zip2zip: Inference-Time Adaptive Tokenization via Online Compression

Research group discussion · 25-minute main talk

Paper by Saibo Geng, Nathan Ranchin, Yunzhen Yao,
Maxime Peyrard, Chris Wendler, Michael Gastpar, Robert West

arXiv:2506.01084v2 · Reading: 12 November 2026

Online tokens save steps, with a quality cost

Central idea

Build context-specific tokens during inference, compute their vectors from base tokens, and generate them directly.

- Tokenizer tests: roughly 13–41% fewer tokens; strongest gains in code and math.
- H100 inference throughput improves; GSM8K accuracy drops sharply.
- Published at NeurIPS 2025, main conference.
- Official implementation and compression library are publicly available.

Paper v2, Tables 3, 5, 9, PDF pp. 8–9, 19. Publication/code checked 2026-11-12.

Fixed vocabularies repeatedly pay for the same fragment

Problem

A domain-specific phrase may require several base tokens every time it appears, even within one context.

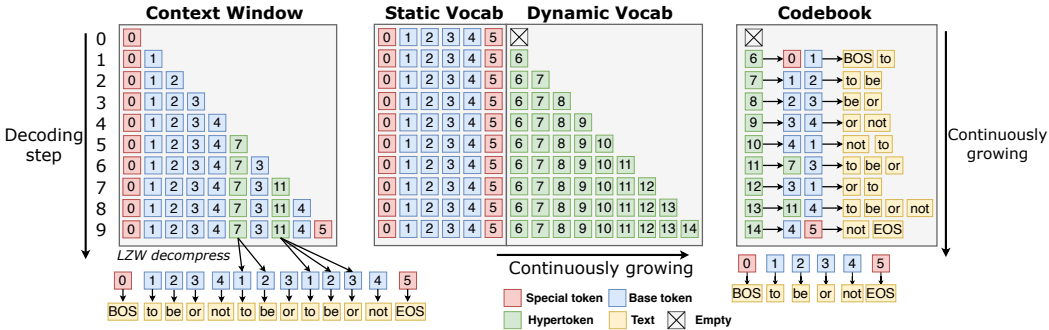
- Larger static vocabularies still cannot cover every local distribution.
- Domain-specific vocabulary expansion requires an adaptation stage.
- Input compression alone leaves output decoding steps unchanged.

zip2zip adapts the active vocabulary in the current session.

The base tokenizer remains in place; an online compression layer sits above it.

Paper, Secs. 1 and 4, PDF pp. 1–3, 9–10.

Context and dictionary evolve together



Blue: base tokens. Green: hypertokens. Pink: special tokens.

Read each row as a decoding step, then follow the codebook on the right.

Original Fig. 1, PDF p. 2. Illustration permits longer entries; main experiments cap merge size at $M = 3$.

One hypertoken replaces several predictions

Dictionary entries in Fig. 1

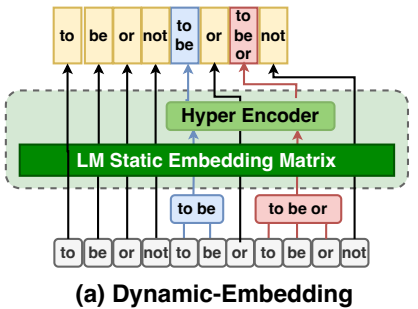
$$1 \mapsto \texttt{to}$$
$$2 \mapsto \texttt{be}$$
$$7 \mapsto (1, 2)$$
$$11 \mapsto (7, 3)$$

Thus 7 means `to be`;
11 means `to be or`.

Paper, Sec. 2.1, PDF p. 3; Fig. 1, PDF p. 2.

- Start each session with $\mathcal{V}_h = \varnothing$.
- LZW grows a dictionary from the token stream; later occurrences reuse entries.
- A single model step can emit multiple base tokens.
- M limits constituent base-token count, including nested merges.

