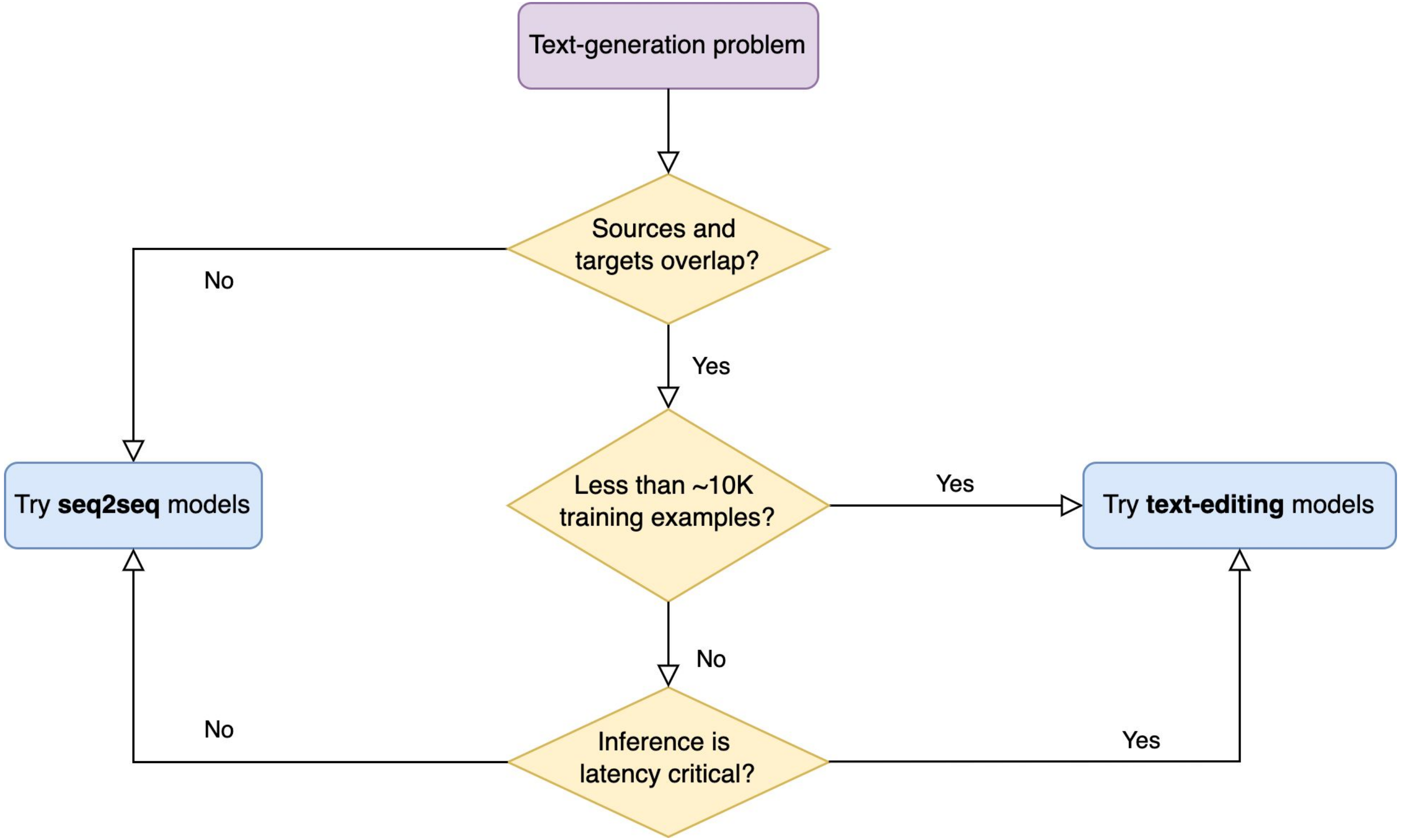
Recommendations and Future Directions

Recommendations



Open-sourced text-editing models

- EdiT5: *(to be released)*
- EditNTS: github.com/YueDongCS/EditNTS
- Felix: github.com/google-research/google-research/tree/master/felix
- GECToR: github.com/grammarly/gector
- LaserTagger: github.com/google-research/lasertagger
- LEWIS: github.com/machelreid/lewis
- PIE: github.com/awasthiabhijeet/PIE
- Seq2Edits: github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/models/research/transformer_seq2edits.py
- (Straka et al., 2021): github.com/ufal/wnut2021_character_transformations_gec

Tutorial Proposal

https://textedit.page.link/paper

Contains:

(1) a summary of the tutorial topics

(2) list of references

Open problems and future directions

1. Learned edit operations
2. Tokenization optimized for editing
3. Text-editing specific pre-training
4. Sampling
5. Scaling up text-editing models

