXL (CNNDM)	T5-SMALL	T=1	0.53
T5-XXL (CNNDM)	T5-BASE	T=1	0.55
T5-XXL (CNNDM)	T5-LARGE	T=1	0.56
LAMDA (137B)	LAMDA (100M)	T=0	0.61
LAMDA (137B)	LAMDA (2B)	T=0	0.71
LAMDA (137B)	LAMDA (8B)	T=0	0.75
LAMDA (137B)	LAMDA (100M)	T=1	0.57
LAMDA (137B)	LAMDA (2B)	T=1	0.71
LAMDA (137B)	LAMDA (8B)	T=1	0.74

Questions?