A shared encoder gives unseen hypertokens a vector



(a) Dynamic-Embedding

$$h = f_\phi(y_{1:M}) \in \mathbb{R}^d$$

- Look up constituent base embeddings: $M \times d$ values.
- Pad shorter compositions; map them to one d-dimensional vector.
- Main configuration: a two-layer Transformer encoder, $M = 3$.
- Base-token lookup stays available.

Original Fig. 2(a), cropped; Secs. 2.2 and 3.1, PDF pp. 3–4, 6–7.

The output head must also admit new tokens

$$p(v \mid z_{\leq t}) = \frac{\exp(s_t^\top u_v)}{\sum_{w \in \mathcal{V} \cup \mathcal{V}_h(t)} \exp(s_t^\top u_w)}$$

- $s_t \in \mathbb{R}^d$: Transformer hidden state.
- u_v: static output vector for a base token; dynamically computed vector for a hypertoken.
- Tied input/output weights permit a shared hyper-encoder; untied weights require separate mappings.
- Keep input dictionary, output candidates, and decoder consistent.

Teaching expansion of the joint-softmax mechanism; paper, Secs. 2.2–2.3, Fig. 3, PDF pp. 3–5.

Train the model on the representation it will generate

$$z = \mathcal{C}(y), \qquad \min_{\theta,\phi} \mathbb{E}_{y\sim\mathcal{D}} \left[-\log \pi_{\theta,\phi}(\mathcal{C}(y)) \right]$$

- $\mathcal{C}$: lossless online compression of base-token sequences.
- Compress documents independently before training.
- Teacher forcing predicts the next base token or hypertoken in z.
- Adapt the backbone with LoRA and train the dynamic embedding modules.

Training/inference interface

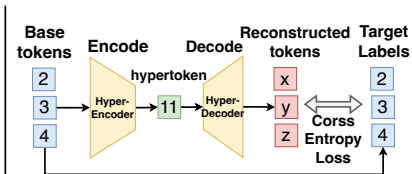
Training precomputes a sequence codebook. The paper does not fully specify per-position masking of dictionary entries created in the future.

Paper, Eq. (2), Sec. 2.4, PDF p. 5; Sec. 3.1, PDF pp. 6–7. Interface observation is reading analysis.

Reconstruction preserves the token composition

$$\mathcal{L} = \mathcal{L}_{\text{CLM}} + \lambda\, \mathbb{E}_{y_{1:M}} \left[\Delta\big(y_{1:M}, f_\psi(f_\phi(y_{1:M}))\big) \right]$$

- f_ψ reconstructs constituent base tokens.
- Δ: reconstruction cross-entropy; $\lambda = 0.1$.
- Exact text decompression uses the symbolic codebook.



(c) AutoEncoding Loss

Paper, Eq. (3), PDF p. 5; original Fig. 2(c), cropped, PDF p. 4.

Caching avoids repeated work; the dictionary still grows

1. Compress the prompt and initialize its dictionary.
2. Compute vectors for new entries; cache them by composition.
3. Run the Transformer on the compressed sequence.
4. Sample over base and current hyper-vocabulary; update and repeat.

- Embeddings are context-independent; hidden states are context-dependent.
- Dictionary and embedding cache grow with context length.
- More output candidates also increase projection and memory costs.
- Implementation uses caching, Triton kernels, and a Rust tokenizer.

Paper, Secs. 2.3 and 3.6, PDF pp. 4–5, 9; App. F, PDF pp. 20–21.

Lossless compression preserves total sequence entropy

For a bijection g onto its image, define transported distributions:

$$P_Z(z) = P_X(g^{-1}(z)), \qquad p_\gamma(z) = p_\theta(g^{-1}(z)).$$

Then, by changing the summation variable,

$$H(P_Z) = H(P_X), \qquad H(P_Z, p_\gamma) = H(P_X, p_\theta).$$

Interpretation

An equivalent model distribution exists in compressed space. The trained finite model must still learn to approximate it.