Speculative Decoding

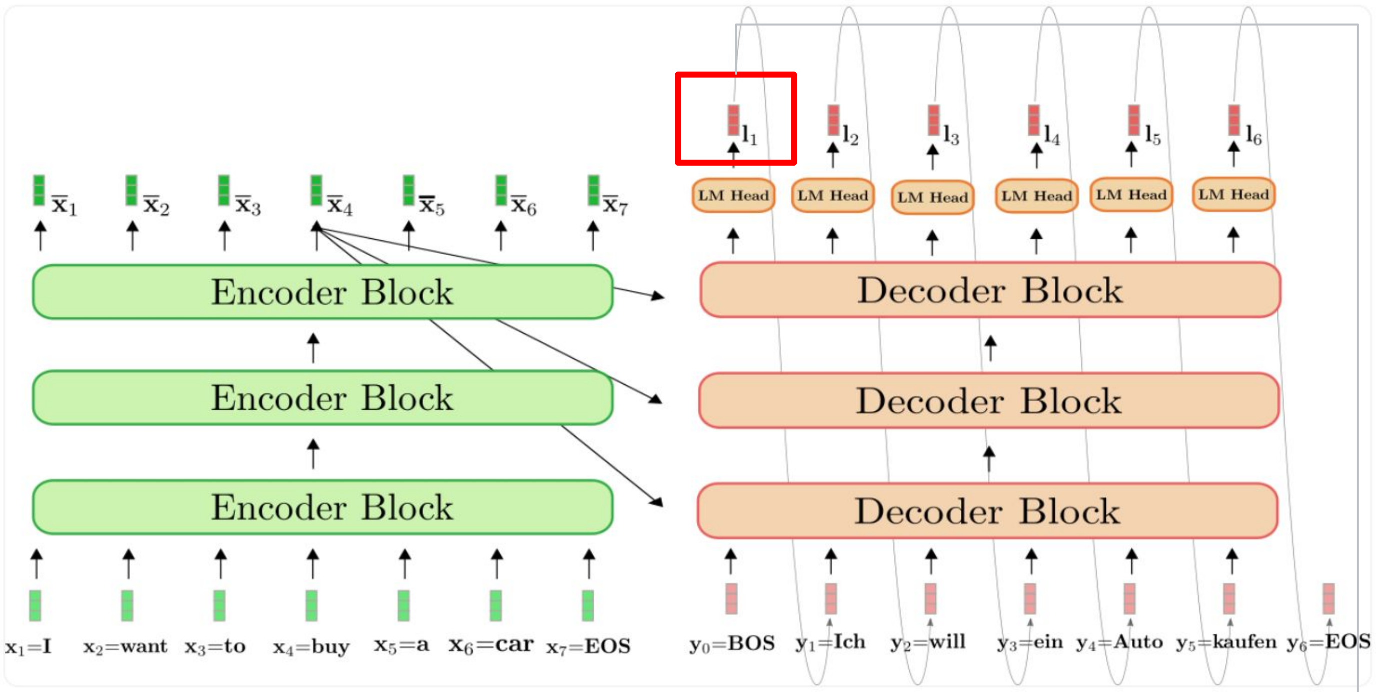
Papers

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]



Converting distribution (“Du”: 70%, “Ich”: 20%, etc.) to token:

- Greedy decoding,
- Sampling,
- TopK sampling, TopP sampling

Result: “Ich”

Transformer decoders are slow

Compared to what?

- The encoder.

Why is decoding from transformers slow?

Solutions

Throughput is a concern?

- Batching (do more compute for the memory we transfer)

Latency is a concern?

- Distillation into smaller model
  - Quality concerns
- Speculative Decoding!

Batching, for latency!

- Have a drafter model, much smaller than the original model

[START] japan ' s benchmark ~~bond~~ n

[START] japan ' s benchmark nikkei 22 ~~,~~5

[START] japan ' s benchmark nikkei 225 index rose 22 ~~.~~6

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~,~~ points

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or ~~0~~ 1

[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]