This concerns total sequence entropy; normalization per token changes.

Paper, Sec. 2.6, PDF p. 6; proof in App. G, PDF pp. 20–21.

Separate adaptation from compression with three variants

Variant	Representation	Updated components
Base	Original tokens	None
Cont. pretrain	Original tokens	LoRA
zip2zip	Compressed tokens	LoRA + hyper-modules

- Reported model families: Phi-3.5-4B and Phi-3-14B (table labels).
- Main setting: $M = 3$, two-layer hyper-encoder, $\lambda = 0.1$.
- Data mix: code 25%; math, chat, web 20% each; multilingual 15%.
- Training cost/configuration reports conflict across sections; details in backup.

Paper, Sec. 3.1, PDF pp. 6–7; Apps. I–J, PDF pp. 22–24.

Structured domains show the largest tokenizer gains

Tokenizer	Domain	Base η	+zip2zip η	Fewer tokens
Llama-3	Code	4.1	6.3	34.9%
Llama-3	Math	2.7	4.0	32.5%
Llama-3	Chat	5.1	6.4	20.3%
Llama-3	Multilingual	3.8	4.7	19.1%
Llama-3	Web	4.6	5.4	14.8%
Gemma-3	Code	3.3	5.6	41.1%

$$\eta = \frac{\text{UTF-8 bytes}}{\text{tokens}} \uparrow, \qquad \text{token reduction} = 1 - \frac{\eta_{\text{base}}}{\eta_{\text{z2z}}}$$

These are tokenizer tests, including vocabularies up to 256K.

Paper, Table 2, PDF p. 8; Table 9, PDF p. 19. Reductions here computed from rounded Table 2 values.

Byte-level perplexity rises on every reported corpus

Model	Method	Wiki	Pile	mC4	dC4
4B	Cont. pretrain	1.59	1.81	1.88	1.74
	zip2zip	1.69	1.95	2.00	1.82
14B	Cont. pretrain	1.47	1.79	1.86	1.68
	zip2zip	1.56	1.90	1.96	1.75

$$\mathrm{BytePPL} = \exp\left[-\frac{\log Q(x)}{B}\right] = \mathrm{PPL}^{1/\eta} \qquad \downarrow$$

B: UTF-8 byte count; $Q(x)$: autoregressive sequence likelihood.
1024 model-token contexts imply different amounts of original text.

Paper, Sec. 3.4 and Table 3, PDF p. 8.

GSM8K reveals a major quality cost

GSM8K accuracy ↑	4B	14B
Base	.82	.84
Cont. pretrain	.40	.52
zip2zip	.15	.25

Extra drop vs. matched adaptation

4B: 25 percentage points.
14B: 27 percentage points.

- Other two-shot benchmarks remain much closer to continued pretraining.
- 4B PIQA: .82 vs. .82; WinoGrande: .75 vs. .75.
- 14B HellaSwag: .68 vs. .66; PIQA: .85 vs. .87.
- Token-boundary sensitivity is an author hypothesis for the math failure.

Paper, Sec. 3.5, Table 3, PDF p. 8. Bootstrap standard deviations reported as approximately .02.

Translation quality also depends on the language pair

Model	Method	Fr BLEU	De BLEU	Ro BLEU	Ro TER
4B	Cont. pretrain	36.5	42.3	16.7	79.7
	zip2zip	34.1	39.7	14.3	93.5
14B	Cont. pretrain	38.9	48.4	21.8	68.3
	zip2zip	36.4	44.8	19.5	72.9

- WMT14 En–Fr; WMT16 En–De and En–Ro; average over both directions.
- BLEU ↑ measures overlap; TER ↓ measures editing needed.
- 4B En–Ro TER worsens by 13.8 points: degradation is not uniformly small.