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

Making the distributions match

Drafter results: _my _favourite _pet _was _a _dog _named _rex

Q - draft model

_dog: 0.5

_cat: 0.2

_the: 0.02

…

P - large model

_dog: 0.4

_cat: 0.35

_the: 0.02

…

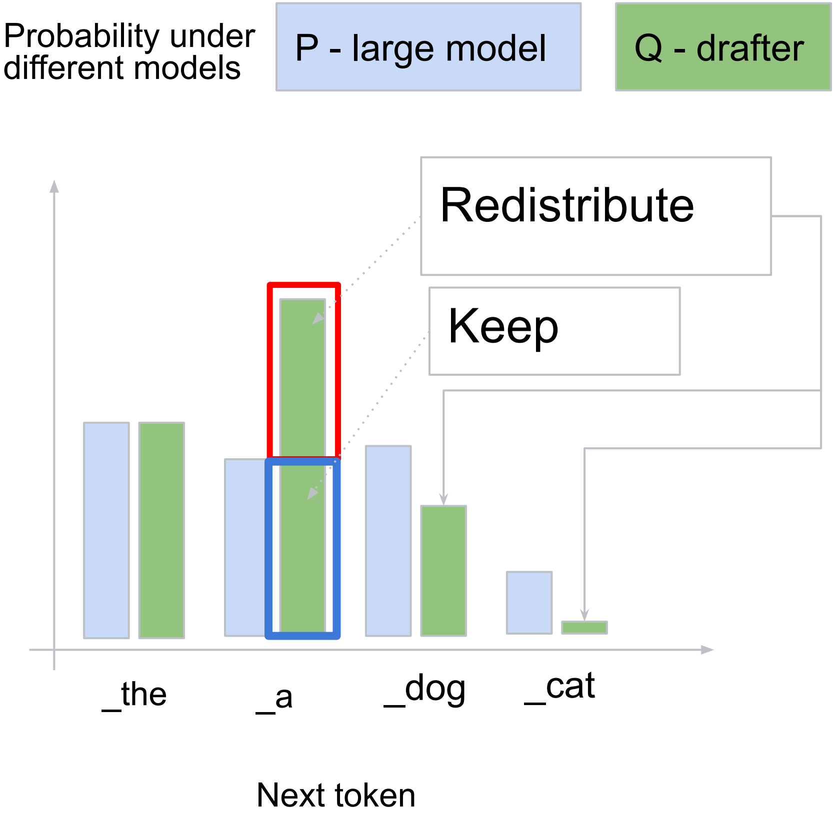
What we have:

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

Distributions can be:

- Vanilla sampling

Making the distributions match



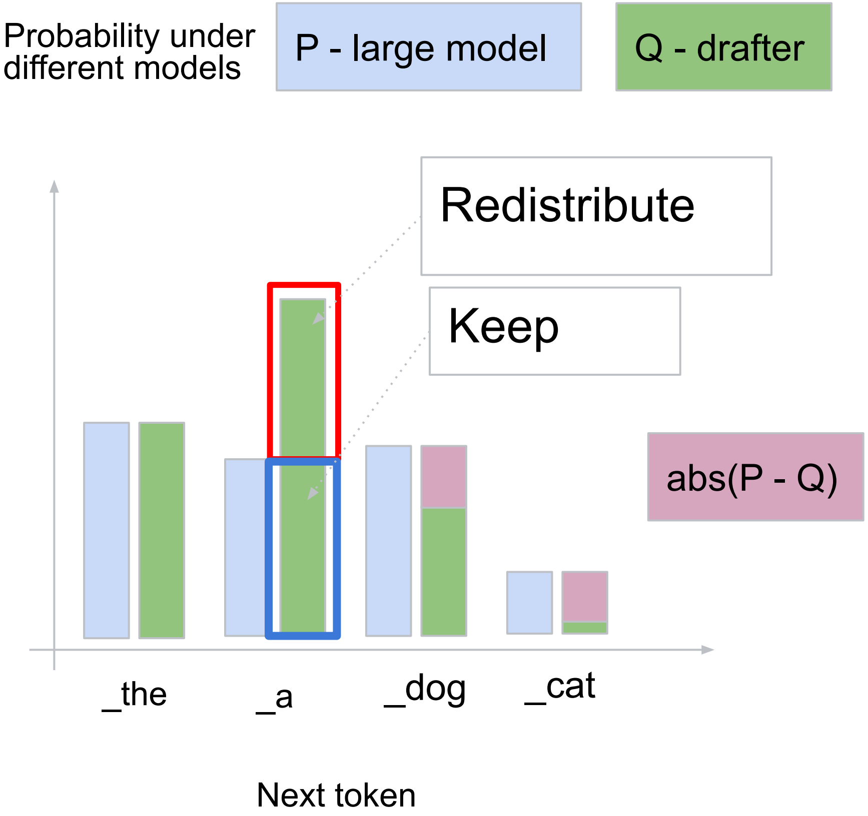
Case 1: Q(token) > P(token)

- 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 prob
- Gamma - Number of tokens we d from the draft model for each token from the large model.
- Latency of drafter / large

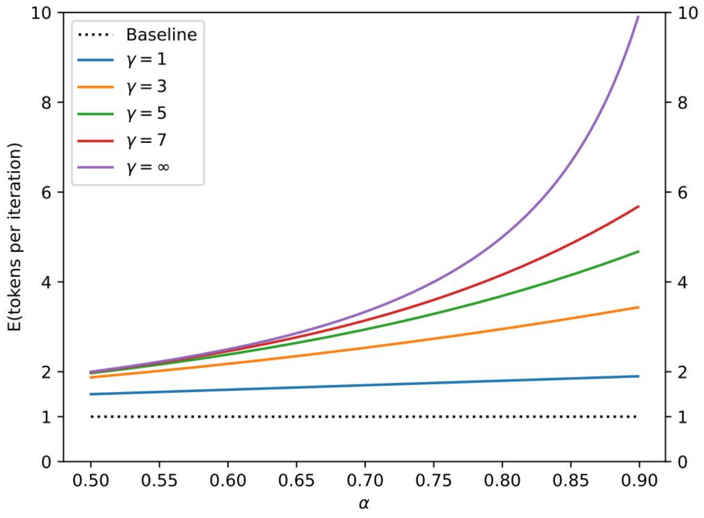


Figure 2. The expected number of tokens generated by Algorithm 1 as a function of α for various values of γ.

Theoretical analysis

C - latency of small/large

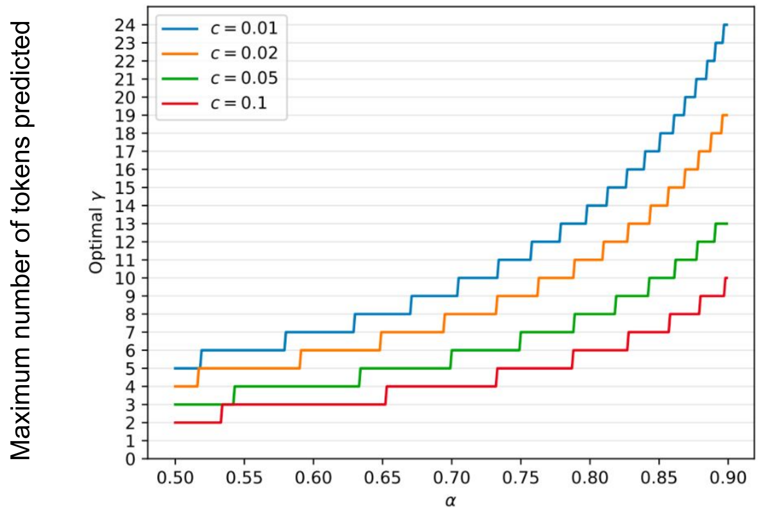


Figure 3. The optimal γ as a function of α for various values of c.

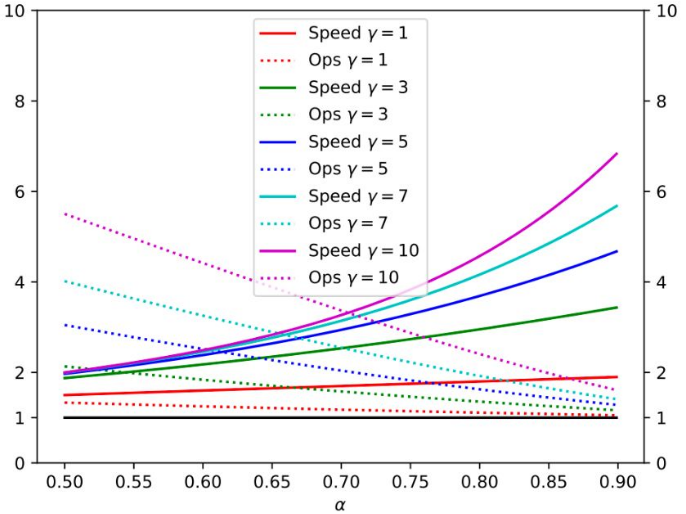


Figure 4. The speed improvement and increase in number of operations as a function of α for various values of γ.

- Quality drops slower tha
latency gains up to T5 sr

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 samplin
- Works (sort of) even with e
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

Table 1 | **Chinchilla performance and speed on XSum and HumanEval with naive and speculative sampling at batch size 1 and $K = 4$.** XSum was executed with nucleus parameter $p = 0.8$, and HumanEval with $p = 0.95$ and temperature 0.8.

Sampling Method	Benchmark	Result	Mean Token Time	Speed Up
ArS (Nucleus)	XSum (ROUGE-2)	0.112	14.1ms/Token	1×
SpS (Nucleus)		0.114	7.52ms/Token	1.92×
ArS (Greedy)	XSum (ROUGE-2)	0.157	14.1ms/Token	1×
SpS (Greedy)		0.156	7.00ms/Token	2.01×
ArS (Nucleus)	HumanEval (100 Shot)	45.1%	14.1ms/Token	1×
SpS (Nucleus)		47.0%	5.73ms/Token	2.46×

Thank You