Paper, Table 4, PDF p. 9; Table 11 bootstrap TER: 79.7 ± 2.3 vs. 93.5 ± 2.5, PDF p. 22.

H100 throughput improves; M1 prefill can become slower

Hardware / model	Input + output	Prefill: Base→z2z	Decode: Base→z2z
H100 / 4B	512+256	1347.2→2722.1	54.4→79.8
H100 / 14B	1024+256	2328.6→3657.0	45.1→66.8
M1 / 4B	256+256	165.0→145.5	7.3→7.9
M1 / 4B	1024+256	200.9→189.6	7.1→7.4

- Values are reported tokens/s, split by inference phase.
- Baselines: HuggingFace Transformers on H100; MLX on M1.
- Token accounting for hypertokens versus expanded base tokens is not fully detailed in Table 5.

Paper, Sec. 3.6, Table 5, PDF p. 9. Absolute values retained; some printed relative percentages differ.

Three efficiency metrics need different denominators

Same text, different token counts

$$\rho = \frac{N_{\text{comp}}}{N_{\text{orig}}} = \frac{\eta_0}{\eta_1}$$

A 50% gain in bytes/token means
$1 - 1/1.5 = 33.3\%$ fewer tokens.

Same work, different processing times

$$\frac{T_1}{T_0} = \frac{R_0}{R_1}$$

45.1→66.8 tokens/s means
48.1% more throughput, but
32.5% less time for equal work.

Equal original-text workload and complete pipeline timing are needed to turn phase throughput into an end-to-end latency claim.

Teaching calculations using the metric definitions and Table 5; PDF pp. 7–9. R: throughput; T: time.

Merge size 3 is a useful compromise

Maximum merge size M	Wiki Byte-PPL ↓	Pile ↓	Retained tokens ρ ↓
1	1.62	1.70	100.00%
2	1.96	2.21	75.30%
3	1.72	1.84	71.21%
4	1.71	1.84	68.93%
5	1.72	1.84	68.41%

- $M = 2$ has the worst quality among compressed settings.
- Moving from $M = 3$ to 5 saves only another 2.8% of original tokens.
- PPL: 1024-token contexts; compression: 2048-token contexts.

Paper, App. B, Table 6, PDF p. 17; training-loss trends in Fig. 5, PDF p. 18. Separate ablation run.

Learned composition matters more than encoder depth

Hyper-encoder	Wiki	Pile	mC4	dC4
Averaging	1.81	1.97	2.29	2.08
One attention layer	1.73	1.86	2.16	2.01
One Transformer layer	1.71	1.83	2.13	1.99
Two Transformer layers	1.72	1.84	2.15	2.00

- All values: byte-level perplexity ↓, 1024-token contexts.
- Trainable composition improves substantially over averaging.
- The one-layer Transformer is slightly better on all four columns; the table does not show a two-layer advantage.

Paper, App. C, Table 7, PDF p. 18; training-loss curves in Fig. 6, PDF p. 19.

A complete interface for dynamic token prediction

1. **Build**: online dictionary compression creates context-specific units.
2. **Represent**: shared hyper-encoders produce vectors for unseen compositions.
3. **Predict**: compressed CLM and a joint output head generate those units directly.

Practical judgment

Compression is substantial, especially for structured text. Quality loss on numerical reasoning and some translation settings is a central cost.

Only about 25% of hypertokens are reused during generation. Selective retention and codebook prefilling are proposed directions, not evaluated solutions.

Paper, Secs. 5–6, PDF p. 10; quantitative evidence in Tables 2–5 and 9.

References and discussion materials

- Geng et al. *zip2zip: Inference-Time Adaptive Tokenization via Online Compression*. arXiv:2506.01084v2, 24 October 2025.
- NeurIPS 2025 official proceedings, volume 38, main conference. DOI: 10.52202/085713-5077.
- epfl-dlab/zip2zip: official model and tokenizer implementation.
- epfl-dlab/zip2zip-compression: official compression component.

This deck discusses the arXiv v2 paper. External publication and code status were checked on 12 November 2026.

Companion Chinese notes: reading-notes.zh.md. Original figures are distributed with this deck.

Backup: multilingual drift has two sources

Task	Language	Base	Cont. pretrain	zip2zip
HellaSwag	Spanish	.66	.57	.55
HellaSwag	Arabic	.44	.27	.27
Winograd	Chinese	.74	.61	.64
TruthfulQA-2	Chinese	.41	.46	.43

- Phi-3.5-4B; accuracy ↑; selected rows from five-language evaluation.
- zip2zip is usually close to continued pretraining on these tasks.
- This does not imply complete preservation of the original model's multilingual performance.

Paper, App. H.2, Table 12, PDF p. 22. Missing task-language combinations in the original are NA.

Backup: training budgets conflict across sections

Item	Main text / one report	Appendix / other report
4B / 14B GPU-hours	About 10 / up to 40	14.0 / 63.0 (Table 13)
Tokens	0.1B training tokens	1B compressed-token dataset
LoRA rank / alpha	32 / 32 (I.1)	16 / 16 (I.2)
Hardware	4×A100 (I.4)	H100 (I.5, Table 13)

- I.1 also lists 32768 tokens/step and 10000 steps: about 0.328B processed tokens if used together.
- Dataset size and consumed tokens can differ; experiments are not explicitly reconciled.
- Treat the abstract's 10 GPU-hours as a headline claim, not every model's measured cost.

Paper, Sec. 3.1, PDF p. 7; Apps. I–J and Table 13, PDF pp. 22–24.

Backup: hypertoken fraction and token savings differ

Example	Code	Biomedical	French
Original tokens	344	322	399
Compressed tokens	265	270	367
Hypertoken count	44	35	26
Hypertoken fraction	16.6%	13.0%	7.1%
Token reduction	23.0%	16.1%	8.0%

- Fraction: hypertoken count / compressed sequence length.
- Reduction: 1 − compressed length / original length.
- Sec. 3.2's larger example ratios are not reconciled with Table 8.

Paper, Table 8, PDF p. 19; reductions computed from its counts. Sec. 3.2 and Fig. 4, PDF p. 7.

Backup: compression must offset module overhead

$$\frac{\Gamma_{\mathrm{z2z}}}{\Gamma} \approx \rho(1+\alpha), \qquad \alpha \approx \frac{lM}{L}$$

- ρ: retained token fraction; l: hyper-encoder layers; L: backbone layers; M: maximum merge size.
- Teaching example: $\rho = .7$, $\alpha = .1$ gives .77 of baseline FLOPs.
- Appendix E prints $12d_{\mathrm{model}} \ll n$ while neglecting quadratic attention cost; the stated condition needs clarification.
- A trainable-parameter count alone also misses frozen-backbone forward work under LoRA.

Measure pipeline time to resolve the competing costs.

Paper, App. E, PDF pp. 19–20; caveat on the printed approximation is reading analysis.

Backup: reindexing proves entropy invariance

$$
\begin{aligned}
H(P_Z, p_\gamma) &= -\sum_{z \in g(\mathcal{X}^*)} P_X(g^{-1}(z)) \log p_\theta(g^{-1}(z)) \\
&= -\sum_{x \in \mathcal{X}^*} P_X(x) \log p_\theta(x) = H(P_X, p_\theta).
\end{aligned}
$$

- Bijection onto the image is the key condition for reindexing the sum.
- The statement preserves sequence likelihood and total cross-entropy.
- It does not establish that the chosen finite Transformer/LoRA family realizes every transported model.
- Auxiliary reconstruction and symbolic decompression have different roles.

Paper, Sec. 2.6 and App. G, PDF pp. 6, 20–21; model-family interpretation is reading analysis.